FLEP-SGS^2: a Flexible and Low-cost Evaluation Platform for Smart Grid Systems Security

Charalambos Konstantinou[*], **Marios Sazos**[†] **Michail Maniatakos**[†]
[*]FAMU-FSU College of Engineering, Center for Advanced Power Systems, Florida State University
[†]Center for Cyber Security, New York University Abu Dhabi
E-mail: ckonstantinou@fsu.edu, {marios.sazos, michail.maniatakos}@nyu.edu

***Abstract*—As digital emerging technologies are being widely integrated in the grid infrastructure, security incidents to these systems could have severe outcomes. In order to understand security vulnerabilities of both field devices and communication protocols used in the newly formed smart grid, testbed frameworks are developed to conduct vulnerability studies. To date, efforts in grid security to develop such testbeds have resulted in costly, non-modular, and proprietary-based environments. In this paper, we present FLEP-SGS^2: a flexible, scalable, multilayer, and low-cost platform to examine security scenarios in smart grid. Power systems simulation tools are integrated with open-source software and code running on Raspberry Pis. FLEP-SGS^2 allows to investigate the performance of the testbed components and evaluate different cyber-attack scenarios on the power system operation while providing a user-friendly interface.**

***Index Terms*—Smart grid, cybersecurity, hardware-in-the-loop, testbed.**

I. Introduction

As information and communication technologies become more embedded into industrial control systems, physical processes are tightly coupled with computation and networking components. These cyber-physical systems (CPSs) are rapidly applied in critical infrastructure such as smart grids, desalination plants, and chemical factories. This integration has reduced the isolation of physical systems and increased their connectivity to the outside world. In addition, control units often incorporate commercial off-the-shelf (COTS) products. This opens up a new range of opportunities for attackers in CPSs and particularly in the energy sector [1].

Due to the complexity and interconnections within smart grid, many attacks use a multilayer threat model. For instance, databases can be connected to computers or other databases located at the business network. An attacker able to access the database on the business network could thereafter exploit the communication channel between the two networks and bypass the security mechanisms protecting the control system.

Since CPSs in real deployments are challenged by motivated and skilled attackers, evaluating a system's robustness in realistic scenarios is of paramount importance. For example, field equipment might be subject to unpredictable conditions after a cyber-attack and cause physical damage to the infrastructure or even to human health. Also, there are often restrictions and concerns in validating the interaction of devices and protocols in a production environment. Thus, the testbed approach is widely applied to perform both modeling and validation in a scaled representation of the system.

This paper presents FLEP-SGS^2, a smart grid testbed designed with low-cost hardware and in which simulation tools are integrated with open-source code. Its current implementation includes a hardware-in-the-loop (HIL) overcurrent protection relay and covers the state estimation process of supervisory control and data acquisition (SCADA) systems. Monitoring and control information is exchanged via Modbus/TCP communication protocol. Multiple hardware components and software tools can be connected to FLEP-SGS^2, making it reconfigurable and interoperable with other systems as well as easily extendable to other CPS cases.

Assessment environments are often difficult to build due to the associated cost and required expertise. FLEP-SGS^2 allows to easily replicate real-world grid behavior. Researchers from various communities can effortlessly develop FLEP-SGS^2 and use it in their own research; and at the same time contribute to the security concerns faced by power system utilities.

The remainder of the paper is organized as follows. Section II presents the background for this work as well as the related efforts on testbed development. Section III focuses on the setup and security design of FLEP-SGS^2. Experiments are carried out in Section IV. Finally, Section V concludes this paper.

II. Background

The function of protection relays is to limit or prevent damage due to overloads and faults. According to North American Electric Reliability Council (NERC), most disturbances are attributed to faulty operation of relays [2]. In our testbed, we implement a HIL environment of an overcurrent relay with features of instantaneous, time definite and inverse definite minimum time (IDMT) characteristics [3].

