Towards a Testbed for Critical Industrial Systems: SunSpec Protocol on DER Systems as a Case Study

Esteban Damián Gutiérrez Mlot*
CIRCE Centro Tecnológico
Zaragoza, Spain
edgutierrez@fcirce.es

Jose Saldana
CIRCE Centro Tecnológico
Zaragoza, Spain
jmsaldana@fcirce.es

Ricardo J. Rodríguez
University of Zaragoza
Zaragoza, Spain
rjrodriguez@unizar.es

***Abstract*—Control systems in critical infrastructures have usually been considered safe as long as they were totally isolated from the outside world. However, today many of these systems are connected to the outside world and use open and standardized communication protocols designed with little or no security measures, such as Modbus or its variants such as SunSpec, widely used in Distributed Energy Resources (DER) systems. This work-in-progress presents a testbed based on open source tools and docker containers to easily evaluate cybersecurity measures against cyberattacks on critical infrastructures without affecting their availability. This testbed is validated in a use case based on the SunSpec protocol on DER systems to detect person-in-the-middle attacks, and is implemented on a hardware-constrained appliance dubbed *Energy Box*.**

***Index Terms*—cybersecurity, critical infrastructure, testbed**

I. Introduction

The Industry 4.0 paradigm arises from the digitalization of industry, with the aim of gradually decentralizing the production control system. Although this evolution brings many advantages (such as improved productivity and efficiency, better flexibility and agility, and increased profitability, among others), it also presents some drawbacks as a result of the adoption of IT equipment and procedures in the world of industrial automation control systems. Difficulties such as the growing connectivity and remote work [1] bring to the table several challenges related to the cybersecurity of industry assets, especially in the field of critical infrastructure [2].

In this regard, critical infrastructures have particularly demanding cybersecurity requirements, especially in terms of availability [3]. However, the need to keep critical systems running, avoiding long maintenance stops, and the long lifespan of devices already installed, have favored the use of legacy equipment, which is frequently interconnected with modern devices using old and insecure protocols.

Modbus is an example of these legacy communication protocols. Originally conceived for use with programmable logic controllers in 1979 [4], today it has become a *de facto* standard communication protocol for industrial electronic devices. Thus, a Modbus variant that works over TCP/IP (called Modbus/TCP) is widely used in Distributed Energy Resources (DER) systems [5]. Recognizing both the popularity of Modbus and the lack of standardized information models, in 2009 the SunSpec Alliance embarked on a project to define standard information models for DER devices [6], creating the SunSpec Modbus specification. This standard allows interoperability between the components of the DER system, defining the appropriate content of the Modbus registers and a series of functions to manage them.

In DER systems, electrical resources are directly connected to the public distribution grid, with a low power range between 3 kW and 50 MW, and are generally located close the end user (for instance, a home or business), such as backup generators, photovoltaic, and batteries. DER systems require extensive data sharing and large communication systems making them highly exposed to cyberattacks [7]. In addition, as DER systems are part of the national electrical systems, they are considered critical infrastructures. Performing penetration tests or testing the cybersecurity of new devices in these environments it is very difficult or even impossible. Therefore, it is important to have an easily deployable platform, on which rigorous, transparent, and replicable tests of the cybersecurity of systems can be carried out.

The latest trends in cybersecurity countermeasures consider the deployment of Intrusion Detection Systems (IDS) in the network to automatically detect potential threats. Working at the network and transport layers, an IDS monitors and analyzes communications in search of anomalies or signatures, which may be signs of violations or threats to computer cybersecurity policies. An IDS can be a powerful defensive tool for critical infrastructures due to their ability to protect against attacks that are unknown and use completely new attack vectors (known as *zero-days attacks*) [8].

In this work-in-progress, we have created a testbed to evaluate the development of cybersecurity mechanisms for the protection of DER devices against attacks. Then, as a case study, the penetration testing platform is used in a controlled environment to assess the security of communication systems and protocols against a controller of a lithium-ion battery bank that is exposed to a conventional attack. Finally, an IDS is presented that has been deployed on an embedded system (dubbed *Energy Box* [9]) and is based on open source tools. This IDS is currently under development.

*Corresponding author. The research of E. D. Gutiérrez and J. Saldana has been supported in part by *the European Union's H2020 Research and Innovation programme* under Grant Agreement No 864459 (UE-19-TALENT-864459). The research of R J. Rodríguez was supported in part by the Aragonese Government under *Programa de Proyectos Estratégicos de Grupos de Investigación* (project reference T21-20R).

