Noname manuscript No.
(will be inserted by the editor)

Investigating Operational Technology Attacks as Code

Franco Callegati · Saverio Giallorenzo · Andrea Melis · Simone Melloni · Marco Prandini · Alessandro Vannini

Received: date / Accepted: date

Abstract Industrial Operational Technology (OT) environments face escalating cybersecurity challenges due to increasing interconnectedness, device heterogeneity, and the integration of legacy systems not designed with modern security requirements. Operators struggle with security validation in OT settings due to the complexity of static reasoning across multilayered architectures and the impracticality of in-production testing, which risks operational disruptions and safety hazards. To address these limitations, we propose SAFARI, a framework that leverages the concepts of digital twin and cyber range to enable Security-Investigation-as-Code for OT environments, automating the creation, deployment, and security testing of faithful OT architecture replicas. SAFARI uses technologies such as Terraform, Proxmox SDN, and MITRE Caldera to provide scalable, reproducible security assessment capabilities while maintaining complete air-gapping for safe malware testing. We demon-

Franco Callegati
Alma Mater Studiorum - Università di Bologna, Bologna, Italy
E-mail: franco.callegati@unibo.it

Saverio Giallorenzo
Alma Mater Studiorum - Università di Bologna, Bologna, Italy
Olas Team, INRIA, Sophia Antipolis, France
E-mail: saverio.giallorenzo2@unibo.it

Andrea Melis
Alma Mater Studiorum - Università di Bologna, Bologna, Italy
E-mail: a.melis@unibo.it

Simone Melloni
ARPAE Emilia-Romagna, Italy
E-mail: smelloni@arpae.it

Marco Prandini
Alma Mater Studiorum - Università di Bologna, Bologna, Italy
E-mail: marco.prandini@unibo.it

Alessandro Vannini
Alma Mater Studiorum - Università di Bologna, Bologna, Italy
E-mail: alessandro.vannini13@unibo.it

strate SAFARI's effectiveness through a comprehensive case study examining three industrial network architectures exhibiting increasing segmentation. Our results show that SAFARI successfully automates complex security scenarios, enables regression testing of architectural refinements, and provides quantifiable insights into attack resistance improvements. The framework represents a significant advancement in OT security testing methodology, offering security operators a practical tool for systematic vulnerability assessment and architectural validation without compromising operational continuity.

Keywords Operational Technology · Industrial Security · Security Investigation as Code · Software-Defined Networks · Infrastructure as Code · Attack and Analysis Automation

1 Introduction

Industrial environments and the so-called Operational Technology (OT) segment are increasingly characterised by integrating interconnected devices and systems, driving efficiency and operational performance advancements [1]. However, this interconnectedness introduces significant challenges, particularly in ensuring robust and comprehensive security measures [2]. Security operators in industrial systems have to deal with the complexity of device heterogeneity, which complicates the implementation of standardised protection mechanisms, and legacy systems not designed with modern cybersecurity requirements in mind [3]. Scalability poses an additional challenge as the number of connected devices expands, further complicating the implementation of adequate security protocols [4]. Furthermore, data privacy concerns exact stringent safeguards against unauthorised access and breaches, while maintaining critical performance and reliability. Other challenges include the secure deployment of firmware updates without operational disruptions, ensuring interoperability across diverse devices and systems, and safeguarding communication channels [5]. The increasing prevalence of cyber-physical threats underscores the need for integrated security strategies encompassing the digital and physical domains.

Towards Digital Twins and Cyber Ranges for OT Leveraging tools like cybersecurity software, such as intrusion detection systems and software-defined networks, is crucial for tackling this complexity and managing threats (e.g., detection, mitigation, recovery) in these industrial settings. One can tackle the problem of validating the correct development and deployment of cybersecurity solutions in a given OT configuration through static reasoning (at the many levels that characterise an OT infrastructure, such as static code analysis for the single software components, control flow analysis for the interactions among them, etc.) on the configurations. However, the multiplicity and interplay of the layers that make up OT architectures make static reasoning complex to pursue effectively, with the task becoming even harder when comparing different versions of the same architecture, under different threat contexts. For instance, when designing a new architectural configuration to mitigate specific threat classes, security experts struggle to determine whether the proposed changes

will maintain protection against previously addressed threats while resolving the vulnerabilities under consideration. This phenomenon parallels software development, where programmers struggle to fix bugs in ways that preserve the expected behaviour of the system while amending the anomalies.

Thus, to complement static reasoning techniques, security operators also resort to testing systems for security vulnerabilities. However, while one could test modifications directly on a given production OT architecture, intervention downtimes would entail disruptions in sensitive, real-time systems, which could hinder productivity and even harm employees, depending on the industrial context — and in-production testing of security threats would further exacerbate the issue.

For these reasons, we see the issue of assessing the resistance against security threats of an OT configuration as similar to assessing the absence of certain classes of bugs from software, where testing represents one of the primary and most direct approaches to study the presence of unwanted behaviour in a given system. Hence, given the limitations of in-production testing, security operators can use simulations to study the runtime behaviour of faithful representations of the functionalities of a given (real, expected) OT architecture (including the industrial traffic it should experience and the operations it is supposed to conduct) and test its level of security against cyberthreats it might be subjected to.

These concepts of industrial simulations have gained momentum with the names of Digital Twin [6] and Cyber Range [7]. A digital twin virtually represents a physical system, enabling experts to implement real-time monitoring, analysis, and optimisations. For example, this technology facilitates predictive maintenance by simulating wear and tear, reducing unplanned downtime and maintenance costs. Cyber ranges allow testing new operational strategies in a controlled, risk-free environment, enhancing decision-making and fostering innovation. These software paradigms can identify inefficiencies, optimise resource utilisation, and improve safety by simulating hazardous scenarios and preparing contingency plans by analysing realistic operational data. Additionally, they can enhance product design and development by providing insights through simulations based on real-world operational data, ensuring better alignment with practical conditions.

Contribution In this article, we propose the application of a recent framework [8] called SAFARI (here rebaptised “Scalable Automated Framework for Air-gapped Risk Investigation”) for the OT setting. Methodologically, we propose the use of SAFARI for OT to support *Security-Investigation-as-Code*, i.e., the specification of the multilayered traits of a given OT configuration security investigation, through code-based infrastructure/operation automation artefacts. To achieve it “as-code” automation, SAFARI uses three main concepts [8]: *Infrastructure-as-Code* (IaC) for the definition of component architectures, *OS-agnostic Task Automation* (OTA) for the specification of process execution independently of the underlying operating system, and *Inspection Tools* (IT) as analysis software built to investigate a target system’s state during the life cycle of an experiment. In the implementation of SAFARI for OT, we propose using tools such as Terraform [9] and Proxmox SDN [10] for IaC and MITRE Caldera [11] for OTA and IT. Thanks to the interaction of these layers of technologies, SAFARI enables the generation of fully fledged digital twins of OT

environments. It automates the execution of comprehensive scenarios, from deploying a twin of the OT system (and variations thereof) to executing attacks that test the security of said systems. One of our proposal's key innovations is that SAFARI scales with the security expert's requirements. If experts only need a digital twin, SAFARI sets it up so they can explore security issues manually. However, if security operators want to automate security test batteries to evaluate different OT versions against these tests in a scalable manner, they can do so. This approach is similar to software engineering practices where standardised test suites verify that software (in this case, the OT architecture) does not exhibit specific problems, allowing experts to test edge cases that automation might struggle to evaluate manually.

Hence, our methodological contribution lies in the quality enhancement of the practice of testing the security of OT architectures. Without SAFARI, implementing these scenarios and tests would indeed require a lot of manual effort to make sure that tests are reliable, i.e., that the testing protocol is carried out equally for each experiment.

Furthermore, SAFARI's architecture helps testers collect metrics from test runs that may concern non-functional properties, like performance, in addition to security properties. Thanks to the reproducibility of tests that is guaranteed by automation, these measurements allow to compare the overall performance of different architectural layouts for the tested system, and to verify that new solutions do not introduce regressions while providing improvements.

To substantiate our claims and demonstrate the usage of our proposal, we present an elaborate case study on a realistic OT architecture. We model the architecture in SAFARI, test its vulnerabilities, and refine it under different network segmentation approaches to mitigate such vulnerabilities. In practice, we test the difference in terms of attack resistance of three incrementally improved industrial network OT architectures, showing how cumulative and structured network segmentation, following the latest standards, can significantly increase the resilience of an industrial plant. We repeat the security tests at each refinement, demonstrating that the subsequent refinements can progressively prevent/mitigate the attacks that would have been successful in the previous versions. We emphasise that our proposed methodology is general: the network scenarios we present are just examples showing how operators can use our contribution to study possible refinement of an existing OT architecture, because this scenario represents a timely and critical challenge in the field [12].

While several approaches have emerged in the literature for OT security testing and digital twin implementations [13, 14, 15, 16]. These solutions typically suffer from limited scope or lack of practical implementation details. Detailed comparison is deferred to Section 6, after introducing the SAFARI concept in Section 2, detailing the technological block and the software choices of the proposed framework in Section 3, describing the validation of the platform with a detailed industrial network case study in Section 4, and providing the results of tests in Section 5. Section 7 discusses final remarks.

2 Concepts and Methods

We proceed by first presenting the concepts behind SAFARI's design and then declining them as the methods that characterise the approach.

The main building blocks of SAFARI are a) *Virtual Machines* (VM) as the components of the environment that runs the elements that make up the digital twin and b) *source code* that automates the life-cycle of the digital twin, more specifically, the VMs and experiments running on the network they form. The digital twin concept is fundamental to SAFARI, allowing for precise virtualised replicas of actual operational technology systems that can be safely analysed and tested under various conditions. SAFARI also implements a *local-first* principle, prioritising on-premises deployments (backed by open-source software that eases the assembly of local testing clusters) over cloud-based solutions. This approach offers significant advantages for cybersecurity investigations in industrial settings: enhanced data sovereignty, reduced latency, protection from internet-based threats, assured compliance with strict regulatory requirements, and continued operation even during Internet outages. While SAFARI can leverage cloud resources when needed, keeping critical testing environments on local infrastructure provide maximum control and security, particularly crucial when working with sensitive industrial control systems. SAFARI is *flexible* because it allows for the orchestration of many physical machines to host the VMs necessary for the experiments. To reproduce the composition of network partitions that characterise OT systems, we equip SAFARI with technologies able to automate the creation of *zoning* environments, useful both to isolate the experiments from one another (to avoid interference) and to analyse the possible propagation of malware infra- and intra-zone. Given this principle of isolation, we avoid using containers as a lightweight alternative to VMs because they cannot guarantee the same level of isolation as VMs [17].

SAFARI implements a comprehensive cyber range approach, functioning as a virtualised (training and) testing environment, where security professionals can safely simulate attacks against digital twins of critical systems. The framework integrates complementary technologies to define experiments and collect and analyse attack data. Concretely, it includes Infrastructure-as-Code solutions such as Terraform [9] and Software-Defined Networking [10] to abstract away the underlying infrastructure (which can consist of physical and/or cloud-based resources). It also uses Operating-System-agnostic Task Automation tools such as Caldera [11] to configure and monitor components, inject exploits, and execute operations. Caldera also offers a platform for security investigation, which allows us one to obtain practical measures for qualitative (the effectiveness of an attack) and quantitative (the number of compromised nodes and the entity of the attack) evaluations. As a cyber range, SAFARI provides a high level of *automation*, thanks to the usage and integration of code-based tools to manage the infrastructure and to operationalise attacks and monitoring. This high level of automation allows investigators to streamline the conduction of parameterised experiments in different contexts within the digital twin environment. Depending on the availability of the infrastructural resources, SAFARI can test batteries in parallel and aggregate their results, further speeding up the analysis process.

