Computers & Security 151 (2025) 104315

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Full length article

SCASS: Breaking into SCADA Systems Security

Nicola d'Ambrosio, Giulio Capodagli, Gaetano Perrone *, Simon Pietro Romano

University of Napoli Federico II, Department of Electrical Engineering and Information Technology, Via Claudio 21, 80125 Napoli, Italy

ARTICLE INFO

Keywords:
ICS cybersecurity
SCADA systems
ICS cyber-ranges
ICS testbeds

ABSTRACT

Industrial Controls Systems (ICS) represent a relevant target for attackers. In order to prevent such critical security threats, ICS security assessment activities should be conducted. Conventional vulnerability assessment and penetration testing within ICSs are not practicable due to safety risks and cost constraints. To overcome these challenges, security researchers have developed cybersecurity testbeds. However, these testbeds commonly rely on closed components, cannot be extended, and are very expensive. This research investigates how a modular, open-source framework can enhance the development of robust cybersecurity testbeds and facilitate the implementation of digital twins for securing Industrial Control Systems. We present SCASS, a fully customizable testbed designed to replicate complex SCADA and ICS environments with high fidelity. SCASS addresses the need for accessible, scalable platforms by supporting both physical and virtual components while enabling the evaluation of heterogeneous attack scenarios and security methodologies. By combining advanced attack scenarios with an objective comparative analysis against existing testbeds, SCASS demonstrates its ability to fill critical gaps in the ICS security landscape, fostering collaboration and advancing security assessment methodologies.

1. Introduction

Cybersecurity incidents significantly damage industrial systems (Fortinet, 2024), as confirmed by the spread of relevant attacks against sensitive *Industrial Control System* (ICS) infrastructures (Langner, 2011; Fidler, 2011; Kara and Aydos, 2019; Anon, 2023j,b,i).

Security researchers aim to address such challenges by developing digital twins, i.e., detailed simulations that accurately replicate the complexities of real-world Industrial Control Systems through virtualization, simulation, and physical reproduction (Dietz and Pernul, 2020; He et al., 2019; Varghese et al., 2022).

Despite the advantages offered by full virtualization, careful considerations are necessary when replicating Industrial Control Systems, particularly with respect to maintaining domain fidelity of the replicated physical system (Howard, 2019). Hybrid testbeds aim to address such a problem by combining both physical and virtual elements. However, most current testbeds neither publicly release their source code nor provide in-depth details for reproducing the networking layer and attack scenarios. We aim to fill this gap by presenting*SCADA Systems Security* (SCASS), a compact, hybrid, and lightweight environment designed for creating security testbeds for ICSs. Housed within a small cabinet, SCASS integrates physical components commonly employed in ICSs alongside a Raspberry Pi – a small, single-board computer capable of virtualizing the networking stack. Leveraging Linux container technology, SCASS creates a lightweight and high-performance virtualization environment that effectively simulates the components and network interactions of ICSs.

The work is structured as follows. Section 2 introduces the domain of Industrial Control Systems and related security practices. Section 3 examines related works, with a particular focus on the EPIC testbed and its digital twin named EPICTWIN. Section 4 illustrates the SCASS architecture, while Section 5 exemplifies the testbed's efficacy through the replication of the EPIC testbed. Section 6 illustrates an ICS attack scenario against the replicated EPIC testbed, validating how SCASS can effectively reproduce real-world cybersecurity challenges. Section 7 provides an in-depth analysis of SCASS, benchmarking it against other hybrid testbeds and highlighting both its limitations and proposed improvements. Section 8 concludes the paper by summarizing the contributions of this research while also offering thoughtful recommendations for future enhancements and extensions of the proposed methodology.

2. Background

This section provides an overview of the foundational concepts related to Industrial Control Systems (ICS), covering key industrial

* Corresponding author.
E-mail addresses: nicola.dambrosio2@unina.it (N. d'Ambrosio), g.capodagli@studenti.unina.it (G. Capodagli), gaetano.perrone@unina.it (G. Perrone), spromano@unina.it (S.P. Romano).

https://doi.org/10.1016/j.cose.2025.104315
Received 3 February 2024; Received in revised form 22 November 2024; Accepted 4 January 2025
Available online 11 January 2025

Nomenclature

AMI	*Advanced Metering Infrastructure*
ARP	*Address Resolution Protocol*
DCS	*Distributed Control System*
DT	*Digital Twin*
ETA	*Event Tree Analysis*
FDI	*False Data Injection*
FMEA	*Failure Mode and Effects Analysis*
FTA	*Failure Tree Analysis*
GOOSE	*Generic Object Oriented Substation Events*
HAZOP	*HAZard and OPerability analysis*
HCA	*Hazardous Control Action*
HMI	*Human Machine Interface*
ICS	*Industrial Control System*
IDE	Integrated Development Environment
IDS	*Intrusion Detection System*
IED	*Intelligent Electronic Device*
IT	*Information Technology*
MITM	*Man In The Middle*
MQTT	*Message Queue Telemetry Transport*
OT	*Operational Technology*
PLC	*Programmable Logic Controller*
RTU	*Remote Terminal Unit*
SCADA	*Supervisory Control And Data Acquisition*
SCASS	*SCADA Systems Security*
STAMP	*Systems Theoretic Accident Model and Process*
STPA	*Systems Theoretic Process Analysis*
STPA-Sec	*Systems Theoretic Process Analysis for Security*
TCP	*Transmission Control Protocol*
VM	*Virtual Machine*

protocols, SCADA systems, and commonly used cybersecurity practices within the ICS domain.

2.1. Industrial control systems

An *Industrial Control System* (ICS) refers to several types of control components (electrical, mechanical, hydraulic, pneumatic) acting together to accomplish an industrial goal (NIST, 2020).

The ICS architecture can be organized into three main groups of components. *Field devices*, such as sensors and actuators, gather data from physical processes and forward them to the *control layer*, which consists of simple components such as *Programmable Logic Controller*s (PLCs) or even more complex setups involving *Distributed Control Systems* (DCSs). The *supervisory layer* gathers data from the control systems and provides interfaces for monitoring and controlling industrial processes through *Human Machine Interfaces* and *Supervisory Control And Data Acquisition* (SCADA) systems, which provide a comprehensive overview of the system's operations (Cui et al., 2024). SCADA systems are designed specifically for remote monitoring and control of industrial processes through a centralized management platform that encompasses supervisory control, data acquisition, and real-time processing capabilities. A typical SCADA system consists of specialized industrial automation computers (*Programmable Logic Controller*s (PLCs)), industrial control devices (*Remote Terminal Unit*s (RTUs)), human operator interfaces (*Human Machine Interface*), data historians for historical records, communication networks for seamless data exchange, and the central control system, which processes data from remote locations and takes control decisions.

2.2. Communication protocols for industrial control systems

While *Operational Technology* (OT) networks share fundamental principles with *Information Technology* (IT) networks, subtle distinctions do exist. OT networks, akin to IT networks, rely on the TCP/IP stack and common application-level protocols. However, over the decades, dedicated industrial protocols based on TCP/IP were designed to facilitate rapid, efficient, and reliable communication between devices (Decotignie, 2005). Table 1 illustrates the most relevant TCP/IP industrial protocols by also underlying their common use cases.

2.3. ICS cybersecurity approaches

ICSs play a crucial role in managing critical infrastructures, making them susceptible to significant attack vectors (Makrakis et al., 2021). Moreover, the contemporary trend toward adopting cloud-based environments and leveraging the cloud computing paradigm escalates the risk of cyber attacks (Chan and Reith, 2022). Consequently, various techniques (often based on machine learning) have been implemented to detect attacks (Bhamare et al., 2020a). Given the impracticality of conducting security testing directly on SCADA systems, the design of formal methodologies becomes essential for defining cyber attacks on ICSs. Numerous methodologies have been developed to model cybersecurity threats in the ICS domain (Bhamare et al., 2020b). However, the predominant approaches often involve integrating security considerations into established safety models (Kriaa et al., 2015).

2.3.1. STAMP, STPA and STP-Sec

Systems Theoretic Accident Model and Process (STAMP) is an accident causality model developed by Leveson (2004) that imposes constraints on the behavior and interactions of system components. The model was extensively adopted across various domains, including air traffic control (Niu et al., 2018), nuclear power (Lee et al., 2020), automobiles, medical devices, and hospital safety.

Systems Theoretic Process Analysis (STPA) is a hazard analysis methodology that builds upon STAMP and is employed to study the dynamics of accidents through several steps.:

- Identifying accidents and hazards (*precondition*);
- Drawing the control structure (*precondition*);
- Identifying unsafe control actions;
- Identifying causal factors and creating scenarios.

The effectiveness of STPA surpasses that of traditional safety models like *Failure Tree Analysis* (FTA), *HAZard and OPerability analysis* (HAZOP), *Failure Mode and Effects Analysis* (FMEA), and *Event Tree Analysis* (ETA) in uncovering overlooked causes, especially in complex systems incorporating computer controls (Yan et al., 2016).

Young and Leveson (2014) address security concerns by proposing the *Systems Theoretic Process Analysis for Security* (STPA-Sec) model, an STPA extension that identifies and analyzes security risks related to critical control processes. Friedberg et al. (2017) have also contributed through *STPA-SafeSec*, a unified process offering a comprehensive approach to address both safety and security aspects.

3. Related works

In this section, we explore pertinent related works on the application of digital twins. We also discuss simulated environments addressing cybersecurity challenges for ICS systems, with a focus on the physical Electrical Power and Intelligent Control (EPIC) testbed (Adepu et al., 2019) and its virtualized twin known as EPICTWIN (Kandasamy et al., 2021).

Table 1
TCP/IP protocols and their use cases in industrial settings.

Protocol	Use case description
Modbus/TCP (Wang et al., 2022)	Facilitates communication among industrial devices, often used for real-time monitoring and control in automation systems.
Profinet (Anon, 2023g)	Supports real-time communication in automation systems, widely used in factory automation.
MQTT (Anon, 2023a)	A lightweight messaging protocol for Industrial Internet of Things (IIoT) environments, enabling communication between remote devices with minimal network bandwidth.
DNP3 (Anon, 2023c)	Reliable protocol for communication between supervisory systems and remote terminal units, commonly used in power substations and water utilities.

3.1. Digital twins

Digital Twin (DT) systems enable the creation of digital representations of systems by mirroring physical entities, while at the same time minimizing costs (Haag and Anderl, 2018). Digital twin systems have been adopted to replicate production or manufacturing systems for cybersecurity purposes (Park et al., 2020/01/01; Zhang et al., 2018).

In the cybersecurity field, digital twins are adopted to model complex systems. Olivares-Rojas et al. (2022) demonstrate their utility in assessing vulnerabilities of an intelligent metering system in a smart home environment. Masi et al. (2023) construct an ICS digital twin based on cybersecurity requirements. They leverage such a system to derive attacks and identify the related countermeasures.

Despite the numerous benefits of digital twins, several authors emphasize their security risks (Holmes et al., 2021; Mark Hearn, 2019). In particular, attackers might exploit Digital Twin vulnerabilities to gather information about the replicated real infrastructure. Besides that, the simulated system may not be able to reproduce severe threats affecting the mirrored physical systems.

SCASS hybrid infrastructure addresses these concerns by integrating both virtual and physical components that allow for defining the degree of realism in the SCASS system. On one hand, this allows to reduce implementation costs, as well as hardware and software expenses. On the other hand, it guarantees adherence to the replicated physical system.

3.2. ICS testbeds

Cyber ranges are simulated platforms mainly developed for training security personnel. Nowadays, they are extensively adopted for other purposes, such as the security evaluation of new solutions and strategies (Yamin et al., 2020).

An extensive analysis of testbeds for industrial cybersecurity is provided by Conti et al. (2021), who categorizes over 50 systems based on parameters like sector, cost, and license. It is possible to classify industrial testbeds into three categories, namely, physical, hybrid, or virtual, with virtual testbeds being more flexible yet less customizable due to the replacement of physical components.

3.2.1. Physical testbeds

Physical testbeds offer a more realistic simulation of events but are often expensive and time-consuming to build. A relevant example is the EPIC (Anon, 2021) physical testbed, which mimics a real-world power system via real PLCs, with communication implemented through the IEC 61850 protocol. In the EPIC system, it is possible to reproduce various security attacks, such as False Data Injection and power supply interruption, and assess related mitigation techniques.

The Smart Grid Test Bed (Anon, 2017) stands out as the world's first full-scale replication of a smart grid within a physical testbed, demonstrating the scalability of such environments from smaller dimensions to real-world counterparts.

Another example of a physical testbed is the Secure Water Treatment (SWaT) (Mathur and Tippenhauer, 2016) system. The platform comprises a six-stage water treatment process, with each stage controlled by a dedicated PLC, evaluated against multiple attack scenarios.

3.2.2. Virtual testbeds

Virtual testbeds are composed of virtual machines or containers and are more cost-effective as they do not require the deployment of real components.

Maynard et al. (2018) propose a scalable framework designed for deploying multiple virtual machines to create a SCADA network. This open-source framework (Maynard, 2018) leverages components like *Human Machine Interfaces* (HMIs) and RTUs, which communicate through OT network protocols and are created with OpenPLC (Anon, 2023f) on VirtualBox virtual machines configured through Vagrant. This framework might be easily extended to support SCASS. For instance, deploying the SCASS network on a cloud infrastructure using Maynard SCADA would enhance scalability and offer a more realistic execution in both hybrid cloud and on-premises environments. We finally observe that the EPIC infrastructure was virtually replicated through the digital twin EPICTWIN, which will be described in detail in Section 3.3.1.

3.2.3. Hybrid testbeds

Hybrid testbeds combine physical and virtual components to leverage the strengths of both approaches, ensuring a realistic simulation within a controlled environment.

Mallouhi et al. (2011) propose a testbed to analyze cyber–physical interactions and vulnerabilities in SCADA environments. The testbed prioritizes the simulation of real-time events and combines physical and virtual elements to imitate SCADA processes.

The Mississippi State University presents a SCADA security laboratory (Morris et al., 2011) which combines process control systems from multiple critical infrastructure industries, enabling laboratory exercises, as well as functional demonstrations and experimentation. No source code is available for this laboratory, and the cited work does not describe the analized attacks in detail.