The rest of this paper is organized as follows. Section II reviews related works. Section III presents the proposed testbed and a demonstration of its use. Finally, Section IV concludes this work and lays out the next steps.

II. Related Work

Maynard et al. proposed in [10] to replicate a Supervisory Control And Data Acquisition (SCADA) network using virtual machines. This testbed included the implementation of several industrial protocols, such as IEC104, OPC-UA, Modbus-TCP, and IEC61850. Based on open source libraries, it is publicly and freely available on GitHub [11]. However, it has several limitations. First, it does not provide a way to emulate the SunSpec Modbus protocol. In addition, it is virtualized on top of Virtualbox, which emulates all the hardware in a computer, resulting in higher execution overhead and configuration effort compared to Docker containers [12].

Khan et al. [13] address the generation of a SCADA testbed based on lightweight containers with reconnaissance and Person-in-The-Middle (PiTM) as cyberattacks. The testbed uses open source software such as Docker, Suricata, and Conpot to run and emulate the devices. However, the work is based on the devices provided by Conpot, which do not follow SunSpec Modbus specifications. In addition, the source code is not publicly available, which makes it difficult to modify the testbed.

Ekisa et al. build in [14] a comprehensive virtualized ICS testbed, based on open source tools and Linux containers, focused on the integration of a 3D visualization tool to simulate the physical consequences of a successful cyberattack on an ICS. Although it promises to deliver a powerful, portable, and flexible testbed, it is at a very early stage, so no technical or implementation information is currently available.

Mahrenholz et al. [15] proposed an anomaly detection method with user feedback algorithm for application in OT networks, although it is focused only on wireless networks.

Unlike these works, our industrial testbed focuses on industrial control systems (in particular, DER systems) and is lightweight, open source, and freely available. Furthermore, our testbed is compliant with the SunSpec Modbus specification. It also facilitates the development and testing of algorithms for defense against cyberattacks. In this regard, we deliberately looked for a virtualized environment capable of easily reproducing different network configurations.

III. A Testbed to Evaluate Cybersecurity in DER systems: Case Study

This section presents the testbed and a use case where we evaluated it. Our testbed is a platform specifically developed to conduct different types of experiments to assess the cybersecurity of DER devices. We first introduce the requirements covered by our testbed and the elements that made up the use case for the testbed. Finally, we present the use case, showing a particular type of attack and its detection.

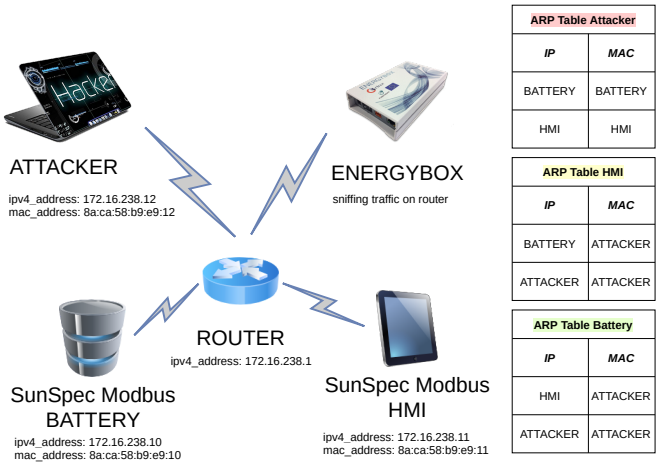

Figure 1. Elements that made up the testbed.

A. Description of the Testbed

Despite the implementation of the best security practices [3], many DER systems still have vulnerabilities that go undetected during cybersecurity assessments. Furthermore, the need for constant operation often makes it difficult to stop these systems and deploy and assess the appropriate cybersecurity countermeasures. Our testbed provides a way to comprehensively test protection measures designed for DER systems.

The goal of the testbed is to allow the generation of attack data sets for DER systems. Rodofile et al. defined ten requirements for creating specific attack data sets for ICS [16]: 1) capable of parsing protocol messages; 2) capable of replicating the protocol stack; 3) capable of sniffing local network traffic; 4) inject anomalous protocol messages into the network; 5) modify protocol message data in real time; 6) provide a protocol master service for masquerading; 7) provide a protocol slave service for masquerading; 8) provide network discovery/reconnaissance to target applications; 9) capable of replaying previous protocol messages; and 10) capable of flooding. Our testbed covers all these requirements.