2.1 Methods

Focusing the discussion on the methods that enable operations in SAFARI, we comment on the practical aspects of the framework and provide an overview of where the introduced components come into play in SAFARI's general functioning.

Infrastructure as Code (IaC) is a paradigm where an orchestrator manages the provisioning of infrastructure components, like computing, network, and storage devices, using code rather than manual configuration. Within our cyber range, IaC ensures that digital twins can be consistently deployed and reconfigured according to experimental needs. With IaC, infrastructure automation, setup, and management become consistent and repeatable, e.g., allowing blue and purple teams [18] to define, deploy, and update virtual hardware components using scripts or configuration files, exploiting the same processes and tools used to keep complex software projects manageable, e.g., versioning, access control, and testing/production branches. By using a virtual version of a computing environment — such as a server or network device — one can make it operate independently of the underlying physical hardware [19]. Overall, IaC and virtualisation support the partition and optimised usage of testing hardware resources, enhancing flexibility, and reducing operational costs. IaC contributes to SAFARI in two ways: formally documenting the hardware and OS specifics of the digital twin experiments, and automating the life-cycle of VMs used for experiments and analyses.

OS-agnostic Task Automation (OTA) offers a uniform interface that allows users to execute processes independently of the underlying operating system. This capability is essential for interacting with heterogeneous digital twin components within the cyber range. Indeed, each operating system (e.g., Windows, macOS, Linux) provides its own set of APIs to interact with the resources it manages and to execute user tasks. This fragmentation leads to high complexity when automating tasks across different environments. By standardising interactions, OTA eliminates the need to navigate the disparate APIs of each system, simplifying task execution and improving cross-platform compatibility.

Inspection Tools (IT) is analysis software built to investigate a target system's state during an experiment's life cycle. These tools examine changes within the experiment system, such as altered configurations, unexpected process behaviours, modified access controls, and other anomalies indicative of security risks or unauthorised modifications. In the context of our cyber range and digital twin approach, these tools are crucial for monitoring the system state and identifying security events. These tools aim to identify the effects of potential security events, helping to assess the extent of system impact and, where applicable, evaluate the effectiveness of protective countermeasures in minimising disruptions.

3 A SAFARI Specialisation for Operational Technology

As mentioned, our prototype reifies IaC with Terraform [9] for the Management of VM infrastructures.

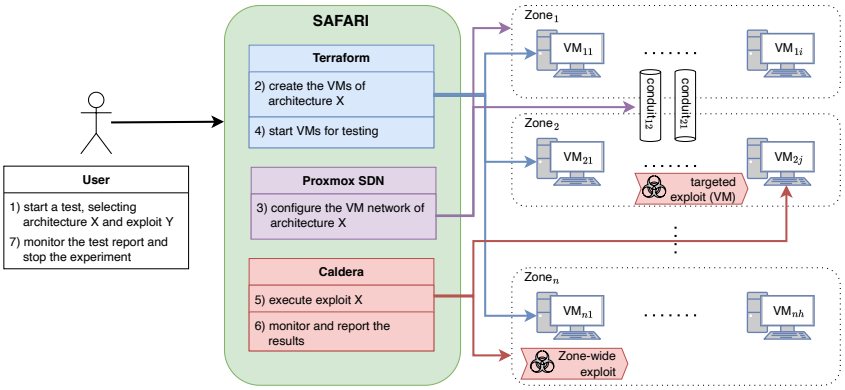

Fig. 1 SAFARI's prototype architecture and functionalities

SAFARI's design abstracts away which virtualisation technology it uses to manage the provisioning of VM through IaC. In this paper, following SAFARI's local-first principle, we choose to use Proxmox [20], which is an open-source, on-premises virtualisation platform lets users run and manage virtual machines. Since Proxmox also provides a Software Defined Network add-on [10], we use that component to complete the "as-code" networking part of the infrastructural aspects of digital twins. Since our application context involves testing the security of the OT segment, we use MITRE Caldera [11] for both the OTA and IT tasks. Caldera is an open-source adversary emulation platform designed to automate security assessments. The platform allows users to simulate different domain attack techniques based on the MITRE ATT&CK framework [21], a comprehensive model of adversary behaviour. Therefore, using Caldera, allows SAFARI's users to define different kinds of attacks on a given infrastructure to test its security posture. Regarding the investigation tools part, Caldera provides a monitoring routine that allows the user to observe the status of the experiment (partitioned into its constituting components) and both obtain live and offline feedback on the results of the experiments.

3.1 Software Architecture

The main components of the architecture are shown in Fig. 1. In the central box, we find the core components of the SAFARI prototype, which orchestrates the operations required to carry out the tests: Terraform, Proxmox SDN, and Caldera. The right-hand side depicts an example test run employing the VMs that run the experiments (VM_{11}, ..., VM_{nh}); these are ephemeral: created, used to run a test session (possibly multiple times and in parallel to gather statistical experimental data), and discarded at its end.

The numbered operations shown in Fig. 1 illustrate the workflow that actuates this process.

At the start of an experiment defined by the user (1), control passes to the IaC domain, specifically, Terraform and ProxMox SDN, that programmatically define the

characteristics of VMs and of the zone(s) each one VM belongs to. The system interacts with the hypervisor deploy the VMs (2), to create conduits that allow the components found in different zones to communicate (3), and to start the VMs (4). After the above steps, control passes to OTA (Caldera), which executes the specified exploit of the experiment (5) and monitors the evolution of the system (6) to provide both live and archival reporting. When running an experiment, the user monitors the system's evolution and the reports produced by Caldera (7).

3.2 Terraform and Proxmox SDN as IaC

Terraform's main components are providers and resources. A provider is a plugin that communicates with a specific virtual infrastructure manager on-premises or in a cloud service provider. In this way, the same Terraform script can work with different virtualisation technologies. Terraform typically supplies one or more providers for a specific technology. We use Telmate [22], which is one of the most widely-used providers for Proxmox. A resource is a component that needs to be instantiated, e.g., a VM, a container, a database. To specify the creation of resources, one needs to typically define two kinds of ".tf" files: one that specifies which provider to use and how to access it, and one or more that describe "resource" blocks. In our prototype, one file of the latter kind defines blocks describing the VMs, with their related OS, disk, and memory requirements. As an example, in Listing 1, we report a snippet of a resource Terraform file. In particular, we point out the indication of a virtual machine template, `windowsVM`, the targeting of a specific node, `pve1`, the assignment of a distinct identifier to the VM, `101`, and the creation of its dedicated disk `local-lvm`.

Listing 1 Terraform script example.

```
resource 'proxmox_vm_qemu' 'windowsVM' {
 name = 'TestVM', target_node = 'pve1', vmid = 101,
 memory = 4096, cores = 4, sockets = 1
 disks { ide { ide0 {
    disk { storage = 'local-lvm', size = '40G'
 }}}}
 network { model = 'e1000', bridge = 'vmbr0' }
}
```

Focusing on the networking deployment, Proxmox's main components for managing software-defined networks are virtual networks, zones, and controllers. A virtual network in Proxmox SDN is a configurable overlay network that can span across different Proxmox nodes (in our case, the physical machines that make up the experiment's cluster), allowing for the specification of isolated and interconnected environments, depending on the configuration. Zones define access and boundaries for these virtual networks, segmenting resources as needed within a virtualised environment.

In Proxmox, the user defines these SDN components in configuration files within the Proxmox interface. Each SDN zone, for instance, can specify settings like the network's VLAN, IP range, and connection to physical interfaces or virtual switches. In our prototype, we configure SDN zones to handle both isolated networks (zones)

and shared ones (connected via conduits), which facilitates secure and manageable network partitioning for experiments.

We show an example Proxmox SDN configuration file in Listing 2, where we define a virtual network with (VLAN) ID "testVLAN", assign it to the node "pve1", and set up an IP range for connected VMs. This configuration creates a virtual network isolated from others, ensuring controlled communication within "testVLAN" zone.

Listing 2 Proxmox SDN configuration example.

```
# Define the SDN zone

zone:
  vlan_zone:
    type: vlan
    vlan-raw-device: enp0s31f6

#  Define VLAN networks within the zone
vnet:
  vlan100:
    zone: vlan_zone
    vlan-id: 100
    bridge: vmbr0

  vlan200:
    zone: vlan_zone
    vlan-id: 200
    bridge: vmbr0
```

In our prototype, such SDN configurations enable the flexible deployment of components in networks, allowing the necessary malleability to both capture existing OT infrastructure and quickly explore what-if/experimental scenarios.

3.3 Caldera as OTA and IT

Using Caldera's API and YAML-based configuration files, one can configure "operations" that run specified adversary tactics on target VMs/zones. These operations implement different OT-specific threats, allowing the user to observe and analyse responses, gather data on system states, and evaluate the effectiveness of infrastructure configuration.

The architecture of Caldera includes a server, which manages agents deployed on target hosts (VMs, in our case), and the agents, which execute "abilities", i.e., predefined actions mapped to specific attack tactics. This modular design provides flexibility in scripting complex attack scenarios and chaining abilities into comprehensive adversary simulations.

As an example, in Listing 3, we show the configuration of a Caldera ability by writing a (simplified) script that simulates a file modification attack — specifically, we emulate an attacker altering a (critical) configuration file, potentially disrupting operations.

Listing 3 Example of a Caldera utility.

```
id: XXXX-XXXX-XXXX
name: Modify Configuration File
description: Emulates modification of a configuration file on the
    target VM.
tactic: impact
technique:
 name: Data Manipulation
platforms:
 linux:
  sh:
   command: echo 'Modified configuration' >> /etc/critical.conf
   cleanup: sed -i '/Modified configuration/d' /etc/critical.conf
```

The above script defines an ability (with a unique ID, a description, and the MITRE ATT&CK attack technique it emulates) that appends a line to the file '/etc/-critical.conf', while the 'cleanup' step reverts this change after the test.

To monitor target VMs, one can configure Caldera to periodically query system parameters, such as open network ports, running processes, or file integrity. Users can configure also this behaviour via an ability script. For example, we report in Listing 4 an ability (simplified) that checks for changes to the target configuration file, given its hash.

Listing 4 Example of a Caldera ability.

```
id: YYYY-YYYY-YYYY
name: Detect Configuration File Modification
description: Monitors for unauthorized modifications to /etc/critical.
    conf
tactic: defense-evasion
technique:
 name: Indicator Removal on Host
platforms:
 linux:
  sh:
   command: |
    original_hash='...'
    current_hash=$(md5sum /etc/critical.conf | awk '{ print $1 }')
    if [ '$current_hash' != '$original_hash' ]; then
     echo 'Configuration file has been modified'
    else
     echo 'No modification detected'
    fi
   parser:
    - source: stdout
      edge: 'host.config_change_detected'
      regex: 'Configuration file has been modified'
   cleanup: 'echo ''Successful integrity check'''
```

The ability in Listing 4 can run periodically within a Caldera operation to continuously monitor the configuration file's integrity. If any unauthorised changes occur, Caldera logs and notifies the operator about the modification event, allowing further investigation or corrective actions. Caldera's "parser" option allows triggering

specific alerts or actions based on outputs, providing visibility on network integrity under simulated attack conditions.

Given a set of abilities, one can configure Caldera operations to automate and streamline both the execution of attack simulations and the collection of corresponding telemetry. For example, we can combine the abilities above to run an operation that deploys the former on a target VM and captures relevant output data, as in the snippet in Listing 5.

