UCLouvain

École polytechnique de Louvain

Build a control system for a mini factory to demonstrate cyber-attacks

Author: **Romain Barbason**
Supervisor: **Ramin Sadre**
Readers: **Cristel Pelsser, N'djonissè Sossou**
Academic year 2025–2026
Master [120] in Computer Science

Abstract

ICS systems are becoming increasingly connected, but this increases the risk of cyber attacks, requiring more vulnerability analysis and penetration testing on systems such as PLCs, which were not designed with security measures in mind. This paper demonstrates our physical testbed made of a Revolution Pi Connect+ running NodeRED and controlling a Fischertechnik mini-factory, and our remote setup, which is used to simulate a cloud server, to demonstrate cyber attacks at various ICS levels. A significant number of these attacks, across ICS levels 0 to 4, had a measurable and impactful effect on the network-level, ranging from ARP poisoning, Denial of Service, RST attacks, to application-level, such as replay attacks and NodeRED Flow Injection. Most of these attacks are possible due to a lack of security measures. These results demonstrate that the primary danger is not sophisticated exploitation, but rather the absence of basic security measures on the network architecture and application configuration, whose presence would have prevented the majority of these attacks.

Acknowledgments

Thank you to Rémi, Pierre, Yinan and François for their enormous and constant help and support this year. I couldn't have done it without all of you, working in your office every day was truly a great time and a blessing.
And thank you to my family, friends and partner who supported me and my long rants while working on this thesis. This would have been much harder without all of you.

Use of Artificial Intelligence

Artificial intelligence was used to improve English phrasing with DeepL Write, as well as for generating and fixing code for different attacks and for finding and refining ideas with Claude AI.

Contents

Chapter 1

Introduction

1.1 Context

Industrial Automation has been and still is a crucial part in the industry, to lessen the workload of humans by letting machine automating the process. With years and years the systems used are becoming more and more connected, more Industrial Control Systems (ICS), such as Programmable Logic Controllers (PLC) and Supervisory Control and Data Acquisition (SCADA), are communicating with TCP/IP, automation protocols or other communication protocols and the possibility of controlling and planning everything remotely from a cloud server is tempting more and more industries. However, these discoveries also bring new dangers. The range of possible attacks on them is increasing, so more security precautions are needed. In particular, PLCs have become a priority target for attackers for a plethora of reasons.
The main one is that many PLCs were built and put into service decades ago without any security measures in place, leaving them without the proper security requirements needed. If breached, this could lead to production shutdowns, economic losses and/or safety hazards, causing significant damage to an industry[1]. In this situation, the demand for testbeds has increased drastically, making it increasingly crucial to perform vulnerability analysis and penetration tests on ICS systems to ensure they are secure enough to prevent accidents or disasters, particularly in important sectors such as energy, including the nuclear sector[2][3][4].

1.2 Objectives

Our objective is to make a functioning testbed, based on the testbed of a paper called LICSTER, standing for Low-cost ICS Security Testbed for Education and Research, to research and test multiple attacks possible and to enlighten about

PLCs security.
Our testbed will be connected to a server, to simulate a cloud server making request remotely, for the only goal to find new attacks regarding that setup.
We will program our own SCADA with NodeRED, an open-source flow programming tool, to control manually and automatically all motors and sensors of our testbed, or "mini factory", and to automatically process given goods with the motors and sensors installed.
These attacks are either already mentioned in the LICSTER paper or "new" ones, explained and/or demonstrated in this paper[4].

Chapter 2

Background

This chapter provides the background information necessary for a good understanding of the rest of this paper. No prior knowledge of ICS systems or networking is assumed.

2.1 Industrial Control Systems

ICS is an umbrella term for multiple types of command and control systems and networks that are used to operate and/or automate industrial processes. Examples include Supervisory Control and Data Acquisition (SCADA) systems and Programmable Logic Controllers (PLCs). These systems are widely used for manufacturing and control at different scales and have many ways to communicate between each other, either with physical hardware like wires and cables, or over a network with TCP/IP or other automation protocols such as Modbus[5][6].

