Wireless Networks
https://doi.org/10.1007/s11276-023-03647-4

On the use of MiniCPS for conducting rigorous security experiments in Software-Defined Industrial Control Systems

Xabier Etxezarreta[1] · Iñaki Garitano[1] · Mikel Iturbe[1] · Urko Zurutuza[1]

Accepted: 22 December 2023

Abstract
Software-Defined Networking (SDN) offers a global view over the network and the ability of centrally and dynamically managing network flows, making them ideal for creating security threat detection and mitigation solutions. Industrial networks possess specific characteristics that make them well-suited for such solutions, leading to extensive research efforts in this area. However, due to the high economic cost and potential risks associated with real equipment interaction, most studies rely on testbeds for demonstration purposes. Therefore, it becomes crucial to understand the limitations and safe operating ranges of testbed environments to ensure the development of scientifically rigorous experiments and accurate result measurements. This study focuses on analyzing MiniCPS-based testbeds in terms of network performance, experiment replicability, and the effects of different attacker implementation modes. The findings demonstrate that utilizing MiniCPS on actual hardware enables the development of highly replicable and high-performance testbeds, as long as they operate within the predefined safe operating ranges. Additionally, this work provides an in-depth analysis of various attacker implementation techniques and their impact on the network.

Keywords Industrial Control Systems · Software-Defined Networking · Network security · Industrial testbeds · MiniCPS · Mininet

1 Introduction

The term Industrial Control System (ICS) encompasses various types of control systems, including Supervisory Control and Data Acquisition Systems (SCADA), Distributed Control Systems (DCS), and other control systems such as Programmable Logic Controllers (PLC) that are commonly found in industrial sectors and Critical Infrastructures (CIs) [1]. CIs are essential for the functioning of modern societies and the well-being of the people within them. Examples of CIs include transportation systems, power grids, hydroelectric dams, and nuclear power plants.

The security of computer and communication networks that supervise and control physical systems is becoming a top priority as many systems and technologies become increasingly interconnected and software-controlled [2]. The number of vulnerabilities threatening CIs is constantly on the rise [3]. In the past, several security incidents have revealed the magnitude of the problem [4], ranging from the destruction of industrial equipment, as in the case of Stuxnet [5], to attacks on the power grid that resulted in a blackout in three provinces of Ukraine [6].

Traditionally, ICSs have been deployed in isolated environments, using proprietary communication protocols and hardware [7, 8]. Compared to Information Technology (IT) networks, industrial network topologies are generally static, and control traffic is inherently repetitive and predictable, as most of the traffic is generated by automated processes [9]. The nature of ICSs makes it challenging for IT security solutions to meet the requirements of these systems. Thus, specific security solutions need to be

✉ Xabier Etxezarreta
xetxezarreta@mondragon.edu

Iñaki Garitano
igaritano@mondragon.edu

Mikel Iturbe
miturbe@mondragon.edu

Urko Zurutuza
uzurutuza@mondragon.edu

[1] Electronics and Computing Department, Mondragon University, Goiru 2, 20500 Arrasate-Mondragón, Spain

developed for these environments. According to the publication NIST SP 800-82 Rev 2 [1], ICSs differ from IT systems in the following aspects:

- ICS are used to control and monitor physical processes.
- Interruptions in ICSs are unacceptable. Availability is prioritized over confidentiality.
- Time is critical in many ICS, and communication latency must remain within established values.
- The replacement and updating period for devices in ICSs is much longer compared to IT devices.
- The application of security patches is often postponed due to availability and reliability needs.
- In many cases, ICSs lack the capability to integrate security mechanisms.

Generally, ICSs are established and designed using strict policies and rules in order to meet high-performance and resilience requirements in critical operations. In many cases, this is achieved through the manual implementation of functions and management rules in custom Command Line Interfaces (CLIs) provided by devices from various industrial vendors. Since most existing industrial network infrastructures are designed for specific applications and implemented in a fixed manner, they are not capable of supporting diverse types of industrial applications with different requirements. This calls for a network infrastructure that allows for dynamic configuration and interoperability between different industrial applications, giving Software-Defined Networking (SDN) the opportunity to be the key technology in building ICSs [10]. RFC 7426 [11] defines SDN as a network programmability approach, which means the ability to dynamically initialize, control, change, and manage network behavior through open interfaces. For this purpose, the control plane is separated and centralized in an external entity called the SDN controller, while the data plane remains in the network devices, focusing its functionality on packet forwarding.

Currently, the integration of SDN in industrial environments is in an early stage of development and requires extensive testing and validation work [12]. However, SDNs have been successfully deployed in IT (e.g., data centers) and telecommunications environments (e.g., wide area networks, 5 G mobile networks), and some studies have already demonstrated their usefulness for developing intelligent and reliable security solutions in ICS, particularly in the field of attack detection and response [13–16]. Therefore, it is necessary to have test environments that accurately and deterministically provide real-world conditions in ICS, including network topology, devices, protocols, and different attack scenarios.

In this work, we investigate the necessary characteristics for conducting scientifically rigorous security experiments within test environments based on MiniCPS [17]. Our main goal is not only to provide guidelines for practitioners using MiniCPS, but also to provide a methodology that can be applied to other test environments, including physical testbeds, to provide behavioral insights and identify the limitations and safe operating ranges of the testbed under study. First, we perform a thorough analysis of the network performance to identify the limits and safe operating ranges for various configurations of packet sizes, bandwidths, delays, and packet queue sizes. Second, we evaluate the ability of MiniCPS to consistently reproduce results over multiple runs and with hardware changes. Finally, we examine the network interference caused by various simulated attacker implementations to gain insight into their impact on the network. All of the tools used to perform the experiments are open source and publicly available to practitioners.

The remainder of the paper is structured as follows: Sect. 2 provides an in-depth analysis of the MiniCPS platform, highlighting its key features and functionalities. In Sect. 3, we present a comprehensive overview of the various experimental scenarios employed in this study, outlining their specific characteristics. Section 4 offers a detailed presentation of the experimental results, including the data analysis and key findings. Lastly, Sect. 5 concludes the paper by summarizing the main insights obtained from the study and discussing their implications for future research in the field.

2 MiniCPS framework and its characteristics

Emulation testbeds enable the creation of complex scenarios in a controlled and secure manner, which may be difficult or impossible to physically recreate due to various factors, including economic constraints. Supervisory Control And Data Acquisition (SCADA) systems, along with their networks, are an example of such scenarios. The recreation of these scenarios, even through emulation, provides the necessary means for conducting research at an affordable cost. Furthermore, the versatility they offer allows for testing systems in a wide range of situations without compromising physical installations or human safety. Simulated environments, on the other hand, leverage software to accurately replicate the behavior of systems.

Unfortunately, real-world industrial topologies and systems are not always accessible to security researchers and developers. An increasingly prevalent trend in industrial environments is the use of emulation, virtualization, and simulation testbeds [18], which provide researchers with the capability to thoroughly test new security mechanisms in a controlled environment before the deployment in real-world systems. This approach allows for the validation of their effectiveness and minimizes potential errors and disruptions during the implementation in real systems.