Listing 5 Example of Caldera operation.

```
name: File integrity attack
adversary_id: WWWW-WWWW-WWWW
phases:
 1:
  - id: XXXX-XXXX-XXXX
 2:
  - id: YYYY-YYYY-YYYY
  - repeat: true
  - sleep: 1
```

As the examples illustrate, Caldera breaks operations down into "phases", where phases run specific abilities on the target host. In the examples, we have two phases. In phase 1, we execute the attack on the configuration file while, in phase 2, we run the integrity check continuously (every 1 second) to detect if the file has been changed.

This multiphase approach allows users to run sequences of actions, simulating complex attacks and measuring system resilience.

4 Case Study: Refining an OT Infrastructure to Enhance its Security Posture

We substantiate our claims and demonstrate SAFARI's usage in the OT context by presenting an elaborate case study on a realistic OT architecture. We model the architecture in SAFARI, test its vulnerabilities, and refine it under different network segmentation approaches to mitigate such vulnerabilities. In practice, we test how three incremental industrial network OT architectures differ in attack resistance, showing how cumulative and structured network segmentation, following the latest standards, can significantly improve an industrial plant's resilience. We repeat the security tests at each refinement, demonstrating that subsequent refinements can progressively prevent or mitigate attacks that proved successful in previous versions. We emphasise that the network segmentation we present provides examples and that our proposed methodology remains general, not tied to any specific segmentation approach or OT architecture.

Concretely, the implementation relies on creating a fully virtualised digital twin of an industrial process, enabling repeatable, automated, and realistic cybersecurity testing. This virtual environment would allow, e.g., researchers and operators of the twin counterpart, to reproduce common industrial scenarios, apply various architectural refinements, and assess their impact on cyber resilience. To evaluate the effect of network segmentation, we selected three representative segmentation scenarios: one with no segmentation, one with a modern micro-segmented architecture aligned with

current industrial best practices, and one with an intermediate level of segmentation. This choice reflects the diverse maturity levels found across industrial environments, as highlighted in the cybersecurity landscape by Schwab and Poujol [23] and further discussed in the context of OT-specific threats by Setola et al. [24].

In particular, micro-segmented scenarios are the most interesting evolution, with recent work demonstrating their effectiveness in limiting lateral movement and improving network visibility in Industrial IoT (IIoT) systems [25, 26]. Micro-segmentation is increasingly recognized as a key enabler of Zero Trust architectures [27], which are progressively adopted across industrial sectors. In contrast, the non-segmented scenario models legacy deployments common in operational networks, where flat architectures lead to increased attack surfaces and detection latency [28, 29]. The intermediate segmentation setup represents transitional environments, where partial security zoning has been implemented (e.g., by subnet or function), but micro-segmentation is not yet fully deployed [30].

By comparing these three setups, we aim to assess how the maturity of network segmentation influences the responsiveness and effectiveness of orchestrated cybersecurity mechanisms.

From these premises, we define the deployment, virtualisation strategy, and security configurations that support the architectural variants. We start by describing the infrastructure used to deploy the case study. Subsequently, we outline the process of virtualising the OT systems that compose the industrial case, followed by the definition and implementation of the three incrementally segmented OT architectures. We give particular attention to the segmented network architecture based on the IEC/ISA 62443 Zones and Conduits model, including the additional definition of realistic user roles for accessing the various OT systems.

Before delving into the details of the case study, we clarify that SAFARI is a tool supporting automated security testing for OT, and the case study we present are instances of such tests included to demonstrate potential uses of SAFARI. Our results do not aim to prove that a given architecture (e.g., the flat one, the one built on the NIST model, etc.) are immune to specific attacks, but rather that SAFARI enables users to implement such verifications quickly and in an automated manner. Indeed, one of the observations we underline in the description of the case study is how easily an operator can modify a pre-existing architecture to refine it into a new one, e.g., when moving from the flat one to the one based on the NIST model, and then repeat the execution of security tests to identify potential security vulnerabilities that the previous might or might not have. For the sake of reproducibility, the full set of scripts and attack code used in our case study is publicly accessible on Zenodo at `https://doi.org/10.5281/zenodo.15593943`.

4.1 Deployment Infrastructure

Besides showcasing the general usage of SAFARI, the case study allows us to also illustrate an on-premises deployment of the framework. Specifically, we assemble the infrastructure shown in Fig. 2 as a companion contribution to SAFARI's definition, designed to run experiments conducted remotely by a group of users safely.

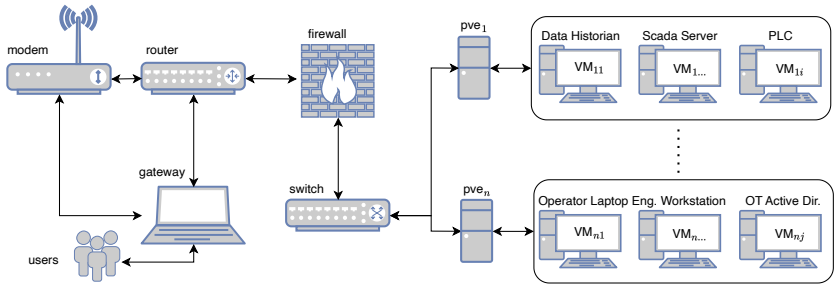

Fig. 2 Schema of Deployment Infrastructure with OT VM components.

Following Fig. 2, users can authenticate and execute the prototype's functionalities through a gateway connected to the Internet. From the gateway, users can interact with the nodes, named $\text{pve}_1, \ldots, \text{pve}_n$ (where PVE stands for Proxmox Virtualisation Environment), that make up the cluster and host the test and analysis VMs. All these nodes and VMs are behind a firewall regulating Internet access through the router and modem.

Below, we detail the main features of the involved components.

The gateway is a machine running Debian v.12 and represents the only network access point that experimenters use to connect to the nodes. Users connect to the gateway via the ZeroTier [31] VPN and use sshuttle [32] to access the nodes directly, e.g., to run Terraform commands as if executed on machines in their local network.

The router is a GL-MT300N-V2 equipped with a MTK 580Mhz CPU, 128MB RAM, and 16MB Flash ROM memory, mounting OpenWrt [33], an open-source Linux distribution designed for network devices that offer advanced functionalities like network filters. The router can give Internet connection to the nodes in the cluster, but by default, its firewall prevents VMs from accessing the Internet to avoid the possible propagation of ransomware.

The hardware of the cluster of pve nodes encompasses a cluster of eight machines, each equipped with Intel i3-4170 (3.70GHz) dual-core, four-thread CPU, 12GB of RAM, and 500GB HDDs. The nodes run Proxmox version 7.0-8.

4.2 Virtualisation of OT systems

An important step during the implementation of our OT case study is the research and definition of virtualised OT Components used in the virtual network infrastructures. In particular, simulating "lower level" systems, the ones mainly involved in the production process, poses the most interesting challenge since open-source simulated Programmable Logic Controllers (PLC) and sensors/actuators are not common. We obtain the virtualisation of systems simulating an industrial process by taking inspiration from the "Graphical Realism Framework for Industrial Control Simulation (GRFICS)" open-source project [34]. Hence, we virtualise the industrial process through the VMs:

- Process Simulation: an Ubuntu VM that runs multiple Python scripts, emulating the behaviour of a cluster of sensors and actuators, and exposes the generated data on multiple network interfaces to be read by the PLC;
- PLC Server: an Ubuntu Server VM that emulates an Open PLC Soft PLC[1]. The PLC configuration dashboard is accessible via localhost port 8080, and, once correctly configured, the PLC retrieves and sends data through the "Process Simulation" VM network interfaces;
- Operator Laptop: a VM running Ubuntu (chosen over Windows for performance and storage reasons), simulating the Laptop of an OT Operator/Engineer, which can connect to the PLC via its local-host port and can consequently configure the PLC program and Modbus configuration through the exposed dashboard;
- Human Machine Interface (HMI)/Local SCADA: A VM running Windows 10 (similar to the majority of Local SCADA/HMI systems), with an open-source HMI/Local SCADA software, ScadaBR[2]. We configure the ScadaBR instance to read data from the PLC and show the overall process status through the HMI dashboard.

The virtualisation of the remaining systems require choosing a suitable operating system and the installation of needed software, together with minor tweaks. Specifically:

- The Remote Access Jump Host VM runs Ubuntu and is configured to access through a remote control solution, Anydesk, the HMI VMs, thus simulating the maintenance process done by third-party operators;
- Data Historians are Ubuntu VMs set up to be able to access data from the local SCADA servers;
- The Inventory Management System (IMS) is a VM running a demo of an open-source IMS software;
- Manufacturing Execution System (MES) is a VM running a demo of an open-source MES software;
- Engineering Workstations are Windows 10 VMs configured with Open PLC software, ideally letting OT engineers write code for the PLCs.

As per SAFARI's definition, we perform the provisioning of virtual machines on the Proxmox platform using Terraform. Since one of the purposes of the case study is testing the effectiveness and propagation of cyberattacks, the whole virtualised infrastructure is built to be easily recovered and restored between tests. Thereby, we save safe-state backups of all VMs to quickly restore the systems after a test. In addition, the Proxmox IP Address Management System (IPAM) automatically reassigns, through DHCP, the same IP address to restored VMs, avoiding possible disruptions of the virtualised industrial processes.

[1] https://autonomylogic.com
[2] https://www.scadabr.com.br

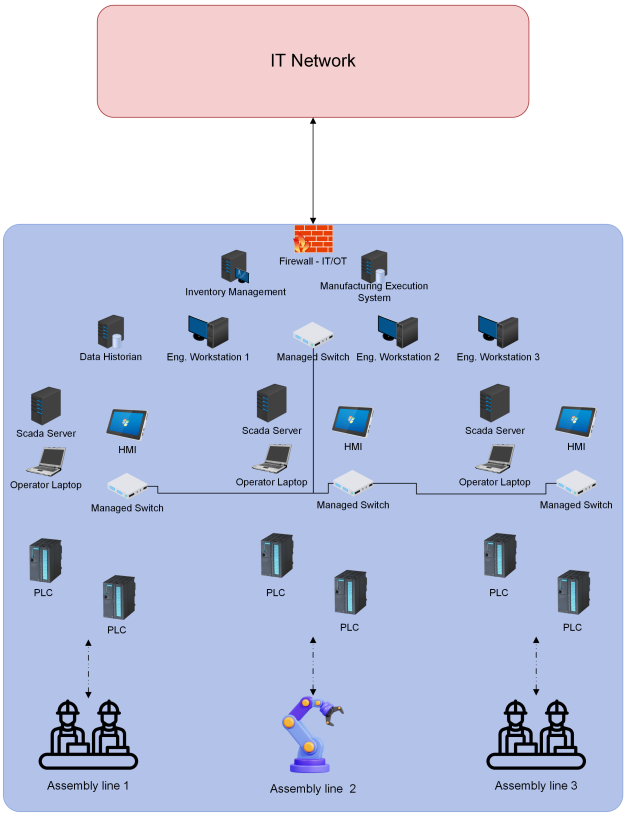

Fig. 3 Case Study: Flat OT Network variant.

4.3 A Basic OT Network Infrastructure

Having defined the case study OT system, developing a secure OT network infrastructure begins with examining the most common infrastructure used in numerous industrial facilities [35]. This network structure features a basic segmentation between the IT network and the plant's OT network, protected by perimeter firewall(s). Therefore, the OT network lacks any internal segmentation, often placing production systems with varying functions and security requirements on the same VLAN. Fig. 3 illustrates such a network architecture.