Singh et al. (2015) simulate a SCADA power system encompassing simulated RTUs that generate traffic through several communication protocols. The testbed introduces interesting security controls, such as comparator modules to detect intrusions, but it is mainly focused on physical attacks and does not take relevant networking scenarios into account.

Koutsandria et al. (2015) developed a hybrid testbed using physical systems and PLCs, Simulink for Operational Technology, and a Raspberry Pi network tap for traffic statistics and packets exchanges. Although the source code is not accessible, they offer an attack scenario example that provides valuable insights for replicating different types of attacks.

Keliris et al. (2016) integrate machine learning defenses into the ICS process by proposing a solution that combines hardware and software components to analyze attacks. It supports comprehensive vulnerability analysis and impact assessment. Unfortunately, the analysis does not include network-specific attacks such as ARP spoofing and packet sniffing.

SCADA-SST (Ghaleb et al., 2016) is a versatile testbed for SCADA security, enabling the modeling of hybrid architectures with both simulated and real components. The testbed is characterized by its light weight, ability to scale, and inclusion of security analysis capabilities, such as the ability to capture traffic and simulate network attacks. It facilitates communication between simulated and real devices, which is essential for conducting realistic tests on SCADA systems under different attack conditions, such as Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. Rosa et al. (2017) describe an active learning approach guiding users through a SCADA cyber-range. The work describes interesting attack scenarios, but technical information about hardware and networking configuration is not detailed enough to facilitate the training reproduction.

HYDRA (Bernieri et al., 2016) is a cost-effective, modular, and flexible open-source testbed that replicates the functioning of a basic water distribution system. The product offers a high level of modularity and flexibility, enabling effortless attachment and detachment of components. Arduino boars are used, in conjunction with the Modbus communication protocol. The paper lacks specific networking details and, though focusing on cyber attacks, does not provide any illustrative example of a real-world attack scenario.

Foglietta et al. (2019) present a testbed that combines physical components with virtualized environments to support integrated fault diagnosis and cybersecurity research. While it effectively addresses fault diagnosis, it does not provide a range of attack scenarios as diverse as other testbeds.

The VTET Virtual Testbed (Xie et al., 2018) offers a hybrid environment combining virtualization with process simulations such as the Tennessee Eastman Process. It supports several ICS protocols, but the attack description lacks relevant ICS attacks, such as ARP spoofing and traffic sniffing. KYPO4INDUSTRY is an educational establishment located at Masaryk University (Čeleda et al., 2020), with the purpose of instructing computer science students on how to tackle cybersecurity risks in Operational Technology networks. The system employs Raspberry Pi devices to emulate Programmable Logic Controllers, enabling the automated creation of attack scenarios using YAML files, as well as the seamless deployment of virtual machines through Ansible playbooks. Despite its scientific relevance and similarity with our work, the source code is not available, and no networking details are provided. The cited work also lacks examples of potential attack scenarios. Cruz and Simões (2021) propose a novel method for active learning by guiding users through a SCADA cyber range, enhancing their understanding of system vulnerabilities and defenses, and equipping them for real-world cybersecurity challenges.

Most of the cited works neither include cost considerations, nor release the source code. Furthermore, just few of them include networking and hardware configuration details, as well as provide a thorough description of the implemented attacks. This makes the replication challenging. The SCASS testbed was designed with the purpose of providing a testbed that embraces all of the relevant features of related works. In Section 7, we provide a comparison between SCASS and related hybrid testbeds.

3.3. A focus on the Electrical Power and Intelligent Control (EPIC) testbed

The Electric Power and Intelligent Control (EPIC) testbed faithfully replicates a smart-grid architecture. The authors present this work to enable researchers to devise new cyber-attacks against smart-grid systems and evaluate the efficacy of defense mechanisms. Fig. 1 illustrates the high-level architecture of EPIC, which comprises four stages,i.e., generation, transmission, smart home, and microgrid. These stages are linked to a controller PLC, establishing connections to a SCADA system. A more detailed description of the testbed is available in Fig. 2. Within each stage, every generator is linked to a motor, and control over the motors is facilitated by Variable Frequency Drive (VFD) devices. The power line can be disconnected from the main grid through Circuit Breakers, which are operated by an array of *Intelligent Electronic Devices* (IEDs).

3.3.1. EPICTWIN

The EPIC architecture serves as an excellent model for reproducing cybersecurity scenarios, but its dependence on expensive physical components complicates experimental reproduction (Tao et al., 2019). To address these limitations, Kandasamy et al. (2022) leverage the digital twin approach to achieve full virtualization of the testbed by realizing EPICTWIN (Kandasamy et al., 2021, 2022), a digital twin allowing the assessment and mitigation of threats and vulnerabilities within the replicated plant. It is based on network protocols such as *Message Queue Telemetry Transport* (MQTT), IEC 61850 *Manufacturing Message Specification* (MMS), and *Generic Object Oriented Substation Events* (GOOSE). The approach proves valuable for generating twins of testbeds or plants for cybersecurity studies. SCASS adopts a similar electric layout with some variations that will be further discussed in the paper.

4. The SCASS architecture

SCASS is implemented as a hybrid testbed, composed of physical and virtual components, which operates within a single cabinet that houses a collection of industrial devices, such as PLCs and RTUs, collaborating with virtual components implemented through the Docker virtualization technology. This allows to reliably replicate a smart grid industrial environment. A hybrid architecture offers cost and size reduction benefits while maintaining a certain level of realism in simulations.

4.1. Physical components

The cabinet is made of metal and has DIN mountings. It includes (Fig. 3) the following physical components:

- One IDS-409-1SFP Industrial Managed Switch by Perle Systems Inc.;
- Two UNO-PS/1AC/24DC/60 W Power supply units by Phoenix Contact;
- Four PM554-TP-ETH PLCs by ABB Ltd (Fig. 4);
- One Raspberry PI4, equipped with a 64 GB SD card by Samsung;
- One C60N magneto thermic switch by Merlin Gerin (Schneider SA);
- One 12HD7 monitor by Beetronic;
- One Stabilized power supply by Mean Well.

The monitor is connected to the Raspberry Pi, which implements virtual elements and manages the virtual networks. The PM554-TP-ETH is a low to mid-end PLC that includes an AC500 e-Co CPU, an Industrial Ethernet port, an RS-485 serial port, and a set of six digital inputs and eight digital outputs. In addition, accessories can be mounted to expand the capabilities of the device. PLCs can be configured through the ABB Automation Builder Integrated Development Environment (IDE), a robust toolset based on CodeSys (CODESTS, 2021), that allows the development, testing, and commissioning of automation solutions and communication networks compliant with the IEC 61131-3 standard programming language.

In this work, we have configured the physical PLCs in such a way as to replicate the EPIC infrastructure. However, the SCASS architecture is general enough and can be arbitrarily extended by configuring PLCs as controllers of either virtualized or physical components, hence realizing any testbed architecture.

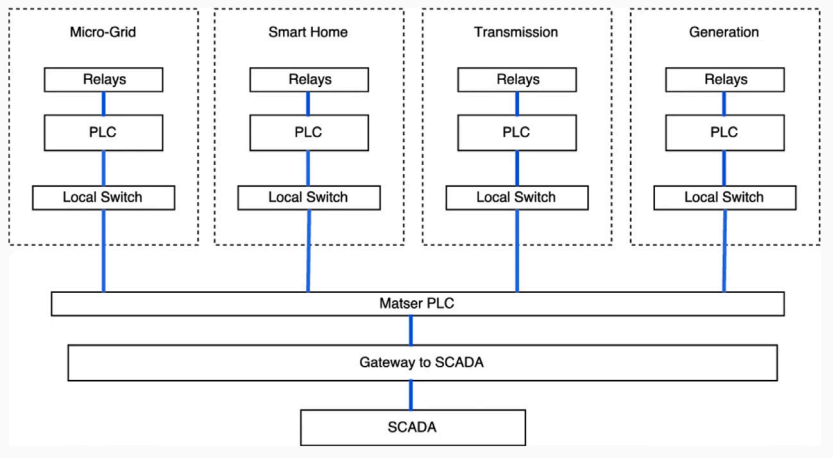


Fig. 1. EPIC high-level architecture.

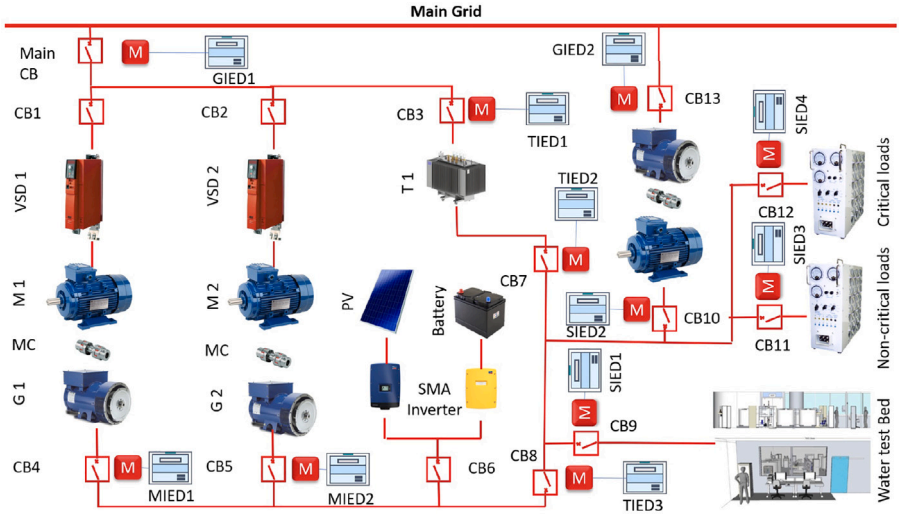


Fig. 2. *Electric layout in the EPIC architecture*. Four areas can be distinguished: Generation, Micro-Grid, Transmission, Smart Home. Prefixes match device names. *Source: EPICTWIN* architecture by Kandasamy et al. (2022, 2021).

4.2. Virtual components

The Raspberry Pi runs a Docker daemon (Boettiger, 2015), facilitating the deployment of lightweight containers based on developed Docker images.

We implement different Docker images:

- "IED" is a Docker image implemented in Python, functioning as a Modbus TCP server leveraging the Pymodbus library (Mehta et al., 2015) that awaits commands on TCP port 501.
- "Router" refers to the router image used to replicate the EPIC configuration. Two instances of the router element are employed: one connects the Controller PLC with the worker PLCs, and the other links the Controller PLC with the SCADA system.
- The "SCADA-System" container that is a part of the overall SCADA system developed using Node-RED, a flow-based programming tool commonly used to model industrial processes (Ferencz and Domokos, 2019; Tabaa et al., 2018; Nicolae and Korodi, 2018).

4.2.1. The HMI

The HMI component of the testbed is implemented through either ad-hoc Node-RED libraries[1] (see Fig. 5(a)), or by leveraging an alternative open-source SCADA/HMI framework named FUXA (frangoteam, 2020) (see Fig. 5(b)). These components are interchangeable, enabling users to select the solution that best suits their needs. In this case, the HMI displays the implemented architecture layout, with all control and monitoring elements that can be managed through a web-enabled interface. Node-RED offers the highest grade of customization but requires higher programming and configuration effort than FUXA. On the other hand, FUXA allows for the effortless integration of new protocols, as well as the addition of new features, with no advanced programming knowledge. However, FUXA does not provide the same depth of customization as its Node-RED counterpart.

[1] The `node-red-contrib-ui-led` library and the `node-red-dashboard` library

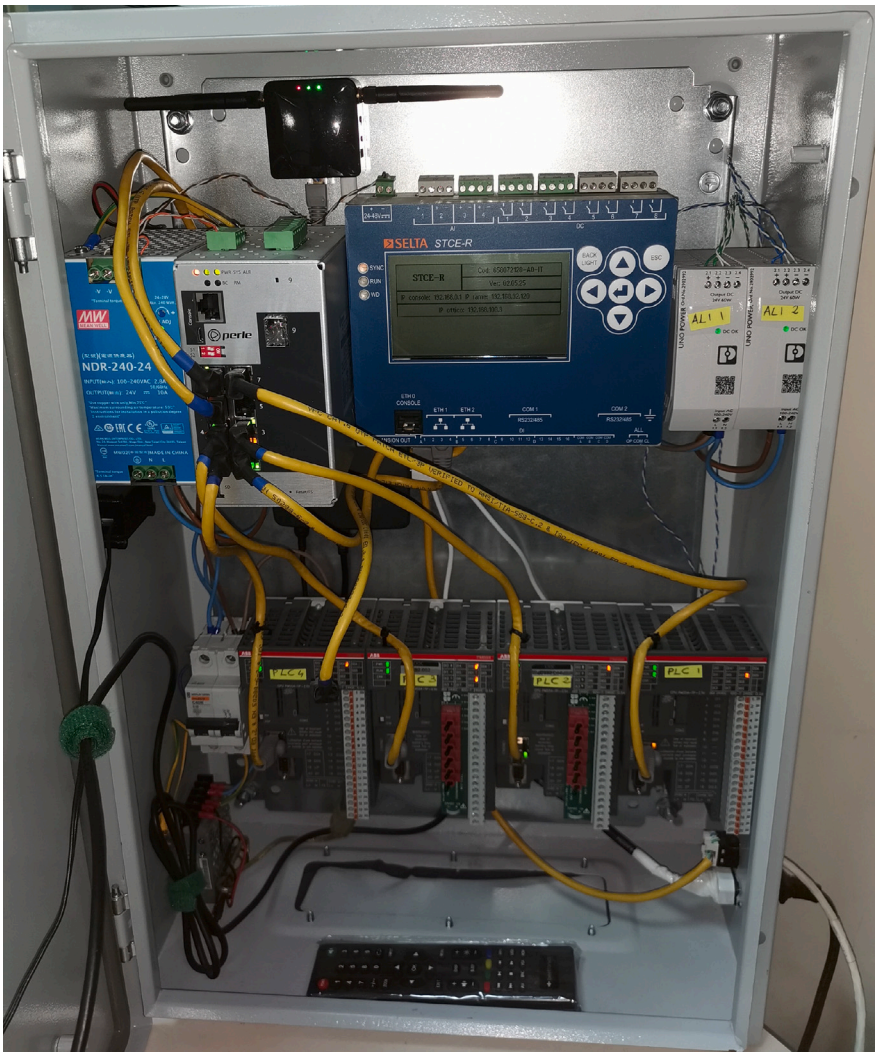


Fig. 3. The cabinet: the array of PLCs can be observed below; the Raspberry PI 4 is located under the industrial switch.