Within energy management systems (EMS), state estimation is a key function for determining the system operation based on the real-time network [4]. The model is extracted from analog and status measurement by filtering errors and leveraging redundant data. The collected information is transmitted through the SCADA network to the control center.

A. Overcurrent Relays

Overcurrent relays operate or pick up when the monitoring current exceeds a predetermined value. In that case, the relay

TABLE I: Values of α and K determine the degree of inverse in the IDMT curves.

Type of curve	α	K
Normal Inverse	0.02	0.14
Very Inverse	1.0	13.5
Extremely Inverse	2.0	80.0
Long-time Inverse	1.0	120.0

issues a trip signal (with or without delay) to the breaker. Overcurrent relays are classified into three groups [3]:

1) *Instantaneous Overcurrent Relay (Definite Current)*: operates instantaneously; its operation criterion is only the current magnitude (without any time delay).
2) *Definite Time Overcurrent Relay*: requires two conditions for operation. Current must exceed the setting value and the fault must be continuous at least a time equal to time setting.
3) *Inverse Definite Minimum Time (IDMT) Overcurrent Relay*: has an inverse time characteristic. The operation time is inversely proportional to the fault current. The current/time characteristics may vary according to the tripping time required and the characteristics of other devices. For these purposes, IEC 60255 defines a number of standard characteristics:

$$T = \frac{K}{\frac{I}{Is}^{\alpha} - 1} \times TMS \tag{1}$$

where T is the relay operation time, I is the measured current, I_s is the current setpoint, K is a constant, $\alpha > 0$ is a constant representing inverse time type, and TMS is the time multiplier setting of the tripping time ($0.1 \leq TMS \leq 1.0$). By varying α and K, four IDMT curves are available. Table I shows theses values and the corresponding curve.

B. Nonlinear State Estimation

In AC state estimation, the algorithmic routine utilizes a nonlinear relationship between the system and measurements:

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{e} \tag{2}$$

$\boldsymbol{x} \in \mathbb{R}_{n\times 1}$ is the vector of state variables, and $\boldsymbol{z} \in \mathbb{R}_{m\times 1}$ is the vector of measured values, where m is the number of meter measurements ($m \geq n$). $\boldsymbol{e} \in \mathbb{R}_{m\times 1}$ is the vector of measurement errors and $\boldsymbol{h}()$ is the nonlinear function determined by the structure of the system. The Jacobian matrix $\boldsymbol{J}_h$ of $\boldsymbol{h}(\boldsymbol{x})$ provides the information with regards to which measurement is dependent on which state variable.

The procedure for solving the nonlinear problem is via iterative techniques such as the Honest or Dishonest Gauss Newton method [4]. For the later and the scenario which the algorithm converges to the true system state $\boldsymbol{x}$, then $\Delta \boldsymbol{x}_k = 0 \implies \boldsymbol{F}(\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}_k)) = 0$, where $\boldsymbol{F} = (\boldsymbol{J}_h{}^T \cdot W \cdot \boldsymbol{J}_h)^{-1} \cdot \boldsymbol{J}_h{}^T \cdot W$ and W is a diagonal weight matrix (typically inverses of measurement noise variance, R^{-1}).

C. Related Work on Testbed Development

There is a number of testbed development platforms reported on CPS applications such as smart grid, water treatment facilities, and chemical plants [5], [6]. The designed models represent a particular view of the system aiming to solve a specific issue [7]. For example, the smart grid CPS testbed in [8] is designed to examine "intelligent" and distributed control algorithms on microgrid's operations.

To date, efforts in developing testbed platforms for smart grid applications have resulted in costly, non-adaptable, and not easily replicated environments. For instance, a real-time CPS testbed for wide area control has been developed that includes hardware devices such a real-time digital simulator (RTDS) and PXI controller [7]. These components however, are expensive, costing tens of thousands of dollars, and therefore out of the reach of many testbed projects. Even developments aim to address this factor [9], are based on costly technologies in order to reproduce the signals on different load conditions. On the other hand, FLEP-SGS^2 implementation supports educational and research-aimed simulation tools integrated with open-source software. The developed models are supported by low-cost hardware platforms such Raspberry Pi (RPi) and data acquisition (DAQ) devices.