We implement this version of the study architecture using Proxmox and the associated Proxmox SDN plugin functionalities. Practically, we define a single Proxmox SDN Zone to host all OT systems. In particular, Proxmox SDN allows the definition of five zone typologies: Simple, VLAN, QinQ, VXLAN, and EVPN. We consider the Simple zone – an "Isolated Bridge. A simple layer 3 routing bridge (NAT)", quoting Proxmox's documentation – suitable for the case study since it provides the essential network isolation and routing capabilities required for our OT architecture modelling without introducing unnecessary complexity. Briefly, VLAN and QinQ require tag-

ging configuration and switch management that would complicate the network topology without adding security assessment value; VXLAN implements overlay networking that would obscure the underlying network security properties we aim to evaluate; EVPN requires route advertisements and distributed control plane coordination that would introduce routing complexities irrelevant to our security testing objectives. The Simple zone's straightforward implementation allows us to focus on the fundamental security aspects of network segmentation whilst maintaining the realistic network behaviour essential for our vulnerability testing scenarios.

The main process for setting up an SDN Simple zone requires the definition of:

1. the zone itself, specifying which PVE nodes of the cluster it covers;
2. VNets, which act as simplified version of network VLANs, and are associated to a specific zone. VMs network interfaces are assigned to a VNet;
3. Subnets for a VNet. Subnets are where a proper IP subnet, gateway, and range of IP addresses are associated with a zone. Simple zones are, at the moment, the only type of SDN zones that support automatic DHCP assignment of IP address from the defined range to VMs attached to the zone.

4.4 Horizontal Refinement of the OT Infrastructure

We can make the proposed flat architecture above more secure through enhanced network segmentation. The variant we first introduce involves horizontal segmentation of the network, following the principles outlined in the NIST Reference Architecture for Smart Manufacturing [36]. Horizontal segmentation refers to the segregation of OT systems within the network based on a hierarchy related to their functionality and the locality level of the managed process. We implement this segmentation through the definition of three zones, as shown in Fig. 4.

4.4.1 Zone 1: Industrial Demilitarised Zone (IDMZ).

The Industrial Demilitarised Zone (IDMZ) is a critical architectural component in secure industrial network design, serving as a buffer zone between operational technology (OT) and information technology (IT) networks. Its primary purpose is to mitigate cybersecurity risks by controlling and monitoring the data flow between these two distinct environments. IDMZs protects critical industrial systems from external threats from IT networks or the broader Internet while allowing necessary data exchanges.

In our case study, the systems included in this zone are:

- A replication of a Data Historian, used to retrieve important process data generated from the OT compartment by the IT compartment;
- A dedicated OT Active Directory Server instance, used for easier and secure Users and Roles definition and assignment;
- A MES for real-time monitoring and control within the facility;
- An IMS for tracking and controlling the inventory of goods and resources within the production environment.

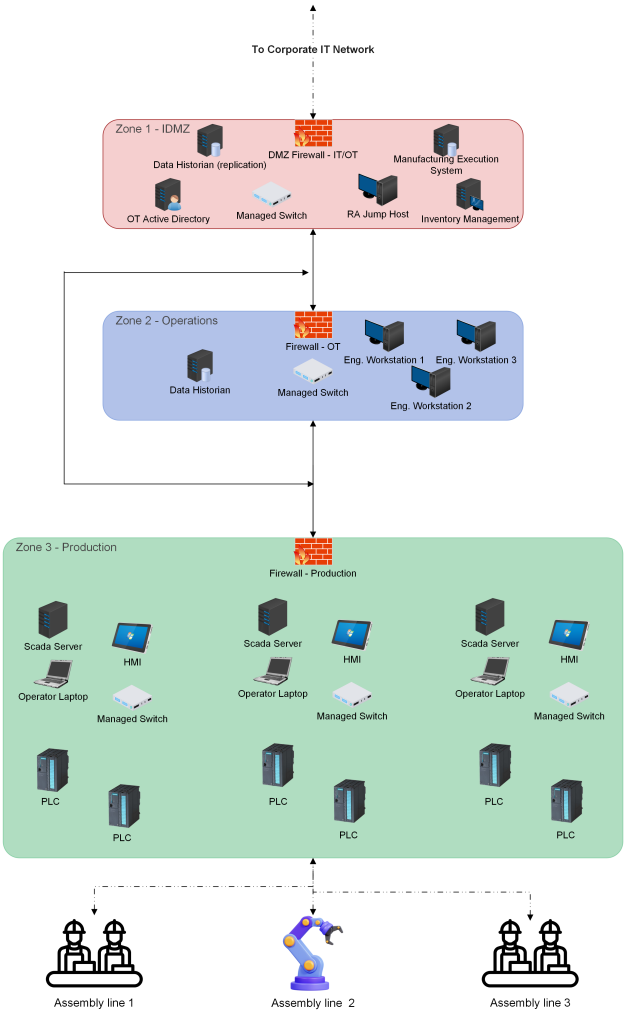

Fig. 4 Case Study: NIST-inspired OT Network variant.

Note that, ideally, the MES and IMS systems in this zone are replicated systems, only used to retrieve data between IT and OT compartments, with their main instances located in the IT compartment. However, since the latter element is not modelled in this case study, these systems have been directly installed in the IDMZ zone for the sake of simplicity.

4.4.2 Zone 2: Operations

The Operations zone regards the high-level supervision and management of production activities. In this zone, we find the the original Data Historian Server, which

collects production data directly from the local SCADA system located in the Production zones. We also find three main Engineering Workstations, managing the PLC software parameters and allowing for direct control of the production processes.

4.4.3 Zone 3: Production Zone

Finally, we find a single Production zone that encapsulates the systems related to production processes, including:

- The PLCs that manage the input/output data directed to the production line sensors and actuators;
- An HMI, which represents an interactive dashboard where an operator can have complete visibility and control of the production information (e.g., temperature and pressure data);
- A Local SCADA Client, which acquires production data from the PLCs and shares them with the SCADA Server (Data Historian), allowing for data analysis and monitoring;
- A Laptop, used by operators to load new programs into the PLCs.

To implement these zones in Proxmox, we define three Simple Proxmox SDN zones, using the same steps mentioned above. Each zone has a dedicated subnet for the hosted systems. We define each of the three implemented zones as a separate Proxmox PVE node, thus allowing proper logical separation.

4.5 Multidimensional Refinement Of The OT Infrastructure: Mixing The NIST Model With IEC/ISA 62443

We follow the guidelines defined by the IEC/ISA 62443 standard – the horizontal standard addressing Industrial Security topics – to implement a further refinement of the OT network infrastructure. Quoting from IEC/ISA 62443's documentation, the standard advocates for implementing a secure OT network infrastructure through the segmentation into various "zones" defined as "the grouping of cyber assets that share the same cybersecurity requirements". These "zones" should then only communicate with each other through "conduits", which are "the grouping of cyber assets dedicated exclusively to communications, and which share the same cybersecurity requirements".

Following these principles, we modified the NIST-model OT infrastructure presented above to achieve an OT network segmented both horizontally (following the NIST Model) and vertically (using the concepts of zones and conduits as defined by IEC/ISA 62443). This two-dimensional segmentation approach shall enhance the security posture of the OT network by isolating different operational levels and ensuring that communication between segments is strictly controlled and aligned with specific cybersecurity requirements.

We show the definition of this version of the OT Network Infrastructure Fig. 5, composed of five zones and four conduits that enable intra-zone communication.

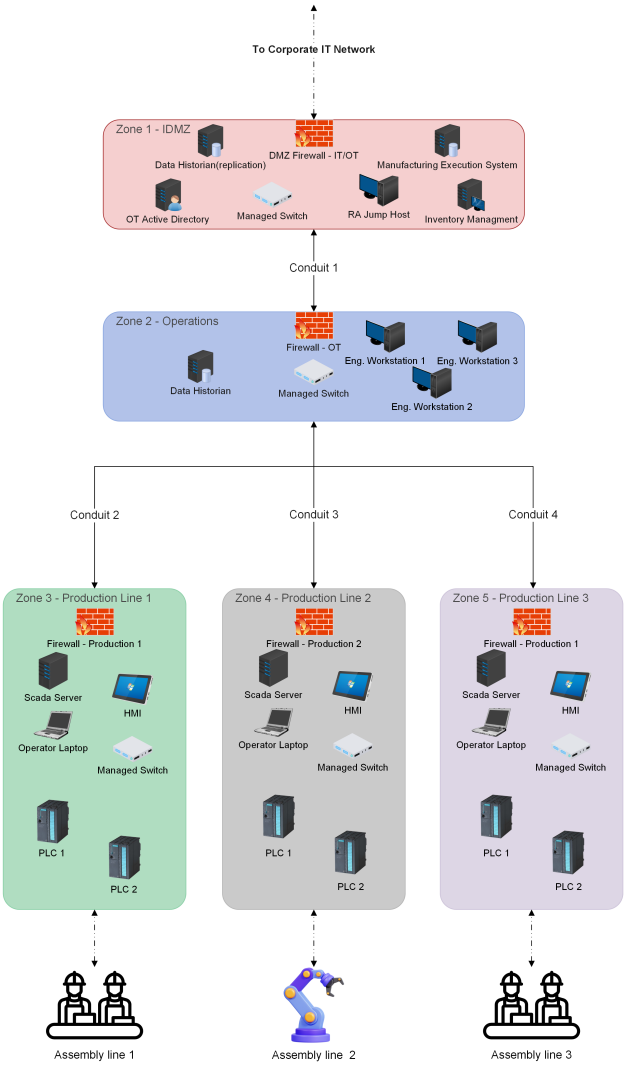

Fig. 5 Case Study: OT Network Architecture segmented following IEC/ISA 62443 (zones and conduits).

In the Figure, we find two zones equivalent to the ones proposed in the NIST Model variant, since the security requirements of the systems, in these cases, do not require a vertical segmentation. Instead, the Production zone found in the NIST-inspired variant, which encapsulated systems related to different production processes, is now segmented horizontally into three zones, each associated with a single Production line and each with specific security requirements. Practically, each of the five implemented zones is a separate Proxmox PVE node, thus allowing proper

Table 1 Mapping of IEC/ISA 62443 Zones to Proxmox SDN configurations.

IEC/ISA 62443 Zone	SDN Zone	Proxmox PVE	Vnet	Subnet	IP Address Range
Zone 1: IDMZ	Zone1	PVE 2	Vnet11	192.168.10.1/24	192.168.10.10-20
Zone 2: Operations	Zone2	PVE 5	Vnet21	192.168.20.1/24	192.168.20.10-20
Zone 3: Production 1	Zone3	PVE 6	Vnet31	192.168.30.1/24, 192.168.35.1/24	192.168.30.10-20, 192.168.35.10-20
Zone 4: Production 2	Zone4	PVE 7	Vnet41	192.168.40.1/24, 192.168.45.1/24	192.168.40.10-20, 192.168.45.10-20
Zone 5: Production 3	Zone5	PVE 8	Vnet51	192.168.50.1/24, 192.168.55.1/24	192.168.50.10-20, 192.168.55.10-20

logical separation. For reference, we report in Table 1 the complete zone mapping scheme.

Theoretically, all systems included in a zone can only share data and have visibility within the systems in that same zone. Hence, the information that needs to flow between systems in different zones must pass through a zone perimeter firewall, which serves as an entry point for the conduit that connects two Zones. Afterward, the information must pass through the destination zone's firewall to exit the conduit and reach the desired systems. Due to the definition of IEC/ISA 62443 zones and Proxmox SDN zones, all VMs inside a zone can communicate without restriction exclusively with other VMs in the same zone.

