School of Electrical Engineering, Computing and Mathematical Sciences

Cyber-Attack Simulation in Industrial Control Systems: Methodologies, Data Generation, and Intrusion Detection

By

Jaxson Brown
BAdvSc (Curtin)
0009-0006-6093-3832

Contents

Abstract

Industrial Control Systems (ICS) is a overarching term that encompasses various components used within the industrial sector. These components include Programmable Logic Controllers (PLC), Human Machine Interfaces (HMI), sensors, actuators, as well as subset control systems such as Supervisory Control and Data Acquisition (SCADA) systems. Manufacturing processes and critical infrastructure commonly consist of such industrial control systems, and can include facilities such as chemical processing, nuclear power generation, electrical grids, and factory process lines. Traditionally, the components of these control systems have been designed on the assumption that their access is limited to a local network or through physical access. Consequently, the security of these systems is often outdated and not fit for an Internet-integrated environment. Due to the critical nature of industrial control systems, it is vital to review existing security services and to develop innovative approaches for enhancing cyber protection.

One approach to improve cyber security is the development of intrusion detection systems (IDS). To develop and refine an intrusion detection system, a real-time industrial control system is required. However, it is often unrealistic and unsafe to utilise an operational control system for this purpose. Due to these limitations, researchers have developed various test-beds for cyber-security research. In this paper we aim to contribute to this research environment by designing and constructing a virtual and semi-physical industrial control system test-bed. We explore the use-cases of this test-bed, such as the execution and analysis of cyber-attacks. Finally, we generate data from the simulation for intrusion detection system development and propose further research that involves test environment.

1

Introduction

The industrial sector relies heavily on automation, scalability and reliability. Industrial control systems (ICS) is a term for infrastructure that provide these functionalities, and are typically used for process control. Industrial control systems encompass systems such as Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS)[Stouffer et al., 2011] as well as related components such as Programmable Logic Controllers (PLC), Remote Terminal Units (RTU) and Human Machine Interfaces (HMI) [Bhamare et al., 2020]. These components often work in unison within a local network to construct a control system.

Critical infrastructure often utilise industrial control systems, and can include facilities such as chemical processing, nuclear power generation, oil and gas, electrical grids, and factory process lines [McLaughlin et al., 2016][Drias et al., 2015]. Such infrastructure are significant to industry and provide essential services on a national level. Furthermore, these systems often involve high coupling between software and physical components. For example, industrial control system components for a hydroelectricity plant may be used to physically adjust water valves to control water level and pressure. In such environments a security compromise can result in disastrous outcomes including extreme physical damage and even harm to human life [McLaughlin et al., 2016].

Despite the critical nature of industrial control systems, it is not uncommon for outdated practices to still be used. This problem stems from the fact that conventional systems were built in isolation and restricted to a local network. As a result, network security has not been a serious consideration [Asghar et al., 2019].

With the continuing adaptation of Internet-based services, the attack surface of these systems has evolved beyond their traditional scope. Threats such as command injection attacks, denial of service attacks and man-in-the-middle attacks are such examples of vulnerabilities that arise as a result of the increased attack surface [Morris and Gao, 2013]. Proprietary protocols and traditional control system components do not have sufficient security safeguards to guard against this increased attack surface [Hu et al., 2018]. It is thus a high priority to explore and evaluate new approaches to industrial control system cyber-security to ensure reliable and safe usage whilst still utilising the benefits of an Internet-integration system.

A recent advancement in security is the development of ICS-specific intrusion detection systems (IDS). These systems utilise machine learning techniques on operational control systems to detect anomalous network behaviour, which can further be classified as malicious behaviour [Dehlaghi-Ghadim et al., 2023b]. To train these detection models, datasets are required that reflect an industrial control system running as normal compared to the same system running under cyber-attacks. In order to obtain such datasets, an operational control system is required. Understandably, running cyber-attacks and collecting confidential data from a real-world control system is unrealistic and posses many limitations [Dehlaghi-Ghadim et al., 2023a]. In response to these limitations, researchers have developed test-beds for the sole purpose of cyber-security research.

Within this paper, we design and develop an industrial control system simulation to contribute to the field and to address limitations with existing test-beds. We construct the simulation virtually and semi-physically, and explore and analyse the effects of realistic cyber-attacks on the test-bed.

1.1 Motivation

Historically, there are many examples of cyber-attacks that have been reported to target industrial control systems. With the continuing Internet exposure of ICSs, the number of attacks in the last few decades have increased substantially [Asghar et al., 2019]. The consequences of these attacks have proved to be disastrous. For example, in 2015 a coordinated cyber-attack on Ukrainian power grids rendered 225000 citizens without electricity [Alert, 2016]. Such attacks on critical infrastructure prove that security research is vital to

ensure the prevention of similar attacks in the future. In the following section we further explore past ICS cyber-attacks and their resulting consequences, solidifying the need for further research into effective security safeguards.

1.1.1 Historical ICS Attacks

BlackEnergy 3

As mentioned previously, in 2015 the Ukrainian power grids were the victim of an ICS cyber-attack. The attack involved the installation of the "BlackEnergy 3" malware through malicious Microsoft Office documents. The malware allowed attackers to gain access through a backdoor and to eventually attack the operation of the power grid [Case, 2016]. This attack caused power outages across Ukraine, resulting in an approximately 225000 customers without power [Alert, 2016].

Stuxnet

In 2010, an attack on Iranian uranium enrichment facilities caused the physical destruction of centrifuges used for the enrichment process [Falliere et al., 2011]. The infamous malware used in this attack, called "Stuxnet", was widely studied by researchers and was titled a "cyber warfare weapon" due to it's unique and violent nature [Langner, 2011]. Stuxent was a highly sophisticated malware that was designed to only target industrial control systems. Once it infected a host it would conduct exploits against multiple unpatched vulnerabilities to modify programmable logic controller code, which caused the enrichment centrifuges to spin and burn out. The Stuxnet malware was characterised by an unusually high level of sophistication. It was estimate that a team of 5 to 10 people worked on the malware over a six month period [Chen and Abu-Nimeh, 2011].

Triton

In 2017 a Saudi Arabian petrochemical processing plant was the victim of a cyber-attack. This attack involved another type of ICS malware named "Triton" that specifically targeted Safety Instrumented System (SIS) controllers [Di Pinto et al., 2018]. The malware would override the controllers and cause them to enter a failed state, resulting in a misdirected shutdown of the plant. Sabotaging the SIS controllers removed a layer of protection that guarded against real malfunction, which hence introduced tremendous safety risks. This attack was of great significant as it was recognised as the first attempted cyber-attack with the possibility of human harm or loss of life [Slowik, 2022].