2.2 PLC

PLCs (Programmable Logic Controllers) are embedded systems that receive discrete and/or continuous inputs, such as temperature, speed and pressure, and generate outputs according to the programming. They are an essential part of industrial automation, and their application can vary greatly depending on the industry. In discrete manufacturing, for example, they can be used to control conveyor belts, elevators, assembly lines, machining processes and material handling. In process manufacturing, meanwhile, PLCs can be used to monitor product quality and enforce safety standards in industries such as oil refining and drug manufacturing[7][8].

2.3 Supervisory Control and Data Acquisition (SCADA)

SCADA is a system that is used to supervise and control a set of physical processes by collecting real-time data from any hardware linked to it. It is usually equipped with:

- A Human Machine Interface (HMI) to display all the information it received.
- A historian, a specialized database to record every information given by all the physical processes.
- A communication network, to communicate with all the components linked to it, either wired or wireless.
- Remote Terminal Units (RTU) linked to a Master Terminal Unit (MTU), or a standalone PLC, that collects the information of the sensors and actuators.

These systems can be either entirely local or accessible via a cloud server[9].

2.4 NodeRED

The control system is made with NodeRED, an open source, graphical, flow-based programming tool made with Node.js, originally made to connect hardware devices to APIs and online services. Flow-based programming is a programming paradigm allowing us to make applications with "black box" reusable modules, linked to each other. An example of a program in NodeRED can be seen in Figure 2.1. These black boxes, or components, are essential building blocks to make an application[10]. This paradigm allows programming novices to easily make applications by simply linking building blocks together.
NodeRED uses this paradigm with a graphical interface and uses "nodes", which are the black boxes cited above. It has a public library of community nodes, created by users, with over 6000 custom nodes, all having different functions and uses. NodeRED allows us to automate multiple workflows and import specific nodes[11].

2.5 ICS Automation Pyramid/ISA 95/IEC 62264

The attacks demonstrated and/or tested on this paper are displayed according to the Industrial Control System (ICS) level they are focusing on. We are using

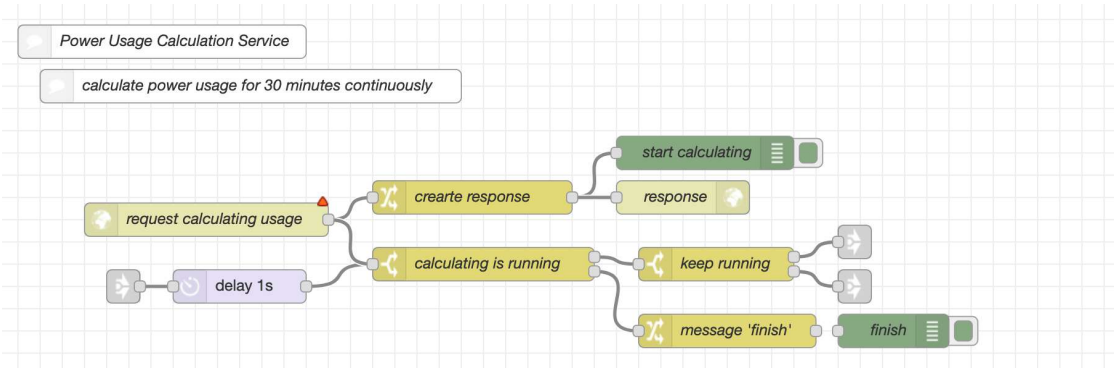

Figure 2.1: An example of a NodeRED flow calculating the power usage[11]

for this the standard structure for ICSs, stated in the ISA-95 and illustrated in Figure 2.2[12].