Thanks to this specific implementation, the fully virtualised industrial process can function normally, since all the systems participating are inside the same zone. It is important to notice that VMs in a Production zone cannot have visibility on VMs in another Production zone, making all three virtualised industrial processes completely separated and segregated, in line with the standard's guidelines. For the same reason, zone 1 and zone 2 are also segregated from each other and from all the Production zones. Nevertheless, as mentioned above, different zones might need to exchange data, which can happen only through IEC/ISA 62443 conduits. In this implementation, the entry points of a conduit are the Proxmox PVE nodes themselves (since each node represents a zone); same for the exit point. We configure the routing tables of these nodes to manage routing between adjacent zones that need to exchange data through a conduit. For this reason, zone 1 (PVE 2) can forward packets to zone 2 (PVE 5). Still, zones 3/4/5 (resp. PVE 6/7/8) can forward packets only to zone 2, and zone 2 is the only one, due to its location in the network infrastructure, configured to allow forwarding of packets to all the other zones.

Routing rules, though, are insufficient to allow a secure implementation of zones and conduits, since they give complete visibility between systems included in two communicating zones. For this reason, firewall rules come as IP tables for each Proxmox PVE Node containing a zone, allowing data exchange only between machines that need to communicate with each other. Specifically, we define five main data streams that run through conduits:

1. one that connects the Data Historian Replica (zone 1) with the main Data Historian (zone 2), and the main Data Historian with all three local SCADA VMs (zone 3/4/5), allowing for sharing of process data;
2. one that allows the Remote Access Jump Host (zone 1) to remotely connect to all three HMI VMs (zone 3/4/5). We realise this stream through the forwarding of packets from zone 1 to 2 and from zone 2 to 3/4/5;
3. three that connect a specific Engineering Workstation (zone 2) with a relative Laptop Operator (zone 3/4/5), allowing the sharing of PLC software for upload.

We represent these main data streams, together with VM IP addresses, in Fig. 6.

Regarding all the zone firewalls, rather than deploying an Open Source Firewall (i.e., PfSense) for each zone, we directly define firewall and routing rules into the Proxmox PVEs that accommodate the different Proxmox SDN zones.

4.6 User Roles and Permissions in the OT Case Study

Since we want to recreate and virtualise a real-life industrial scenario (its digital twin), we give attention to authentication and authorization methods and processes. In a secured real industrial plant, ideally, roles and permissions are not managed locally for each machine but, e.g., in a Windows-based environment, an Active Directory Server dedicated to OT roles (hence, separated from the IT one) manages roles and accesses. Since we do not have a Windows-based system, we do not include an OT Active Directory Server for managing user roles. Instead, we leverage the built-in access management capabilities of Proxmox to define roles and permissions that emulate our realistic industrial scenario. Specifically:

- OT Engineer: role dedicated to industrial engineers that needs to write PLC software and upload it into process systems. Each engineer has access only to his Engineering Workstation, located in zone 2, and to the relative Operator Laptop, PLC, and Process Simulation system located in one of the Production zones.
- HMI Operator: role dedicated to operators that manage the HMI of production line. Each HMI operator can only access to the HMI located in a specific Production zone;
- Maintenance: role dedicated to third party technicians that need to carry out maintenance operations on HMI systems. They only have access to the Remote Access Jump Host located in the IDMZ zone;
- Plant Production Supervisor: role dedicated to the supervisor of the production process, with access to all systems that contains production data, thus meaning the Data Historian (zone 2) and his replica (zone 1), the MES and IMS (zone 1), and all local SCADA systems in the Production Zones;
- IT Data Analyst: role that represents the most common type of plant data needed by corporate IT, production data. This role can access through the IDMZ the MES system and the Data Historian replica, but cannot monitor other types of OT data.

To clarify the permissions/actions of roles discussed above, we visualise them in the scheme in Fig. 7.

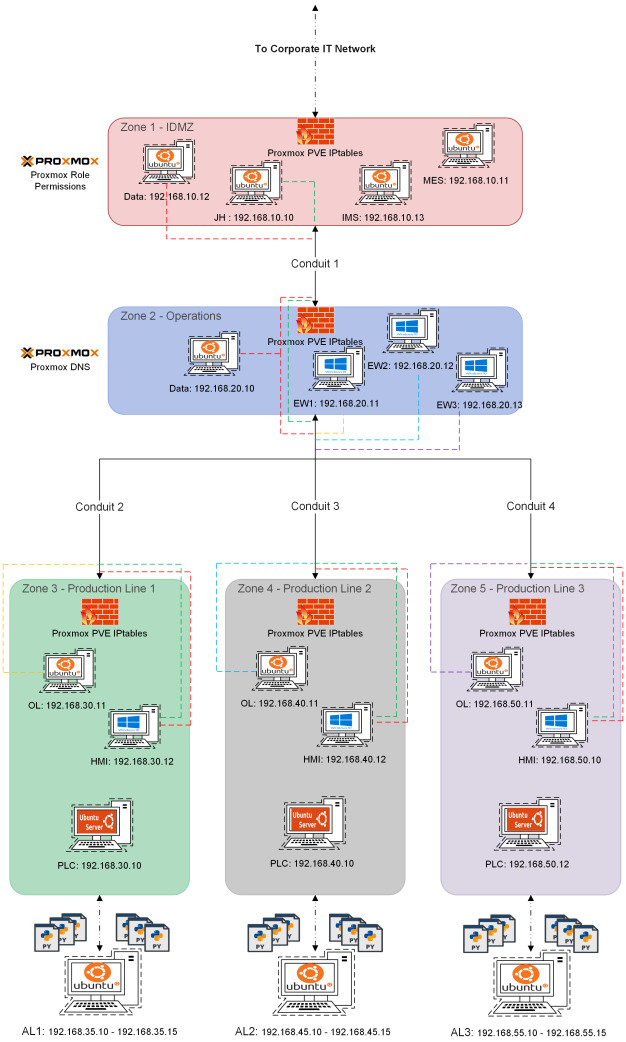

Fig. 6 Case study: zones and conduits segmentation with allowed extra-zone communications.

4.7 Summary of the Case Study

To summarize, we define three, incremental variants of OT network architectures to test segmentation effectiveness against cyberattacks in SAFARI for OT.

We start with the flat environment, which simulates an entirely unsegmented network where we apply no zones or other segmentation measures. Then, we progress to the NIST Model implementation, which follows the loose horizontal segmentation that the NIST Model hierarchy proposes, resulting in three zones where we divide VMs by functionality rather than security requirements. We maintain an IDMZ Zone

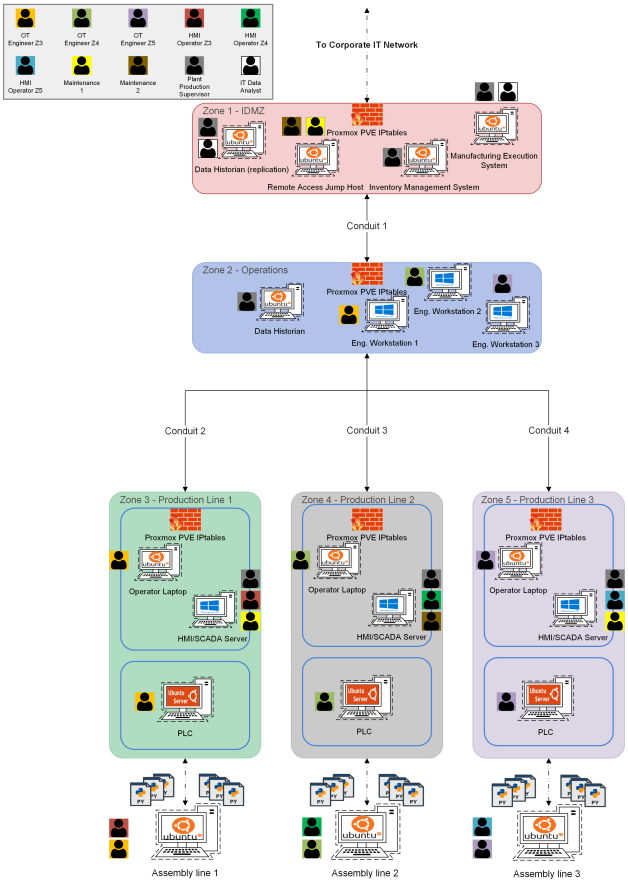

Fig. 7 Mapping of OT User Roles to the Infrastructure

and an Operations zone, while we define a single Production zone containing all production management VMs, since the NIST Model architecture does not propose vertical segmentation. Finally, we implement the virtualised OT network infrastructure following the zones and conduits paradigm that the IEC/ISA 62443 standard proposes, where we divide VMs according to security requirements rather than merely functionality, incorporating both horizontal and vertical segmentation principles.

5 Analysing the Security Posture of the Case Study Variants through Exploits

We start by defining the threat model that motivates the definition of the attacks that characterise our tests. Following these definitions, we model the selected attacks in SAFARI using Caldera. Then, we run the tests using SAFARI, reporting the resulting data and commenting on the security analysis of the different versions.

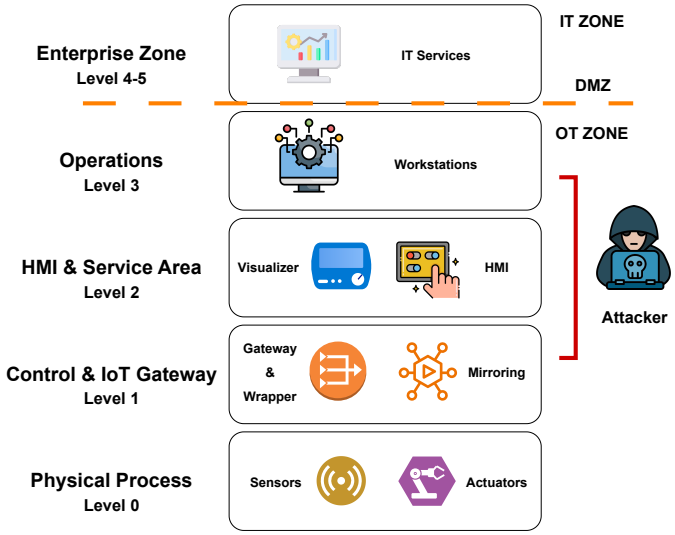

Fig. 8 Representation of the case study threat model, based on the ICS Purdue Model Reference Architecture. We assume the attacker to have Dolev-Yao capabilities between level 2 and 3, to intercept, tamper with, and modify the network traffic on every zone.

5.1 Threat Model

The threat model underpins our security assessment study, defining the adversarial capabilities, attack vectors, and security assumptions we consider when evaluating the three network architectures. We establish the scope of potential threats, the attacker's knowledge and access levels, and the specific vulnerabilities.

We illustrate the system and threat model behind this work through the Purdue Model Reference Architecture [36] for Industrial Control System (ICS). As depicted in Fig. 8, the Purdue Architecture organises the ICS network into six levels: levels 4 and 5 form the Information Technology (IT) network, while the lower levels constitute the Operational Technology network. The latter handles the control, monitoring, and automation of physical processes. At level 0, sensors and actuators interact with the physical processes: we refer to them as IIoT devices. They are directly connected to level 1, which comprises various PLCs, Industrial Gateway aimed to aggregate and expose level 0 data, and specific network devices for traffic management such as Mirroring. These devices implement systems control logic by observing sensor readings and updating actuator signals.

At the enterprise level (level 4 and 5), threats primarily originate from IT-based attacks, including phishing campaigns, ransomware, and unauthorised data exfiltration. Attackers may compromise enterprise systems to move laterally toward operational technology, exploiting weak segmentation between IT and OT networks. The demilitarised zone (DMZ) at level 3–5 plays a critical role in isolating these two domains,

but misconfigurations or vulnerabilities in firewall policies can serve as an entry point for adversaries.