Year	Target	Description
2003	US Davis-Besse Nuclear Plant	An attack which compromised integrated system safety features [Evancich and Li, 2016].
2012	US Electric Utility Turbine Control System	A botnet called "Mariposa" initiated an attack causing 3 weeks of system downtime [Miller et al., 2021].
2013	Bowman Dam, New York	An attack which rendered the dams SCADA system uncontrollable [Hemsley et al., 2018].
2016	Ukraine Electrical Grids	Malware titled "CrashOverride" induced a second attack on Ukrainian electrical grids following Stuxnet [Slowik, 2018].
2021	US Colonial Pipeline	Hacking group "DarkSide" initiated a ransomware attack resulting in a 4.4 million USD payout [Kilovaty, 2022].

TABLE 1.1: Brief list of recent significant ICS cyber-attacks

Other ICS Attacks

Table 1.1 presents a few other significant attacks on ICS. Note the diverse nature of the systems targeted and the varying consequences of these attacks.

1.2 Research Objective

The primary objective of this paper is to provide a realistic test-bed environment for industrial control system cyber-security. The purpose of this test environment is to enable researchers to investigate ICS-specific cyber-attacks, and to develop innovative mitigation solutions. Our overarching objective is to enhance the current ICS cyber-security landscape.

1.2.1 Deliverables

Considering the primary objective of this paper, we aim to produce the following deliverables:

- An industrial control system simulation capable of running both virtually and semi-physically. To elaborate, by "virtually" we imply that the simulation can fully run as software on a single host machine, and by "semi-physically" we mean that the simulation can run on multiple physical components.

- Various realistic cyber-attack scripts that specifically target industrial control systems. These attack scripts will be developed will a distinct focus on replicating how real-world cyber-attacks occur.
- Datasets comprising of both benign and malicious industrial control system network traffic. These datasets will be produced from subjecting the simulation to the developed attacks. The purpose of this deliverable is to demonstrate the simulation's ability to provide data that can potentially be used to develop an intrusion detection system [Morris et al., 2015].

1.3 Structure of This Paper

The rest of this paper is structured as follows. In Chapter 2 we discuss the background of industrial control systems, including the architecture and the current cyber-security climate of these systems. In Chapter 3 we discuss this paper's methodology for the design and implementation of an industrial control system test-bed and the development of related cyber-attacks. We also elaborate on the approaches used to generate datasets. In Chapter 4 we evaluate the methodology used to build the simulation and the results of the cyber-attacks and related datasets. Finally, in Chapter 5 we conclude the overall work of this paper and describe potential future considerations.

2 Background

2.1 Industrial Control System Architecture

Industrial control systems consist of various different components. These components have unique jobs and purposes, which vary between systems. A typical control system setup involves a network of these components communicating with each other to perform overall process control. Common systems that are built in this fashion include Supervisory Control and Data Acquisition (SCADA) systems and Distributed Control Systems (DCS) [Asghar et al., 2019].

As previously mentioned, the variety of industrial control systems differs significantly. As such, the configuration of these devices will inevitably differ from plant to plant. Despite these differences in configuration, there are several shared characteristics amongst all industrial control systems. For instance, many systems utilise similar types of components in their networks. Additionally, they often rely on the same communication protocol to facilitate inter-device communication.

2.1.1 ICS Components

Here we list components and devices commonly found in industrial control systems.

- **Programmable Logic Controller (PLC)**. These devices are responsible for handling complex logic and switching operations [Drias et al., 2015, Alphonsus and Abdullah, 2016]. They oversee field devices such as sensors and actuators for process control and data acquisition. PLCs operate by receiving input from either a central control centre or from other devices (including other PLCs). Depending on this input, the PLC performs predetermined logic and sends output signals to other devices, which dictate their behaviour. For instance, consider a setup where a temperature sensor is connected as input to a PLC, and a cooling fan is connected as corresponding output. The PLC can be programmed to activate the fan if the temperature sensor detects an abnormally high temperature.

- **Remote Terminal Unit (RTU)**. These are devices that are primarily used for measurement and data acquisition of remote field devices [Drias et al., 2015]. They have logical I/O functionalities similar to PLCs, but are considered different components as they are designed more as "self-contained" computers that are primarily used for long distance monitoring.

- **Human Machine Interfaces (HMI)**. These components act as a centralised control point within an ICS [Igure et al., 2006]. They act as clients to PLCs and RTUs, being able to request data for visualisation and send commands for manual control of field devices. HMIs are built with a physical display and user-interface to assist operators with visualising and controlling the operation of a system.

- **Sensors and Actuators**. These devices represent field components that physically operate the control system. "Sensors" refer to devices that primarily retrieve input data, such as switches, temperature/pressure sensors and power meters. "Actuators" refer to devices that produce physical motion, such as electrical motors and valves.

2.1.2 The Modbus Protocol

Modbus is the most widely used communication protocol in industrial control systems [Chen et al., 2015]. The Modbus protocol is an open-source protocol developed with simplicity as a priority. As such, it has become a protocol favoured by industry.

Modbus has a master/slave architecture. Modbus-compliant devices can either act as a master or as a slave: master devices send Modbus "requests" and slave devices reply with Modbus "responses" [Dutertre, 2007].

Modbus comes in three variations: RTU, ASCII and TCP. Modbus-RTU and ASCII variants both operate through a serial communication line. The main difference between

these two variants is that Modbus-RTU uses binary encoding whereas Modbus ASCII uses human-readable characters [Thomas, 2008]. RTU also uses cyclic redundancy check (CRC) for error detection whilst ASCII uses longitudinal redundancy check (LRC). Modbus-RTU is preferred in industry for serial communication due to its speed and more reliable CRC. As such, this paper only addresses Modbus-RTU for serial communication.

The Modbus-TCP variant is an encapsulation of the RTU packet that allows the protocol to operate on networks using TCP/IP protocols [Swales et al., 1999]. Modbus-TCP makes use of features of the TCP/IP stack such as IP addressing and seamless error detection/correction.

Modbus-RTU Structure

Being a communication protocol, Modbus defines the structure of data frames that get sent between different Modbus-compliant devices [Thomas, 2008]. Figure 2.1 gives a visual representation of the different fields within a Modbus-RTU frame. The structure and purpose of each field for an RTU frame is described in detail below:

- Address: This field takes up two bytes and is used to assigned an address to devices. The address itself is simply an integer ranging from 1 to 247.
- Function Code: This field takes up two bytes and references a predefined function which dictates the purpose of a Modbus request. For example, function code "1" (or in hexadecimal format, "0x01") is the "Read Coils" function, and is used to read coils or single bit values from a device.
- Data: The data field varies in content and length depending on the function code of the request. Typically it holds function parameters for Modbus requests or return values for Modbus responses.
- CRC: This field contains two bytes used for the CRC error detection.

Address	Function Code	Data	CRC
1 byte	1 byte	N bytes	1 byte

Figure 2.1: Modbus-RTU frame structure

Transaction Identifier	Protocol Identifier	Length (number of remaining bytes)	Unit ID	Function Code	Data
2 bytes	2 bytes	2 bytes	1 byte	1 byte	N bytes

FIGURE 2.2: Modbus-TCP frame structure

Modbus-TCP Structure

Modbus-TCP is an encapsulation of a Modbus-RTU packet within a TCP header. Essentially, it offers the same functionality as the RTU variation but with all the features of the TCP/IP stack. The benefit here is that Modbus-TCP can operate on network mediums such as Ethernet or through wireless access points.