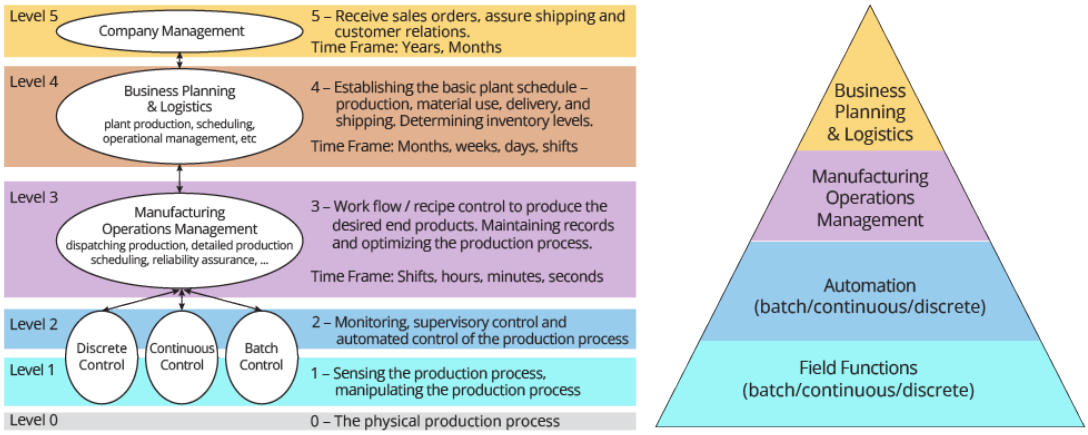

Figure 2.2: Industrial Automation Pyramid[12]

- **Level 4: Business planning and logistics**: Includes ERP (Entreprise Resource Planning) and all other activities that make a business run smoothly.
- **Level 3: Manufacturing operations management**: includes MES (Manufacturing Execution Systems) and SCADA (Supervisory Control And Data Acquisition) systems. It manages operations and optimize them.
- **Level 2: Monitoring and supervising control**: Monitoring and control of the lower layers, with in our setup, PLCs or other elements.
- **Level 1: Sensing and manipulating the production process**: Sensors,actuators,... Remote I/Os devices that read and write inputs.

- **Level 0: Physical production processes**: The machinery in itself and other assets on the same level.

It is important to note that not all sources and pyramids representing ISA-95 will look or say the same.
Some pyramids will allocate functions to lower or upper layers. For example, some will put the PLC at level 2, while others will put it at level 1. This is because the pyramid shows each layer's function, not specific systems. Some PLCs handle the monitoring and control of the layers below them (like our setup), while others only focus on retrieving data and passing it to other SCADA systems at level 2. Therefore, all differences are technically valid, depending on how the system is used.
In this paper, we will argue that the PLC is at level 2. This is because our PLC can handle both monitoring and control, which is also consistent with the official ISA-95 explanation[13].

2.6 Transmission Control Protocol (TCP)

The TCP protocol is a network protocol that allows programs and applications to communicate over a network. It ensures that every packet sent must be delivered properly and that their data is untouched. It is one of the most used protocols on the entire Internet.
Before transmitting data between a server and a user, it ensures that the connection between them will not be interrupted until said notice, this is called a Three-Way Handshake, seen in Figure 2.3:

1. The client sends a Synchronous Sequence Number (SYN) packet, to inform the server that it wants to establish a connection, and share what segment numbers it starts its segment with.

2. The server sends a SYN+ACK packet, the ACK represents the acknowledgment of the packet it received, and SYN to share what segment numbers it starts its segment with.

3. The client acknowledges the packet it received, and establish a connection with the server, data can now be transmitted