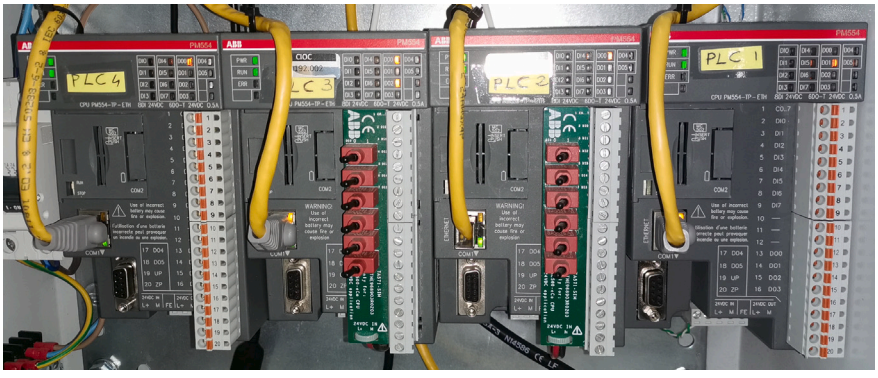
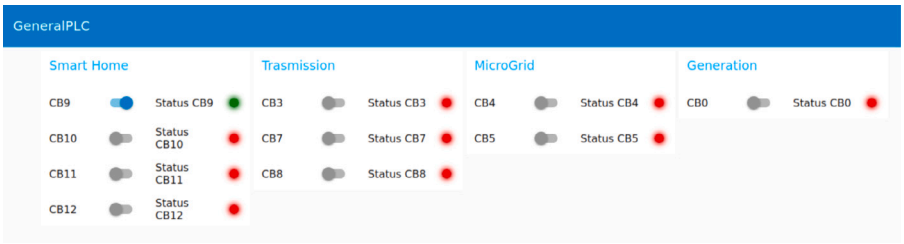


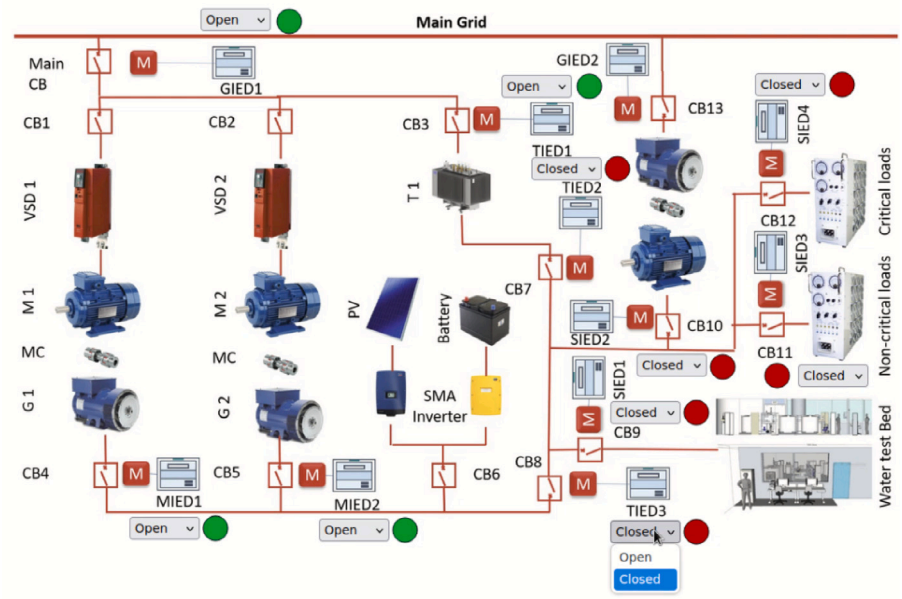
Fig. 4. The array of PLCs. In the current SCASS configuration, physical PLCs are used as workers, while the general PLC controller is virtualized. Such a configuration can be changed, and it is possible to configure physical PLCs as both controllers and workers.

4.3. Networking

Layer-2 interconnection of SCASS components is facilitated by the IDS-404-1SFP Industrial Managed Switch, that is responsible for managing communications among all of the physical PLCs in the cabinet. A dedicated "OUT" line allows the integration of external systems for more complex attack scenarios. VLANs are configured within the physical switch, with each physical PLC residing on a specific VLAN. The Docker engine manages VLANs, and the Macvlan network driver assigns each Docker container an IP address associated with a specific VLAN. This approach facilitates the establishment of a unified networking stack, seamlessly interconnecting physical and virtual components. Certain containers serve as default gateways, enabling the construction of intricate infrastructures comprising multiple networks. Fig. 6 provides an illustrative example of the networking configuration that interconnects both virtual and physical components.



(a) HMI realized through Node-RED GUI capability.



(b) HMI realized through FUXA capability.

Fig. 5. HMI designs using Node-RED and FUXA.

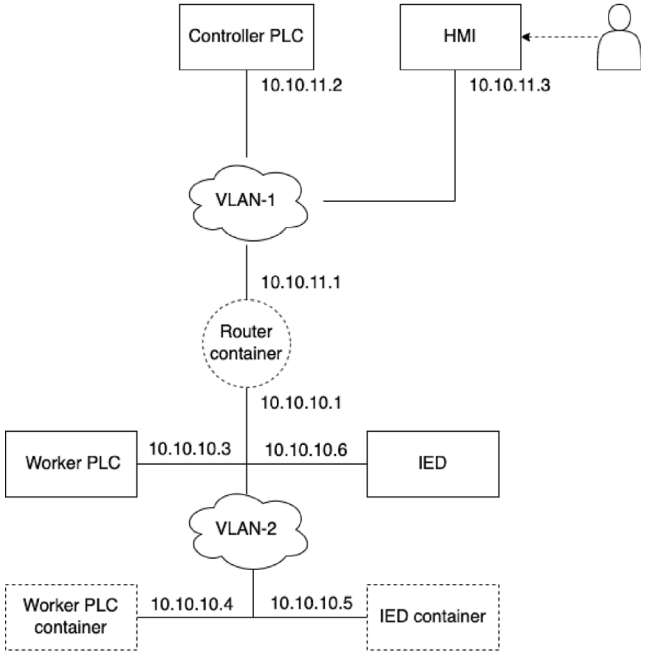


Fig. 6. SCASS networking layer example. The SCASS switch is configured with two VLANs that implement a controller–worker architecture commonly adopted in SCADA systems. The controller PLC is responsible for issuing commands to the worker PLCs, which manage the IEDs. A dedicated protocol such as Modbus TCP can take care of all message transfers, and all the components can be either containerized or physical.

4.3.1. *Containers and cyber-attack scenarios*

The Raspberry Pi, being a compact embedded device, benefits from an efficient container-based virtualization. However, containers present different challenges, such as replicating specific attack scenarios involving Windows machines or exploiting vulnerabilities in the Linux kernel. One further significant challenge, highlighted by Kandasamy et al. (2021), is the difficulty to simulate physical Man In The Middle (MITM) attacks at the data link layer using containers. This feature is fundamental for replicating cyber attacks against ICSs. To address this issue, we leverage the Macvlan network driver in Docker, a solution we previously employed in our work (Caturano et al., 2020). This feature allows the creation of a shared network that connects physical and virtual components by configuring containers with a virtual Network Interface Controller (NIC) that emulates a direct connection to the physical network. By integrating concepts from our previous work, we establish an ICS networking stack that seamlessly connects physical and lightweight virtual elements.

5. Reproducing the EPIC architecture through SCASS

To showcase the system's adaptability, we configure SCASS to replicate the Electric Power and Intelligent Control (EPIC) testbed (Adepu et al., 2019) using a methodology akin to that proposed by Kandasamy et al. (2021). The virtual components are created using the "Infrastructure as Code (IaC)" methodology (Artac et al., 2017). Specifically, a "docker-compose"[2] file is employed to define the virtual components. In the compose file, the following 'services' are specified:

[2] https://docs.docker.com/compose/

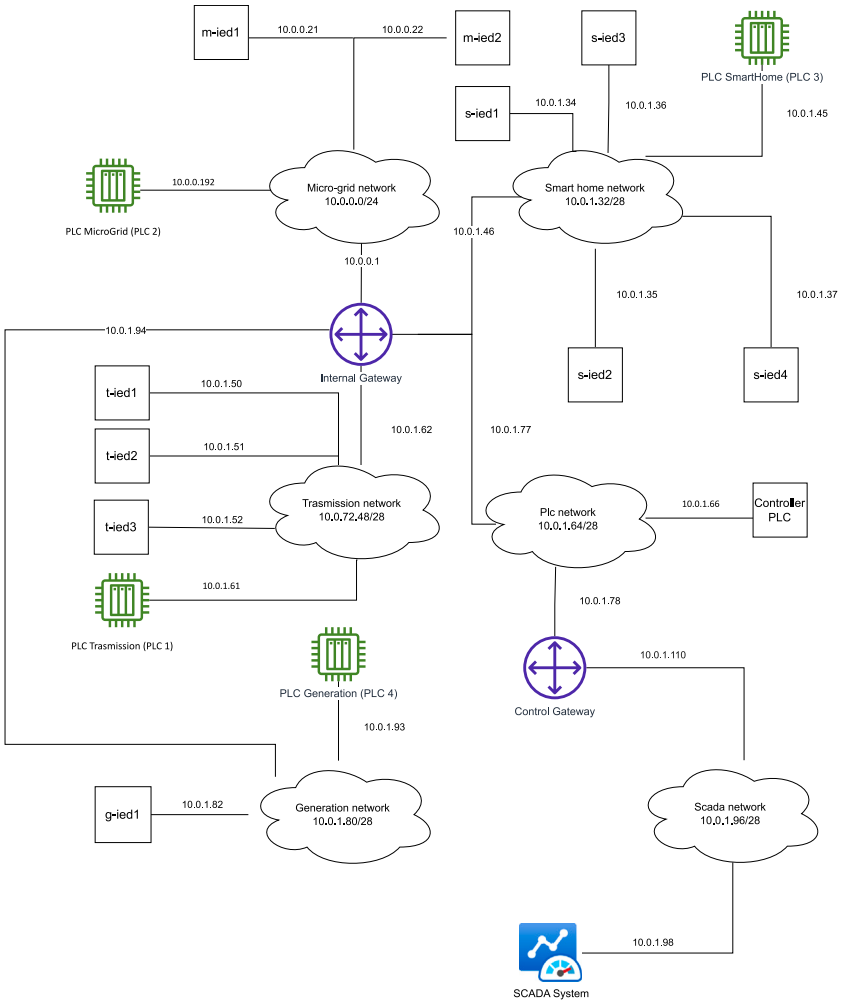


Fig. 7. EPIC networking through SCASS.

- one IED for the generation phase;
- three IEDs for the transmission phase;
- four IEDs for the smart-home phase;
- two IEDs for the micro-grid phase.

The networking is planned based on the concepts elucidated in Section 4.3. Six VLANs are defined in the physical switch and mapped to virtual networks generated through the Docker engine. Physical elements connect to tagged VLAN ports, while virtual components acquire the designated VLAN through the Docker configuration. Specifically, the Raspberry Pi is configured to access all VLANs in trunk mode, enabling a generic configuration that links virtual and physical devices. Virtual networks have either 24-bit (such as the "scass-microgrid" network below) or 28-bit (for all other networks) subnet masks and are organized as follows:

- "scass-microgrid": 10.0.0.0-10.0.0.0.255
- "scass-smarthome": 10.0.1.32-10.0.1.47
- "scass-trasmission": 10.0.1.48-10.0.1.63
- "scass-plc": 10.0.1.64-10.0.1.79
- "scass-generation": 10.0.1.80-10.0.1.95
- "scass-scada": 10.0.1.96-10.0.1.111

The network architecture is reported in Fig. 7. Two router containers serve as default gateways for the multiple VLANs defined in the physical switch. The internal gateway segregates the four networks, enabling the realization of the four stages in the EPIC architecture. Each network features a physical PLC controlling a set of virtual IEDs simulating circuit breakers. The Control Gateway separates the SCADA environment from the other networks, facilitating communication between the SCADA system and the Controller PLC. Fig. 8 illustrates the communication flows among components, utilizing a controller–worker paradigm. The SCADA system issues commands to the containerized Controller PLC, which oversees physical PLCs. Each PLC manages multiple virtual IEDs, emulating the EPIC architecture.

Communications within the system rely on the Modbus TCP protocol. Modbus TCP was chosen for two primary reasons: (i) it is one of the most widely used protocols in the ICS domain, and (ii) it is the protocol utilized by the ABB devices available in the cabinet we used, which is of particular interest to the COR interforce command. That being said, the SCASS architecture is designed to be general enough to allow the seamless replacement of Modbus TCP with any other protocol for communication between industrial electronic devices.

The physical PLCs are programmed using CodeSys, whereas the virtual elements are created using Python modules that implement Modbus TCP servers, as detailed in Section 4.

5.1. Differences with EPICTWIN

The SCASS architecture differs from the EPICTWIN testbed in various aspects. Table 2 provides a comparison of the EPICTWIN and SCASS implementations.