There are a few important distinctions between Modbus-TCP and Modbus-RTU. For Modbus-TCP, devices are recognised through an IP address, and error checking is performed on the Transport and Network Access layers of the TCP/IP model [Dutertre, 2007]. The architecture for TCP follows a connection-oriented client/server architecture, where slave devices are now clients and master devices are now servers. There are a few changes to the structure of a Modbus-TCP frame.

The Modbus-TCP frame consists of a transaction identifier, protocol identifier, length field, and unit ID, as well as the regular function code and variable data fields present in Modbus-RTU [Swales et al., 1999]. Notice how the address field from an RTU frame has changed to the "unit ID" field. This unit ID field is only used for compatibility purposes as addressing in Modbus-TCP is handled through IP addresses. The CRC error detection fields are also removed as error detection is handled through the TCP/IP headers. Figure 2.2 displays the Modbus-TCP frame structure (excluding the headers from the TCP/IP stack).

In further chapters we design and implement an industrial control system simulation that uses the Modbus-RTU and TCP protocol. Therefore, it is necessary to outline in detail how the protocol works.

2.1.3 The Purdue Enterprise Reference Architecture

Industrial control systems have different setups and configurations depending on the infrastructure of the system. Despite these differences, the overall architecture of an ICS often remains similar across different systems.

Many industrial control systems follow an architecture called the "Purdue Enterprise

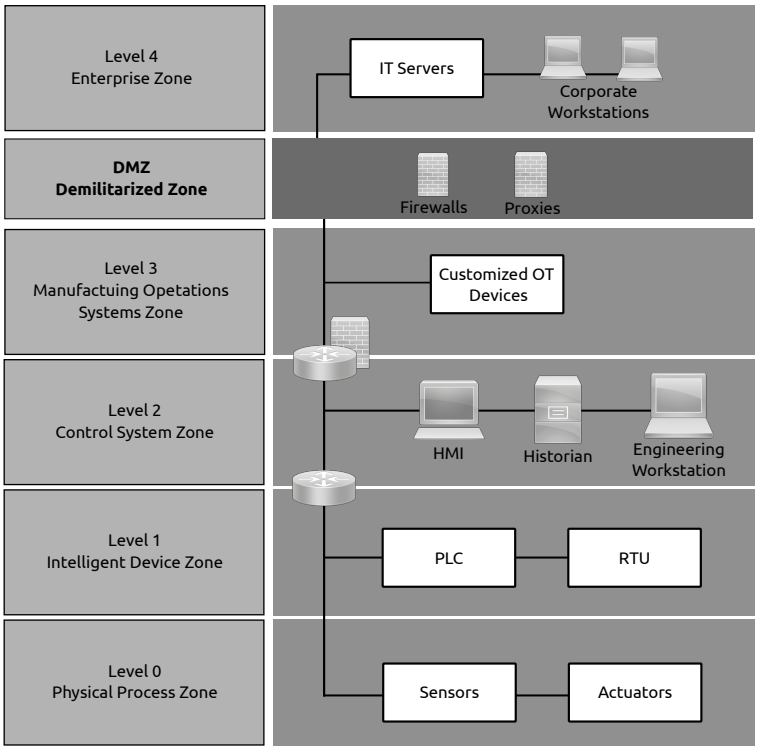

Figure 2.3: Levels of the Purdue Enterprise Reference Architecture

Reference Architecture" [Williams, 1994]. This model describes an ICS in a layered approach, with each layer representing a different level of component interaction. Lower layers interact more with the physical environment of the system, whilst higher layers deal with more logical and application level interactions. Figure 2.3 shows the different layers of this architecture, and displays where the components mentioned in Section 2.1.1 sit within the model. Communication between these components use various protocols, including the Modbus protocol as mentioned in Section 2.1.2.

In regards to the level of the architecture, Layer 0 is concerned with physical interactions with devices that directly affect or monitor the field, such as sensors and actuators. Level 1 consists of intelligent devices such as PLCs or RTUs that often control the logic of the field devices. Level 2 is where supervisory control is performed. Devices such as HMIs and regular engineering workstations belong to this level. These devices collect data and execute commands for overall system management. Level 3 consists of components for higher management, and can often include devices such as other data historians and plant-specific production control devices. From here, most control systems implement a Demilitarised Zone (DMZ) [Abdelghani, 2019]. This zone consists of firewalls, proxies,

and other security measures used to limit and protect from unauthorised remote access. Levels above the DMZ, including level 4 and other higher levels, are usually connected to corporate networks and the Internet.

Further along in this paper we mention we design a simulation based on levels 0 to 2 of this architecture.

3 Methodology

There are various approaches to simulating an industrial control system. In this chapter we outline our own methodology for developing an ICS simulation and the reasoning behind key design choices. We elaborate on the specific type of control system we simulate, and the different approaches taken to simulate it both virtually and semi-physically. Additionally, this section discusses the design of cyber-attack scripts tailored to target our simulation, describing certain differences between our attack scripts and similar scripts presented in other works. Finally, we analyse the methodology used for IDS dataset generation.

3.1 Simulation Design

The ICS simulation designed in this paper follows a smart grid model. Given past ICS cyber-attacks on electric utilities, such as the devastating "BlackEnergy" malware attack [Case, 2016], there was a clear need to address potential vulnerabilities within similar systems. Therefore, it was fitting to construct an ICS test-bed that replicated an intelligent electrical grid design.

The design follows an electrical grid of an embedded network for a small collection of households, similar to those found in environments such as apartments or retirement villages [Uslar et al., 2013]. These environments typically consist of several closely situated

households, each fitted with power management devices for central monitoring. For the intelligent aspects of the grid, each household is equipped with a simulated solar panel system. The smart grid is responsible for monitoring power generation per household and input power allocation during peak solar generation times.

The ICS setup dynamically manages solar power allocation and determines when a household can run entirely on solar power alone. It records solar power generation through simulated power meters, and when sufficient power is produced it triggers a transfer switch to automatically shift from the default mains power to solar power input (and vice-versa). In this way, the smart grid acts as an automatic transfer switch, ensuring optimal energy use by seamlessly switching to solar power when available.

These monitoring features and switching operations of the test-bed provide an environment for effective ICS cyber-security study. The implemented logic forces the components to communicate to each other in various ways. Having this dynamic form of inter-device communication allows us to capture a wide range of network data, similar to a real ICS network. This ensures that methodologies applied to the test-bed can be transferred over to a real-world ICS.

3.1.1 Simulation Architecture

As mentioned in Section 2.1.3, the architecture of industrial control systems often resemble the "Purdue Enterprise Reference Architecture" [Dehlaghi-Ghadim et al., 2023a, Williams, 1994]. To ensure a realistic design, the smart grid simulation has been built in relation to this architecture.