To achieve this, tools such as MiniCPS offer a framework for developing extensible and reproducible security research environments focused on simulating network communications and physical layer interactions of industrial systems. MiniCPS is built upon Mininet [19, 20], a tool that emulates a complete network of hosts, links, and switches on a single machine, utilizing process-based virtualization and network namespaces available in the Linux kernel. Mininet is particularly useful for interactive development, testing, and demonstrations, especially those involving OpenFlow [21] (a communication protocol that provides access to the data plane of a switch over the network) and SDN. This makes MiniCPS an ideal choice for testing detection and response mechanisms to attacks in SDN-based industrial environments. Furthermore, the different SDN controller implementations provided by Mininet (and subsequently by MiniCPS) can easily be transferred to production industrial SDN networks once the necessary testing has been completed. Figure 1 depicts the architectural relationship between MiniCPS and Mininet. The main components of MiniCPS are the following:

- **Hosts:** Hosts are emulated as *bash* processes running in a network namespace. This setup enables the execution of any code that would typically run on a Linux server, such as a web service, within one of these hosts. Each host possesses its own network interface and can solely access its own processes. Moreover, hosts have the capability to execute any command or application accessible to the underlying Linux system, along with its file system. Within MiniCPS, these emulated hosts are leveraged to run various real-time ICS simulation Python scripts. The accuracy of MiniCPS simulations of physical systems depends on the precision of the user-developed code. Researchers have the flexibility to progress from basic scripts for simple simulations to complex models that faithfully represent physical systems and closely resemble their real-world counterparts. In network-based experimentation scenarios where process dynamics are not the primary focus, simple simulations may be sufficient. However, for experiments that require precise, real-world-like simulations, MiniCPS allows the extension and implementation of new logic to achieve this goal. Furthermore, it is worth noting while MiniCPS provides a framework for ICS simulations, researchers can extend its capabilities by integrating other physical system simulations into the framework. Examples of simulations that could be implemented inside MiniCPS include pyTEP [22], ICSSIM [23] or DHALSIM [24], to name a few. This integration can be accomplished by running the corresponding code on the hosts deployed within the MiniCPS environment.
- **Switches:** Thanks to its integration with Mininet, MiniCPS allows the deployment of software-based switches, including options such as Open vSwitch (OVS) [25] or Indigo Virtual Switch (IVS). Unlike hosts, switches are deployed by default in the main namespace, but gives the option to assign their own namespace.
- **SDN Controller:** An SDN controller is responsible for managing the control flow to switches using specific protocols like OpenFlow. Although Mininet includes several default SDN controller implementations (e.g., NOX, ovs-controller), allows users to use any other controller by explicitly specifying the use of an external SDN controller and providing the IP address and port where it is deployed (whether on the same machine or on an external server). Similar to switches, if a controller included in Mininet is used, the SDN controller will be deployed in the main namespace.
- **Links:** The links in MiniCPS, whether between host-switch, switch-switch, or host-host, are established as virtual Ethernet pairs (vEth). It is possible to customize the characteristics of each link, such as bandwidth, delay, packet loss, among others, using the system management program in the user space Traffic Control (TC). This way, different network conditions can be simulated to evaluate the performance and effectiveness of experiments in a controlled environment.

When conducting scientifically rigorous experiments on an emulated and simulated testbed using MiniCPS, it is necessary to ensure their usage within confidence boundaries. In other words, it is necessary to guarantee that the

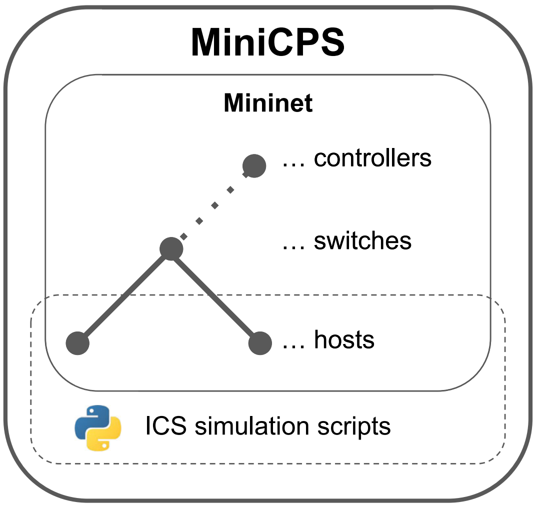


Fig. 1 Architectural relationship between MiniCPS and Mininet

obtained results correspond to reality and not to an out-of-bounds usage that may alter the results, such as running experiments with the CPU or RAM at 100%. To achieve this, it is essential to establish confidence margins before conducting any experiment. This work analyzes these properties through the following features:

- **Performance:** Network performance is critical as it allows evaluating the capacity and quality of the network under controlled situations. If the network does not perform well, issues such as latency, packet loss, congestion, and other problems may arise, negatively impacting both the overall network performance and the performance of applications and services running on it, thus altering the results. Therefore, it is essential to continuously measure and optimize network performance to ensure optimal and reliable operation in the testing environment.
- **Reproducibility:** Reproducibility refers to the ability to repeat an experiment and obtain consistent or statistically coherent results. To achieve reproducibility, the researcher must be able of configuring the experimental platform to its initial state and perform all required actions in the established order and appropriate moments. This ensures that any changes in the results are due to variations in the variables being studied and not to external factors or experimental errors. Reproducibility is essential to guarantee the validity and reliability of the results.
- **Simulated Attacker's Interference:** Researchers can conduct security experiments in MiniCPS by implementing and simulating attackers with different techniques and locations. It is crucial for researchers to consider that different implementation approaches can affect the accuracy and reliability of the measurements of the obtained results. Therefore, researchers must take measures to minimize for the interference caused by different attacker implementations and ensure the reliability of the results.

3 Experimental scenarios

The topology used in the conducted experiments has been deployed using the MiniCPS tool and its simulation of the SWaT (Secure Water Treatment) testbed [26]. The SWaT dataset is one of the most popular in the field of industrial cybersecurity [18]. Part of the SWaT test environment simulation is already implemented within MiniCPS. As depicted in Fig. 2, the industrial topology consists of the following devices:

1. *End-devices:* Three hosts (PLC1, PLC2, and PLC3) implemented through three Mininet hosts where different Python scripts are running to simulate industrial PLCs and their corresponding physical processes.
2. *Switch:* An OpenFlow switch (Open vSwitch) that interconnects all the devices in the topology.
3. *SDN Controller:* An SDN controller responsible for controlling network flows by installing (or removing) flow rules in the OpenFlow switch. In the experiments, the Ryu[1] controller has been used, which communicates with the switch using the OpenFlow protocol.
4. *Attacker:* A network attacker capable of intercepting and modifying network traffic between PLC1 and PLC3.

For the experimentation, we defined four scenarios (shown in Fig. 2). Three of them involve different locations and implementation techniques of an attacker, while one scenario is configured in the absence of an attacker:

- Scenario 1 (Fig. 2a): The attacker is deployed on a Mininet host and connected to a port on the OpenFlow switch. We developed a script using the Ettercap[2] tool to intercept the network traffic between PLC1 and PLC3 through an ARP spoofing attack. This forces the network traffic between PLC1 and PLC3 to pass through the attacker's host before reaching its destination.
- Scenario 2 (Fig. 2b): The attacker is deployed in the middle of the link between PLC3 and the OpenFlow switch. The attacker has two network interfaces, one for connecting to PLC3 and another for connecting to the OpenFlow switch. Using the Scapy[3] tool, the attacker bridges the two interfaces so that PLC3 can communicate with the rest of the network, while allowing the attacker to collect or manipulate network traffic.
- Scenario 3 (Fig. 2c): The attacker is implemented as a module in the SDN controller. By installing different flow rules in the OpenFlow switch, the traffic is sent to the SDN controller to execute any attacker's actions and make decisions regarding the forwarding of network traffic.
- Scenario 4 (Fig. 2d): This topology is considered as a reference in our experimentation. It is assumed that this is the normal operation of the network in the absence of an attacker.