As evident, our implementation extensively employs container-based virtualization for networking and controllers. In EPICTWIN, the

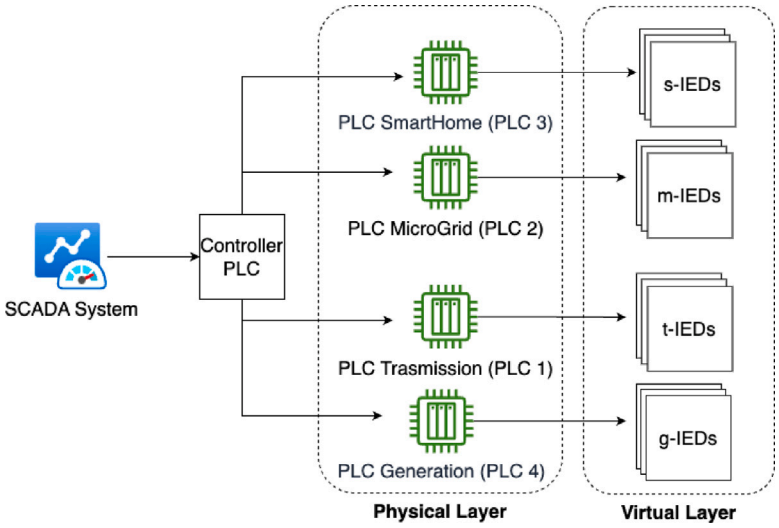


Fig. 8. EPIC components' flows.

Table 2
EPICTWIN - SCASS comparison.

EPIC testbed's element	EPICTWIN	SCASS Implementation
Network switch	Ubuntu VM	Docker container hosted on a Raspberry Pi
PLC controller	Ubuntu VM	Physical (except for Controller PLC, not present in EPICTWIN)
IED controllers	Ubuntu VM	Docker containers hosted on a Raspberry Pi
AMI controllers	Docker	Not implemented
SCADA	Node-RED process running on a Linux VM	Node-red process running on a Windows VM

authors acknowledge the advantages of Docker virtualization but face limitations when using containers to fully replicate a network stack environment capable of reproducing *Man In The Middle* (MITM) attacks. As mentioned earlier, we utilize the Docker Macvlan network driver to directly connect the container to the physical network, fully virtualizing the network stack and connecting real PLCs with virtual components in the Raspberry Pi.

This approach allows us to leverage the benefits of container-based virtualization, including performance, scalability, and portability. In our case, the networking stack runs on a small Raspberry Pi, and the entire infrastructure is contained in the compact cabinet shown in Fig. 3.

We do not emulate the *Advanced Metering Infrastructure* (AMI) controllers since we do not simulate any physical process. Indeed, SCASS employs physical PLCs that can connect to real devices, offering a faithful replication of a real-world scenario. A significant distinction in our EPIC replication is the inclusion of the Controller PLC. Our system replicates the typical "controller-workers" architecture presented in the EPIC testbed. In the EPICTWIN architecture, the "Controller PLC" does not seem to be present.

EPICTWIN employs the Open vSwitch technology for virtualization, whereas our approach utilizes Docker's networking layer, configured to work in tandem with the physical switch that supports multiple VLANs. This distinction in networking technologies underscores the flexibility and adaptability of virtualization solutions across different implementations.

At the protocol layer, EPICTWIN leverages the IEC 61850 (GOOSE) potocol, whereas SCASS relies on Modbus TCP, as already discussed above. This is actually a minor difference since both digital testbeds can easily support a wide set of ICS communication suites.

Unfortunately, the implementation details of the EPICTWIN infrastructure are not publicly available. In our case, to facilitate the replication and extension of the SCASS architecture within the community, we have released the source code, encompassing both real and virtual PLCs. The source code for the EPIC replication is openly accessible on GitHub.[3] This allows security researchers to enhance the foundational EPIC infrastructure for replicating more intricate scenarios.

We can summarize the distinguishing features of SCASS as follows:

- Hybrid Architecture: the SCASS infrastructure integrates both physical and virtual elements, leveraging the advantages of hybrid architectures (Qassim et al., 2017);
- Container-Based Virtualization: SCASS implements its virtual layer using container-based virtualization, overcoming the networking limitations inherent to this approach by utilizing the Macvlan feature (Caturano et al., 2020);
- Open Source: the SCASS source code is publicly available, enhancing the reproducibility of the testbed.

6. Breaking into SCASS

In this section, we demonstrate the capability of the SCASS system to replicate realistic scenarios in SCADA systems. This involves the entire process, from identifying an attack path for the target infrastructure to executing attack operations.

6.1. Threat analysis

A threat analysis utilizing the STAMP/STPA method has been conducted to identify vulnerabilities within SCASS. By leveraging the insights and framework provided by Castiglione et al. (2022), we aim to effectively identify and address potential security vulnerabilities and threats within the SCASS system. The STAMP model for SCASS can be established by considering the components of the target infrastructure. In this model, the control chain of the system is represented as a sequence of associations, as illustrated in Fig. 9. These associations are established based on the specific responsibilities assigned to each individual component. For brevity, this analysis will focus exclusively on the MicroGrid zone.

[3] https://github.com/NS-unina/SCASS

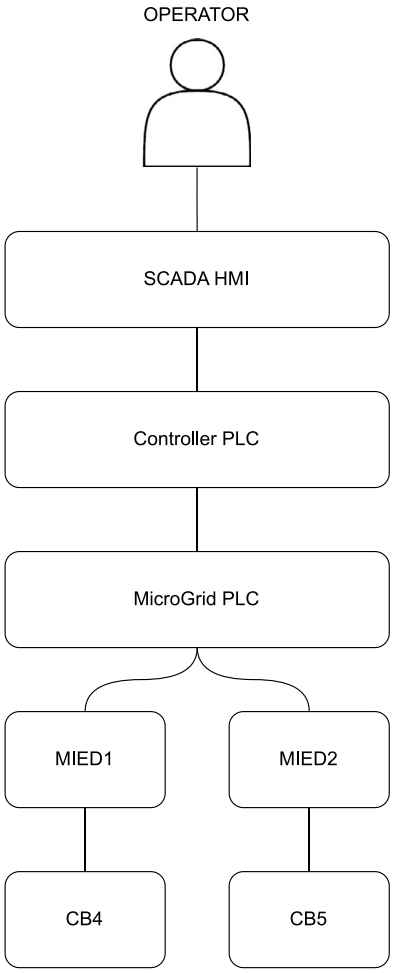


Fig. 9. Entities associations to identify the control chain in SCASS. The operator interacts with the SCADA HMI to monitor and control the status of all Circuit Breakers (CBs) within the testbed. The Controller PLC relays the commands received from the SCADA HMI to the corresponding zone-specific PLCs. Notably, the figure illustrates only the MicroGrid PLC, which is responsible for disconnecting generator units from the electrical network. The intelligent devices MIED1 and MIED2 execute commands from the MicroGrid PLC to open or close Circuit Breakers CB4 and CB5, respectively.

Once the control chain has been identified, the next step involves executing the hazard analysis using STPA, conducted across a series of steps.

Step 1: Identify accidents and hazards

The first step entails identifying potential accidents and hazards within the target architecture. Accidents refer to unintended and undesired events that can lead to adverse consequences, while hazards are conditions or situations that can contribute to the occurrence of accidents (Leveson, 2016).

The accident list is as follows:

- A_1: uncontrolled generator overload (with damage);
- A_2: undesired generator shutdown.

A list of hazards is given below:

- H_1: Controller PLC not synchronized with physical process;
- H_2: MicroGrid PLC not synchronized with physical process;
- H_3: unsafe configuration of MicroGrid PLC;
- H_4: unsafe configuration of MIED1;
- H_5: unsafe configuration of MIED2.

Step 2: define the control structure

In this step, Control Actions and Feedback will be specified, starting from the STAMP model (Fig. 10).

The Control Actions for the system are:

- CA_1: sends open/close signals to Controller PLC in order to control MIEDs;
- CA_2: relays open/close signals from Controller PLC to MicroGrid PLC;
- CA_3: relays open/close signals from MicroGrid PLC to MIED1;
- CA_4: relays open/close signals from MicroGrid PLC to MIED2;
- CA_5: applies open/close actions at CB4;
- CA_6: applies open/close actions at CB5.

The Feedback signals are:

- F_1: receive measurements from generators, state from IEDs;
- F_2: relay measurements from generators, state from IEDs;
- F_3: MIED1 state, measurements from the generator controlled by MIED1;
- F_4: MIED2 state, measurements from the generator controlled by MIED2.

Step 3: identify hazardous control actions

The Hazardous Control Actions leading to accidents A1 and A2 are listed below:

- HCA_1: CA_1 is applied with incorrect parameters, shutting down the generator controlled by MIED1 → H_3, H_4;
- HCA_2: CA_1 is applied with incorrect parameters, shutting down the generator controlled by MIED2 → H_3, H_5.
- HCA_3: CA_2 is applied with incorrect parameters, shutting down the generator → H_3, H_4, H_5;
- HCA_4: F_2 is applied with incorrect parameters, giving incorrect feedback to the Controller PLC → H_1;
- HCA_5: F_3 is applied with incorrect parameters, giving incorrect feedback to the MicroGrid PLC → H_1, H_2;
- HCA_6: F_4 is applied with incorrect parameters, giving incorrect feedback to the MicroGrid PLC → H_1, H_2;
- HCA_7: CA_3 is applied with incorrect parameters, shutting down the generator controlled by MIED1 → H_4;
- HCA_8: CA_4 is applied with incorrect parameters, shutting down the generator controlled by MIED2 → H_5.

Step 4: associate attack steps with hazardous control actions

In this phase, attack steps are associated with HCAs. This operation leads to the definition of a list of attack steps. An attack step is a tuple $a_i = (t, e, v)_i$ where:

- t is the type of threat;
- e is the targeted control action or feedback;
- v is the value injected.

Values can be injected through forgery. The element can be a Control Action or Feedback and, therefore, either a command or a measurement. Possible attack steps against SCASS can be grouped in a table (Table 3) and associated with HCAs.

These attack steps can be properly combined to reach a possible attack goal through an attack path.

6.2. Deriving the attack graph with MulVal

The attack graph for the discussed SCASS configuration can be derived using *MulVAL* (Anon, 2023e), a logic-based enterprise network security analyzer. MulVAL requires a rule file containing the rules to generate the attack graph and an input file describing network elements

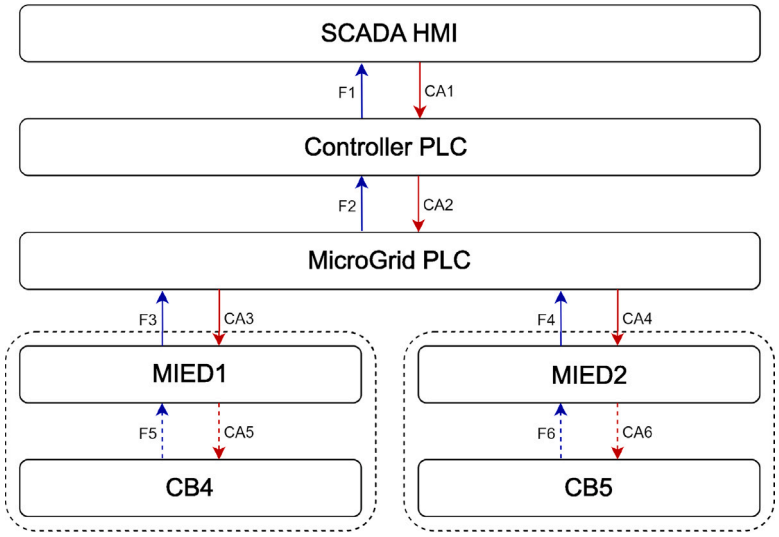


Fig. 10. STAMP model, identifying *Control Actions* and *Feedback*.

Table 3
Attack steps and related HCAs.

Attack step	Description	HCAs
a_1	$(forgery, CA_1, cmd)$	*1, 2*
a_2	$(forgery, CA_2, cmd)$	*3*
a_3	$(forgery, CA_3, cmd)$	*7*
a_4	$(forgery, CA_4, cmd)$	*8*
a_5	$(forgery, F_1, \bar{v})$	*4*
a_6	$(forgery, F_2, \bar{v})$	*5*
a_7	$(forgery, F_3, \bar{v})$	*6*

and vulnerabilities.[4] The output is an attack graph that visualizes all possible steps an adversary might leverage to alter a specific Control Action or Feedback and, therefore, trigger one of the aforementioned HCAs. In this context, the leaves represent the necessary preconditions for the attack, while the root represents the desired attack goal. Each step toward the attack goal corresponds to a successful exploitation of vulnerabilities, which may have a certain probability of execution. The generated attack graphs are depicted in Fig. 11 and are further analyzed in Section 6.2.1 and Section 6.2.2.

6.2.1. False data injection through ARP poisoning

The tree in Fig. 11(a) models a False Data Injection (FDI) attack. To successfully execute the attack on SCASS, the attacker needs to satisfy specific preconditions within the OT network, where the IEDs and the area supervisory control device are located. The preconditions for the attack include:

- the attacker must have access to the OT network;
- the devices within the network must support the *Address Resolution Protocol* (ARP) protocol.

To forge messages to the area supervisory control device, the following preconditions need to be met:

- the ARP Poisoning attack must have been successfully executed;
- a data flow should exist between the IED and the area supervisory control device;
- the protocol being used does not employ encryption.

[4] The complete source file associated with the architecture description can be found on the project's github repository

To disrupt legitimate feedback from reaching the MicroGrid PLC device, the following preconditions must be fulfilled:

- a data flow between the MIED and the MicroGrid PLC device must be present;
- the attacker should have the ability to forge feedback messages to the area supervisory control device;
- the protocol being utilized lacks authentication mechanisms.

MulVAL, indeed, provides a method to clearly identify how to exploit these preconditions and attack SCASS successfully. By understanding and addressing these vulnerabilities, appropriate countermeasures can be implemented to enhance the security and resilience of the SCASS system.

6.2.2. Multi-stage false data injection attack

This scenario shows a multi-stage attack against the Main Router to take the control of Control Action CA_3. In detail, there are two stages in the attack graph shown in Fig. 11(a):

1. the attacker takes control of the General-PLC to gain network visibility on the Main Router (*blue path*);
2. the attacker gets access to the network device through the SSH server (*red path*).