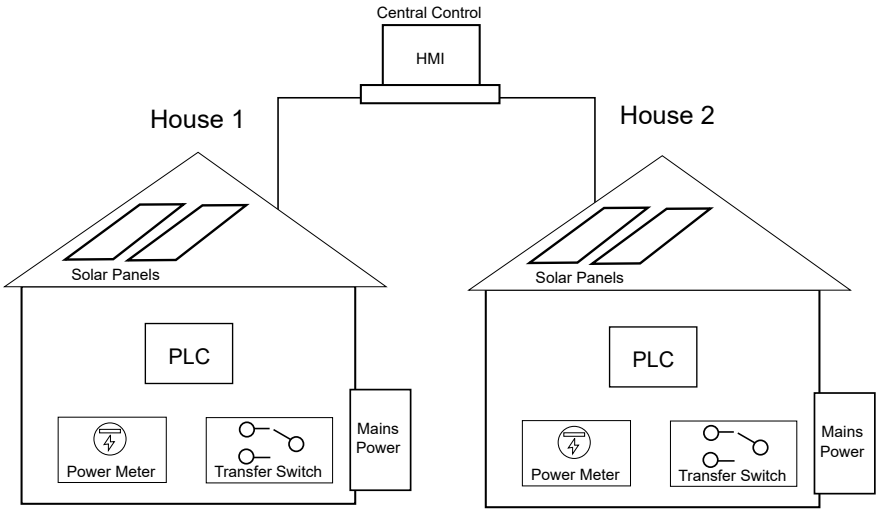

FIGURE 3.1: Diagram of the ICS smart grid simulation design

The smart grid setup consists of two households, shown in Figure 3.1. Whilst it is possible for the test-bed to simulate more than two households, we have restricted it to just two units for simplicity. Each household has been fitted with two field devices: a solar panel power meter acting as a sensor, and a transfer switch acting as an actuator. These components belong to level 0 of the Purdue reference model as they directly monitor physical processes. A PLC is assigned to each house, with each being connected to the sensors and actuators of their respective house. The PLCs are within level 1 of the Purdue model as they handle the logic and data acquisition of the field devices. Further upwards, the PLCs are connected to a HMI for central monitoring and control. The HMI is part of the control system zone and belongs to level 2 of the Purdue model. Figure 3.2 shows the simulation setup in relation to the Purdue architecture.

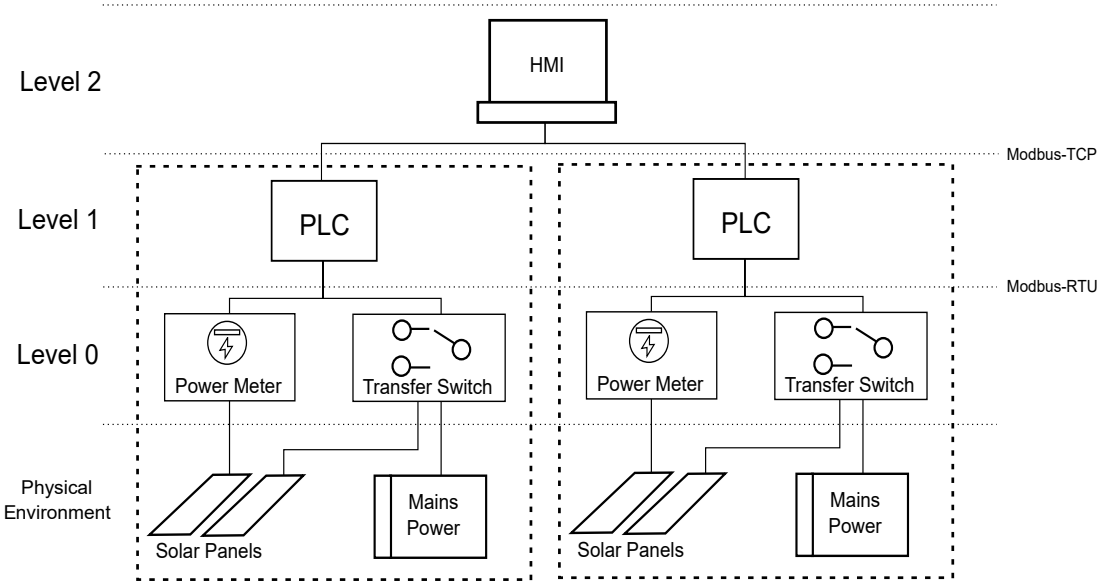

Figure 3.2: Components of the simulation in relation to the Purdue Reference Architecture

Modbus-RTU is used for communication between the field devices and the PLCs, as field devices typically use serial communication lines for legacy reasons. For communication amongst the PLCs and the HMI, Modbus-TCP is utilised. These devices are configured within in a local area network (LAN).

3.2 Simulation Development

3.2.1 Component Implementation

To simulate the components of the simulation, Python modules were used in conjunction with Docker technology. This approach reflects the methods used by previous researchers to simulate ICS test environments [Dehlaghi-Ghadim et al., 2023a]. Docker uses OS-level virtualisation to create containers, which are self-contained environments that allow for packaged code to run in isolation from other services. Such an environment is suitable for simulating ICS components, which are often physically segregated. Furthermore, these containers can be configured to operate within the TCP/IP network stack, offering settings such as configurable static IP addresses. Each container runs an Ubuntu operating system and executes a single Python script to manage the ICS component operations.

Since control systems consist of many devices, we required a method to run multiple containers simultaneously. To achieve this, Docker Compose was used. Compose is an extension of the standard Docker services that provide the necessary functionality to run several containers at once. It also offers convenient configuration settings that simplify and streamline the container creation process, along with options to build isolated virtual networks.

Different containers were designed to simulate the different components. Table 3.1 presents all the simulated devices, as well as their respective container configurations. Note that "House Number" column refers to the specific household that the components are situated in (with reference to the smart grid design). The table shows that the power meters and transfer switches do not have IP addresses. This is because these devices communicate through serial communication lines. For these components, virtual serial ports were exposed to their respective containers to facilitate serial communication.

3.2.2 Physical Environment Replication

Some of the simulated components require interaction with a physical environment. For instance, the power meter component needs electrical power readings for it to actually measure something. Table 3.2 shows the components relating to the physical environment. Obviously, it is unreasonable to expect real implementations of a physical environment for a virtual simulation, such as having a working solar panel. As a workaround, software-defined Hardware-in-the-Loop (HIL) systems have been implemented that replicate real-world data readings.

The Hardware-in-the-Loop mechanism is built into the Python execution environment of the field devices. For example, the power meter component has an in-built execution

House Number	Component	Container Name	IP Address
N/A	Human Machine Interface	HMI	192.168.0.11
House 1	Programmable Logic Controller	PLC1	192.168.0.21
House 2	Programmable Logic Controller	PLC2	192.168.0.22
House 1	Power Meter	PowerMeter1	N/A
House 1	Transfer Switch	TransferSwitch1	N/A
House 2	Power Meter	PowerMeter2	N/A
House 2	Transfer Switch	TransferSwitch2	N/A

Table 3.1: Names and IP address of the simulated components

thread that serves simulated solar panel power readings. To ensure realistic and accurate outputs, real-world datasets have been utilised within the HIL software to generate sample readings. Specifically, for the power meter sensor, a dataset containing readings from 300 houses were used to generate authentic solar power generation readings [Ratnam et al., 2017].