Within the operations and control layers (level 2 and 3), the attack surface expands to SCADA systems, HMIs, and PLCs. At this level, cyber threats manifest as remote code execution, manipulation of process logic, or injection of rogue commands. Threat actors may exploit protocol weaknesses (e.g., Modbus or specific IoT ones such as MQTT) to interfere with automation processes, leading to process instability or production downtime.

At the field device and physical process levels (0 and 1), IIoT sensors and actuators become primary targets. Attackers can manipulate sensor readings through spoofing or jamming techniques, leading to erroneous process adjustments or even catastrophic failures in industrial systems. These low-level attacks are particularly challenging to detect, as they often blend with normal operational variances.

An effective attack detection mechanism must integrate anomaly-based monitoring, AI-driven threat detection, and zero-trust security principles across all Purdue levels to counter these threats. Leveraging real-time analytics and industrial threat intelligence can enhance visibility and resilience against evolving cyber threats.

Our system model assumes that communication between operational levels of an industrial environment – specifically between Purdue levels 1, 2, and 3 – is facilitated through managed industrial switches and segmented network zones. These zones are typically governed by predefined policies for isolating traffic across control, monitoring, and supervision functions.

Regarding adversarial capabilities, we assume the presence of a Dolev-Yao intruder [37] with partial access to the ICS network infrastructure. The Dolev-Yao model grants the attacker complete control over the network communication channel: the ability to read, intercept, inject, replay, delay, or drop any message transmitted between nodes. However, the model assumes *perfect cryptography*, which prevents the attacker from breaking cryptographic primitives – attackers can decrypt messages or forge digital signatures only if they have the corresponding keys. Therefore, brute-force attacks on private keys, password guessing, or side-channel cryptanalysis are outside the scope of this model.

Following the assumptions established in previous work [38,39,40,41], we restrict the attacker's operational domain to the industrial network layers, specifically at levels 2 and 3 of the Purdue Enterprise Reference Architecture. The attacker has no physical access to level 0 and 1 (e.g., sensors and actuators) nor can compromise enterprise systems in level 4 and 5.

Within the 4-5-level perimeter, the attacker seeks to disrupt or compromise industrial operations by targeting vulnerabilities in the network infrastructure and its segmentation mechanisms. We consider the following representative attack scenarios:

1. **Traffic Manipulation and Injection:** The attacker intercepts and alters communication between industrial control systems (e.g., SCADA servers, HMIs, and PLCs). By modifying command or telemetry packets in transit, the adversary can influence control logic outcomes, cause misreporting of system status, or degrade

the trustworthiness of monitoring dashboards. Such attacks often exploit weak protocol implementations or misconfigured firewall policies at level 3.
2. **Network-Level Denial of Service (DoS):** Targeting the communication fabric, the attacker floods industrial switches or key communication endpoints with illegitimate traffic. These attacks overload buffers or saturate bandwidth, resulting in delayed or lost packets. Unlike classical DoS targeting internet-facing servers, these attacks reduce determinism and responsiveness in the control loop, undermining the real-time behaviour expected in OT networks.
3. **Unauthorised Service Discovery and Access:** Through horizontal or vertical scanning, the attacker maps out industrial services exposed across segmented network zones. Exploiting default credentials, unencrypted protocols, or weak access controls, the attacker may gain unauthorised access to level-2/3 services, such as configuration panels, log endpoints, or engineering workstations. These attacks pave the way for subsequent privilege escalation or manipulation of the operational environment.

In summary, the threat model focuses on adversaries that exploit segmentation misconfigurations, network protocol weaknesses, or poor access governance, rather than those performing low-level device exploitation.

5.2 Definition of Attack Simulation Tests

We proceed by defining the attack simulations based on the attack model – we remind, to test the resistance of the alternative OT networks against those cyberattacks. Concretely, we implement five types of IT and OT-related attacks with SAFARI (as elements of the MITRE Caldera platform, cf. Section 3.3):

5.2.1 Recon/Remote Discovery Attack

Once an attacker has established a foothold on a system within the OT network perimeter, the next logical step is to initiate internal reconnaissance. This phase is critical for gaining situational awareness, identifying other assets on the same network segment, and determining potential targets for lateral movement. Simulating such an attack is crucial for validating detection capabilities, as it mirrors real-world adversarial behaviour commonly observed in targeted attacks against industrial environments. We simulate this kind of attack as a two-step action in Caldera.

First, through an arp command, we query the compromised machine's ARP (Address Resolution Protocol) cache. This action lists IP and MAC address mappings for devices recently communicated on the local network segment. Since ARP is a link-layer protocol, we obtain an accurate snapshot of the reachable hosts, including IP addresses of VMs or embedded devices (e.g., PLCs, HMIs) within the same broadcast domain. This step is crucial for attackers to enumerate immediate neighbours without triggering alarms, as it uses data stored locally and avoids generating active network traffic. The second phase involves executing a script that scans the subnet previously inferred. This script uses tools such as nmap to identify live hosts within the given subnet.

This scanning phase enables: a) the enumeration of all reachable devices (not only those in recent communication); b) the detection of open ports and running services, if extended to a port scan; and c) the validation of which systems are active and potentially accessible for lateral movement.

Combined, these two steps simulate a realistic attacker workflow for subnet enumeration and provide a low-noise, effective way to identify lateral movement targets.

5.2.2 Modbus Attack

Modbus is one of the most widely used protocols in industrial environments due to its simplicity and broad support by PLCs, HMIs, and SCADA systems. However, in its default implementation, Modbus lacks fundamental security features: it transmits messages in plain text, does not support authentication, and does not provide integrity checks or encryption. These characteristics make Modbus-based systems vulnerable to passive eavesdropping, unauthorised command injection, and data manipulation.

Therefore, simulating an attack against Modbus is highly relevant and realistic for assessing the resilience of industrial monitoring systems. It reflects a typical scenario in which an adversary, after gaining access to the network, targets a Modbus slave device to read process values or actively alter system behaviour. These attacks are not hypothetical: several real-world incidents [42] and research demonstrations [43] have shown how manipulating Modbus traffic can lead to process disruption, equipment damage, or safety risks Using Caldera's abilities, we implement a multistep attack. This attack exploits Modbus' lack of encryption and authentication by reading, first, and then writing/overwriting a coil and a register. In Modbus, coils are binary outputs that one can turn ON/OFF, typically representing physical devices like relays or switches. Registers are data storage locations, allowing read/write operations for parameters or settings. It is possible to disrupt or alter a production process by manipulating these values. The attack concretises in the following steps:

1. Read Coils: issue a Modbus function code 0x01, reading binary outputs (coils) starting at a given address for a given number of elements. Coils are typically associated with actuators or relays, so retrieving their state gives insight into the current logic of the control process.
2. Fuzz Coils: exploit the lack of write restrictions. The instruction sends multiple write coil (0x05) or multiple write coils (0x0F) commands, targeting the address range between a given begin address and an end address , with a given number of iterations.
3. Read Discrete Inputs: This uses Modbus function code 0x02 to read read-only binary inputs (discrete inputs) for monitoring purposes. Although they are not writable, they can be used to verify system state after coil manipulation or to craft more informed attacks.
4. Fuzz Registers: This issues Modbus write commands (likely 0x06 or 0x10, depending on implementation) to alter register values. The attack sweeps through the given range of registers, modifying registers with values between a given minimum and maximum value. These registers may control critical parameters (e.g., temperature thresholds, motor speeds), so overwriting them can induce system malfunction or unsafe behaviour.

5.2.3 Worm Attack

One of the most disruptive and realistic [44] attack scenarios in OT environments involves the propagation of a worm, a self-replicating payload designed to spread autonomously across multiple systems. Worms have historically been responsible for some of the most damaging cyber incidents in industrial networks (e.g., Stuxnet [45], WannaCry [46]), as they can quickly scale their impact by compromising not just one machine but the entire OT infrastructure. Simulating such behaviour is crucial for evaluating the ability of detection and response mechanisms to handle automated lateral movement, payload delivery, and multi-host compromise. We use Caldera's built-in "Worm" adversary profile to simulate this kind of attacks. In its Linux-based form, the attack aims to collect the history of SSH connections and execute SSH commands, intending to propagate and execute a payload on all identified SSH servers. To do that, it uses Stormssh, a tool for managing SSH hosts and credentials that may expose pre-configured SSH targets when installed and queried. Consequently, the arp command enables the worm to identify neighbouring hosts, refining its lateral movement targets. Then, through a bash command, the worm payload is copied onto remote systems via SSH. Finally, the payload is executed in the background, detaching from the session.

5.2.4 Data Exfiltration Attack

A critical phase of a cyberattack in an OT environment occurs when the adversary has successfully gained access to a machine containing sensitive data, such as production parameters, control logic, configuration files, or intellectual property. In such cases, the attacker likely attempts to exfiltrate the data, potentially leading to industrial espionage, competitive damage, or regulatory violations.

Simulating this type of behaviour is crucial for validating detection mechanisms that monitor unauthorised file access, compression, staging, and outbound transfer operations. To simulate this type of attack, we use Caldera's built-in "Thief", "Advanced Thief", "Superspy", and "Ransack" adversary profiles. While having a different structure, the core purpose of all the aforementioned attacks is to gain sensitive information in the victim system and exfiltrate it outside the network. Specifically:

- "Thief" directly looks in the system for all files with a specified extension(e.g., .st for PLC software files), stage the found files in a directory, compress them, and exfiltrate them;
- "Advanced Thief" looks for a list of files with extensions that could possibly indicate the presence of sensitive information;
- "Superspy" monitors the active user and navigates through their files before the exfiltration;
- "Ransack" tries to gather information (e.g., network information) before the exfiltration phase.

5.2.5 Local Attack(s)

It is realistic for the attacker to also execute families of local attacks on the infected machines, such as local discovery attacks and defence evasions, to gather more information on the system or to establish a more secure and undetectable foothold. To simulate other local attacks, we use Caldera's built-in "Discovery", "Check", and "Defence Evasion" adversary profiles. Specifically, Discovery and Check find network and host information and configurations, while Defence Evasion tries to disable or elude security measures active on the infected host.

5.3 Tests and Results

As last ingredient for running our tests, we deploy a MITRE Caldera agent on target VMs to test the defined scenarios. Specifically, in the "flat" network, we deploy the agent in an Operator laptop VM. Regarding the other infrastructures, in zone 1, we deploy the agent on the Remote Access Jump Host VM, the most exposed system to external threats and in zone 2, on one of the Engineering Workstations. Regarding the Production zones, we deploy different agents to simulate the various scenarios: on an HMI/Scada VM, a PLC VM, and an Operator Laptop VM. After the validation of the successful deployment on the Caldera Server, we execute the defined tests sequentially and atomically, i.e., we run no attacks or attack steps simultaneously, for accurate and repeatable results.

5.3.1 Recon/Remote Discovery Attack

The recon/remote discovery attack, shown in Fig. 9, demonstrate the effectiveness of network segmentation measures. Discovery attacks allow visibility into all VMs in an infrastructure without zones. With the introduction of partial segmentation, zones 1 and 2 lack visibility into other zones, while the zone containing production VMs retains visibility into VMs associated with different production processes – entirely unnecessary visibility. With segmentation based on the IEC/ISA 62443 zones and conduits model, each zone has no visibility beyond the VMs within its own zone. In general, the greater an attacker's visibility over the network infrastructure, the higher his ability is to perform lateral movement and gain additional information and privileges. Consequently, restricting an attacker's visibility to the single zone it can have accessed to significantly reduces the attacker's potential threat.