The experiments were conducted on an HP computer with an Intel i5-10210U processor and 16 GB of RAM, running Ubuntu 22.04 as the operating system. To minimize the impact of potential hardware dependencies on the results, we employed a single computer. For the reproducibility study, we added a second MSI computer and a server to the

[1] Available at https://github.com/faucetsdn/ryu.
[2] Available at https://www.ettercap-project.org/
[3] Available at https://scapy.net/

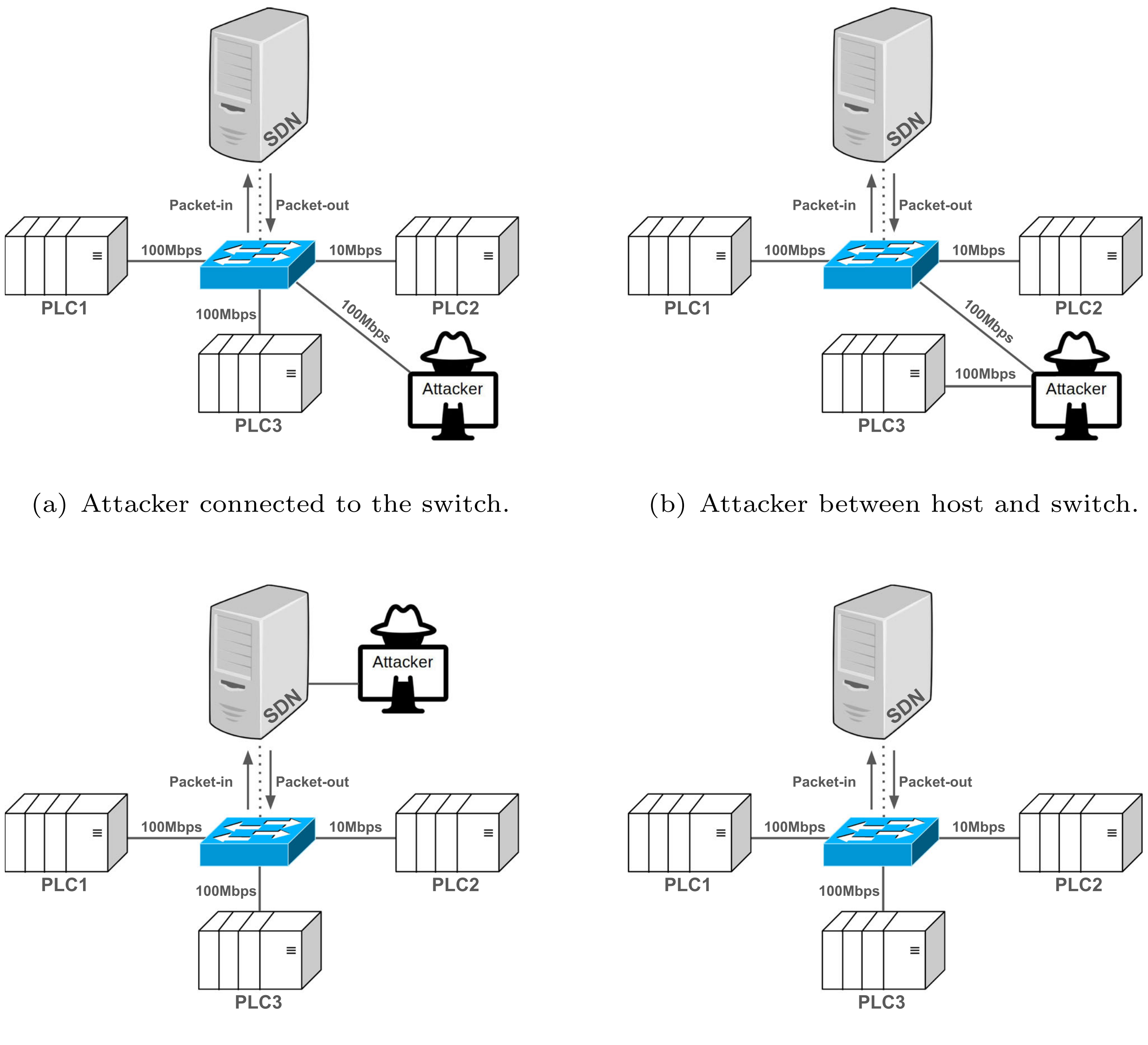
experimentation, resulting in a total of three devices with different hardware. The MSI computer, equipped with an Intel i7-10710U processor and 16 GB of RAM, and the server, equipped with an Intel Xeon E5-2630 processor and 8 GB of RAM, were both running Ubuntu 22.04.

4 Experimental results

This section studies the MiniCPS framework as a tool for conducting rigorous scientific experiments through three fundamental characteristics: (1) network performance, (2) reproducibility, and (3) the interference caused by the implementation of different attackers. Based on the obtained results, we developed a list of guidelines and warnings for the use of MiniCPS as an experimentation platform. Drawing from the results and experiences obtained, the aim is to provide guidance and useful recommendations for future users of MiniCPS.

4.1 Network performance

In this section, we present an in-depth analysis of network performance in the context of MiniCPS. Our investigation



(a) Attacker connected to the switch.

(b) Attacker between host and switch.

(c) Attacker in the SDN controller.

(d) Base scenario without attacker.

Fig. 2 The four scenarios used in the experimentation. Figures 2(**a**), 2(**b**) and 2(**c**) represent different implementations of attackers within MiniCPS. Figure 2(**d**) represent the reference scenario without attacker

encompasses several configurable features, enabling us to assess the system's behavior under different scenarios. Specifically, we delve into three key aspects: bandwidth analysis, the effect of delay configuration on bandwidth and packet loss, and the impact of varying packet queue sizes.

4.1.1 Bandwidth analysis

To analyze network performance, we have considered the traffic sent by a source host (PLC1) and the traffic received by the destination hosts (PLC2 and PLC3). By default, Mininet does not impose bandwidth limits, relying on the hardware used for experimentation. To ensure that performance measurements are independent of the hardware, we have defined specific bandwidth limits in the network. In this setup, we configured a bandwidth limit of 10Mbps between PLC2 and the switch, and a limit of 100Mbps between PLC3 and the switch. For testing the bandwidth, we generated UDP traffic by utilizing Iperf[4] tool, PLC1 acting as the client and PLC2 and PLC3 as the servers. We generated network traffic for 120 s, varying the packet sizes with payloads of 64, 128, 256, 512, 1024, and 1472 bytes, and bandwidths ranging from 0 to exceeding the limit of each scenario (10Mbps and 100Mbps). It is important to note that the maximum size of an Ethernet frame is 1518 bytes (which can reach up to 1522 bytes with a VLAN tag), with 18 bytes allocated to the frame header and checksum. Consequently, the maximum transmission unit (MTU) is 1500 bytes. The minimum Ethernet frame size is 64 bytes. Industrial protocols implemented over Ethernet operate with packets within these size ranges. In our tests, a packet with a payload of 1472 bytes represents the limit before packets are fragmented for transmission. For each packet size and bandwidth, we performed measurements 30 times on 3 different days to ensure that the results were not affected by specific system states. Additionally, we halted unnecessary processes to prevent interference with the measurements. The network performance experienced in the reference scenario and the two emulation strategies with different bandwidths is depicted in Fig. 3(a, b), respectively.

The ideal behavior would be a straight diagonal line ($x = y$) that reaches the maximum speed of the interface, followed by a horizontal line once the theoretical bandwidth limit is reached (taking into account the overhead introduced by header bytes transmission). In reality, as shown in Fig. 3, the behavior of the configuration without an attacker approximates the ideal, at least for packets with a payload larger than 256 bytes, reaching over 80% of the available bandwidth. In the worst-case scenario, with a payload size of 64 bytes, the performance reaches 60% of the interface speed. When the bandwidth limit is exceeded, packets are buffered and transmitted at the maximum speed achievable by the hardware used for experimentation.

Based on these results, we can conclude that MiniCPS realistically replicates good performance quality within the specified limits for each tested packet size and type, as long as the condition $x \approx y$ is satisfied. In the worst-case scenario of 64-byte packets, the bandwidth usage reaches approximately 60% of the maximum available capacity.

4.1.2 Delay analysis

ICSs are comprised of interconnected devices, including sensors, actuators, controllers, and computational units, which rely on wired or wireless connections to interact with each other. The delay experienced during this communication process can vary based on factors such as component distance, network connection quality, and traffic volume. Longer distances can introduce higher latency, potentially impacting real-time responsiveness and synchronization. Unstable network connections may result in increased latency and data loss, leading to disruptions in system operation. Furthermore, higher traffic volumes can cause congestion and delays in data transmission.

In MiniCPS, the delay parameter on a link can be set to a specific value to emulate a desired delay to simulate different network conditions. The delay parameter is used to model the time it takes for data to propagate from the sender to the receiver over a particular link. It represents the time delay introduced by the physical characteristics of the link, such as the transmission medium and network congestion. Researchers and developers can simulate real-world communication conditions and study the impact of latency on the behavior of the ICS. Delays can affect the timeliness and reliability of data transmission, and they can have significant implications on the performance and stability of the ICS.