3.2.3 Communication Protocol Implementation

As mentioned previously, the communication protocols used in this simulation are Modbus-RTU and Modbus-TCP. Whilst there many ICS communication protocols used in industry, Modbus is the most common [Chen et al., 2015].

To implement these protocols within the Python software, the libraries `pyModbusTCP` and `PyModbus` were used. `pyModbusTCP` provides extensive functionality for Modbus-TCP implementation, including classes for server/client connectivity and automatic TCP connection handling. `PyModbus` provides functionality for all the Modbus variants, but focuses on serial communication especially. Accordingly, `PyModbus` was used for Modbus-RTU implementation whilst `pyModbusTCP` was used for Modbus-TCP.

For network IP-aware communication, a virtual network was created with Docker Compose. The containers on this network have their own virtual network interface statically set to be part of the 192.168.0.0/24 network. As such, the devices with an IP address mentioned in Table 3.1 are able to communicate via this network.

For serial communication, virtual serial ports were established on the host machine that linked between certain containers. Selected containers (i.e the PLCs and sensor/actuator containers) were configured to be exposed to these virtual serial ports, allowing them to

communicate via these lines. To construct the virtual serial ports on the host machine, the Linux tool `socat` was used. `socat` is a command-line utility used for establishing bidirectional data transfer streams. For this simulation, it is was to create virtual serial lines for inter-container communication. `socat` is not a part of Docker Compose services, and so a separate bash script was written that automatically created the `socat` serial ports to run alongside the containers.

3.2.4 Semi-Physical Implementation

The previous sections describes the simulation implemented on a single host machine using techniques such as containerisation and network virtualisation. For increased accuracy and flexibility, as well as for improved virtualisation, we also developed a semi-physical test-bed. In this setup, the different ICS components run on dedicated devices and communicate via real Ethernet cables and serial lines. We refer to this implementation as "semi-physical" as the sensor and actuator components are still virtual simulations. Specifically, the power meter sensors and transfer switch actuators do not interact with or record any real-world values. Aside from this limitation, the semi-physical test-bed has been developed in accordance with relevant industrial standards.

For the physical devices, Raspberry Pis have been used. Each Pi has been flashed with the Ubuntu operating system and have had Docker installed. The containers used in the virtual simulation have simply been transferred to the Raspberry Pis, allowing for streamlined implementation with little configuration change. The primary difference between the virtual simulation and the semi-physical is the use of a real network and real serial communication lines.

The Raspberry Pis replicating the HMI and PLC components are connected to a network switch via Ethernet. Each of these Pis have been configured with a static IP address that matches it's Docker container counterpart from the virtual simulation. For the all Raspberry Pis that require serial communication, such as the field devices, a USB to RS485 converter is used to enable serial connectivity. Finally, all components have been packaged into a portable and durable case, as shown in Figure 3.3.

Figure 3.3: Semi-physical simulation featuring the Raspberry Pis in a contained in a case

3.3 Cyber-Attack Designs

Once the ICS smart grid simulation was built, cyber-attacks were designed and executed on the test-bed. The purpose of running these attacks was to obtain network data representing an ICS running under attack, which would later contribute to building a dataset for developing an intrusion detection system. The attacks were designed following a methodology proposed by Morris and Gao (2013), which depicted different approaches for creating ICS-specific cyber-attacks. This paper describes various attack scenarios that could be executed on a general ICS. The design of these attacks had to be slightly modified to work with the smart grid simulation. Such modifications included minor adjustments, such as changing the address values for malicious Modbus requests. Whilst the attacks implemented were specific to our test-bed, the overall methodology for each attack can be mapped to any other ICS, provided the attacker has the relevant information.

12 cyber-attacks were developed that targeted the smart grid simulation. Table 3.3 describes each of these attacks. The attacks can be classified into 4 categories:

1. Reconnaissance. Attacks belonging to this category are designed to gather information on a system, such as control system network information and device characteristics. Sophisticated attacks can map out network architectures and identify potential vulnerabilities within specific components.

2. Response and Measurement Injection. These attacks involve observing regular network packets and crafting malicious packets based on recorded responses.

3. Command Injection. These attacks also involve packet injections, but have been designed around manipulating actuators and other output devices through carefully crafted commands. Knowledge of device memory is required to perform successful command injection attacks.

4. Denial of Service (DOS). The objective of DOS attacks is to render a portion of the system unavailable. These attacks involve flooding a network with large amounts of malicious packets in order to obstruct real packets from reaching their destination properly.

Many previous papers design attack scripts that run individual exploits in isolation. However, this may not properly replicate how real-world attacks are performed. Analysis of historic attacks demonstrate the use of multiple attack vectors through different stages of attacks. For example, the Struxnet attack involved an initial stage of reconnaissance, followed by execution of multiple exploits [Langner, 2011]. To mimic this behaviour, another script was developed that ran multiple individual attacks together in a defined procedure. These procedures had a main objective to them, replicating how real-world attacks are motivated by a certain goal. Table 3.4 shows the 8 objective-based staged attacks that have been developed. The table gives details on which attacks were performed together to achieve the objective.

Attack Name	Category	Description
Address Scan	Reconnaissance	Performs a port scan on a specified network to find services running Modbus.
Function Code Scan	Reconnaissance	Sends Modbus requests of differing function codes to a device to identify which codes it supports. An exception is returned for the invalid function codes.
Device Identification Attack	Reconnaissance	Sends a Modbus device identification request of function code 0x2B to a device. If the device supports this code, it will return vendor information on the device.
Naive Sensor Read	Response and Measurement Injection	Reads memory address values at random to attempt and find used memory.
Sporadic Sensor Measurement Injection	Response and Measurement Injection	Sporadically writes to random memory addresses with randomly generated data. The attack is very noisy and the affects can easily be observed.
Calculated Sensor Measure Injection	Response and Measurement Injection	Injects falsified data into a device that closely mimics real operation. Can be used to skew the data into false but seemingly realistic operation.
Replayed Measurement Injection	Response and Measurement Injection	Involves capturing packets and replaying the same data.
Altered Actuator State	Command Injection	Modifies actuator values through malicious Modbus write commands.
Altered Control Set Points	Command Injection	Modifies control set points of a devices through Modbus write commands. These points usually determine safety bounds for a system.
Force Listen Mode	Command Injection	Devices often feature a listen mode, which is activated by sending a Modbus request with function code 0x08 with a sub-function code of 0x04. When activated the device will stop responding to requests.
Restart Communication	Command Injection	Involves sending requests with function code 0x08 and sub-function code 0x01, which forces the device to restart. Sending many of these requests can render a device unavailable.
Data Flood Attack	Denial of Service	Floods a network with random Modbus read requests.
Connection Flood Attack	Denial of Service	Floods a network with TCP connection request packets.