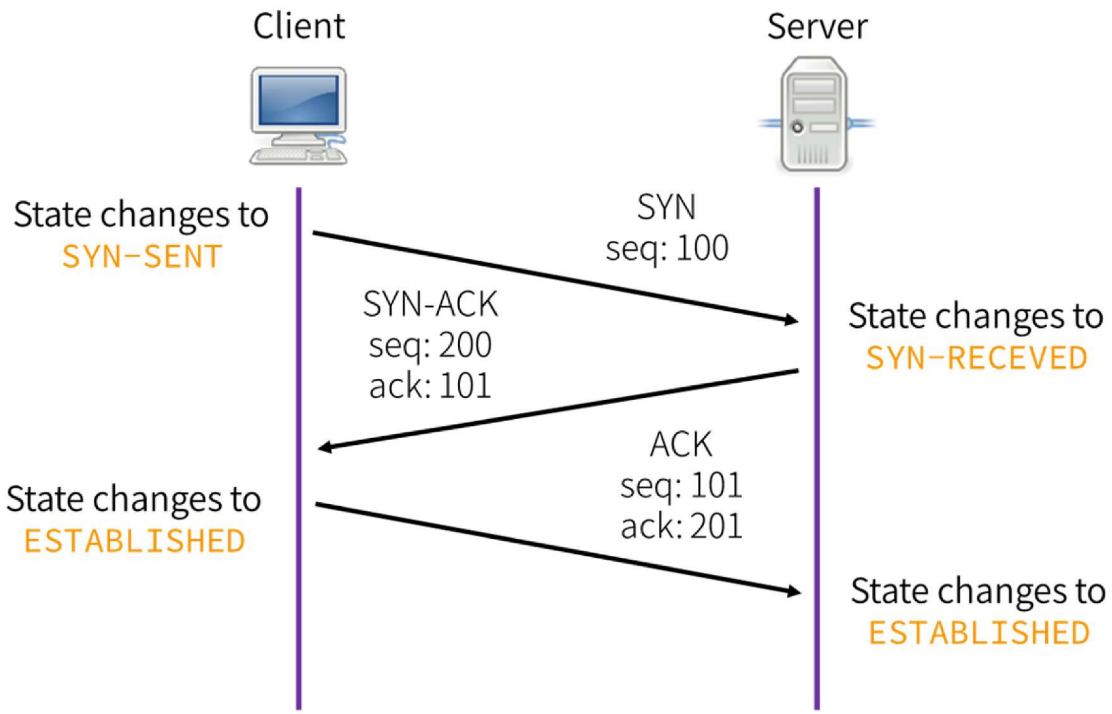

Figure 2.3: A TCP Three-Way handshake[14]

These packets are not secured by default, unless it is coupled with other protocols, such as Transport Layer Security (TLS)[15].

2.7 Transport Layer Security (TLS)

The TLS protocol is a cryptographic protocol whose goal is to provide a secure communication between two entities, a client and a server, to avoid eavesdropping and/or tampering of the data passed between them. It is widely used to encrypt communications for emails, instant messaging or Voice over IP, but is also widely used for web security with HTTPS.
Before transmitting any data, the client communicates with the server to properly secure their communications, with a TLS handshake:

1. The client sends a Client Hello to the server and shares the TLS version it supports, a random number and a list of cipher suites the client supports.

2. The server replies with a Server Hello to the client, containing the cipher suite chosen by the server, a session ID, a random number and its digital certificate, which contains its public key

3. The client verifies the server's certificate and sends a random byte string, encrypted with the server's public key, called the "premaster secret". If the server requested a certificate from the client, the client gives it with a random byte string encrypted with the client's private key.

4. If the server receives the client's certificate, it verifies it. It decrypts the premaster secret and both generates the session keys from the client random, server random and the premaster secret.

5. The client sends a "Finished" message, encrypted with a session key.

6. The server sends a "Finished" message, encrypted with a session key.

From now, and during the entire TLS session, the client and the server can now safely encrypt their communications[16][17][18].

2.8 Address Resolution Protocol (ARP)

The ARP is a communication protocol, used to link an internet layer address, such as an Internet Protocol (IP) address, to a link layer address, such as a Media Access Control (MAC) address. The internet and link layer addresses refers to the Internet protocol suite, or TCP/IP.
If a device, AKA the sender, wants to send an Ethernet frame, for example an IPv4 packet, to another device on the same local network, for which it knows the IP address:

1. The sender broadcasts on the entire network an ARP request, asking "What MAC address is linked to searchIP? Says senderIP".