To assess the impact of the delay configuration in the network performance, we conducted an analysis by implementing various delay configurations on the link between the switch and PLC3. We considered four distinct delay scenarios: 5ms, 25ms, 50ms, and 100ms. In each scenario, we utilized the Iperf tool to generate network from PLC1 to PLC3, ranging from 0Mbps to 120Mbps and with packet payload sizes of 64, 128, 256, 512, 1024, and 1472 bytes. The experimental results are illustrated in Fig. 4(a–d). The results show that as the delay increases, the traffic received by the target device decreases significantly. When the delay increases, it generally leads to a decrease in bandwidth due to the inherent relationship between delay and data transmission rates. With a lower delay, the sender can transmit packets more frequently.

[4] Available at https://iperf.fr/

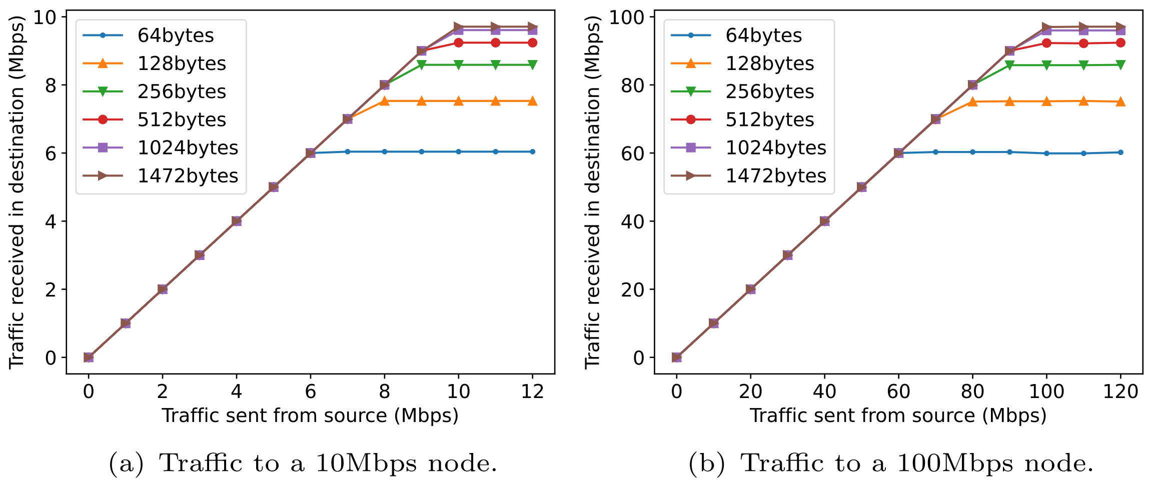


Fig. 3 Network performance in the base reference scenario configuration without attacker (Scenario 4)

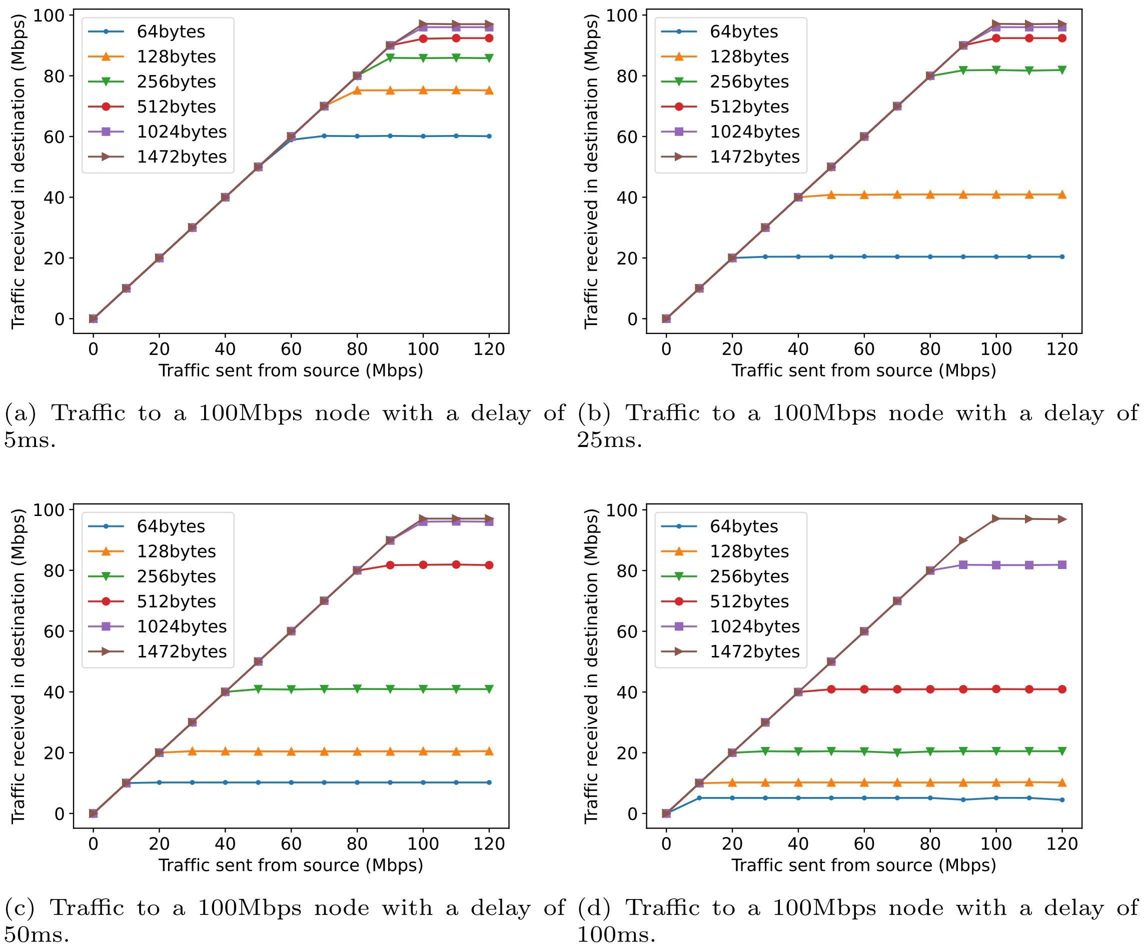


Fig. 4 Impact on the network performance with different delays configurations in the base reference scenario configuration without attacker (Scenario 4)

This results in a higher transmission rate and, therefore, a higher bandwidth. Conversely, when the delay increases, the transmission rate decreases due to the longer waiting time. As a result, the bandwidth decreases.

When packets arrive at a network device for forwarding, they are often stored in queues or buffers temporarily before being forwarded to the next hop. If the queues become full due to high network delay, subsequent incoming packets may be dropped, resulting in packet loss. Therefore, the relationship between delay and packet loss is fundamental to the development of rigorous experiments within identified confidence values. Figures 5(a–d) represent the packet loss ratio on different bandwidth, delay and packet payload sizes configurations. If the packet queue is not fully populated during data transmission, no packet loss occurs, ensuring that all packets generated at the source successfully reach the destination device. This happens in the scenario with 5ms delay configuration (Fig. 5a). Contrarily, in scenarios where the latency is set to 25ms, 50ms, and 100ms, the packet queue becomes congested with some packet payload size configurations, resulting in packet loss when transmitting traffic.

In conclusion, understanding the interplay between bandwidth usage, transmission delay, and packet loss is crucial for the successful development of scientifically rigorous experiments on MiniCPS. On one side, reducing the packet payload size while increasing bandwidth and delay leads to an increase in packet loss and a decrease in bandwidth usage. On the other side, increasing the packet payload size results in a decrease in packet loss and an increase in bandwidth usage.