Simulated testbeds, although they offer a low-cost means to model CPSs, do not consider a HIL setup and hence lack the ability to model multilayer interactions [10], [11]. Software-only techniques often fail to capture the complexity and recreate the real-world conditions of CPSs. In comparison, the developed HIL testbed connects devices with the simulation and HMI platforms providing an accurate environment that represents all the layers of the architecture. As a result, security tests can be examined at each layer and across layers.

III. FLEP-SGS^2 Description

The proposed platform has been designed with four main modules to demonstrate major functions of grid operation. Fig. 1 presents the setup of our HIL testbed. Power system analysis is performed with MATLAB/Simulink on IEEE 9-bus system. An overcurrent protection relay, implemented on a Raspberry Pi 2 (RPi2) with a UniPi extension board, is connected to the simulation in a HIL setup. RPi is considered a cost effective, portable, and widely-used solution which its performance is good enough to allow it to function as a basic PC and be easily integrated with other systems. The measurements from the simulation and HIL relay setup are transferred to both SCADA human-machine interface (HMI) as well as to the state estimation of the EMS. The SCADA RPi2-based workstation is designed using *pvbrowser*, an open-source SCADA software suite [12]. The state estimation algorithm is implemented on a different RPi2 and our code is freely available online [13]. Additional testbed code and data are available upon request. The current implementation of FLEP-SGS^2 supports Modbus/TCP protocol.

Software packages for power system analysis typically are divided into commercial and educational/research-aimed packages. MATLAB and Simulink have become popular in

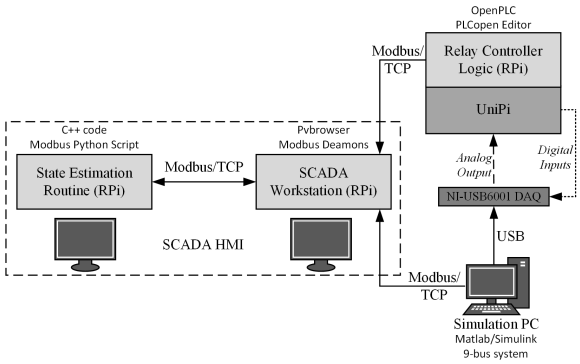


Fig. 1: Setup of smart grid hardware-in-the-loop system testbed.

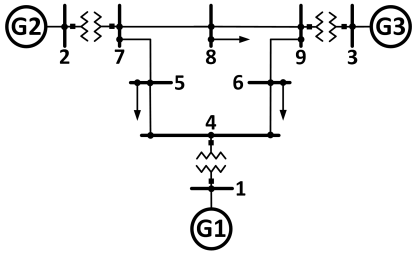


Fig. 2: IEEE 9-bus system (WSCC test case).

the design and development of power systems applications. For our testbed purposes, we utilize Simulink to implement a model of IEEE/WSCC 9-bus system (Fig. 2).

In the FLEP-SGS[2] setup, we run a load flow study of the 9-bus system. The soft real-time simulation is running on a 64-bit host-machine with 2.9GHz Intel Core i7-6700T single-core processor with 16GB RAM. Several large test systems can be used for the power system simulation since the platform scalability depends on the host-simulation PC. The results from the simulation are re-transmitted to the SCADA platform via NI LabVIEW. We establish a UDP connection between MATLAB/Simulink and a custom LabVIEW interface on the host-machine to send the per unit values of voltage, current, active and reactive power from each bus of the system.