Table 3.2: All cyber-attacks implemented for the smart grid simulation

3.4 Dataset Generation

For capturing network data from the test-bed, Wireshark was used. Wireshark sniffs packets traversing across a network, allowing for deep packet inspection and analysis. In the simulation, it was used to capture network traffic between the PLC and HMI components, providing insights into the communication patterns.

To create a dataset representing benign and malicious operations, the simulation was launched alongside a separate process that automatically launched attacks at random intervals. These attacks used in this process were the objective-based attacks defined in Table 3.4. Once both the simulation and attacking process started, Wireshark was initialised to begin capturing network packets. The format of the files containing the packet information were exported in the standard Wireshark PCAP file output.

Using this methodology, three PCAP files were generated containing malicious and benign network operations. The first file was generated over a 30 minutes period, while the latter two files were recorded over a 3 hour period. The difference between the last two files was that the random time intervals between the attacks were adjusted to provide variation.

3.4.1 Formatting Dataset

Currently, the datasets are in the form of a PCAP file, containing all information from the inspected packets. Most of this information is irrelevant and adds unnecessary size to the dataset files. In addition, presenting a PCAP file as a dataset has many issues, especially when it comes to using the dataset for machine learning purposes. Therefore, in future work these PCAP files will be parsed and formatted using standard data engineering techniques. Datasets such as the widely used KDD Cup dataset [Tavallaee et al., 2009] use CSV or ARFF formats for dataset presentation, allowing for effective machine-readable files. The PCAP files generated through this research work will be cleaned to remove irrelevant packets fields and formatted to align with these existing datasets. This will also allow for better comparison and analysis in regards to existing work.

Objective	Executed Attacks
Reconnaissance	1. Address Scan 2. Function Code Scan 3. Device Identification Attack 4. Naive Sensor Read
Sporadic injections	1. Address Scan 2. Sporadic Sensor Measurement
Disable service through Force Listen Mode	1. Address Scan 2. Force Listen Mode
Disable service through Restart Communication	1. Address Scan 2. Restart Communication
Denial service to devices	1. Connection Flood Attack 2. Data Flood Attack
Attempt to find device-related exploits	1. Address Scan 2. Device Identification Attack
Cause power outage	1. Address Scan 2. Function Code Scan 3. Naive Sensor Read 4. Altered Control Set Points
Burn out transfer switch	1. Address Scan 2. Function Code Scan 3. Naive Sensor Read 4. Altered Actuator State

Table 3.3: Objective-based attacks designed and implemented for the smart grid test-bed

4 Results

This chapter discusses the findings of this research project. These findings include the methodologies and approaches that have been effective for streamlined ICS simulation development, and the results of developing the simulation virtually and semi-physically. We analyse the performance of the cyber-attacks of the test-bed environment and describe the implications of the successful attacks. Finally, we describe the generated network data and how it relates to real-world operational control systems.

4.1 Simulation Development

After exploring many approaches to developing ICS simulations, a methodology proposed by [Dehlaghi-Ghadim et al., 2023a] found to be an effective approach for several reasons. Firstly, the utilisation of containerisation with technology such as Docker allowed for multiple ICS components to be simulated on a single host machine. Whilst similar technology exist that provide similar functionalities, such as virtual machines, the use of containers allowed for more flexibility and granularity especially with how they can be configured. Containers can also be quickly created and launched, allowing for streamlined development. In addition, the extended Docker service, Docker Compose, allowed for virtual networks to be easily created for the containers.

Furthermore, using containers to encapsulate software for the device simulation allowed for easy transfer to physical components. This allowed us to effortlessly build a semi-physical simulation, which allowed for better visualisation of the process as well as more accurate network and serial line implementation. Other ICS simulation software, such as MiniCPS [Etxezarreta et al., 2024], can only simulate a network on a single host making them unsuitable for creating a test-bed that can be run on multiple machines.

4.2 Cyber-Attack Results

The developed cyber-attacks, as defined in Table 3.3, all have a clear affect on the smart grid simulation. By designing objective-based attacks, as shown in Table 3.4, we were able to more accurately replicate real-world attack scenarios. It is never the case that cyber-attacks occur from a singular instance, rather they unfold in a series of stages. Through creating staged cyber-attacks that aim to achieve an overall objective, we managed to create cyber-attacks that better mirrored real scenarios.

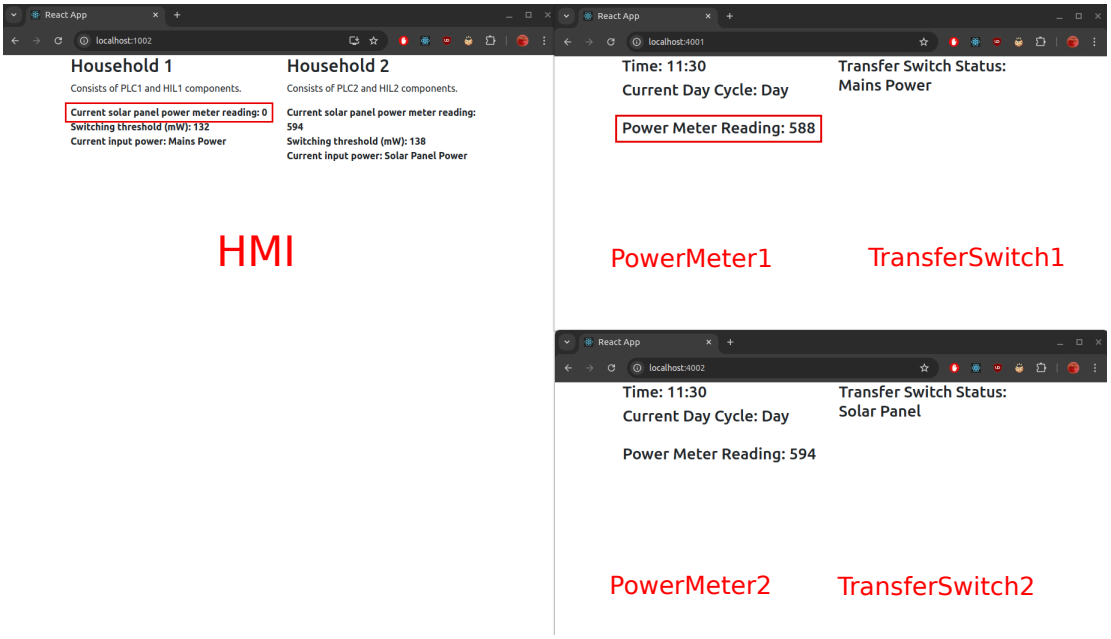

FIGURE 4.1: Example of a successful "Calculated Sensor Measurement Injection" attack falsifying a power meter reading value