4.1.3 Packet queuing analysis

In networking, when a link becomes congested, meaning it receives traffic at a higher rate than it can transmit, packets are typically stored in a queue. The queue acts as a temporary buffer, holding packets until they can be forwarded or transmitted. However, if the queue becomes full and reaches its maximum capacity, any additional incoming packets will be dropped to avoid further congestion and potential performance degradation. MiniCPS allows customizing the packet queue size by configuring a link with Traffic Control (TC) type interfaces. In our experiments, we observed that if the packet queue size is not explicitly configured on a link with TC interfaces, it defaults to a size of 1000.

To evaluate the impact of various queue size configurations on network performance, we conducted experiments involving five scenarios with packet queue sizes of 50, 250, 500, 750, and 1000 slots. In each configuration, we generated 120Mbps traffic from PLC1 to PLC3, surpassing the maximum 100Mbps bandwidth of the destination node. Throughout the experiments, we maintained a fixed packet payload size of 64 bytes and introduced network delays of 5ms, 25ms, 50ms, and 100ms. We aimed to assess how different queue sizes and network delays affect bandwidth utilization and packet loss.

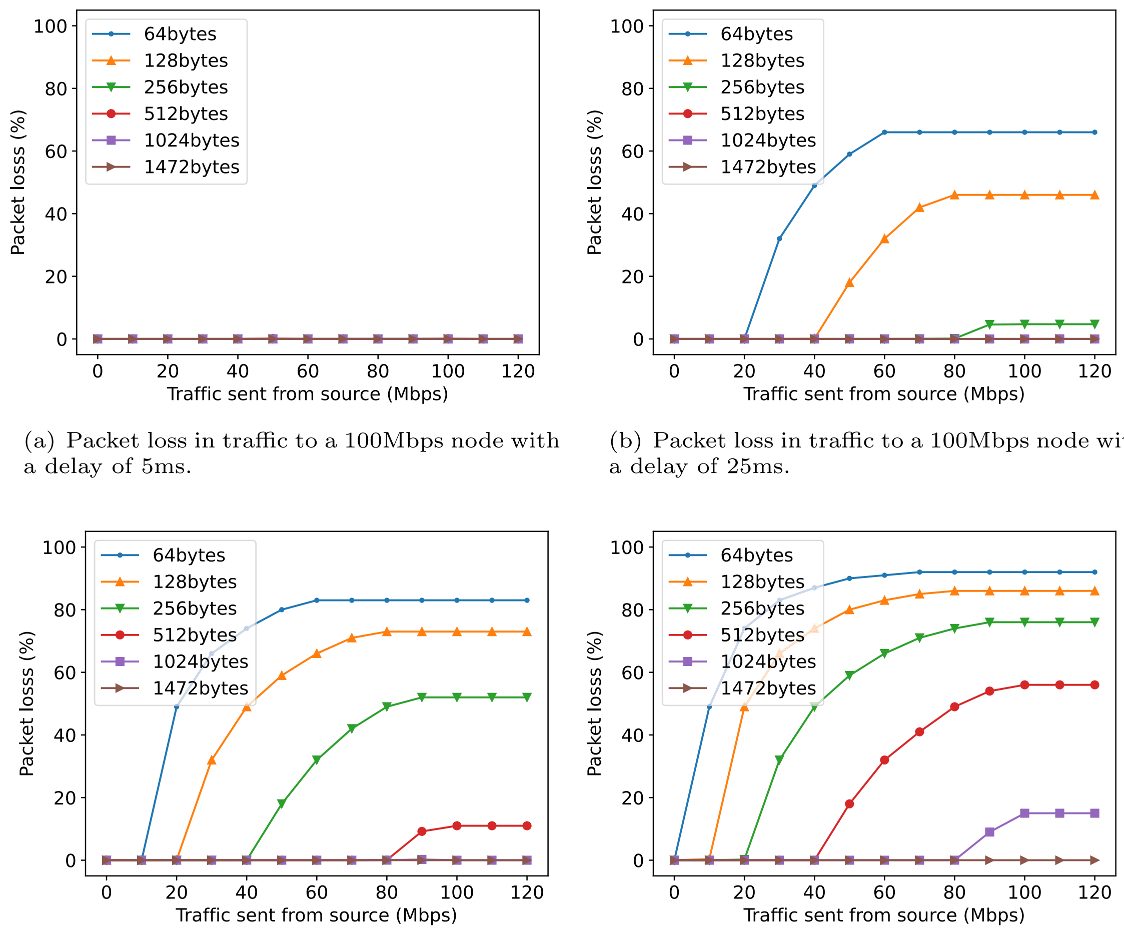
Figure 6(a) illustrates the maximum bandwidth utilization achieved in our hardware. It indicates that a packet queue size of 750 slots or larger results in optimal bandwidth utilization. Conversely, reducing the packet queue size leads to a significant decrease in bandwidth usage across all scenarios with different delay configurations. This decrease in bandwidth can be attributed to the packet queue becoming overwhelmed, resulting in substantial packet loss. The observation is visually depicted in Fig. 6(b). When a queue size of 50 slots is employed, packet loss exceeds 90% in all network delay scenarios. However, as the packet queue size is increased, the occurrence of packet loss diminishes. For example, when a queue size of 750 slots is coupled with a 5ms network delay, the packet loss is reduced to 0%.

Overall, our findings highlight the importance of appropriately sizing the packet queue to ensure optimal network performance and minimal packet loss. With a smaller queue size, the network experiences higher levels of congestion and packet loss, resulting in decreased bandwidth utilization. This is especially evident when network delays are introduced, as the smaller queue size struggles to handle the incoming traffic effectively. The choice of queue size in research experiments should align with the specific objectives and characteristics of the study. It requires careful consideration of the network environment, traffic patterns, and performance requirements. By taking these factors into account and appropriately configuring the queue size, researchers can ensure that their experiments yield accurate and reliable results.

4.2 Reproducibility analysis

To achieve scientifically rigorous experimentation, it is necessary not only to obtain realistic performance but also to be able to reproduce the results at any time, even by another researcher using the same tools and configurations. However, hardware can vary from one experiment to another, which can affect the results obtained due to various reasons such as resource constraints, system randomness, or different requirements for each use case. For this reason, we conducted different tests to study the impact of reproducibility with multiple runs and different hardware configurations. To do this, we measured the network performance using the Iperf tool in several repetitive experiments (traffic sent from one device and received on another device). In particular, we considered the following two experiments:

- *Same hardware:* Involves multiple runs on the same hardware. We used the HP computer to perform these experiments.
- *Different hardware:* Involves multiple runs on different hardware. We used three devices with different hardware, as mentioned in Sect. 3.



(a) Packet loss in traffic to a 100Mbps node with a delay of 5ms.

(b) Packet loss in traffic to a 100Mbps node with a delay of 25ms.

(c) Packet loss in traffic to a 100Mbps node with a delay of 50ms.

(d) Packet loss in traffic to a 100Mbps node with a delay of 100ms.

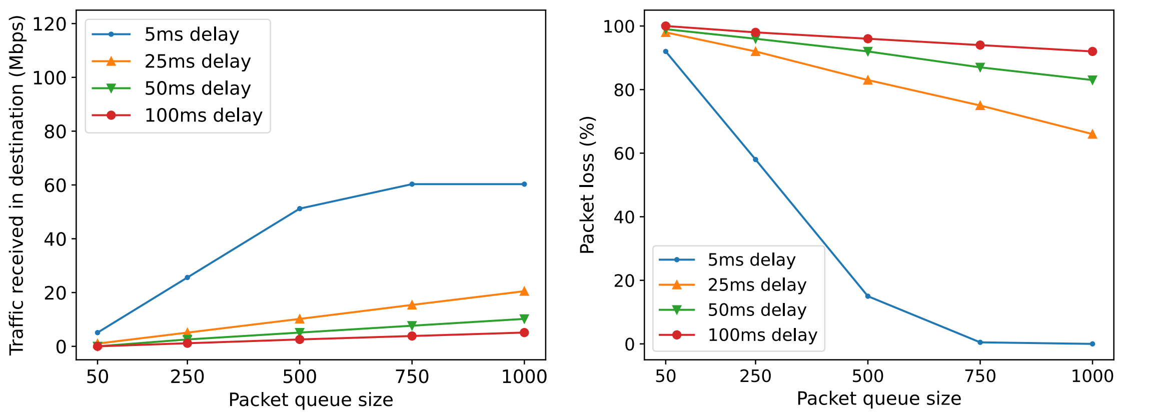
Fig. 5 Packet loss with different delays and packet sizes configurations in the base reference scenario configuration without attacker (Scenario 4)

In all experiments, we utilized the Iperf tool to generate UDP traffic ranging from 0Mbps to 120Mbps between the PLC1 and PLC3 hosts, which had a maximum bandwidth of 100Mbps. Each packet in the experiment consisted of a 64-byte payload. We conducted a total of 20 experiment runs when using the same hardware setup. Additionally, we performed 20 runs for each device in the case of different hardware, resulting in a total of 60 runs. The obtained results are presented in Fig. 7(a–b). Notably, we observed consistent performance across all runs when the traffic remained below 60Mbps. However, as the traffic exceeded the maximum bandwidth of 60Mbps, packets started to buffer, waiting for their turn to be transmitted. Consequently, this led to increased variability in the obtained results.