The 9-bus simulation model is modified to include a HIL bus with an overcuccrent single-phase[1] protection relay on the load bus 6. The relay, implemented on a RPi2 connected to a UniPi extension board, has the features of instantaneous, time definite, and IDMT characteristics; providing the option to the user to select the relay category. The logic of the HIL relay is developed in ladder diagrams using PLCopen Editor, an open-source software able to write programmable logic controllers (PLCs) programs according to IEC-61131-3 [14]. In order to compile and upload the PLC-based relay logic to the HIL RPi/UniPi, we utilize OpenPLC, an open-source PLC software that includes a program development environment [15].

[1]The monitoring of a single phase is due to the limited number (2) of analog inputs of the UniPi board.

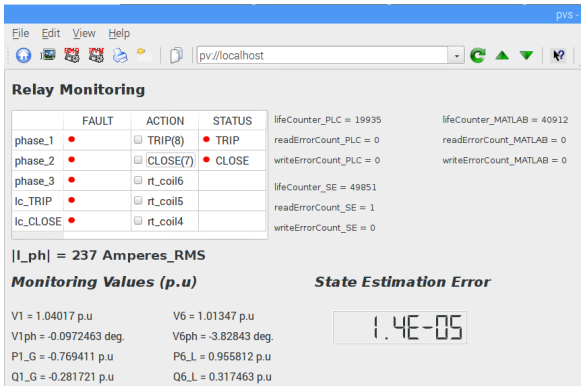


Fig. 3: SCADA interface (*pvbrowser*).

The analog/digital interaction between the simulation environment and the HIL relay is achieved via a DAQ device (NI-USB6001). The *rms* value of phase current at bus 6 is transferred via the DAQ to the RPi2/UniPi HIL relay. In case of an overcurrent condition, based on the current value and the type of the relay, the trip command is forwarded from the digital output of UniPi to the DAQ system and from there to MATLAB/Simulink. At the simulation environment, a simulated circuit breaker takes action and isolates the faulty load bus. In addition, we have hardwired at the UniPi trip and close contacts and configured accordingly the digital inputs. This setup allows to locally initiate the trip or close signal to the breaker, e.g., for maintenance purposes.

FLEP-SGS[2] includes a SCADA platform that collects all monitoring and state messages from both the simulation and HIL testbed. The SCADA master and HMI is implemented on a RPi2 platform and developed using *pvbrowser*. The SCADA platform gives a global view of the state of the system. As shown in Fig. 3, the SCADA HMI supports control and alerting capabilities. The SCADA center communicates with the rest of the testbed with Modbus/TCP, a variant of the Modbus family of simple, vendor-neutral communication protocols intended for SCADA equipment.

The Modbus/TCP protocol employs a client-server communication mode. Only one device can be designated as the client (HMI) while the remaining devices are servers. In our testbed setup, Modbus over TCP/IP traffic is routed through a gigabit Ethernet switch.

Modbus data is read and written as “registers” which are 16-bit pieces of data. Most often, the register is either a signed or unsigned 16-bit integer. If a 32-bit integer or floating point is required, these values are actually read as a pair of registers. Therefore, since we transfer floating type values of per unit data from the simulation environment to the SCADA platform, we pair each per unit value into two 16-bit Modbus registers per 32-bit floating type number.

In order to collect data at the SCADA platform, three independent Modbus daemons are running on the SCADA-RPi2 pulling data from all other components: *a)* a daemon pulls the simulation data from MATLAB/Simulink, *b)* a daemon pulls

information from the overcurrent relay, and *c)* the last daemon forwards the simulation data to the state estimation algorithm. A daemon consists of two threads where one thread reads data cyclically and writes the result to a shared memory. The other thread is waiting on a mailbox for commands to output data to the interface. The visualization can read the shared memory, and therefore all the collected information can be accessed from the designed pvbrowser web browser HMI for monitoring purposes. To output data, i.e., send control and write requests from the SCADA HMI to the other testbed elements, the visualization sends a message to the daemons via mailbox. An advantage of this architecture is that data acquisition and visualization are separated from each other.