In the first step, the hacker takes advantage of the Shellshock vulnerability (*node 8-9*) found in the web HMI operating on the General PLC. The web HMI (*node 7*) shows the real-time device status remotely using a dynamic web page. Specifically, the HTTP server employs the CGI (*Common Gateway Interface*) technology to run a bash script that gathers information to be displayed on the web page. However, our system runs the script using a bash version vulnerable to Shellshock, which enables the attacker to execute unauthorized code on the machine (*node 5-6*). Then the attacker can use the General PLC as a stepping stone to compromise the Main Router. Indeed, this attack step gives the attacker network access to all OT network subnets. Next, the attacker can contact the *Secure Shell* (SSH) service (*node 13*) exposed by Main Router. SSH is a widely-used protocol for secure remote access and control of computer systems and servers (*node 13 to 15*). Unfortunately, the configuration made by an inexperienced technician (*node 10 to 12*) maintained the default credentials (*node 18*), making the system vulnerable to unauthorized access and compromising its security. By exploiting this security issue, the attacker can run arbitrary code with elevated privileges on this machine (*node 13*). In this context,

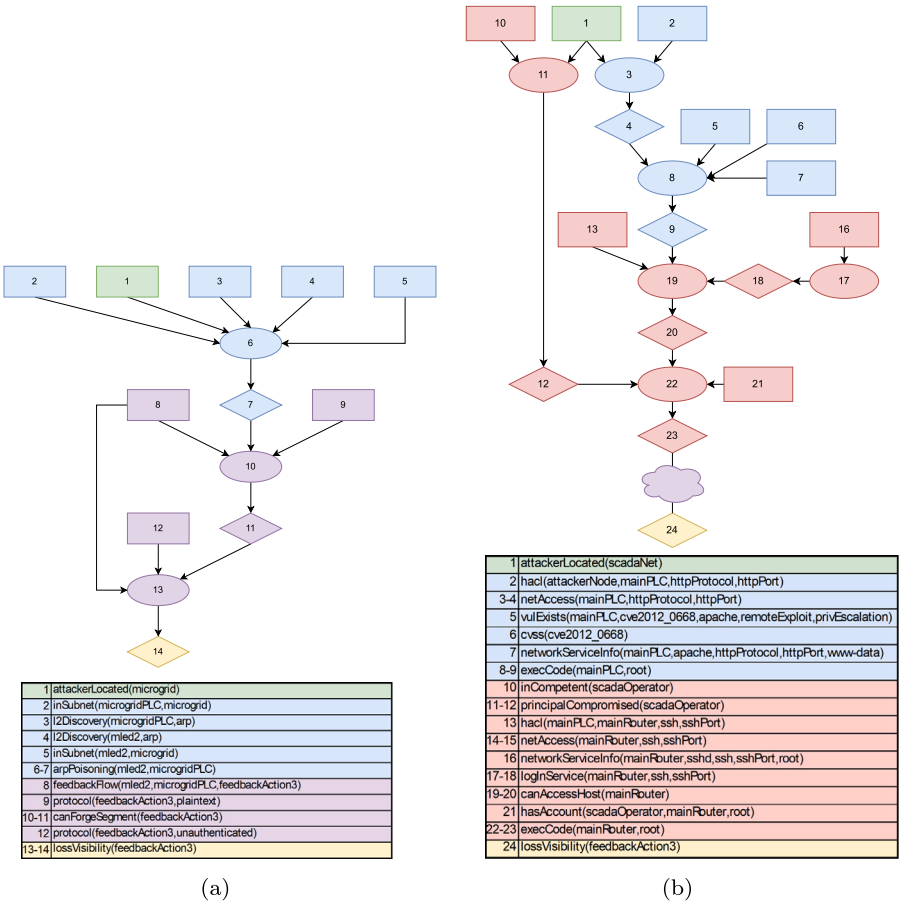


Fig. 11. The SCASS attack graph. The attack path can be represented by connecting leaves with a root.

the attacker can intercept all network traffic on the machine and alter the control flow data (*node 22-23*). The final steps of the attack graph (highlighted in purple) follow the same sequence described in Section 6.2.1.

6.3. Attacking SCASS

In this section, we implement the two attack scenarios associated with the attack paths identified through the previously descried methodology.[5]

6.3.1. False data injection through ARP poisoning

The attack involves a malicious actor that employs a TCP injection attack to alter generator operations and manipulate monitor-control cycles by injecting false Modbus TCP messages containing fabricated measurements. The architecture implemented to replicate an ICS cybersecurity attack is depicted in Fig. 12. We suppose that the attacker already compromised the microgrid network (i.e., 10.0.0.0/24 in Fig. 7), and their purpose is now to harm the system by manipulating the commands between the microgrid PLC and the intelligent device MIED1 driving one of the circuit breakers (CB4).

To conduct the attacks, a Kali Linux (Anon, 2023d) *Virtual Machine* (VM) instance is utilized (the "Attacker" rectangle in Fig. 12), to execute TCP injection attacks and manipulate the operations of generators within the target architecture. In the context of this study, we use Scapy (Anon, 2023h) to implement a *Transmission Control Protocol* (TCP) Injection technique for the *False Data Injection* (FDI) attack. The attack is carried out between the microgrid PLC and one of the IEDs responsible for controlling the generators. The objective is to manipulate the data flow, specifically introducing false measurements into the physical PLC, thereby causing the generation of incorrect commands.

The first step involves conducting a MITM attack using ARP spoofing and Modbus TCP server impersonation (Fig. 13). A rogue Modbus TCP server can be easily implemented using Py-Modbus,[6] a Python library specifically designed for Modbus TCP communication and manipulation. By employing this setup, the attacker can intercept Modbus TCP messages and redirect them to his/her own Modbus TCP server, ensuring that the operator terminal receives plausible values and remains unaware of the injected data. This allows the attacker to maintain control and prevent detection while manipulating the operational data within the system.

The second step involves packet sniffing, which is aimed at understanding the communications within the plant, especially the control actions. By examining the captured traffic, the attacker may notice a periodic exchange of values between the IP addresses 10.0.0.192 (the microgrid PLC) and 10.0.0.21 (the microgrid IED). The attacker may infer that the data sent by the IED represents some form of measurement, while the data flowing in the opposite direction may correspond to commands issued by the physical PLC to the IED, aimed at maintaining the generator G1 properly powered through the closure of the circuit breaker CB4. The circuit breaker in question is, in fact, driven by the intelligent electronic device MIED1, which is in turn controlled through the microgrid PLC (see Fig. 2).

The third step involves executing TCP Injection to introduce false measurements into the physical PLC, thereby triggering the generation

[5] For detailed instructions and examples of all the exploits discussed, refer to the walkthrough provided in the GitHub repository

[6] https://pymodbus.readthedocs.io

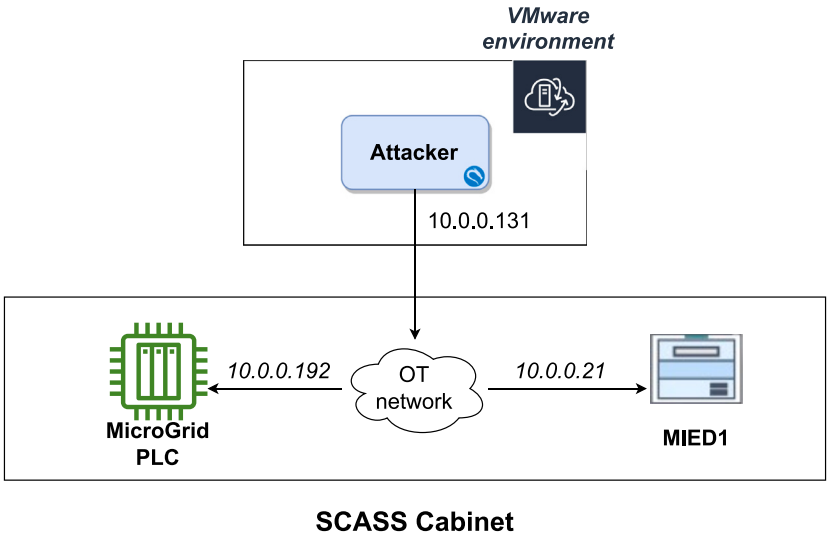


Fig. 12. Deployed architecture for attacking SCASS.

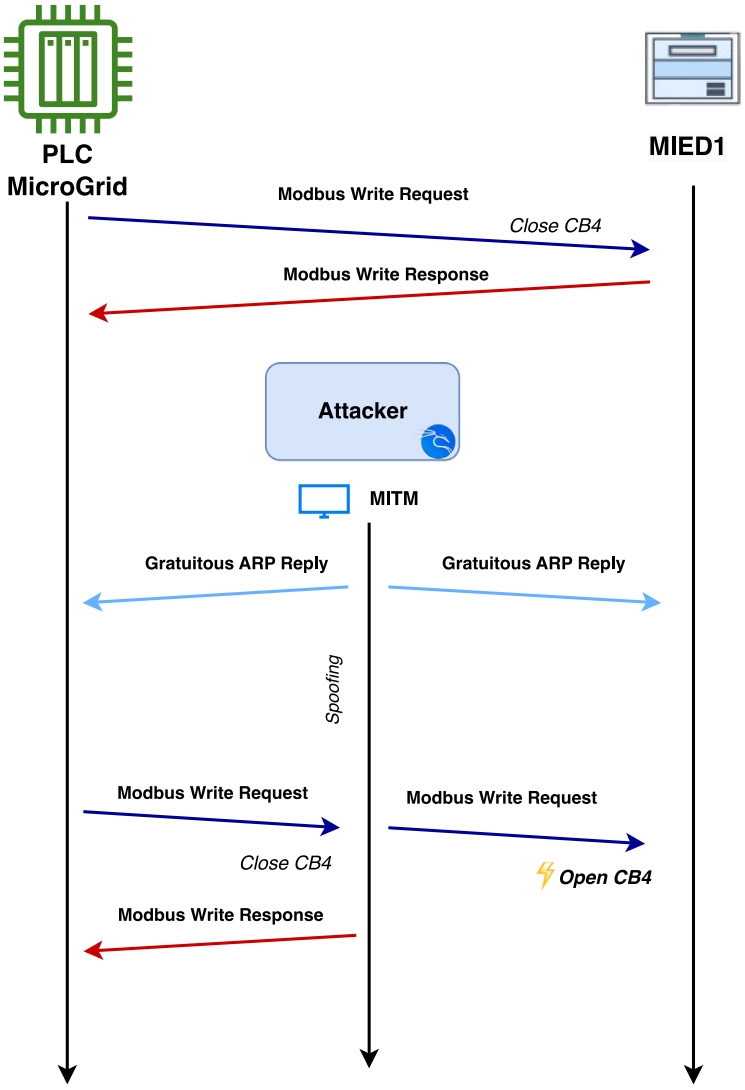


Fig. 13. Man-in-the-middle attack scenario. In this scenario, the attacker blocks the legitimate Modbus TCP write requests sent to keep CB4 closed and rather forces it to open, thus disconnecting the generator.

of incorrect values into the physical PLC, and eventually disrupting the control process. As a result, the attacker intercepts and blocks legitimate commands sent through the microgrid PLC, generating false Modbus TCP write commands to MIED1. This malicious action forces the opening of CB4, disconnecting the controlled generator and disrupting the system, which can no longer be properly powered, potentially causing significant damage.

6.3.2. Multi-stage false data injection attack

The attack scenario begins with a scanning of the Industrial Network under attack. This activity reveals the presence of hosts within the SCADA and PLC networks. These findings align with our expectations, since the attacker cannot contact internal networks, such as the Micro-Grid subnet, from the SCADA network. Furthermore, the service enumeration analysis reveals that the General-PLC, located in the PLC network, is vulnerable to the Shellshock RCE vulnerability (Panchal et al., 2018). As a result, the attacker can craft a malformed HTTP request to the URL http://10.0.1.66/cgi-bin/stats/ to execute arbitrary code on the vulnerable machine. Specifically, they can exploit this vulnerability to open a reverse shell and gain control over the target endpoint.

The next step in the attack scenario focuses on redirecting network traffic using the compromised machine as a pivot to exploit other hosts that before were unreachable. This task is achieved by establishing a port forwarding session between the compromised machine and the attacker node. After completing the pivoting process, the attacker performs reconnaissance activities on the newly accessible network segment. In this case, they discover that the Internal Gateway has a misconfigured SSH service allowing login with default credentials. In detail, the attacker can exploit this vulnerability to gain root access to this endpoint. With these elevated privileges, it is possible to upload a malicious payload that can manipulate the encrypted traffic passing through the router. Consequently, the attacker gains control over control action CA_3 within the Microgrid Subnet.

7. Discussion

A SCADA testbed evaluation is a complex task that requires a detailed analysis of various aspects. Alanazi et al. (2023) conducted a comprehensive review of SCADA vulnerabilities and attacks while also proposing a structured methodology for evaluating SCADA testbeds. Their approach builds on the requirements proposed by Mallouhi et al. (2011), which serve as a foundation for assessing the capabilities of SCADA security testbeds. In addition, they examine the security focus (application area) and security testing capabilities of each reviewed testbed.

Following this methodology, we evaluate the SCASS features from three distinct perspectives:

- **Characteristics**: the testbed functionality according to the requirements defined by Mallouhi et al. (2011) (as detailed in Table 4).
- **Application areas**: which type of security analysis is conducted on the proposed testbed (as detailed in Table 5).
- **Testing capability**: which types of attack scenarios can be implemented with the testbed (as detailed in Table 6).

7.1. SCASS features in detail

The SCASS testbed is evaluated against the illustrated features, providing a comprehensive analysis of its strengths, limitations, and potential in the ICS cybersecurity domain.

7.1.1. Characteristics

The SCASS source code is publicly available, and we have included electrical schematics and hardware details of the physical testbed components in A. This transparency is intended to enhance the reusability and extendibility of the testbed, making it easier for researchers and practitioners to adapt and build upon our work.

SCASS scalability is ensured through the use of container-based virtualization, which offers several advantages, including flexibility, resource efficiency, and ease of deployment. This approach allows for the rapid scaling of the testbed to accommodate diverse and complex industrial control system configurations without compromising performance. The application of a hybrid design improves domain fidelity by combining both virtual and physical components. This allows for more realistic simulations. Fidelity can be further enhanced by either integrating additional components or connecting the system to real physical processes, hence providing an even closer representation of real-world ICS environments.