The obtained results were analyzed numerically by calculating the coefficient of variation (C_v), which measures the relationship between the mean and standard deviation of a given variable. An ideal C_v is 0, indicating that the arithmetic mean accurately represents the dataset. The coefficient of variation (C_v) is calculated using the equation $C_v = \frac{\sigma}{\bar{x}} * 100$, where σ represents the standard deviation and $\bar{x}$ represents the mean. Figure 7(c) illustrates the C_v for each bandwidth configuration in the two considered scenarios: same hardware and different hardware.

The results clearly demonstrate that conducting experiments on the same hardware leads to better reproducibility compared to using different hardware. It is crucial to emphasize the high precision of these experiments, with a coefficient of variation (C_v) below 0.2 when the bandwidth utilization remains at or below 40Mbps. However, as the bandwidth increases to 60Mbps, the C_v progressively rises, reaching relatively high values in both scenarios. In the case of the same hardware, C_v exceeds 0.3, while when different hardware is used, C_v surpasses 0.6. Notably, the maximum value of C_v in the same hardware scenario is 0.62, whereas in the different hardware scenario, it reaches 1.01.

The obtained results show that MiniCPS provides precise reproducibility capabilities in experiments. Under



(a) Bandwidth utilization with different delay and queue size configurations.

(b) Packet loss with different delay and queue size configurations.

Fig. 6 Queue sizes configurations impact on the network performance in the base reference scenario configuration without attacker (Scenario 4)

moderate network load conditions, the variation in the results is very low, ensuring high reproducibility regardless of the hardware used. However, as the network load increases, it is recommended to use the same hardware for conducting the experiments, as this offers better replicability with lower variation among different runs.

4.3 Evaluating attacker's implementation interference

When testing attack detection or mitigation systems, it is critical to integrate a device that can execute or launch attacks. This integration is necessary to evaluate the effectiveness of security systems. Various methods can be used to simulate an attacker, especially in the context of MiniCPS. However, these simulations can affect the measurement of experimental results. In this section, we first analyze the impact of different attacker implementations on network latency. Then, we analyze the CPU utilization by the SDN controller under different implementations.

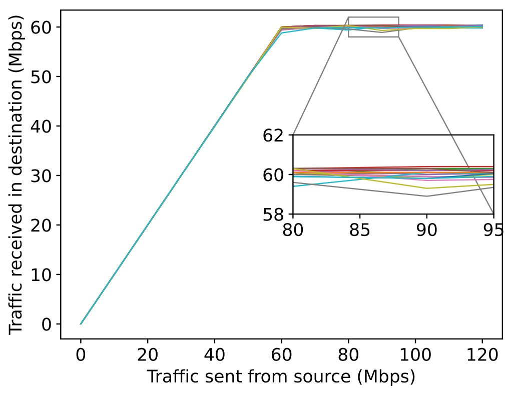
4.3.1 Network latency interference analysis

To measure the network latency impact of different attacker implementations, we focus on analyzing a well-known attack in industrial environments known as Man-in-the-Middle (MitM) attacks [18, 27]. In a MitM attack, the attacker intercepts messages between two parties who believe they are communicating directly. A major concern is the use of unencrypted industrial protocols to transmit data between ICS devices [28]. ICSs are used to control and automate industrial processes, and a successful MitM attack could disrupt, damage, or even compromise the security of critical infrastructure. An attacker who is able to intercept traffic through a MitM attack will be able to gain extensive knowledge of the operational process and, in the worst case, manipulate ICS traffic. An example of such disruptive attacks are False Data Injection Attacks (FDIA), where an adversary injects false or manipulated data into the process [29–31], causing severe consequences.

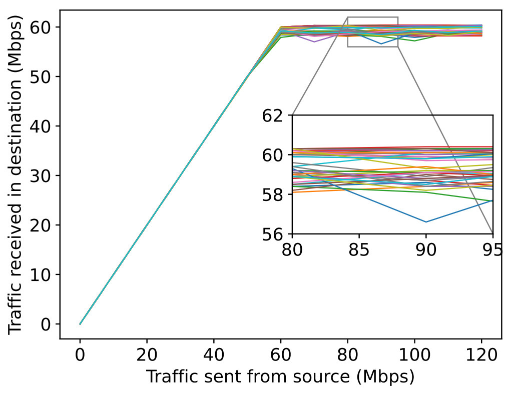
We examine three scenarios (detailed in Section 3) where an attacker capable of intercepting traffic between PLC1 and PLC3 is implemented in three different locations and follows different MitM attack strategies. To quantify the impact of each implementation on the network, we assume that the attacker is performing a passive MitM attack. The attacker intercepts the communication between PLC1 and PLC3, and vice versa, without making any changes. The attacker collects the transmitted packets and forwards them to their original destination without any manipulation.

To evaluate the effect of different implementations, we measured the Round-Trip Time (RTT) between PLC1 and PLC3. RTT measures the time it takes for a packet to travel from the sender to the receiver and back. We analyzed the traffic for a duration of 10 min in each scenario and collected the resulting data, which is shown in Fig. 8. To provide a comprehensive overview of the RTT measurements for each scenario, we calculated the average, standard deviation, and minimum and maximum values, all of which are summarized in Table 1.

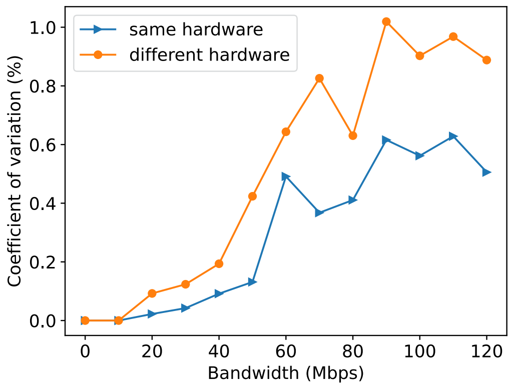
The observed differences in RTT measurements between the various scenarios is attributed to the different methods used in each attack strategy and their impact on network traffic. The baseline scenario (Scenario 4) serves as a reference point without any attacker-induced interference, allowing the impact of the different attack strategies to be compared. The lower RTTs in this scenario represent the baseline performance without any intentional attacker implementation.



(a) Different executions on the same hardware.



(b) Different executions on different hardware.



(c) Comparison of the coefficient of variation for the two hardware scenarios.

Fig. 7 Reproducibility of network performance in different hardware usage scenarios

In Scenario 1, where the attacking device is connected to the switch and ARP poisoning is used, the increased RTT and variability is explained by the additional steps introduced into the communication process. ARP poisoning involves the manipulation of the ARP cache on the target devices, resulting in potential delays in the normal flow of network traffic. This interference results in increased RTT and a less predictable communications environment.