Our testbed has been built for an energy storage system that uses the communication and control protocol SunSpec model number 803. This model specifies how the Modbus registers are configured for a Lithium-ion Battery Bank. This scenario has been selected from a use case used in the EU H2020 TALENT project [17], and is a battery application commonly found in DER monitoring networks, where the battery is routinely queried and returns a value.

Figure 1 shows the network diagram for this use case. The example testbed consists of three virtual nodes and one physical appliance. Virtual nodes are created using three configuration profiles (*Human Machine Interface* (HMI), *Battery*, and *Attacker*) and deployed on a physical machine using docker technology. Regarding the physical device, Energy Box is a multipurpose data concentrator for the operation of Smart Grids. It contains several communication interfaces and an integrated CPU mainly for capturing and storing information,

running network algorithms, and controlling electrical installations. We are currently working on a virtualized version of this node to make the testbed scalable and easily extensible with techniques like Hardware-in-the-loop. The source code is available in the repository created on GitHub [18].

Once all virtual nodes are deployed, the HMI initiates communication with the battery using the SunSpec Modbus protocol and polls the node for information at a specified polling interval. All nodes are connected using docker engine, which exposes their network interface to the Energy Box. Hence, the docker engine acts as a "router," i.e., it is used for routing packets and assigning IP addresses.

The *HMI* node has a web interface that can be accessed through a browser. The interface polls the battery for data every 5 seconds and displays three different battery measurements provided by SunSpec model considered in this use case: average voltage; average current; and average temperature. The *Battery* node simulates a set of Lithium-ion batteries operating under normal conditions.

Finally, the *Attacker* node is deployed on a machine running Kali Linux [19], a Debian-based Linux distribution designed for digital forensics and penetration testing that has a rich collection of software tools for different tasks related to cybersecurity assessment.

B. Simulated Attack

In this testbed, the *attacker* will primarily perform a PiTM attack. We assume that they are already within the LAN network. In this attack, the communication between the HMI and the *Battery* is intercepted, and the messages sent from the battery to the HMI are modified by replacing the original temperature values with fake values created by the attacker.

To perform this attack successfully, the attacker will follow certain steps. First, they have to scan the network. To do this, an active and passive scan is carried out to identify possible attack surfaces, vulnerabilities, services, protocols, addresses and ports that can be used to compromise systems. The information that can be obtained about the network includes source and destination addresses, ports used, and control and configuration commands. In addition, information about industrial devices such as manufacturer, model number, allowed commands, and memory maps can also be obtained.

This information is processed by the attacker to prepare the next step of the attack. After learning the IP address and the source and destination ports, the attacker connects to the devices, making them believe that they are directly communicating with each other. To do this, they perform an ARP spoofing attack [20]. In this attack, forged ARP packets are sent to modify the ARP tables of network devices. These tables store the ARP addresses that are most frequently needed during communication. In this way, the attacker's MAC address is associated with the IP addresses of the victims. Thus, the packets that are sent between two machines have the correct destination IP address but with the attacker's MAC address, so the network switch receives them and sends them to the physical port to which the attacker is connected.

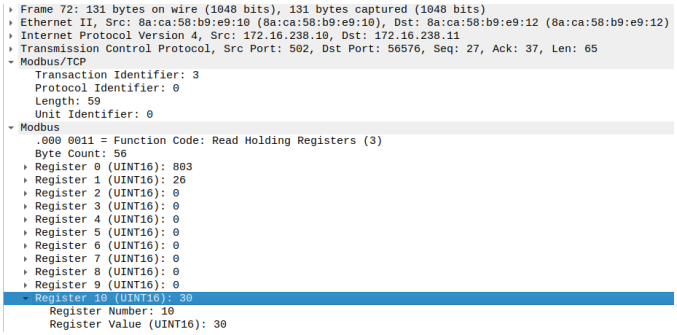

Figure 2. Snippet of Wireshark output

Therefore, the attacker can carry out different attacks such as: *sniffing*, which allows the attacker to obtain all the communication traffic and search for specific information such as emails, passwords, or web cookies to carry out impersonation or identity theft attacks; *proxy*, allowing packets to be modified to change protocol information, terminate connections, remove packets, etc.; or *packet injection*, inserting network packets that change the expected behavior of industrial devices.