Through SCASS, we demonstrate the ability to replicate the EPIC testbed, which simulates smart-grid processes. However, our goal extends beyond mere replication. We aim to enhance the testbed by conducting a detailed analysis of process simulations and providing

Table 4
Criteria for SCADA testbed evaluation - Characteristics.

Feature	Description
Reproducibility	The testbed is reproducible and reusable by other researchers.
Scalability	The testbed supports a diverse array of devices without any need to redesign the system.
Domain fidelity	The testbed uses physical hardware and is configured realistically.
Process simulation	The testbed mimics a real-time process.
Multiple ICS protocols	The testbed supports more than one ICS protocol.
Cost-effectiveness	The testbed is affordable for use in SCADA security research.

Table 5
Criteria for SCADA testbed evaluation - Application areas.

Feature	Description
Vulnerability analysis	The testbed is designed for discovering weaknesses in simulated SCADA systems.
Security validation	The testbed is designed for evaluating the vulnerability impact reduction after the application of security controls.
Impact analysis	The testbed can be used to analyze the impact of cyberattacks against a SCADA system.

Table 6
Criteria for SCADA testbed evaluation - Testing capabilities.

Feature	Description
Reconnaissance	Capability to replicate network scanning and probing activities.
(Cloud-based) brute force attack	Can simulate brute force attacks from cloud-based platforms.
Command injection	Supports the emulation of command injection attacks.
MITM (Man-In-The-Middle)	Enables the replication of Man-In-The-Middle attacks.
ARP Spoofing	Capable of simulating ARP spoofing to mimic network-level attacks.
DoS (Denial of Service)	Can reproduce Denial of Service attack scenarios.
Wireless Attacks	Allows replication of various wireless-based attacks.
Sniffing Attacks	Supports the simulation of packet sniffing activities.
Vulnerability discovery	Provides tools for simulating vulnerability discovery techniques.

Table 7
Characteristics evaluation components.

Reproducibility		
Source code availability	○ ●	No Yes
Hardware configuration details	○ ◐ ●	Few details Some details Many details
Network configuration details	○ ◐ ●	Few details Some details Many details
Scalability		
Virtualization type	V C	Classic virtualization Container-based virtualization
Domain fidelity		
Presence of real devices	○ ●	No Yes
Hardware configuration details	○ ◐ ●	Few details Some details Many details
Process simulation		
Component	*Values*	*Details*
Process type	S P	Simulated Physical
Multiple ICS protocols		
Component	*Values*	*Details*
Implemented protocols	Modbus, DNP3, etc.	The implemented protocols
Cost-effectiveness		
Component	*Values*	*Details*
Cost details	○ ●	No details Cost details

formal verifications to ensure the fidelity of the representation of real-world processes. As discussed in previous sections, this work currently supports only the Modbus protocol, which represents a limitation given the presence of newer protocols in modern SCADA infrastructures. However, in Section 7.3.1, we provide detailed instructions on how SCASS can be configured to either replace physical devices or integrate virtual ones, without compromising the overall architecture of the testbed. This flexibility allows for future adaptability to support additional protocols as needed.

SCASS has been designed with portability and cost-effectiveness in mind, making it relatively affordable for security researchers. Table A.13 provides an estimated breakdown of the total cost. It is important to note that the overall cost can be further reduced by replacing physical devices with virtual counterparts, although this would come at the expense of domain fidelity.

7.1.2. Application areas

The modular design of the SCASS testbed facilitates the creation of digital twins, as demonstrated in Section 5. This capability allows for the vulnerability analysis of real SCADA systems, with an example scenario detailed in Section 6.3. While our focus was not exclusively on validating security control solutions, we provide insights into integrating security solutions, such as Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), within the SCASS framework in Section 7.3.1. We are also exploring the use of SCASS for experimenting with Moving Target Defense (MTD) techniques, particularly through the application of the Software Defined Networking (SDN) paradigm. These aspects will be addressed in future developments of SCASS.

7.1.3. Testing capabilities

Reconnaissance, ARP spoofing, and Man In The Middle (MITM) attacks are feasible within the SCASS network architecture due to the unencrypted nature of the Modbus protocol, allowing for the replication of sniffing attacks. However, the testbed has limitations when it comes to cloud-based brute force attacks and wireless scenarios. Indeed, the testbed is wired and lacks cloud components, which restricts the implementation of cloud-based attacks. To incorporate cloud-based offensive scenarios, integration with the MQTT protocol could be considered, enabling publish–subscribe architectures with a cloud-based broker. Nevertheless, this integration might conflict with the design goals of cost-effectiveness and portability. As to wireless attacks, while they are relatively simple to implement given the onboard WiFi capabilities of Raspberry Pi 3 (or through cost-effective USB WiFi adapters), we have not focused on them within SCASS, as of yet. This aspect requires further study to create representative wireless attack scenarios for SCADA systems. These limitations will be explored in future works.

7.2. Comparison with other testbeds

In Section 3, we have reviewed various related works that employ different approaches, including physical, virtual, and hybrid designs. According to the analysis by Qassim et al. (2017), hybrid architectures offer significant advantages by combining the scalability and cost benefits of virtual components with the domain fidelity provided by physical elements.

In this section, we compare our SCASS solution with other hybrid solutions, utilizing the features outlined in Section 7.1 to evaluate and contrast their capabilities and benefits.

For each feature of each category, we define a set of parameters to evaluate the testbeds and obtain a rating score of their characteristics, application areas, and testing capabilities, as per the mentioned taxonomy defined in Mallouhi et al. (2011). Tables 7, 8, and 9 illustrate the detailed criteria adopted for evaluating hybrid testbeds.

The features related to the testbed characteristics are benchmarked in Tables 10 and 11. Table 12 instead summarizes the results of the

Table 8
Application areas evaluation components.

Component	Values	Details
Vulnerability analysis	○	No attack scenarios
	●	Attack scenarios presented
Security validation	○	No security controls integrated into the testbed
	◐	Not present in the current architecture, but suggestions on how to integrate them.
	●	Security controls integrated in the testbed
Impact analysis	○	Impact analysis not performed
	●	Impact analysis performed

Table 9
Testing capabilities evaluation components.

Component	Values	Details
Coverage level (per attack category)	○	Troublesome implementation
	◐	Potentially implementable
	◐	Not implemented but discussed
	●	Covered

comparative evaluation for both the "Application areas" category and the "Testing capabilities" category.

The SCASS testbed demonstrates significant strengths in reproducibility and transparency. Its source code is publicly available, which facilitates replication and further development by other researchers. Additionally, SCASS provides detailed information about both hardware and networking configurations, which enhances its usability and transparency.

Another key advantage of SCASS is its cost-effectiveness. By including a comprehensive cost estimation of the physical components (as detailed in Table A.13), the testbed allows researchers and security practitioners to assess the budgetary implications and feasibility of replicating the setup. Cost transparency aids in informed decision-making and reduces entry barriers for those utilizing or building upon SCASS for research or security testing purposes. Despite SCASS advantages, there are limitations that need to be addressed. One such limitation is the current domain fidelity of SCASS. The testbed primarily simulates industrial control processes rather than replicating them with high accuracy. Although this approach provides valuable insights, the fidelity could be significantly enhanced by incorporating additional physical devices into the SCASS setup. However, this enhancement would come with increased costs, which must be carefully weighed against the benefits of improved domain fidelity.

Furthermore, SCASS is currently designed with support for a single ICS protocol, specifically Modbus. As discussed in the previous sections of the paper, this choice reflects both the widespread use of Modbus in ICS environments and the practical constraints of the testbed. However, the inclusion of other modern protocols could broaden the applicability of SCASS. Additionally, the current implementation does not incorporate integrated security controls, such as intrusion detection or prevention systems. Addressing this gap could enhance the testbed's utility in evaluating and developing comprehensive security solutions.

These limitations and potential improvements are discussed in detail in the following section, where we explore strategies for enhancing the capabilities of our testbed and address its current constraints.

Additionally, it is important to note that while the testbeds reviewed in our analysis, including SCASS, offer various features and functionality, none of them specifically addresses cloud-based brute force attacks. This is an area that remains underexplored, yet presents significant research opportunities. Furthermore, although some testbeds could potentially be adapted to include wireless attacks, there remains a need for dedicated solutions that address the unique challenges posed by this specific type of attack scenarios. We encourage future research to focus on these areas to advance the development of comprehensive and adaptable security testing environments.

7.3. SCASS limitations and proposed solutions

A careful analysis of the SCASS testbed allows us to identify a few issues and challenges that are still to be faced. In the following subsections, we summarize our findings and propose potential solutions.

7.3.1. Limited set of communication protocols

SCASS components rely exclusively on the Modbus communication protocol. Although Modbus is an old protocol, its continued prevalence in industrial systems (largely due to the widespread deployment of legacy infrastructures) justifies this choice. Supporting this decision, a study (Irmak and Erkek, 2018) reports that Modbus still accounts for 50% of protocol usage in SCADA enterprise systems. Though, recent studies show that industrial systems are gradually shifting toward newer communication protocols. To remain effective, SCASS should hence cover them. To address this limitation, we investigated two possible approaches for substituting industrial protocols. The first approach involves replacing the existing physical devices with ones that inherently support the desired modern protocols. This method would enhance domain fidelity, even though it might increase costs. The second approach involves virtualizing all physical PLCs. This technique enables the system to support multiple communication protocols without requiring additional physical hardware. A variety of community-developed libraries are available to help construct the necessary programming logic.

Modbus and DNP3 utilize master/slave communication protocols. Consequently, this substitution would require little effort. However, integrating Profinet may demand additional adjustments due to its consumer/producer communication paradigm. Notably, zone PLCs act as I/O controllers in a Profinet-based network. Indeed, they produce the configuration working parameters for the I/O devices, while simultaneously consuming the state information coming from them. On the other hand, IEDs act as I/O devices within the network. Therefore, they will be consumers of output data from I/O controllers and producers of state information. Similarly, integrating MQTT requires modifications to the network infrastructure due to its publisher/subscriber nature. In this context, each node operates both as a publisher and a subscriber, depending on the topics. Controlled devices would publish their state information on specific topics to which the controller device subscribes. Conversely, the controller device publishes configuration parameters to topics to which the controlled devices subscribe. GOOSE follows the same publish/subscribe architecture but does not require a centralized message broker as it relies on multicast messaging over Ethernet for communicating.

Table 10
Comparison in terms of characteristics evaluation components.

	Mallouhi et al. (2011)	Morris et al. (2011)	Singh et al. (2015)	Koutsandria et al. (2015)	Keliris et al. (2016)	Ghaleb et al. (2016)	Rosa et al. (2017)	Bernieri et al. (2016)	Foglietta et al. (2019)	Xie et al. (2018)	Čeleda et al. (2020)	Cruz and Simões (2021)	SCASS
Source code availability	○	○	○	○	○	●	○	●	○	○	○	○	●
Hardware configuration details	●	◐	◐	●	●	●	◐	●	◐	◐	◐	◐	●
Network configuration details	○	◐	●	○	◐	◐	◐	◐	●	●	○	●	●
Virtualization type	V	V	V	V	V	V	V	V	V	V	C	C	C
Presence of real devices	●	●	○	●	●	●	●	●	●	●	●	●	●
Hardware configuration details	●	◐	◐	●	◐	●	●	●	◐	◐	◐	◐	●
Process type	S	P	S	P	P	S	P	P	P	S	P	P	S
Cost details	○	○	○	○	○	○	○	○	○	○	○	○	●

Table 11
Comparison in terms of implemented ICS protocols.

Work	Supported ICS protocols
Mallouhi et al. (2011)	Modbus, DNP3
Morris et al. (2011)	Modbus, DNP3
Koutsandria et al. (2015)	Modbus, IEC 61580
Ghaleb et al. (2016)	Modbus
Keliris et al. (2016)	Modbus
Singh et al. (2015)	DNP3, IEC-60870-5-101, and Blowfish encryption.
Rosa et al. (2017)	Modbus
Bernieri et al. (2016)	Modbus
(Foglietta et al., 2019)	Modbus
Xie et al. (2018)	OPC, Modbus, Siemens S7
(Čeleda et al., 2020)	Modbus
Cruz and Simões (2021)	Modbus
SCASS	Modbus

7.3.2. Security validation

Unlike other testbeds, SCASS does not include integrated security controls. This is because SCASS was not specifically designed to evaluate the effectiveness of protection measures in reducing the risks associated with ICS systems. As such, the "Security controls presence" application area is not a key focus of the SCASS platform. However, this does not mean that security capabilities cannot be incorporated into the system. On the contrary, SCASS is designed with flexibility in mind, allowing for the seamless integration of network security measures without altering the core architecture.

In SCASS, routing is managed by headless containers, which simplifies the integration of security mechanisms like *Intrusion Detection Systems* (IDSs). Thanks to container-based networking, incorporating intrusion detection systems into the platform is straightforward and can enhance the overall security analysis.

7.3.3. Attack complexity

One of the attacks (namely, the False Data Injection with ARP poisoning) illustrated in Section 6 is intentionally basic and does not capture the full complexity of modern SCADA systems. The second attack (i.e., the Multi-stage False Data Injection) does represent a concrete example of a more advanced scenario that proceeds through multiple stages of the so-called cyber kill chain, including lateral movement and privilege escalation. We chose these two attack categories in order to demonstrate SCASS ability to seamlessly reproduce scenarios with different complexity levels. We have demonstrated in the paper that the threat analysis task leveraging the STAMP/STPA-Sec framework serves as a functional validation of the testbed's effectiveness. The flexibility of the approach allows it to be easily employed both for basic and for more advanced attack simulations.

8. Conclusions

This work has introduced SCASS as a robust and versatile architecture for replicating heterogeneous security testbeds. The SCASS design is based on a hybrid approach, combining both physical and virtual elements. Our contributions can be summarized as follows:

- We propose a hybrid and modular (Alves et al., 2018) testbed for SCADA cybersecurity based on lightweight virtualization, a combination that offers several benefits in terms of repeatability, safety, cost-savings, fidelity, and scalability (Qassim et al., 2017).
- We replicate the EPIC architecture through SCASS by integrating container-based virtualization through the adoption of the Macvlan feature, which addresses the networking challenges identified by the authors of the related EPICTWIN (Kandasamy et al., 2021) testbed.
- We showcase a threat analysis methodology and a practical example of how it is possible to configure a system to realize offensive scenarios.
- We benchmark our testbed against valuable alternatives proposed in the literature, by relying on the features identified by the inspiring works of Mallouhi et al. (2011) and Alanazi et al. (2023).
- We identify and discuss SCASS limitations, while also proposing effective solutions for addressing them.