Scenario 2, which utilizes a computer with two interfaces and employs Scapy to bridge them for the MitM attack, introduces a more controlled environment compared to Scenario 1. The lower increase in RTT and reduced variation can be attributed to Scapy's management of the bridging process and the elimination of the need to add new links, which can introduce additional delays. However, it is important to note that while this scenario is more stable, it still introduces some level of interference due to the active manipulation (interception and forwarding) of network traffic.

In Scenario 3, where the attacker is located in the SDN controller, the implementation of forwarding rules on switches to control the traffic path results in a more optimized MitM attack compared to Scenario 1. This approach also minimizes variations in RTT, as the SDN controller can efficiently manage and redirect the traffic flow without introducing unnecessary delays. A longer delay in this scenario compared to Scenario 2 is due to the traffic having to go to and from the SDN controller. In Scenario 2, traffic passes through a host without adding additional communication links.

In conclusion, the choice of attack strategy significantly influences the RTT measurements and their variability. While Scenario 1 (ARP poisoning) may be suitable in certain situations, scenarios 2 and 3 demonstrate more controlled and efficient approaches, with Scenario 2 having the least impact on average RTT and variability. The observed differences highlight the importance of considering the specific requirements and priorities of a given network experiment or application when selecting the attack implementation strategy.

4.3.2 SDN controller CPU usage interference analysis

In addition to evaluating the RTT between PLC1 and PLC3, we conducted an analysis of the CPU utilization

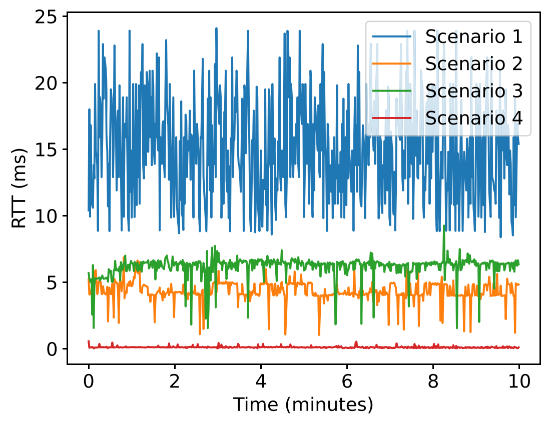


Fig. 8 RTT measurements in the considered four different scenarios

Table 1 RTT measurements between PLC1 and PLC3 in different scenarios

		Scenarios			
		4 (Normal)	1	2	3
RTT (ms)	avg	0.110	15.029	4.325	6.163
	stdev	0.062	3.776	0.713	0.796
	min	0.033	8.384	1.031	1.542
	max	0.560	24.084	6.865	9.255

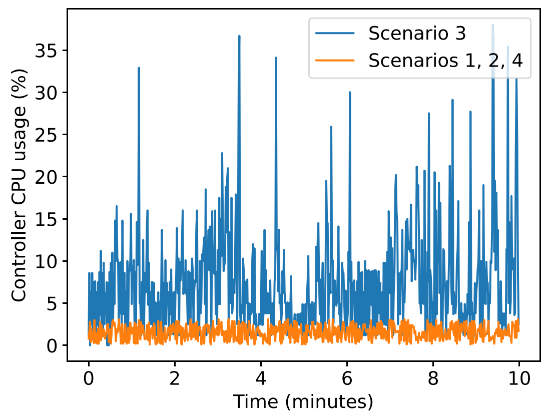


Fig. 9 CPU usage of the SDN controller in the different scenarios

within the SDN controller during each 10-minute experiment. Figure 9 provides a visual representation of the observed CPU utilization trends. We have grouped scenarios 1, 2, and 4 together because the role of the SDN controller in these scenarios is limited to network provisioning, which includes installing flow rules and establishing communications.

In scenarios 1, 2, and 4, CPU utilization remains consistently low. This is due to the fact that once the flow rules are installed in the OpenFlow switch, the traffic is processed within the data plane without requiring the involvement of the SDN controller.

In contrast, Scenario 3 shows a more intensive CPU usage pattern. The attacker's specific implementation forces all traffic to be forwarded from the switch to the SDN controller, resulting in a significant increase in CPU utilization. This distinction highlights the impact of malicious activity on the resource utilization of the SDN controller.

In conclusion, our evaluation shows that the CPU utilization of the SDN controller is significantly affected by the attacker implementation location. While network packets processing in the data plane maintain low CPU utilization, the introduction of an adversarial element in the SDN controller, as seen in Scenario 3, causes a significant increase in CPU usage. This finding highlights the importance of considering this resource demand in experiments where the CPU usage of the SDN controller is critical.

5 Conclusion and future work

Testing and evaluating the resilience and security of Industrial Control Systems (ICSs) pose significant challenges. In order to address these challenges, researchers commonly employ testbeds such as MiniCPS, which are designed for conducting research and security testing in SDN-based industrial networks. This study aims to analyze several crucial features of a MiniCPS-based testbed that are essential for conducting scientifically rigorous experiments. Specifically, it focuses on network performance, replicability, and the impact of different attacker implementations. The experimental results demonstrate the following key findings regarding MiniCPS: (1) provides a consistent performance within established confidence intervals, (2) exhibits a high degree of reproducibility in experiments, although it is recommended to use the same hardware for experiments with high network load, and (3) enables the use of different methods for implementing attackers, thereby influencing the realism of experiments and the resulting outcomes.

As part of future directions, this research aims to expand its scope by analyzing the impact of employing various measurement tools, SDN controllers, and software-based switches. Additional investigations will further enhance the understanding of the testbeds capabilities and provide valuable insights into the broader applicability of MiniCPS.

Acknowledgements This work has been developed by the Intelligent Systems for Industrial Systems group, supported by the Department of Education, Language Policy, and Culture of the Basque Government (IT1676-22). This project has received support from the Department of Economic Development, Sustainability, and Environment of the Basque Government, within the ELKARTEK 2023 call, under the BEACON project (registration number 2023RTE00242510).