Figure 4.1 illustrates the results of executing these attacks. It shows a successful Calculated Sensor Measurement Injection attack overriding a power meter's reading. On the right on the figure, one of the power meter components is displayed with it's correct

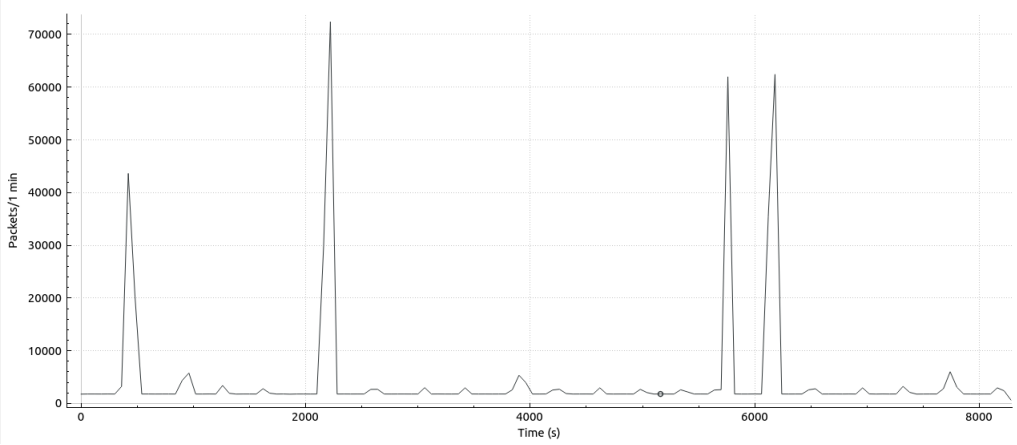

Figure 4.2: Graph showing the number of network packets captured over a period of time

power reading (labelled with the read box), which has a value of 588. However, to the left the HMI component falsely reads this value as 0, which is a result of the injection attack.

Additionally, Figure 4.2 shows packet flow over a period of test-bed operation. The graph shows distinct spikes of heavy packet collection, indicating extreme periods of network activity. These spikes point to successful denial of service attacks.

4.3 Dataset Generation

From this work, three PCAP files of were generated that captured the network activity of the simulation in operation. Figure 4.3 displays a portion of the captured packets, showing some of the Modbus packets. These files provide a detailed record of network interactions, giving a comparison between the system's normal behaviour and when the subject to cyber-attacks. As mentioned before, these files will contribute to creating a fully formatted dataset that can be used for intrusion detection system development and training.

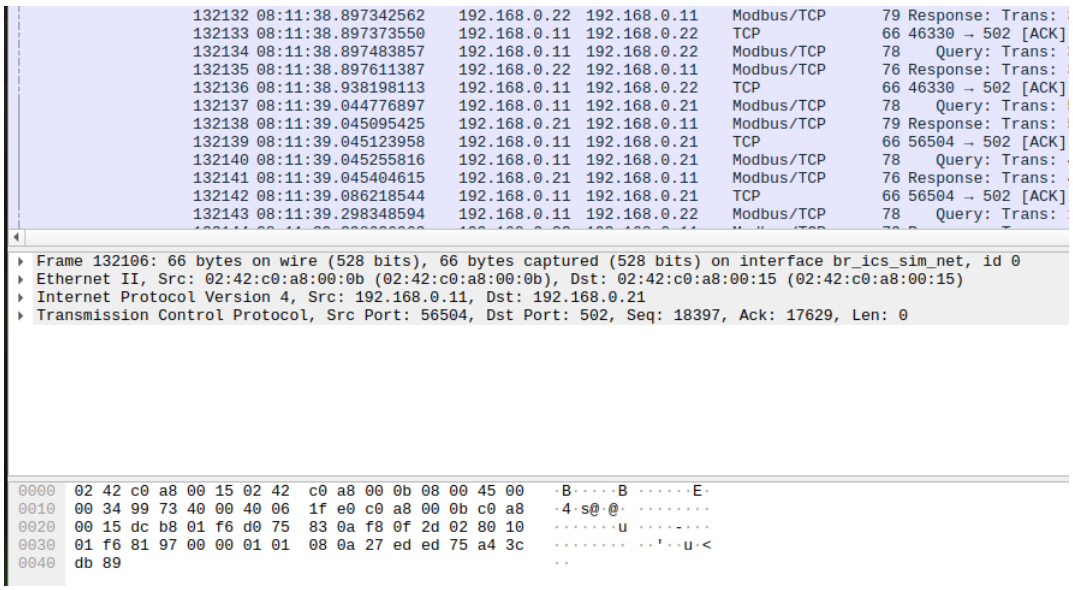

FIGURE 4.3: Screenshot of the Wireshark application showing a portion of captured network packets

5 Discussion

This paper has presented an industrial control system test-bed environment. The test-bed followed an intelligent electrical grid design and utilised industry-standard communication protocols and architecture. The methodology used to build the simulation enabled us to develop it virtually and semi-physically, enhancing the design's flexibility and versatility. To make use of the test-bed, we developed 12 singular cyber-attacks and 10 objective-based attacks. These attacks proved effective on the simulation, and allowed us to generate data to be used for developing intrusion detection systems. All methodologies used in this paper adhered to industrial and research standard, ensuring that the work produced can be applied to similar applications.

5.1 Improvements

There are a several areas of this project that could be improved upon. While the design of the smart electrical grid simulation follows industrial standards, the control system structure could be further adjusted to more closely mimic a real-world system. For instance, an ICS could be taken from an operational plant and it's design could be used within the simulation. This way, the simulation would have the added credibility from following an actual ICS environment. It would also yield more accurate network activity.

In addition, the cyber-attacks developed in this paper could be refined to better match realistic scenarios. Ideally, the cyber-attacks would be conducted in real-time by a penetration tester, with all the attacks conducted being recorded to identify the corresponding malicious network packets. This approach would be challenging to execute, but would provide an optimal cyber-attack scenario.

Another factor to consider is extending the functionality of the ICS simulation so that it becomes a working, miniaturised control system. This would provide the most accurate results for data collection and cyber-attacks. Similar test-beds already exist in research, such as the SWaT ICS test-bed [Mathur and Tippenhauer, 2016], but most have limitations such as restricted access and high cost of operation. Ideally, but unrealistically, access to historical data from a real ICS would provide the best and most accurate data.

5.2 Future Work

Future work for this project involves implementing the improvements as mentioned in Section 5.1. Additionally, the datasets presented in Section 4.3 should be filtered and parsed to retain only the useful collected data. The datasets should also be formatted into file types such as CSV and ARFF files to align with existing datasets, ensuring consisting within the research environment. Furthermore, additional cyber-attacks could be implemented to extend the range of collected malicious data.

Finally, as a significant motivation for this project, future work could involve the development of an intrusion detection system based on the data collected from the simulation. Techniques such as machine learning could be employed and explored to develop such a system. As ICS-specific intrusion detection systems are a fairly new concept, this work could pose to be highly valuable for enhancing the ICS cyber-security environment.