The system state can be obtained at any given moment at the EMS; running the state estimation algorithm on a RPi2 after collecting all the testbed data. Our *C++* code implements the nonlinear state estimation, performed via the Dishonest Gauss Newton method. The state estimation code leverages *libmodbus*, a free software library to send/receive data with a device which supports the Modbus protocol. In order to pull and send data from and to the SCADA RPi2 via Modbus/TCP, we have developed a python script based on *pymodbus*, a Modbus protocol implementation in python. The input data from SCADA are also stored locally on a historian database, and the estimation error is transferred to the SCADA HMI.

IV. Results

The setup of FLEP-SGS^2 was used to investigate the testbed performance as well as the effect of cyber-attack scenarios.

The operation settings of the relay such as the time multiplier, pickup value, etc. have been selected carefully to ensure realistic operation of both the relay and the testbed setup. For example, the pickup setpoint I_s of the instantaneous, definite, or IDMT relay has been chosen after calculating the full load current (I_{fl}), the short-circuit current, and also taking into consideration the turn ratio of the current transformer (CT). The selected pickup current is based on the relation: $1.2 \times I_{fl} \leq I_s \leq 0.9 \times I_{min.fault}$ [16]. Following these steps, fault analysis of the 9-bus test system was conducted.

In order to validate our testbed setup, we apply three-phase to ground fault at $t = 5sec$. The results for instantaneous, definite time, and IDMT standard inverse curves are shown in Fig. 4. Similar to the IDMT standard inverse curve are the results for very inverse, extremely inverse, and long-time inverse settings. Also, only a single phase is shown as its a case of symmetrical fault. The additional delay (from the moment the fault is applied until the trip signal is transferred back to the simulation) can be attributed to the absence of a real-time operating system and digital simulator (RTOS/RTDS) and to the communication delay between the system components.

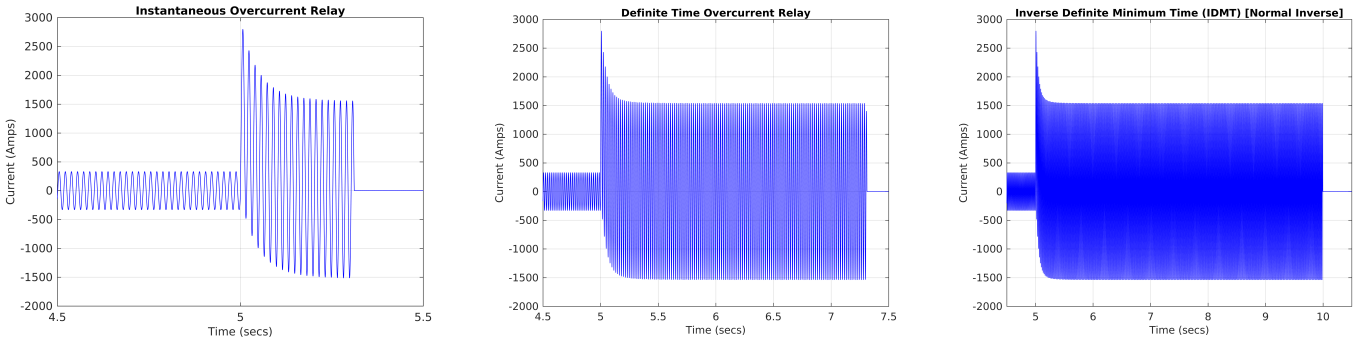
In the developed testbed, vulnerability assessment can be performed by simulating cyber-attack scenarios such as denial-of-service (DoS) and man-in-the-middle (MitM) attacks. In the following case study, a MitM attack is launched that causes a DoS. In MitM attacks, an adversary intercepts the communication between two systems, A and B [17]. The attacker "splits" the original communication link into two connections, one between A and the attacker, and the other between the attacker and B. Although the two systems believe that they directly communicate with each other, the adversary acts as a proxy who eavesdrops on the connection. The execution of a MitM attack requires to modify the exchanged data or inject new traffic into the communication channel.