Declarations

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

References

1. Stouffer, K., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V., & Lightman, S. (2023). *Guide to operational technology (ot) security*. National Institute of Standards and Technology: Gaithersburg, MD, USA.
2. Giraldo, J., Sarkar, E., Cardenas, A. A., Maniatakos, M., & Kantarcioglu, M. (2017). Security and privacy in cyber-physical systems: A survey of surveys. *IEEE Design & Test, 34*(4), 7–17. https://doi.org/10.1109/MDAT.2017.2709310
3. Dragos: 2022 ics/ot cybersecurity year in review. Technical report (2022)
4. Alladi, T., Chamola, V., & Zeadally, S. (2020). Industrial control systems: Cyberattack trends and countermeasures. *Computer Communications, 155*, 1–8. https://doi.org/10.1016/j.comcom.2020.03.007
5. Nourian, A., & Madnick, S. (2018). A systems theoretic approach to the security threats in cyber physical systems applied to stuxnet. *IEEE Transactions on Dependable and Secure Computing, 15*(1), 2–13. https://doi.org/10.1109/TDSC.2015.2509994
6. Xiang, Y., Wang, L., & Liu, N. (2017). Coordinated attacks on electric power systems in a cyber-physical environment. *Electric Power Systems Research, 149*, 156–168. https://doi.org/10.1016/j.epsr.2017.04.023
7. Boyes, H., Hallaq, B., Cunningham, J., & Watson, T. (2018). The industrial internet of things (iiot): An analysis framework. *Computers in Industry, 101*, 1–12. https://doi.org/10.1016/j.compind.2018.04.015
8. Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K., & Meskin, N. (2020). Cybersecurity for industrial control systems: A survey. *Computers & Security, 89*, 101677. https://doi.org/10.1016/j.cose.2019.101677
9. Iturbe, M., Garitano, I., Zurutuza, U., Uribeetxeberria, R.: Visualizing network flows and related anomalies in industrial networks using chord diagrams and whitelisting. In: VISIGRAPP (2: IVAPP), pp. 101–108 (2016)
10. Xu, H., Yu, W., Griffith, D., & Golmie, N. (2018). A survey on industrial internet of things: A cyber-physical systems perspective. *IEEE Access, 6*, 78238–78259. https://doi.org/10.1109/ACCESS.2018.2884906
11. Haleplidis, E., Pentikousis, K., Denazis, S., Salim, J. H., Meyer, D., & Koufopavlou, O. (2015). Software-Defined Networking (SDN): Layers and Architecture Terminology. *RFC Editor*. https://doi.org/10.17487/RFC7426
12. Etxezarreta, X., Garitano, I., Iturbe, M., Zurutuza, U.: Software-defined networking approaches for intrusion response in industrial control systems: A survey. International Journal of Critical Infrastructure Protection, 100615 (2023). doi: https://doi.org/10.1016/j.ijcip.2023.100615
13. Etxezarreta, X., Garitano, I., Iturbe, M., & Zurutuza, U. (2023). Low delay network attributes randomization to proactively mitigate reconnaissance attacks in industrial control systems. *Wireless Networks*. https://doi.org/10.1007/s11276-022-03212-5
14. Sándor, H., Genge, B., Szántó, Z., Márton, L., & Haller, P. (2019). Cyber attack detection and mitigation: Software defined survivable industrial control systems. *International Journal of Critical Infrastructure Protection, 25*, 152–168. https://doi.org/10.1016/j.ijcip.2019.04.002
15. Kim, S., Eun, Y., & Park, K.-J. (2021). Stealthy sensor attack detection and real-time performance recovery for resilient cps. *IEEE Transactions on Industrial Informatics, 17*(11), 7412–7422. https://doi.org/10.1109/TII.2021.3052182
16. Cai, T., Jia, T., Adepu, S., Li, Y., Yang, Z.: Adam: An adaptive ddos attack mitigation scheme in software-defined cyber-physical system. IEEE Transactions on Industrial Informatics, 1–12 (2023). doi: https://doi.org/10.1109/TII.2023.3240586
17. Antonioli, D., Tippenhauer, N.O.: Minicps: A toolkit for security research on cps networks. In: Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security And/or PrivaCy. CPS-SPC '15, pp. 91–100. Association for Computing Machinery, New York, NY, USA (2015). doi: https://doi.org/10.1145/2808705.2808715.
18. Conti, M., Donadel, D., & Turrin, F. (2021). A survey on industrial control system testbeds and datasets for security research. *IEEE Communications Surveys & Tutorials, 23*(4), 2248–2294. https://doi.org/10.1109/COMST.2021.3094360
19. Kaur, K., Singh, J., Ghumman, N.S.: Mininet as software defined networking testing platform. In: International Conference on Communication, Computing & Systems (ICCCS), pp. 139–42 (2014)
20. Oliveira, R.L.S., Schweitzer, C.M., Shinoda, A.A., Prete, L.R.: Using Mininet for Emulation and Prototyping Software-Defined Networks. doi: https://doi.org/10.1109/ColComCon.2014.6860404
21. Foundation, O.N.: OpenFlow Switch Specification, Version 1.5.1. https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf Accessed 2023-03-14
22. Reinartz, C., & Enevoldsen, T. T. (2022). pytep: A python package for interactive simulations of the tennessee eastman process. *SoftwareX, 18*, 101053. https://doi.org/10.1016/j.softx.2022.101053
23. Dehlaghi-Ghadim, A., Balador, A., Moghadam, M. H., Hansson, H., & Conti, M. (2023). Icssim - a framework for building industrial control systems security testbeds. *Computers in Industry, 148*, 103906. https://doi.org/10.1016/j.compind.2023.103906
24. Murillo, A., Taormina, R., Tippenhauer, N., Galelli, S.: Co-simulating physical processes and network data for high-fidelity cyber-security experiments. In: Sixth Annual Industrial Control System Security (ICSS) Workshop. ICSS 2020, pp. 13–20. Association for Computing Machinery, New York, NY, USA (2021). doi: https://doi.org/10.1145/3442144.3442147.
25. Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., et al.: The design and implementation of open vswitch. In: 12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15), pp. 117–130 (2015)
26. Mathur, A.P., Tippenhauer, N.O.: Swat: a water treatment testbed for research and training on ics security. In: 2016 International Workshop on Cyber-physical Systems for Smart Water Networks

(CySWater), pp. 31–36 (2016). doi: https://doi.org/10.1109/CySWater.2016.7469060
27. Gómez, Á. L. P., Maimó, L. F., Celdrán, A. H., Clemente, F. J. G., Sarmiento, C. C., Masa, C. J. D. C., & Nistal, R. M. (2019). On the generation of anomaly detection datasets in industrial control systems. *IEEE Access, 7*, 177460–177473.
28. Barbieri, G., Conti, M., Tippenhauer, N.O., Turrin, F.: Assessing the use of insecure ics protocols via ixp network traffic analysis. In: 2021 International Conference on Computer Communications and Networks (ICCCN), pp. 1–9 (2021). doi: https://doi.org/10.1109/ICCCN52240.2021.9522219
29. Huang, D., Shi, X., & Zhang, W.-A. (2020). False data injection attack detection for industrial control systems based on both time- and frequency-domain analysis of sensor data. *IEEE Internet of Things Journal, 8*(1), 585–595.
30. Aoudi, W., Iturbe, M., Almgren, M.: Truth will out: Departure-based process-level detection of stealthy attacks on control systems. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. CCS '18, pp. 817–831. Association for Computing Machinery, New York, NY, USA (2018). doi: https://doi.org/10.1145/3243734.3243781.
31. Giraldo, J. A., El Hariri, M., & Parvania, M. (2022). Moving target defense for cyber-physical systems using iot-enabled data replication. *IEEE Internet of Things Journal, 9*(15), 13223–13232. https://doi.org/10.1109/JIOT.2022.3144937

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Xabier Etxezarreta received the bachelor's degree in Computer Engineering in 2019 and the master's degree in Data Analytics, Cybersecurity and Cloud Computing in 2021, both at Mondragon Unibertsitatea, Arrasate-Mondragón, Spain. During the course of his studies, he has participated in numerous research projects related to artificial intelligence and multi-objective optimization. He is currently pursuing his doctoral studies, researching on intrusion detection and response techniques based on SDN for industrial control systems.

Iñaki Garitano is a researcher/lecturer in the Intelligent Systems for Industrial Systems research group, and member of the Data Analysis and Cybersecurity research area of Mondragon Unibertsitatea. He holds a PhD titled Behavioral Modeling for Anomaly Detection in Industrial Control Systems, and MSc in Telecommunication Engineering. Before joining Mondragon Unibertsitatea in 2015, he worked in Universitetssenteret på Kjeller (UNIK) as postdoctoral researcher. In addition, he is the CTO of the Basic Internet Foundation. His main research interest is in the area of Cybersecurity applied to Industrial Automation and Control Systems and Information and Communication Networks including Internet of Things, and Application Container technologies. He currently participates in several European, national and regional level public funding projects.

Mikel Iturbe received the M.Sc. degree in ICT security from the Open University of Catalonia, Barcelona, Spain, in 2013, and the Ph.D. degree from Mondragon Unibertsitatea, Arrasate-Mondragón, Spain, in 2017, where he worked on data-driven intrusion detection in industrial networks. He is a Lecturer and a Researcher with Mondragon Unibertsitatea. He is currently part of the Data Analysis and Cybersecurity Research Group. His main research interest is related to cybersecurity, primarily in the industrial sector. The main lines he works on are industrial control system security, embedded security, and software security. He also works in exploring novel data-driven applications for cybersecurity.

Urko Zurutuza is the principal investigator of the Intelligent Systems for Industrial Systems research group, and coordinator of the Data Analysis and Cybersecurity research team. He obtained his PhD in January 2008 at Mondragon Unibertsitatea, in collaboration with the Zürich IBM Research Lab. His research interests revolve around applications of Machine Learning to real world problems, and specially Network Security. He has published more than 25 articles in high impact journals, more than 44 publications in blind peer-reviewed conferences, edited 3 books (2 of them as conference proceedings), and coauthored 9 book chapters. He serves as Program Committee member in conferences such as RAID, DIMVA or SECRYPT, and as Steering Committee member in RAID and DIMVA.