5.3.2 Modbus Attack

The results of the Modbus attacks show, in Fig. 10, that in a network infrastructure with no to poor segmentation, once the IP address of a PLC has been identified, it is possible to sniff and even tamper Modbus traffic from, possibly, any system in the network. Indeed, in the infrastructure with no network segmentation, once the attacker obtains access, every PLC is accessible from any system, making Modbus traffic tampering straightforward. By increasing the level of segmentation, the PLCs

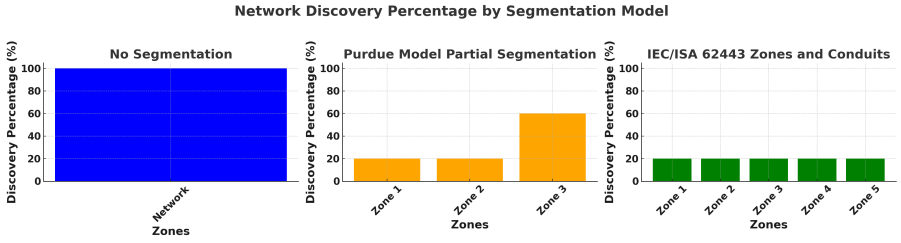

Fig. 9 Remote Discovery Attack results for each OT architecture variant.

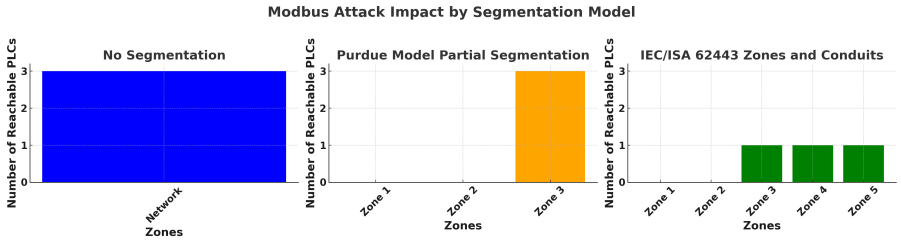

Fig. 10 Modbus Attack results for each OT Architecture.

become unreachable from zones 1 and 2, whereas, in the case of access to zone 3, it is still possible to access the traffic of all PLCs. With the maximum level of segmentation, the attacker can access at most one PLC, precisely the one in the zone they gained access to, such as zones 3, 4, or 5. In this way, Modbus traffic tampering is significantly reduced and, in the worst-case scenario, restricted to the zone to which access was gained.

5.3.3 Worm Attack

The Worm attacks exploit SSH connections. In a flat architecture, with all the systems in the same zone, worms can exploit all SSH connections found since there is no traffic control through a perimeter firewall. When increasing segmentation, traffic travelling from one zone to another would need to pass through a firewall that, if correctly configured, can limit improper use of SSH traffic. In addition, IEC/ISA 62443 requires to disable that all ports and services (such as SSH) that are not strictly necessary, other than allowing only predefined connections through the zone entry/exit point. During tests, in the architecture segmented following the 62443 guidelines, the Worm attack could only be successfully achieved if the attacker happened to get access to a VM that was allowed to communicate via SSH, due to operational reasons, with VMs situated in other zones, thus limiting the success rate of the attack.

5.3.4 Data Exfiltration Attack

The Data Exfiltration tests involve searching for sensitive files on the victim VMs and exfiltrating them to the Caldera server VM. However, in a realistic scenario, exfiltration requires the attacker to export the obtained data from the OT network. To

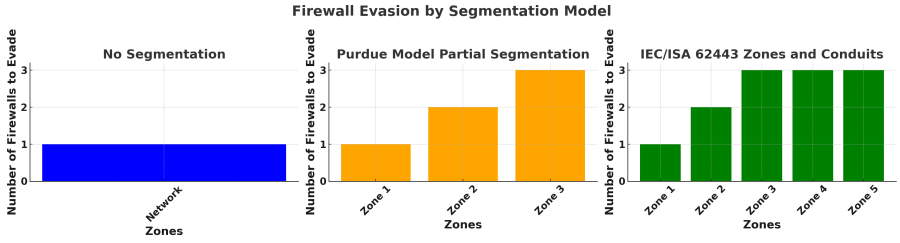

Fig. 11 Data Exfiltration Attack results for each OT architecture variant.

achieve this result, depending on the zone that has been accessed, it is necessary to traverse a specific number of firewalls, as shown in Fig. 11. If properly configured, these firewalls should restrict exfiltration attempts. Starting from a flat network and progressively increasing segmentation by introducing firewalls for each zone, the difficulty of exfiltrating data from the network rises accordingly.

In the IEC/ISA 62443 variant, even if the number of firewalls does not increase from the precedent level of segmentation, the difficulty of the Exfiltration process is still higher than the alternatives, considering the stricter firewall rules and overall complexity of the Network.

5.3.5 Local Attack(s)

Local attacks on the VMs highlight the limitations of the segmentation measures, even the stricter IEC/ISA 62443, since none affect the configurations of individual VMs. Consequently, the local attacks tested on the machines are all successful irrespective of the considered variant. This means that proper OT Security requires other security measures.

5.4 Summary of the Tests

We summarise all test results in Table 2, where we report the three architecture variants in the rows (further subdivided by zones, where applicable) and the five types of attacks tested in the columns. The symbols used in the cells represent the outcome of each attack on the specific variant (zone), with the following meanings:

- ✓ indicates a successful outcome for the attack;
- ≈ denotes a partially successful outcome or a result with constraints;
- \- indicates a complete failure of the attack or one with significant limitations.

Analysing the table, it is evident that, as we move down the rows, we increase the segmentation of the network and see fewer attacks (partially) succeed. Specifically, starting from the flat architecture, where all attacks succeed, the results progress to failed or minimal outcomes for the zones segmented according to the IEC/ISA 62443 model – the only exception is Local Attacks, which, as mentioned, network segmentation does not affect.

Network Segmentation	Zones	Recon/Discovery Attack	Modbus Attack	Worm Attack	Data Exfiltration	Local Attack
Flat	-	✓	✓	✓	✓	✓
NIST-based Model	Zone 1	-	-	≈	✓	✓
	Zone 2	-	-	≈	≈	✓
	Zone 3	≈	✓	✓	≈	✓
IEC/ISA 62443	Zone 1	-	-	≈	✓	✓
	Zone 2	-	-	≈	✓	✓
	Zone 3	-	-	-	-	✓
	Zone 4	-	-	-	-	✓
	Zone 5	-	-	-	-	✓

Table 2 Case study OT Network variants and associated attack impacts.

6 Related work

The main purpose of SAFARI for OT is to use code to reproducibly and reliability simulate real-world OT networks in an emulated air-gapped environment and run security tests on them. We argue that this kind of solution can be interpreted as an implementation of the concept of Digital Twin (DT); a transformative technology that enhances processes, predicts system failures, and identifies anomalies. Moreover, our architecture fits into the technological paradigm of Cyber Ranges. A Cyber Range [47], when viewed as a DT application, is a virtualised environment designed to replicate real-world IT systems, networks, and infrastructures for cybersecurity training, testing, and research [48]. By creating a high-fidelity digital replica of an organisation's environment, a Cyber Range enables users to simulate cyber-attacks and defensive strategies without impacting live operations. This DT approach enhances preparedness by supporting vulnerability analysis, the development of incident response protocols, and the testing of cybersecurity tools in a safe, controlled setting [49].

In the literature, practical technological implementations of this type are scarce, or are developed around a vertical use-case scenario.

For example, De Benedictis et al. [50] introduced an architecture for IIoT anomaly detection based on DT and autonomic computing paradigms. The architecture derives from the MAPE-K (Monitor-Analyse-Plan-Execute over a shared Knowledge) feedback loop to monitor, analyse, plan, and execute appropriate reconfiguration or mitigation strategies based on the detected deviation from prescriptive behaviour stored as shared knowledge. These features allow the architecture to work on the IIoT anomaly detection but are limited to this scope, without the possibility to test malicious software, such as ransomware, since it lacks air-gapping. In contrast, SAFARI allows for

a broader, general purpose, type of architecture virtualisation, allowing for the implementation of multiple types of case study, including the OT case study we propose.

Another related work is the one presented by Masi et al. [51], who derive a cybersecurity DT as part of the security-by-design practice for industrial automation and control systems used in critical infrastructures. Masi et al.'s DT not only serves to simulate cyber-attacks and devise countermeasures but is also directly tied to the system's architecture model for which the cybersecurity requirements are established. While Masi et al.'s work is conceptually similar to our solution, it provides only an architectural overview, without elaborating on or specifying the enabling technologies required to implement the architecture. In contrast, our proposal is a concrete one, which delves into the technological specifics, detailing how it is implemented and used both in practical terms and experiments.

SCASS [13], although completely open-source, proposes a DT oriented to the implementation of a specific industrial component system use case, which includes a mix of virtualised components and "Hardware-in-the-loop" physical components. The authors use this testbed to test cyberattacks specific to the ICS context. Similar to SCASS, EPICTWIN [14] and ICSrange [15] are an open source virtualisations of highly specific ICS use cases and allows for live attack simulations on SCADA systems. Differently from the last two cited works, SAFARI does not aim to reproduce a specific use case, but allows the user to configure any kind of (OT) virtualised network. For example, in the case study proposed in this article, we define, implement, and test three types of increasingly refined OT architectures.

Regarding structure and implementation, another work close to ours is PANDORA [16], which is a safe testing environment that allows users to conduct experiments on automated cyber-attack tools. The differences between the two proposals lie in their focus. PANDORA is designed mainly for testing automated cybersecurity tools, such as scanners or IDS systems. In contrast, SAFARI focuses on creating test scenarios that are as close as possible to real-world case studies, prioritizing open tools and virtualisation technologies, ensuring greater fidelity to practical application environments and enhancing flexibility in adapting to various testing needs.

We summarise in Table 3 the key points of the comparison discussed above.

7 Conclusion

The increasing interconnectedness of industrial environments through OT systems has fundamentally transformed the landscape of cybersecurity challenges. Traditional approaches to security validation in OT environments face significant limitations, particularly when attempting to balance the need for comprehensive testing against the operational constraints of production systems.

SAFARI for OT addresses these challenges by providing a concrete implementation of the Security-Investigation-as-Code methodology, bridging the gap between theoretical security analysis and practical implementation. Through the integration of Infrastructure-as-Code, OS-agnostic Task Automation, and Inspection Tools, SAFARI enables security operators to create faithful digital twins of their OT archi-

Compared solutions	Configurable Virtualized Infrastructure	Sandboxing Capabilities	General Purpose	Hardware Agnostic	Air-gapped by design	Open Source Code
De Benedictis et al. [50]	●	○	◐	◐	○	○
Masi et al. [51]	●	○	◐	●	○	○
PANDORA [16]	●	●	●	◐	○	○
SCASS [13]	●	●	○	○	○	●
EPIC [14]	●	●	○	◐	○	●
ICSrange [15]	●	●	○	◐	○	●
SAFARI for OT	●	●	●	●	●	●

Table 3 Comparison of related work (one work per row) against SAFARI for OT (last row) under its main characteristics.

tectures while maintaining the flexibility to test various security configurations and threat scenarios in a controlled environment.

The case study we present illustrates the practical value of our approach through the systematic evaluation of network segmentation strategies across three incremental OT architecture refinements. The results demonstrate that structured network segmentation, when properly implemented following current standards, can significantly enhance the security posture of industrial systems. More importantly, our methodology enables security operators to validate these improvements through repeatable, automated testing protocols that would be impractical or impossible to implement in production environments. The scalability of SAFARI represents a particular strength of our approach. Security experts can leverage the framework at different levels of sophistication, from manual exploration of digital twins to fully automated security test batteries. This flexibility ensures that organizations with varying levels of cybersecurity maturity and resources can benefit from the methodology while maintaining consistency and reproducibility in their security assessment practices.