In our future investigations, we aspire to enhance the capabilities of the testbed by reproducing a complete grid within the cabinet, utilizing a, heterogeneous set of physical devices. Additionally, we aim to broaden the scope of the target architecture by incorporating other commonly used protocols in OT systems, such as MQTT or IEC 61850. We plan to diversify the range of both physical and virtual devices, exploring alternatives like IFTTT and n8n.io, and incorporating libraries such as PyModbusTCP and OpenPLC.

In the context of modeling cybersecurity threats in the ICS domain, various methodologies and approaches have been developed (Bhamare et al., 2020b). Notably, security contextualization of well-established safety models (Kriaa et al., 2015; Leveson, 2004; Yan et al., 2016; Young and Leveson, 2013, 2014) has been widely adopted. In our upcoming work, we aim to practically apply such methodologies within SCASS to identify comprehensive attack graphs.

Through the SCASS platform, we have explored attack scenarios within the ICS domain. Alanazi et al. (2023) offer a comprehensive taxonomy of attacks against SCADA systems. In future work, we plan to leverage this taxonomy to formally verify the SCASS testbed's ability to reproduce diverse and complex attack scenarios. This will provide a more rigorous validation of SCASS as a platform for cybersecurity experimentation and defense analysis in the SCADA domain.

Moreover, the SCASS environment benefits from the integration of cyber-range functionality, including scoring or monitoring systems. The inclusion of these features would facilitate the reproduction of red-team versus blue-team cyber simulation scenarios, offering a holistic testing ground for cybersecurity measures. This can be further enhanced by integrating security controls in the architecture.

Looking ahead, we plan to expand SCASS by deploying lightweight components, such as Wazuh agents (Katulić et al., 2024), across the testbed infrastructure. These agents will systematically collect critical data, including raw network traffic, system performance metrics, and logs. This data will be invaluable for deepening our understanding of attacker activities, as well as assessing the impact of cyberattacks on ICS environments. The insights gained through such improvements will support more comprehensive security evaluations and contribute to a better understanding of threats targeting both SCADA and ICS systems.

Finally, we are also planning to enhance SCASS by incorporating Moving Target Defense (MTD) functionality through the application of Software Defined Networking (SDN). This approach will add a dynamic security layer to SCASS, allowing for continuous changes in the system's attack surface to make it more resilient against cyberattacks. The current organization of the testbed lends itself well to this upgrade, as the integration of SDN can be easily accomplished using containerized versions of Open vSwitches.[7]

The continuous evolution and refinement of SCASS align with the dynamic landscape of cybersecurity, ensuring its relevance and effectiveness in addressing emerging threats and challenges in the ICS domain.

[7] https://www.openvswitch.org/

Table 12
Comparison in terms of Application areas and Testing capabilities.

	Mallouhi et al. (2011)	Morris et al. (2011)	Singh et al. (2015)	(Koutsandria et al., 2015)	Keliris et al. (2016)	Ghaleb et al. (2016)	Rosa et al. (2017)	Bernieri et al. (2016)	Foglietta et al. (2019)	Xie et al. (2018)	Čeleda et al. (2020)	Cruz and Simões (2021)	SCASS
Application areas													
Vulnerability analysis	●	●	●	○	●	●	●	○	●	●	○	●	●
Security validation	●	●	●	●	●	○	○	●	○	◐	○	●	◐
Impact analysis	○	○	●	○	●	●	○	●	●	●	○	○	●
Testing capabilities													
Reconnaissance	◐	◐	◐	◐	●	◐	●	◐	◐	●	◐	◐	●
Cloud-based brute force	○	○	○	○	○	○	○	○	○	○	○	○	○
Command injection	◐	◐	◐	◐	●	◐	◐	◐	◐	◐	◐	◐	◐
MITM	●	○	◐	●	◐	●	●	◐	●	◐	●	◐	●
ARP Spoofing	●	○	◐	●	◐	●	●	◐	●	◐	●	◐	●
Wireless attacks	◐	○	○	◐	◐	◐	○	◐	◐	◐	◐	◐	◐
Sniffing attacks	◐	○	◐	◐	◐	◐	●	◐	◐	◐	◐	◐	●
Vulnerability discovery	◐	◐	◐	◐	◐	◐	●	◐	◐	◐	◐	◐	●

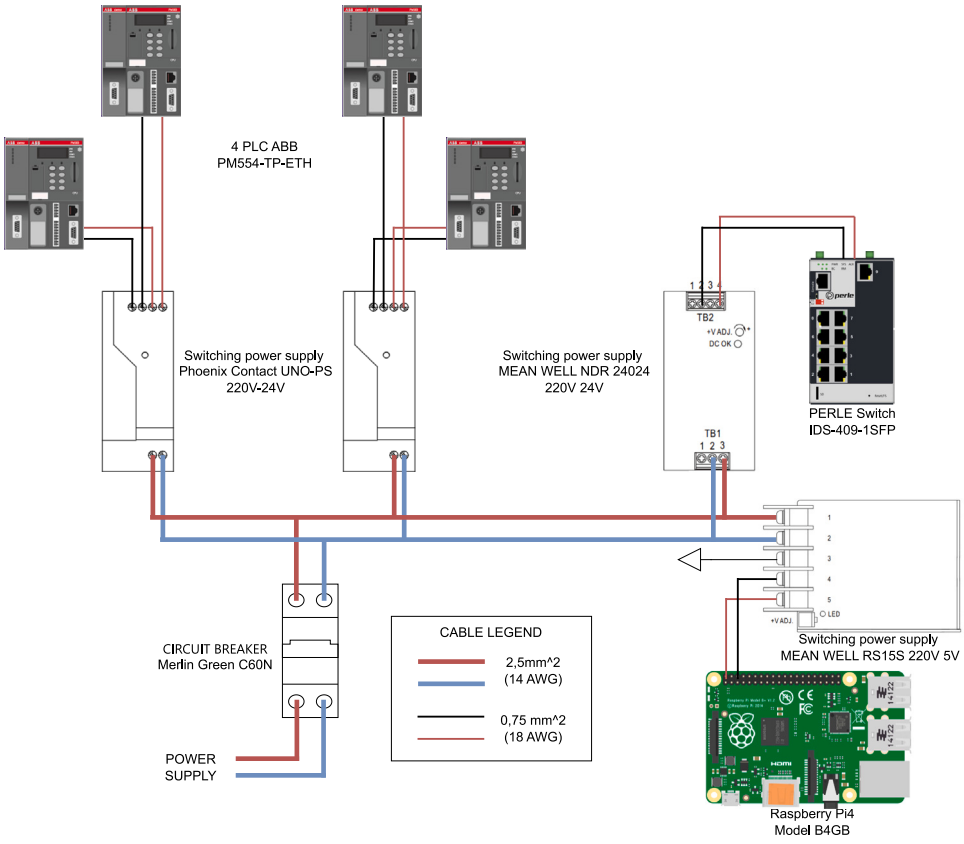


Fig. A.14. SCASS circuit diagram.

CRediT authorship contribution statement

Nicola d'Ambrosio: Writing – review & editing, Validation, Software, Methodology, Conceptualization. **Giulio Capodagli:** Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Conceptualization. **Gaetano Perrone:** Writing – original draft, Validation, Software, Methodology, Formal analysis, Conceptualization. **Simon Pietro Romano:** Writing – review & editing, Supervision, Project administration, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The development of SCASS has been carried out in collaboration with "Comando per le Operazioni in Rete" (COR), the main Italian military interforce unit specializing in cyber defense and cyber security operations. COR has provided the necessary hardware required for the project's implementation. This work has been funded by the project Azione IV.4 - Dottorati e contratti di ricerca su tematiche dell'innovazione - DM 1061 del 10 agosto 2021, Italy (CUP: E65F21003690003).

Appendix A. Hardware configuration details

In this appendix, we provide detailed information about SCASS hardware components. In particular, Fig. A.14 illustrates the SCASS circuit diagram, while Table A.13 provides the list of currently employed hardware components with associated costs. Moreover, in order to facilitate the testbed replicability, we developed virtual components of programmable logic controllers (PLCs) to enable the deployment of the testbed without relying on physical PLCs, albeit with reduced fidelity. Finally, we released several video clips with deployment instructions.[8]

Table A.13
SCASS physical components and related costs.

Number	Component	Estimated price in dollars (2024)
1	CIRCUIT BREAKER Merlin Green C60N	6,7
4	PLC ABB PM554-TP-ETH	660
1	PERLE Switch IDS-409-1SFP	1.605,00
1	Raspberry Pi 4	60
2	Switching power supply Phoenix Contact UNO-PS 220V-24V	55
1	Switching power supply MEAN WELL RS15S 220V 5V	13,9
1	Switching power supply MEAN WELL NDR 24024 220V 24V	66,72

Data availability

We aim to share publicly the source code on GitHub.

References

Adepu, S., Kandasamy, N.K., Mathur, A., 2019. EPIC: An electric power testbed for research and training in cyber physical systems security. In: Computer Security. Springer International Publishing, pp. 37–52. http://dx.doi.org/10.1007/978-3-030-12786-2_3.

[8] https://github.com/NS-unina/SCASS/tree/master/docs/video

Alanazi, M., Mahmood, A., Chowdhury, M.J.M., 2023. SCADA vulnerabilities and attacks: A review of the state-of-the-art and open issues. Comput. Secur. 125, 103028. http://dx.doi.org/10.1016/j.cose.2022.103028, URL https://www.sciencedirect.com/science/article/pii/S0167404822004205.

Alves, T., Das, R., Werth, A., Morris, T., 2018. Virtualization of SCADA testbeds for cybersecurity research: A modular approach. Comput. Secur. 77, 531–546. http://dx.doi.org/10.1016/j.cose.2018.05.002, URL https://www.sciencedirect.com/science/article/pii/S0167404818304905.

Anon, 2017. Idaho national laboratory (INL) smart grid test bed revision 2. IdahoSmartGrid, https://www.energy.gov/sites/prod/files/2017/10/f38/CX-270322.pdf.

Anon, 2021. ElectricPower and intelligent control (EPIC) testbed. EPICiTrust, https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_epic.

Anon, 2023a. ISO/IEC 20922:2016(en) information technology — Message queuing telemetry transport (MQTT) v3.1.1 (preview). MQTTprotocol https://www.iso.org/obp/ui/#iso:std:iso-iec:20922:ed-1:v1:en.

Anon, 2023b. Drago, chernovite's PIPEDREAM malware targeting industrial control systems (ICS). Drago https://www.dragos.com/blog/industry-news/chernovite-pipedream-malware-targeting-industrial-control-systems/.

Anon, 2023c. IEEE standard for electric power systems communications-distributed network protocol (DNP3). http://dx.doi.org/10.1109/ieeestd.2012.6327578, DNP3protocol.

Anon, 2023d. Kali linux home page. KaliLinuxDistribution. https://www.kali.org/.

Anon, 2023e. MulVal, a multi stage Vulnerability Analysis tool. MulVal. https://github.com/risksense/mulval/.

Anon, 2023f. OpenPLC site. OpenPLC https://openplcproject.com/.

Anon, 2023g. PROFINET system description. PROFINETprotocol http://us.profinet.com/wp-content/uploads/2012/11/PROFINET_SystemDescription_ENG_2014_web.pdf.

Anon, 2023h. Scapy project. Scapyproject https://scapy.net/.

Anon, 2023i. Wired, fed's uncover a 'swiss army knife' for hacking industrial control systems. WiredPipedream, https://www.wired.com/story/pipedream-ics-malware/.

Anon, 2023j. Wired, Russia's sandworm hackers attempted a third blackout in Ukraine. WiredIndustroyer https://www.wired.com/story/sandworm-russia-ukraine-blackout-gru/.

Artac, M., Borovssak, T., Di Nitto, E., Guerriero, M., Tamburri, D.A., 2017. DevOps: Introducing infrastructure-as-code. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). pp. 497–498. http://dx.doi.org/10.1109/ICSE-C.2017.162.

Bernieri, G., Moro, F.D., Faramondi, L., Pascucci, F., 2016. A testbed for integrated fault diagnosis and cyber security investigation. In: 2016 International Conference on Control, Decision and Information Technologies (CoDIT). IEEE, pp. 454–459. http://dx.doi.org/10.1109/codit.2016.7593605.

Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K., Meskin, N., 2020a. Cybersecurity for industrial control systems: A survey. Comput. Secur. 89, 101677. http://dx.doi.org/10.1016/j.cose.2019.101677, URL https://www.sciencedirect.com/science/article/pii/S0167404819302172.

Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K., Meskin, N., 2020b. Cybersecurity for industrial control systems: A survey. Comput. Secur. 89, 101677. http://dx.doi.org/10.1016/j.cose.2019.101677.

Boettiger, C., 2015. An introduction to docker for reproducible research. Oper. Syst. Rev. 49 (1), 71–79.

Castiglione, L.M., Hau, Z., Get, P., Co, K.T., Munoz-Gonzalez, L., Teng, F., Lupu, E., 2022. HA-grid: Security aware hazard analysis for smart grids. In: 2022 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). IEEE, pp. 446–452. http://dx.doi.org/10.1109/smartgridcomm52983.2022.9961003.

Caturano, F., Perrone, G., Romano, S.P., 2020. Capturing flags in a dynamically deployed microservices-based heterogeneous environment. In: 2020 Principles, Systems and Applications of IP Telecommunications (IPTComm). pp. 1–7. http://dx.doi.org/10.1109/IPTComm50535.2020.9261519.

Čeleda, P., Vykopal, J., Švábenský, V., Slavíček, K., 2020. KYPO4indUStry. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education. ACM, pp. 1026–1032. http://dx.doi.org/10.1145/3328778.3366908.