The network traffic of Modbus communication is monitored at the SCADA station using Wireshark and shown in Fig. 5(a) [18]. The tick interval for the x-axis is $0.1sec$ and $10\ pixels/tick$. Since Modbus is a request/reply protocol and the pulling data from the relay to SCADA workstation is exchanged in defined intervals, we observe that the traffic between the two units is periodic and constant. Similarly, the traffic between the simulation and the SCADA is periodic. Here, despite we transmit more Modbus registers, the exchanged packets are less than the traffic between the SCADA and relay modules because we utilize one Modbus command to transmit all registers content. Finally, the communication of SCADA and the state estimation routine depends on the bidirectional read/write commands between these two functions.

Two scenarios, with and without MITM attack, were considered. First, we examine the scenario in which trip and close commands are send from the SCADA master to the overcurrent relay. Fig. 5(b) presents the traffic collected between the components of FLEP-SGS^2 and the SCADA center. With this scenario we aim to profile the exchanged traffic in our testbed under eventful operating conditions. We observe that the trip and close signals appear as four spikes in the traffic between the relay and the SCADA units. The first spike at $5.3sec$ represents the signal 1 of the trip command of *TRIP coil 0*. After $1sec$, the second spike shows the reset request signal 0 of the same *TRIP coil 0*. The close command and its corresponding signal 1 is transmitted at $t = 10sec$ via the *CLOSE coil 1*. Finally, the forth spike corresponds to the reset signal 0 of *coil 1* which sets the relay back to its initial state.

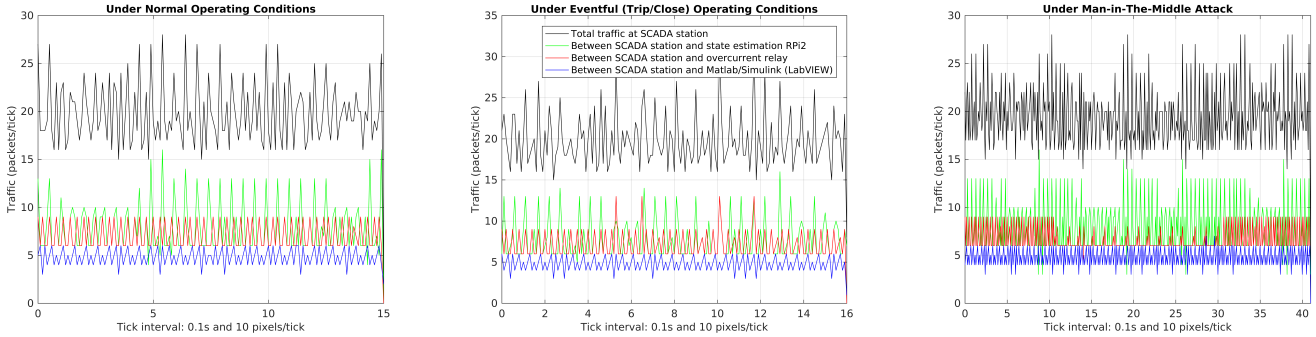
Next, a MitM attack was performed with Ettercap tool that maliciously modifies the Modbus/TCP commands at the communication link between the SCADA station and the HIL overcurrent relay. Ettercap is a free and open source network security tool for MitM attacks on a local area network (LAN) [19]. An Ettercap filter was created to modify Modbus/TCP communications coming from the SCADA workstation with a destination to the HIL overcurrent relay. The traffic modification was achieved via Address Resolution Protocol (ARP) poisoning: an attacker changes the MAC address and attacks an Ethernet LAN by changing the target computer's ARP cache with a forged ARP request and reply packets.