Beyond the immediate practical benefits, our work contributes to the broader evolution of security testing methodologies in industrial contexts. By adapting software engineering practices such as regression testing and continuous integration to the OT domain, we provide a foundation for more systematic and rigorous approaches to industrial cybersecurity. The ability to test architectural refinements while ensuring that existing security properties are preserved represents a significant advancement in the field, particularly as OT systems continue to evolve and integrate with broader digital transformation initiatives.

Looking toward future developments, several directions emerge from this research. We envision expanding SAFARI's capabilities to support hybrid on-premises-cloud deployments, enabling users to dynamically scale their testing infrastructure by leveraging cloud resources when local capacity becomes insufficient for comprehen-

sive testing requirements. This hybrid approach would provide organizations with the flexibility to conduct large-scale security assessments while maintaining control over sensitive industrial data and configurations.

Our current prototype implementation presents opportunities for further enhancement through the integration and automation of additional analysis tools, expanding the framework's analytical capabilities and providing security operators with a more comprehensive toolkit for threat assessment. To further democratise access to this technology, we are considering the development of user-friendly interfaces that would enable non-technical users to define high-level testing plans, which the framework would automatically translate into the corresponding Infrastructure-as-Code and OS-agnostic Task Automation components. This abstraction layer would significantly lower the technical barrier to entry while maintaining the rigour and reproducibility that characterise the current implementation.

The methodology presented here also opens possibilities for collaborative security research within the industrial cybersecurity community. The standardised, code-based approach to security investigation could facilitate the sharing of threat scenarios, defensive configurations, and testing protocols across organizations and research institutions, potentially accelerating the development of more effective security solutions for OT environments.

Conflict of interest

The authors declare that they have no conflict of interest.

Data Availability Statement

All data supporting the findings of this study and in particular, the source code for the SAFARI framework and the testbed configuration files are available at: `https://github.com/Flooding-against-Ransomware/SAFARI`.

In addition to that, the code of the attack scripts is available on Zenodo at the link: `https://doi.org/10.5281/zenodo.15593943`

References

1. Z. Wan, Z. Gao, M. Di Renzo, L. Hanzo, IEEE Network **36**(6), 157 (2022). DOI 10.1109/MNET.008.2100484
2. K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, S. Lightman, A. Hahn, S. Saravia, A. Sherule, M. Thompson, Guide to operational technology (ot) security. NIST Special Publication NIST SP 800-82r3, US Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, USA (2023). URL `https://doi.org/10.6028/NIST.SP.800-82r3`
3. K. Awuson-David, J. Thompson, K. Tuner, T. Al-Hadhrami, in *Advances on Intelligent Informatics and Computing*, ed. by F. Saeed, F. Mohammed, F. Ghaleb (Springer International Publishing, Cham, 2022), pp. 461–472
4. S. Hollerer, B. Brenner, P.R. Bhosale, C. Fischer, A.M. Hosseini, S. Maragkou, M. Papa, S. Schlund, T. Sauter, W. Kastner, *Challenges in OT Security and Their Impacts on Safety-Related Cyber-Physical Production Systems* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2023), pp. 171–202. DOI 10.1007/978-3-662-65004-2_7. URL `https://doi.org/10.1007/978-3-662-65004-2_7`

5. L. Rinieri, A. Iacobelli, A. Al Sadi, A. Melis, F. Callegati, M. Prandini, in *2024 IEEE 10th International Conference on Network Softwarization (NetSoft)* (2024), pp. 190–194. DOI 10.1109/NetSoft60951.2024.10588908
6. C. Grasselli, A. Melis, L. Rinieri, D. Berardi, G. Gori, A.A. Sadi, in *2022 International Symposium on Networks, Computers and Communications (ISNCC)* (2022), pp. 1–7. DOI 10.1109/ISNCC55209.2022.9851731
7. A. Melis, A. Piroddi, O. Kaya, R. Girau, in *2024 IEEE 29th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)* (2024), pp. 1–6. DOI 10.1109/CAMAD62243.2024.10942708
8. T. Compagnucci, F. Callegati, S. Giallorenzo, A. Melis, S. Melloni, A. Vannini. SAFARI: A scalable air-gapped framework for automated ransomware investigation (2025). DOI 10.1007/978-3-031-92882-6_15. URL https://doi.org/10.1007/978-3-031-92882-6_15
9. Y. Brikman, *Terraform: Up and Running* (O'Reilly Media, Inc., 2022)
10. Proxmox Team. Proxmox software-defined network. https://pve.proxmox.com/pve-docs/chapter-pvesdn.html. [Online; accessed Sept. 2024]
11. MITRE Corporation. Mitre caldera: Automated adversary emulation platform. https://caldera.mitre.org/ (2024). Accessed: Nov. 2024
12. Z.T. Almulla, H. Rahman, in *2025 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)* (2025), pp. 0342–0347. DOI 10.1109/ICAIIC64266.2025.10920695
13. N. d'Ambrosio, G. Capodagli, G. Perrone, S.P. Romano, Comp. Sec. **151**, 104315 (2025). DOI 10.1016/j.cose.2025.104315
14. N.K. Kandasamy, S. Venugopalan, T.K. Wong, N.J. Leu, Computers and Electrical Engineering **101**, 108061 (2022). DOI 10.1016/j.compeleceng.2022.108061
15. V. Giuliano, V. Formicola. Icsrange: A simulation-based cyber range platform for industrial control systems (2019). URL https://arxiv.org/abs/1909.01910
16. H. Jiang, T. Choi, R.K. Ko, in *Security in Computing and Communications: 8th International Symposium, SSCC 2020, Chennai, India, October 14–17, 2020, Revised Selected Papers 8* (Springer, 2021), pp. 1–20
17. D.P.V. S, S.C. Sethuraman, M.K. Khan, Comput. Secur. **135**, 103490 (2023). DOI 10.1016/J.COSE.2023.103490. URL https://doi.org/10.1016/j.cose.2023.103490
18. A. White, (No Title) (2017)
19. K. Abuelenain, J. Doyle, A. Karneliuk, V. Jain, *Network Programmability and Automation Fundamentals* (Cisco Press, 2021)
20. R. Goldman, *Learning Proxmox VE* (Packt Publishing Ltd, 2016)
21. MITRE Corporation. Mitre att&ck framework. https://attack.mitre.org/ (2024). Accessed: Nov. 2024
22. Proxmox Team. Proxmox provider. https://registry.terraform.io/providers/Telmate/proxmox/latest/docs. [Online; accessed Sept. 2024]
23. W. Schwab, M. Poujol, Schneider Electric (2018)
24. R. Setola, G. Oliva, G. Assenza, L. Faramondi, International Journal of System of Systems Engineering **10** (2020)
25. M. Arifeen, A. Petrovski, S. Petrovski, in *2021 International Conference on Security of Information and Networks (SIN)* (2021). DOI 10.1109/sin54109.2021.9699232
26. M. Kallatsa, arXiv preprint (2024). ArXiv:2401.xxxxx
27. N. Basta, M. Ikram, M.A. Kâafar, A. Walker, Proceedings of the 2022 International Symposium on Security pp. 1–7 (2022)
28. N. Wagner, C. Şahin, M. Winterrose, J. Riordan, J. Peña, D. Hanson, et al., in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (2016). DOI 10.1109/SSCI.2016.7849908
29. W. Shang, Q. Qiao, M. Wan, P. Zeng, Journal of Software pp. 432–438 (2016)
30. H.A. Al-Ofeishat, R. Alshorman, International Journal of Computer Applications **16**(1), 1499 (2024)
31. Zerotier Team. Zerotier. https://www.zerotier.com/. [Online; accessed Sept. 2024]
32. Sshuttle Team. Sshuttle. https://github.com/sshuttle/sshuttle. [Online; accessed Sept. 2024]
33. OpenWrt Team. Openwrt. https://openwrt.org/. [Online; accessed Sept. 2024]
34. D. Formby, M. Rad, R. Beyah, in *2018 USENIX Workshop on Advances in Security Education (ASE 18)* (USENIX Association, Baltimore, MD, 2018). URL https://www.usenix.org/conference/ase18/presentation/formby
35. D. Berardi, F. Callegati, A. Giovine, A. Melis, M. Prandini, L. Rinieri, Future Internet **15**(3) (2023). DOI 10.3390/fi15030095. URL https://www.mdpi.com/1999-5903/15/3/95

36. E. Barkmeyer, E. Barkmeyer, E.K. Wallace, *Reference architecture for smart manufacturing part 1: Functional models* (US Department of Commerce, National Institute of Standards and Technology, 2016)
37. D. Dolev, A. Yao, IEEE Transactions on information theory **29**(2), 198 (1983)
38. S.G. Abbas, M.O. Ozmen, A. Alsaheel, A. Khan, Z.B. Celik, D. Xu, in *33rd USENIX Security Symposium (USENIX Security 24)* (2024), pp. 6597–6613
39. M. Ike, K. Phan, K. Sadoski, R. Valme, W. Lee, in *2023 IEEE Symposium on Security and Privacy (SP)* (IEEE, 2023), pp. 20–37
40. R. Pickren, T. Shekari, S. Zonouz, R. Beyah, in *Network and Distributed System Security (NDSS) Symposium* (2024)
41. A. Erba, N.O. Tippenhauer, in *Proceedings of the 38th Annual Computer Security Applications Conference* (2022), pp. 412–426
42. P. Huitsing, R. Chandia, M. Papa, S. Shenoi, International Journal of Critical Infrastructure Protection **1**, 37 (2008)
43. C. Parian, T. Guldimann, S. Bhatia, Procedia Computer Science **171**, 2453 (2020). DOI https://doi.org/10.1016/j.procs.2020.04.265. URL https://www.sciencedirect.com/science/article/pii/S1877050920312576. Third International Conference on Computing and Network Communications (CoCoNet'19)
44. S. Chaudhary, P.K. Mishra, Computer Networks **236**, 110015 (2023)
45. R. Langner, IEEE Security & Privacy **9**(3), 49 (2011). DOI 10.1109/MSP.2011.67
46. G. Martin, S. Ghafur, J. Kinross, C. Hankin, A. Darzi, BMJ **361** (2018). DOI 10.1136/bmj.k2381. URL https://www.bmj.com/content/361/bmj.k2381
47. A. Pokhrel, V. Katta, R. Colomo-Palacios, in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (Association for Computing Machinery, New York, NY, USA, 2020), ICSEW'20, pp. 671–678. DOI 10.1145/3387940.3392199. URL https://doi.org/10.1145/3387940.3392199
48. R.V. Mahmoud, M. Anagnostopoulos, J.M. Pedersen, IEEE Instrumentation & Measurement Magazine **25**(6), 31 (2022). DOI 10.1109/MIM.2022.9847127
49. M.M. Yamin, B. Katt, Computers & Security **122**, 102892 (2022). DOI https://doi.org/10.1016/j.cose.2022.102892. URL https://www.sciencedirect.com/science/article/pii/S0167404822002644
50. A. De Benedictis, F. Flammini, N. Mazzocca, A. Somma, F. Vitale, IEEE Transactions on Industrial Informatics **19**(12), 11553 (2023). DOI 10.1109/TII.2023.3246983
51. M. Masi, G.P. Sellitto, H. Aranha, T. Pavleska, Software and Systems Modeling **22**(2), 689 (2023)