Chan, J., Reith, M., 2022. Cyber concerns with cloud computing. Eur. Conf. Cyber Warf. Secur. http://dx.doi.org/10.34190/eccws.21.1.429.

CODESTS, 2021. CODESYS development system. https://help.codesys.com/webapp/_cds_f_development_system_introduction;product=codesys;version=3.5.17.0 Online; 27-Jul-2023.

Conti, M., Donadel, D., Turrin, F., 2021. A survey on industrial control system testbeds and datasets for security research. IEEE Commun. Surv. Tutor. 23 (4), 2248–2294. http://dx.doi.org/10.1109/comst.2021.3094360.

Cruz, T., Simões, P., 2021. Down the rabbit hole: Fostering active learning through guided exploration of a SCADA cyber range. Appl. Sci. 11 (20), 9509. http://dx.doi.org/10.3390/app11209509.

Cui, H., Hong, J., Louden, R., 2024. An overview of the security of programmable logic controllers in industrial control systems. Encyclopedia 4 (2), 874–887. http://dx.doi.org/10.3390/encyclopedia4020056, URL https://www.mdpi.com/2673-8392/4/2/56.

Decotignie, J., 2005. Ethernet-based real-time and industrial communications. Proc. IEEE 93, 1102–1117. http://dx.doi.org/10.1109/JPROC.2005.849721.

Dietz, M., Pernul, G., 2020. Unleashing the digital twin's potential for ICS security. IEEE Secur. Priv. 18, 20–27. http://dx.doi.org/10.1109/MSEC.2019.2961650.

Ferencz, K., Domokos, J., 2019. Using node-RED platform in an industrial environment. In: XXXV. Jubileumi Kandó Konferencia, Budapest. pp. 52–63.

Fidler, D.P., 2011. Was stuxnet an act of war? Decoding a cyberattack. IEEE Secur. Priv. Mag. 9 (4), 56–59. http://dx.doi.org/10.1109/msp.2011.96.

Foglietta, C., Masucci, D., Palazzo, C., Santini, R., Panzieri, S., Rosa, L., Cruz, T., Lev, L., 2019. From detecting cyber-attacks to mitigating risk within a hybrid environment. IEEE Syst. J. 13 (1), 424–435. http://dx.doi.org/10.1109/JSYST.2018.2824252.

Fortinet, 2024. 2024 state of operational technology and cybersecurity report. https://www.fortinet.com/content/dam/fortinet/assets/reports/report-state-ot-cybersecurity.pdf Online; 15-Nov-2024.

frangoteam, 2020. FUXA. https://github.com/frangoteam/FUXA Online; 18-Nov-2024.

Friedberg, I., McLaughlin, K., Smith, P., Laverty, D., Sezer, S., 2017. STPA-SafeSec: Safety and security analysis for cyber-physical systems. J. Inf. Secur. Appl. 34, 183–196. http://dx.doi.org/10.1016/j.jisa.2016.05.008.

Ghaleb, A., Zhioua, S., Almulhem, A., 2016. SCADA-SST: a SCADA security testbed. In: 2016 World Congress on Industrial Control Systems Security. WCICSS, pp. 1–6. http://dx.doi.org/10.1109/WCICSS.2016.7882610.

Haag, S., Anderl, R., 2018. Digital twin – proof of concept. Manuf. Lett. 15, 64–66. http://dx.doi.org/10.1016/j.mfglet.2018.02.006.

He, R., Chen, G., Dong, C., Sun, S., Shen, X., 2019. Data-driven digital twin technology for optimized control in process systems. ISA Trans. http://dx.doi.org/10.1016/j.isatra.2019.05.011.

Holmes, D., Papathanasaki, M., Maglaras, L., Ferrag, M.A., Nepal, S., Janicke, H., 2021. Digital twins and cyber security – solution or challenge? In: 2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM). pp. 1–8. http://dx.doi.org/10.1109/SEEDA-CECNSM53056.2021.9566277.

Howard, D., 2019. The digital twin: Virtual validation in electronics development and design. In: 2019 Pan Pacific Microelectronics Symposium (Pan Pacific). pp. 1–9. http://dx.doi.org/10.23919/PanPacific.2019.8696712.

Irmak, E., Erkek, I., 2018. An overview of cyber-attack vectors on SCADA systems. In: 2018 6th International Symposium on Digital Forensic and Security. ISDFS, pp. 1–5. http://dx.doi.org/10.1109/ISDFS.2018.8355379.

Kandasamy, N.K., Venugopalan, S., Wong, T.K., Leu, N.J., 2022. An electric power digital twin for cyber security testing, research and education. Comput. Electr. Eng. 101, 108061. http://dx.doi.org/10.1016/j.compeleceng.2022.108061.

Kandasamy, N.K., Venugopalan, S., Wong, T.K., Nicholas, L.J., 2021. EPICTWIN: An electric power digital twin for cyber security testing, research and education. arXiv:arXiv:2105.04260.

Kara, İ., Aydos, M., 2019. The ghost in the system: technical analysis of remote access trojan. Int. J. Inf. Technol. Secur. 11 (1), 73–84.

Katulić, F., Groš, S., Sumina, D., Erceg, I., 2024. Enhancing industrial automation and control systems cybersecurity using endpoint detection and response tools. In: Auer, M.E., Langmann, R., May, D., Roos, K. (Eds.), Smart Technologies for a Sustainable Future. Springer Nature Switzerland, Cham, pp. 186–197.

Keliris, A., Salehghaffari, H., Cairl, B., Krishnamurthy, P., Maniatakos, M., Khorrami, F., 2016. Machine learning-based defense against process-aware attacks on industrial control systems. In: 2016 IEEE International Test Conference. ITC, pp. 1–10. http://dx.doi.org/10.1109/TEST.2016.7805855.

Koutsandria, G., Gentz, R., Jamei, M., Scaglione, A., Peisert, S., McParland, C., 2015. A real-time testbed environment for cyber-physical security on the power grid. In: Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/Or PrivaCy. ACM, pp. 67–78. http://dx.doi.org/10.1145/2808705.2808707.

Kriaa, S., Pietre-Cambacedes, L., Bouissou, M., Halgand, Y., 2015. A survey of approaches combining safety and security for industrial control systems. Reliab. Eng. Syst. Saf. 139, 156–178. http://dx.doi.org/10.1016/j.ress.2015.02.008.

Langner, R., 2011. Stuxnet: Dissecting a cyberwarfare weapon. IEEE Secur. Priv. Mag. 9 (3), 49–51. http://dx.doi.org/10.1109/msp.2011.67.

Lee, S.H., Shin, S.-M., Hwang, J.S., Park, J., 2020. Operational vulnerability identification procedure for nuclear facilities using STAMP/STPA. IEEE Access 8, 166034–166046. http://dx.doi.org/10.1109/access.2020.3021741.

Leveson, N., 2004. A new accident model for engineering safer systems. Saf. Sci. 42 (4), 237–270. http://dx.doi.org/10.1016/s0925-7535(03)00047-x.

Leveson, N., 2016. Engineering a Safer World. The MIT Press.

Makrakis, G.M., Kolias, C., Kambourakis, G., Rieger, C., Benjamin, J., 2021. Industrial and critical infrastructure security: Technical analysis of real-life security incidents. IEEE Access PP, http://dx.doi.org/10.1109/ACCESS.2021.3133348, 1–1.

Mallouhi, M., Al-Nashif, Y., Cox, D., Chadaga, T., Hariri, S., 2011. A testbed for analyzing security of SCADA control systems (TASSCS). In: ISGT 2011. pp. 1–7. http://dx.doi.org/10.1109/ISGT.2011.5759169.

Mark Hearn, S.R., 2019. Cybersecurity Considerations for Digital Twin. Industrial Internet Consortium.

Masi, M., Sellitto, G.P., Aranha, H., Pavleska, T., 2023. Securing critical infrastructures with a cybersecurity digital twin. Softw. Syst. Model. 22 (2), 689–707.

Mathur, A.P., 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). IEEE, pp. 31–36.

Maynard, P., 2018. Peter Maynard's GitHub. https://github.com/PMaynard/ICS-TestBed-Framework.

Maynard, P., McLaughlin, K., Sezer, S., 2018. An open framework for deploying experimental SCADA testbed networks. In: Electronic Workshops in Computing. BCS Learning & Development, pp. 92–101. http://dx.doi.org/10.14236/ewic/ics2018.11.

Mehta, K., Joshi, R., Kulkarni, S., Soni, B., 2015. Implementation of PLC based software prototype for 45.6 MHz, 100 kW, ICRH DAC using EPICS control system. Int. J. Sci. Eng. Res. 6.

Morris, T., Vaughn, R., Dandass, Y.S., 2011. A testbed for SCADA control system cybersecurity research and pedagogy. In: Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research. CSIIRW '11, Association for Computing Machinery, New York, NY, USA, http://dx.doi.org/10.1145/2179298.2179327.

Nicolae, A., Korodi, A., 2018. Node-red and OPC UA based lightweight and low-cost historian with application in the water industry. In: 2018 IEEE 16th International Conference on Industrial Informatics. INDIN, pp. 1012–1017. http://dx.doi.org/10.1109/INDIN.2018.8471928.

NIST, 2020. Industrial control system. https://csrc.nist.gov/glossary/term/industrial_control_system Online; 27-Aug-2024.

Niu, H., Ma, C., Wang, C., Han, P., 2018. Hazard analysis of traffic collision avoidance system based on STAMP model. In: 2018 IEEE International Conference on Progress in Informatics and Computing (PIC). IEEE, pp. 445–450. http://dx.doi.org/10.1109/pic.2018.8706283.

Olivares-Rojas, J.C., Reyes-Archundia, E., Gutiérrez-Gnecchi, J.A., Molina-Moreno, I., Cerda-Jacobo, J., Méndez-Patiño, A., 2022. Towards cybersecurity of the smart grid using digital twins. IEEE Internet Comput. 26 (3), 52–57. http://dx.doi.org/10.1109/MIC.2021.3063674.

Panchal, A.C., Khadse, V.M., Mahalle, P.N., 2018. Security issues in iIoT: A comprehensive survey of attacks on iIoT and its countermeasures. In: 2018 IEEE Global Conference on Wireless Computing and Networking. GCWCN, pp. 124–130. http://dx.doi.org/10.1109/GCWCN.2018.8668630.

Park, K.T., Lee, J., Kim, H.-J., Noh, S.D., 2020/01/01. Digital twin-based cyber physical production system architectural framework for personalized production. Int. J. Adv. Manuf. Technol. 106 (5), 1787–1810. http://dx.doi.org/10.1007/s00170-019-04653-7.

Qassim, Q., Jamil, N., Zainal Abidin, I., Ezanee Rusli, M., Yussof, S., Ismail, R., Abdullah, F., Ja'afar, N., Che Hasan, H., Daud, M., 2017. A survey of SCADA testbed implementation approaches. Indian J. Sci. Technol. 10 (26), 1–8. http://dx.doi.org/10.17485/ijst/2017/v10i26/116775.

Rosa, L., Cruz, T., Simões, P., Monteiro, E., Lev, L., 2017. Attacking SCADA systems: A practical perspective. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management. IM, pp. 741–746. http://dx.doi.org/10.23919/INM.2017.7987369.

Singh, P., Garg, S., Kumar, V., Saquib, Z., 2015. A testbed for SCADA cyber security and intrusion detection. In: 2015 International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications. SSIC, pp. 1–6. http://dx.doi.org/10.1109/SSIC.2015.7245683.

Tabaa, M., Chouri, B., Saadaoui, S., Alami, K., 2018. Industrial communication based on modbus and node-RED. Procedia Comput. Sci. 130, 583–588. http://dx.doi.org/10.1016/j.procs.2018.04.107, URL https://www.sciencedirect.com/science/article/pii/S1877050918304691. The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops.

Tao, Y., Xu, W., Li, H., Ji, S., 2019. Experience and lessons in building an ICS security testbed. In: 2019 1st International Conference on Industrial Artificial Intelligence (IAI). pp. 1–6. http://dx.doi.org/10.1109/ICIAI.2019.8850804.

Varghese, S.A., Ghadim, A.D., Balador, A., Alimadadi, Z., Papadimitratos, P., 2022. Digital twin-based intrusion detection for industrial control systems. In: 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops). pp. 611–617. http://dx.doi.org/10.1109/PerComWorkshops53856.2022.9767492.

Wang, Y., Peng, D., Qi, E., Wang, H., 2022. Modbus/TCP data acquisition system based on industrial ethernet technology. In: 2022 4th International Conference on Electrical Engineering and Control Technologies. CEECT, pp. 746–750. http://dx.doi.org/10.1109/CEECT55960.2022.10030131.

Xie, Y., Wang, W., Wang, F., Chang, R., 2018. VTET: A virtual industrial control system testbed for cyber security research. In: 2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications. SSIC, pp. 1–7. http://dx.doi.org/10.1109/SSIC.2018.8556732.

Yamin, M., Katt, B., Gkioulos, V., 2020. Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. Comput. Secur. 88, http://dx.doi.org/10.1016/j.cose.2019.101636.

Yan, F., Tang, T., Yan, H., 2016. Scenario based STPA analysis in automated urban guided transport system. In: 2016 IEEE International Conference on Intelligent Rail Transportation (ICIRT). IEEE, pp. 425–431. http://dx.doi.org/10.1109/icirt.2016.7588764.

Young, W., Leveson, N., 2013. Systems thinking for safety and security. In: Proceedings of the 29th Annual Computer Security Applications Conference. ACSAC '13, Association for Computing Machinery, New York, NY, USA, pp. 1–8. http://dx.doi.org/10.1145/2523649.2530277.

Young, W., Leveson, N.G., 2014. An integrated approach to safety and security based on systems theory. Commun. ACM 57 (2), 31–35. http://dx.doi.org/10.1145/2556938.

Zhang, H., Zhang, G., Yan, Q., 2018. Digital twin-driven cyber-physical production system towards smart shop-floor. J. Ambient Intell. Humaniz. Comput. 10 (11), 4439–4453. http://dx.doi.org/10.1007/s12652-018-1125-4.