References

[Abdelghani, 2019] Abdelghani, T. (2019). Implementation of defense in depth strategy to secure industrial control system in critical infrastructures. *American Journal of Artificial Intelligence*, 3(2):17–22. 11

[Alert, 2016] Alert, D. (2016). Cyber-attack against ukrainian critical infrastructure. *Cybersecurity Infrastruct. Secur. Agency, Washington, DC, USA, Tech. Rep. ICS Alert (IR-ALERT-H-16-056-01)*. 2, 3

[Alphonsus and Abdullah, 2016] Alphonsus, E. R. and Abdullah, M. O. (2016). A review on the applications of programmable logic controllers (plcs). *Renewable and Sustainable Energy Reviews*, 60:1185–1205. 8

[Asghar et al., 2019] Asghar, M. R., Hu, Q., and Zeadally, S. (2019). Cybersecurity in industrial control systems: Issues, technologies, and challenges. *Computer Networks*, 165:106946. 2, 7

[Bhamare et al., 2020] Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K., and Meskin, N. (2020). Cybersecurity for industrial control systems: A survey. *computers & security*, 89:101677. 1

[Case, 2016] Case, D. U. (2016). Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 388(1-29):3. 3, 13

[Chen et al., 2015] Chen, B., Pattanaik, N., Goulart, A., Butler-purry, K. L., and Kundur, D. (2015). Implementing attacks for modbus/tcp protocol in a real-time cyber physical system test bed. In *2015 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pages 1–6. 8, 17

[Chen and Abu-Nimeh, 2011] Chen, T. M. and Abu-Nimeh, S. (2011). Lessons from stuxnet. *Computer*, 44(4):91–93. 3

[Dehlaghi-Ghadim et al., 2023a] Dehlaghi-Ghadim, A., Balador, A., Moghadam, M. H., Hansson, H., and Conti, M. (2023a). Icssim—a framework for building industrial control systems security testbeds. *Computers in Industry*, 148:103906. 2, 14, 16, 25

[Dehlaghi-Ghadim et al., 2023b] Dehlaghi-Ghadim, A., Moghadam, M. H., Balador, A., and Hansson, H. (2023b). Anomaly detection dataset for industrial control systems. *IEEE Access*. 2

[Di Pinto et al., 2018] Di Pinto, A., Dragoni, Y., and Carcano, A. (2018). Triton: The first ics cyber attack on safety instrument systems. *Proc. Black Hat USA*, 2018:1–26. 3

[Drias et al., 2015] Drias, Z., Serhrouchni, A., and Vogel, O. (2015). Analysis of cyber security for industrial control systems. In *2015 international conference on cyber security of smart cities, industrial control system and communications (ssic)*, pages 1–8. IEEE. 1, 8

[Dutertre, 2007] Dutertre, B. (2007). Formal modeling and analysis of the modbus protocol. In *International Conference on Critical Infrastructure Protection*, pages 189–204. Springer. 8, 10

[Etxezarreta et al., 2024] Etxezarreta, X., Garitano, I., Iturbe, M., and Zurutuza, U. (2024). On the use of minicps for conducting rigorous security experiments in software-defined industrial control systems. *Wireless Networks*, pages 1–14. 26

[Evancich and Li, 2016] Evancich, N. and Li, J. (2016). Attacks on industrial control systems. *Cyber-security of SCADA and other industrial control systems*, pages 95–110. 4

[Falliere et al., 2011] Falliere, N., Murchu, L. O., Chien, E., et al. (2011). W32. stuxnet dossier. *White paper, symantec corp., security response*, 5(6):29. 3

[Hemsley et al., 2018] Hemsley, K. E., Fisher, E., et al. (2018). History of industrial control system cyber incidents. Technical report, Idaho National Lab.(INL), Idaho Falls, ID (United States). 4

[Hu et al., 2018] Hu, Y., Yang, A., Li, H., Sun, Y., and Sun, L. (2018). A survey of intrusion detection on industrial control systems. *International Journal of Distributed Sensor Networks*, 14(8):1550147718794615. 2

[Igure et al., 2006] Igure, V. M., Laughter, S. A., and Williams, R. D. (2006). Security issues in scada networks. *computers & security*, 25(7):498–506. 8

[Kilovaty, 2022] Kilovaty, I. (2022). Cybersecuring the pipeline. *Hous. L. Rev.*, 60:605. 4

[Langner, 2011] Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51. 3, 20

[Mathur and Tippenhauer, 2016] Mathur, A. P. and Tippenhauer, N. O. (2016). Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pages 31–36. IEEE. 30

[McLaughlin et al., 2016] McLaughlin, S., Konstantinou, C., Wang, X., Davi, L., Sadeghi, A.-R., Maniatakos, M., and Karri, R. (2016). The cybersecurity landscape in industrial control systems. *Proceedings of the IEEE*, 104(5):1039–1057. 1

[Miller et al., 2021] Miller, T., Staves, A., Maesschalck, S., Sturdee, M., and Green, B. (2021). Looking back to look forward: Lessons learnt from cyber-attacks on industrial control systems. *International Journal of Critical Infrastructure Protection*, 35:100464. 4

[Morris and Gao, 2013] Morris, T. H. and Gao, W. (2013). Industrial control system cyber attacks. In *1st International Symposium for ICS & SCADA Cyber Security Research 2013 (ICS-CSR 2013)*. BCS Learning & Development. 2

[Morris et al., 2015] Morris, T. H., Thornton, Z., and Turnipseed, I. (2015). Industrial control system simulation and data logging for intrusion detection system research. *7th annual southeastern cyber security summit*, pages 3–4. 5

[Ratnam et al., 2017] Ratnam, E. L., Weller, S. R., Kellett, C. M., and Murray, A. T. (2017). Residential load and rooftop pv generation: an australian distribution network dataset. *International Journal of Sustainable Energy*, 36(8):787–806. 17

[Slowik, 2018] Slowik, J. (2018). Anatomy of an attack: Detecting and defeating crashoverride. *VB2018, October*. 4

[Slowik, 2022] Slowik, J. (2022). Zeroing in on xenotime: Analysis of the entities responsible for the triton event. 3

[Stouffer et al., 2011] Stouffer, K., Falco, J., Scarfone, K., et al. (2011). Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16. 1

[Swales et al., 1999] Swales, A. et al. (1999). Open modbus/tcp specification. *Schneider Electric*, 29(3):19. 9, 10

[Tavallaee et al., 2009] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee. 22

[Thomas, 2008] Thomas, G. (2008). Introduction to the modbus protocol. *The Extension*, 9(4):1–4. 9

[Uslar et al., 2013] Uslar, M., Specht, M., Dänekas, C., Trefke, J., Rohjans, S., González, J. M., Rosinger, C., and Bleiker, R. (2013). *Standardization in smart grids: introduction to IT-related methodologies, architectures and standards*. Springer. 13

[Williams, 1994] Williams, T. J. (1994). The purdue enterprise reference architecture. *Computers in industry*, 24(2-3):141–158. 11, 14