The penetration tests performed on the battery controller are detailed below. Note that this attack scenario assumes that the attacker has access to the local network or the attack originates within the local network through a legitimate network node controlled by the attacker, who can execute commands remotely.

As explained before, in this use case the commands used by the attacker are executed on a node with the Kali Linux operating system. To perform the network scan, the attacker follows these steps:

- Identification of the devices on the local network using `nmap` (for instance, `nmap -sP 172.16.238.0/24`).
- Identification of open ports in the battery controller, also using `nmap` (for instance, `nmap -Pn -p 502 172.16.238.10`).
- Identification of Modbus addresses, ports, commands, and memory using data-network packet analyzers such as `tcpdump` or `Wireshark`. A snippet of a Wireshark capture is shown in Figure 2. As shown, the connections between the battery and the HMI, as well as control commands and memory maps can be identified.

To perform the ARP spoofing attack, the attacker uses the `arpspoof` tool with the IP addresses of the previously detected devices as input parameters (for instance, `arpspoof -t 172.16.238.10 172.16.238.11`).

After making this attack, the attacker begins to receive information. This information is quite useful to execute later attacks because it allows knowing the addresses, ports, commands and memory maps of the battery and the HMI.

To complete the PiTM attack, the attacker has an interest in changing a battery value (specifically, the average temperature value). This value is found in register number 10 of the Lithium-ion Battery Bank model used in this scenario. In particular, the temperature value will be changed from 30 to

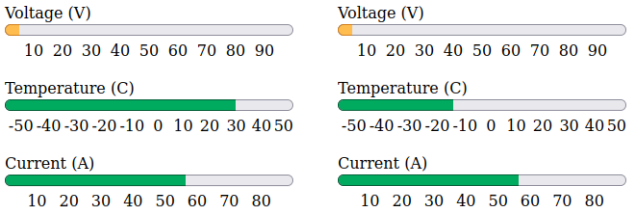

Figure 3. HMI original (left-hand side) and fake values (right-hand).

-10 using Python3 and `Scapy`, which is a powerful interactive packet manipulation library for Python.

When the Python script runs, it inspects each packet received from the *Battery*, looking for the register 10. If found, it will change its associated value to the new value. After changing it, the checksum of the packet will be recalculated (so that it is not discarded for transmission errors) and finally sent to the HMI as if it were sent by the *Battery* itself.

To check if the attack is successful, we are going to use the HMI device itself, which has a web interface that shows three values (voltage, temperature, and intensity) received from the battery, updated every 3 seconds. Figure 3 shows two screenshots of the HMI device web interface before and after the attack.

C. Detection of the PiTM Attack

Detection of PiTM attacks is performed on the embedded Energy Box device, running an open source network IDS called Snort [21]. This IDS uses a series of rules that help define malicious network activity and are used to find packets that match the rules and to generate the corresponding alerts.

Specific rules are created to detect ARP spoofing attacks. In this use case, the rules are defined at the time of placing the Energy Box as a network element, associating each existing device in the network with its current MAC address. In this way, when network traffic with unexpected MAC and IP addresses is detected, the system generates an alarm in the administrator console. Listing 1 shows an example of the alarm generated by Snort in this use case. As can be seen, the attack is successfully detected and the user is informed about the IP address of the victim (the *Battery*) and the attacker.

```
02/06-19:05:53.997061 [**] [1:472:4] ICMP
    redirect host [**] [Classification:
    Potentially Bad Traffic] [Priority: 2] {
    ICMP} 172.16.238.12 -> 172.16.238.10
```

Listing 1. Snort alarm example

IV. Discussion and Future Work

In this work-in-progress, we have presented a testbed created to assess the cybersecurity of DER systems against attacks on SunSpec-based communication devices. Using our testbed in a controlled environment composed of an HMI and a battery controller, we have shown that the industrial system is vulnerable to the same attacks as conventional information systems. In addition, we have used the embedded Energy Box system to deploy an open source IDS that allows us to mitigate attacks by issuing alarms when threat events are detected.

As future work, our goal is to expand the testbed with a new dataset created from the testing of new use cases related to critical infrastructures. We also intend to improve the rules-based traffic anomaly detector to use dynamic algorithms (using artificial intelligence) optimized to run on embedded devices with limited hardware resources instead of static rules.