The implemented filter monitors the read holding register requests that hold the relay input current I. These requests are replaced with "spoofed" set commands (signal 1) to the *TRIP coil 0*. As a result, the SCADA workstation is oblivious to the measured current and at the same time the attacker continuously sends the trip command to the relay leading to a DoS-type of attack. Fig. 5(c) shows the traffic between the testbed components at the SCADA center during the attack.



(a) (b) (c)

Fig. 4: Results for HIL relay with (a) instantaneous, (b) definite time ($2sec$), and (c) IDMT normal inverse overcurrent relay.



(a) (b) (c)

Fig. 5: Traffic between testbed components at the SCADA center: (a) under normal operating conditions, (b) under eventful operating conditions (trip/close), and (c) under man-in-the-middle (MitM) attack.

The packets between the relay and the SCADA station show that MitM was carried out successfully starting at $t = 11sec$.

V. Conclusion

In this paper, we demonstrate a low-cost, scalable, and flexible testbed able to examine smart grid test cases. FLEP-SGS^2 utilizes low-cost hardware and open-source software. We aim to provide an easy-to-develop platform replicating a realistic grid environment that allows to conduct cybersecurity studies. We examine the performance of a HIL relay and present a DoS attack. Future work will reduce the operating delays (e.g., with RTOS), and address more attack scenarios.

References

[1] ICS-CERT, "ICS-CERT Year in Review: 2016."

[2] North American Electric Reliability Council, New Jersey, "NERC Disturbance Reports," 1992-2009.

[3] L. Blackburn and T. Domin, *Protective relaying: principles and applications*. CRC press, 2006.

[4] A. Gomez-Exposito and A. Abur, *Power system state estimation: theory and implementation*. CRC press, 2004.

[5] NIST, "Measurement challenges and opportunities for developing Smart Grid testbeds," 2014.

[6] A. Mathur and N. Tippenhauer, "Swat: A water treatment testbed for research and training on ics security," in *CySWater, International Workshop on*. IEEE, 2016, pp. 31–36.

[7] B. Chen *et al.*, "Implementing a real-time cyber-physical system test bed in rtds and opnet," in *NAPS*. IEEE, 2014, pp. 1–6.

[8] M. Stanovich *et al.*, "Development of a smart-grid cyber-physical systems testbed," in *Innovative Smart Grid Technologies, IEEE*, 2013.

[9] M. Hernandez *et al.*, "Embedded real-time simulation platform for power distribution systems," *IEEE Access*, vol. 6, pp. 6243–6256, 2018.

[10] C. Queiroz *et al.*, "Building a scada security testbed," in *Network and System Security Conference*. IEEE, 2009, pp. 357–364.

[11] X. KoutsouKos *et al.*, "Sure: A modeling and simulation integration platform for evaluation of secure and resilient cyber–physical systems," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 93–112, 2018.

[12] Lehrig Software Engineering, "The process visualiation browser (pvbrowser," [Online]. Available: https://pvbrowser.de/.

[13] C. Konstantinou, "State Estimation in a HIL Testbed ," [Online]. Available: https://github.com/harryskon/FLEP-SGS-2.

[14] E. Tisserant and L. Bessard, "PLCopen Editor," [Online]. Available: http://www.openplcproject.com/plcopen-editor.

[15] T. Alves *et al.*, "Openplc: An open source alternative to automation," in *GHTC, IEEE*, 2014, pp. 585–589.

[16] M. Almas, R. Leelaruji, and L. Vanfretti, "Over-current relay model implementation for real time simulation & hardware-in-the-loop (hil) validation," in *IECON*. IEEE, 2012, pp. 4789–4796.

[17] C. Konstantinou and M. Maniatakos, "Security analysis of smart grid," *Chapter 15 – Communication, Control and Security Challenges for the Smart Grid*, vol. 2, p. 451, 2017.

[18] "Wireshark Software," [Online]. Available: https://www.wireshark.org/.

[19] "Ettercap Project," [Online]. Available: https://www.ettercap-project.org/.