References

[1] T. Menze, "The State of Industrial Cybersecurity in the Era of Digitalization," [Online, https://ics.kaspersky.com/media/Kaspersky_ARC_ICS-2020-Trend-Report.pdf, 2020, accessed on April 7, 2022.

[2] E. D. Knapp, *Industrial network security: securing critical infrastructure networks for smart grid, scada, and other industrial control systems*, 2nd ed. Waltham, MA: Elsevier, 2014.

[3] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to Industrial Control Systems (ICS) Security," National Institute of Standards and Technology, Tech. Rep., NIST SP 800-82 Revision 2.

[4] Modbus Organization, Inc., "Modbus FAQ," [Online, https://modbus.org/faq.php], accessed on April 7, 2022.

[5] R. Chauhan and K. Chauhan, *Distributed Energy Resources in Microgrids: Integration, Challenges and Optimization*. Elsevier, 2019.

[6] SunSpec Alliance, "SunSpec Energy Storage Models," [Online, https://sunspec.org/sunspec-energy-storage-model-description], accessed on April 9, 2022.

[7] A. Sundararajan, A. Chavan, D. Saleem, and A. Sarwat, "A Survey of Protocol-Level Challenges and Solutions for Distributed Energy Resource Cyber-Physical Security," *Energies*, vol. 11, no. 9, p. 2360, Sep. 2018.

[8] R. Leszczyna, *Cybersecurity in the Electricity Sector: Managing Critical Infrastructure*. Cham: Springer International Publishing, 2019.

[9] CIRCE Foundation. CIRCE Building / Campus Río Ebro. Zaragoza. Spain, G. Fernández, H. Bludszuweit, J. Torres, J. Almajano, I. Machin, J. Sanz, R. Serrano, and M. Garín, "Optimal demand-side management with a multi-technology battery storage system," *Renewable Energy and Power Quality Journal*, vol. 1, pp. 345–349, Apr. 2018.

[10] P. Maynard, K. McLaughlin, and S. Sezer, "An Open Framework for Deploying Experimental SCADA Testbed Networks," Aug. 2018.

[11] P. Maynard, "ICS TestBed Framework," [Online, https://github.com/PMaynard/ICS-TestBed-Framework], accessed on April 15, 2022.

[12] W. Elmenreich, P. Moll, S. Theuermann, and M. Lux, "Making computer science results reproducible - A case study using Gradle and Docker," *PeerJ Prepr*, vol. 6, p. e27082, 2018.

[13] M. Khan, O. Rehman, I. M. H. Rahman, and S. Ali, "Lightweight Testbed for Cybersecurity Experiments in SCADA-based Systems," in *2020 International Conference on Computing and Information Technology (ICCIT-1441)*. Tabuk, Saudi Arabia: IEEE, Sep. 2020, pp. 1–5.

[14] C. Ekisa, D. O. Briain, and Y. Kavanagh, "An Open-Source Testbed to Visualise ICS Cybersecurity Weaknesses and Remediation Strategies - A Research Agenda Proposal," in *2021 32nd Irish Signals and Systems Conference (ISSC)*. Athlone, Ireland: IEEE, Jun. 2021, pp. 1–6.

[15] D. Mahrenholz, G. Lukas, J. Paffrath and K. Detken, "Detecting Low-Level Attacks on Wireless OT Networks," in *IDAACS-SWS*. IEEE, 2020, pp. 1–6.

[16] N. R. Rodofile, K. Radke, and E. Foo, "Framework for SCADA cyber-attack dataset creation," in *Proceedings of the Australasian Computer Science Week Multiconference*. ACM, Jan. 2017, pp. 1–10.

[17] European Union's Horizon 2020, "TALENT project," [Online, https://talentproject.eu], accessed on April 27, 2022.

[18] G. M. Esteban Damián, "DER Testbed," [Online, https://github.com/esguti/TestbedDER], accessed on April 27, 2022.

[19] OffSec Services, "Kali Linux — Penetration Testing and Ethical Hacking Linux Distribution," [Online, https://www.kali.org], accessed on April 27, 2022.

[20] Sumit Kumar and Shashikala Tapaswi, "A centralized detection and prevention technique against ARP poisoning," in *CyberSec*. IEEE, 2012, pp. 259–264.

[21] Snort, "Snort - Network Intrusion Detection and Prevention System," [Online, https://www.snort.org], accessed on April 29, 2022.