2. Every device on the local network receives the request.

3. The device with the searchIP address sends to the sender an ARP response, saying "I am searchIP, my MAC address is aa:bb:cc:dd:ee".

4. The two devices update their ARP cache, and add the IP address of the other with its linked MAC address.

The ARP cache is automatically purged regularly to free up space (unless this is a fixed ARP cache)[19].

2.9 Secure Shell (SSH)

SSH is a connection protocol whose goal is to provide secure remote login and other secure network services over an insecure network.
It encrypts all data packets sent between two devices with a public key cryptography, allowing tunnelling to transmit packets to networks that do not support their type. It also enables port forwarding, which transfers data from one port to another. It is widely used to control servers remotely, transferring files, etc. [20][21].

2.10 Internet Control Message Protocol (ICMP)

ICMP is a network protocol, this protocol unlike many others like TCP or UDP, is not primarly used to transfer data, but rather to signal any network communication problem between two peers in a network and diagnose these problems. It is widely used in routers, switches and other network devices[22][23].

2.11 Representational State Transfer (REST API)

A REST API is an application programming interface (API) that follows the design principles of the Representational State Transfer (REST) architecture style, which are rules on how to build an efficient web API, these APIs are often called "RESTful APIs".
An API allows a user or an application to get data or service from another application. In order for an API to be considered RESTful it must follow these constraints:

- Stateless client-server communication: The server can't store any data from a client request.
- Cacheability: Data can be cacheable on either side, as long as it is allowed by the server.
- Client/Server independence: The client and the server must be independent from each other, one can't alter the application of the other, and vice versa.
- Uniform interface: An API request for a certain data should look the same if this request is sent from somewhere else.
- Layered system architecture: The system needs to organize each type of server into layers, or hierarchies. This should not be noticeable by the client.

- Code on demand (optional): It is possible to send executable code if requested.

REST APIs send request with HTTP requests, it uses GET requests to get specific data, POST requests to create specific data, PUT requests to modify specific data and DELETE request to delete specific data[24][25].

Chapter 3

State of the art

This chapter presents the research that has already been conducted on this topic, in order to contextualise our work. This includes the LICSTER paper, on which this paper and our testbed are based, as well as the various defensive techniques that are relevant to the attacks tested later in this paper.

3.1 Testbeds

A testbed is a platform that reproduces real scenarios to perform testing of new technologies, tools and theories. There are multiple types of testbed, each offering their pros and cons:

- Physical testbeds: Uses real hardware and software, perfect to have realistic results and to use certain devices, but expensive in construction and maintenance and they are not easily replicable.
- Hybrid testbeds: Uses both real hardware and software simulations. Only some of the components are simulated, while the rest are real. This is an effective way of achieving more realistic results while reducing the cost of the testbed.
- Virtual testbeds: Uses software simulations and emulations with multiple devices to reproduce every component and its network. While it is much cheaper and safer for simulating dangerous procedures, such as nuclear ones, it is difficult to simulate physical processes correctly in order to achieve realistic results.

Its use in ICS systems is particularly important, since many industries simply cannot be stopped for months of testing on PLCs, due to their function, such as dams or power plants, or due to the risk of financial loss, such as with a production

line. In the context of ICS systems, testbeds are mostly used for vulnerability analysis, education and testing of defense mechanisms, as seen in [2].
It is also widely used outside of ICS systems, particularly for the Internet of Things (IoT), for example in robots, smartphones and sensors, and wireless sensor networks (WSN)[26][27][28].

3.2 Low-cost ICS Security Testbed for Education and Research (LICSTER)

This thesis references the LICSTER paper numerous times because its content is highly relevant to our testbed. At the time of writing, it is the only low-cost, open-source ICS physical testbed, making it perfect for education and research, whose feasibility was proven by its numerous tests.
Their setup consists of multiple components:

- 2 custom-made remote IOs, communicating with Modbus/TCP.
- A self-made Human Machine Interface (HMI) running on a dedicated Raspberry Pi with a touchscreen.
- A Raspberry Pi 3 with OpenPLC acting as the PLC.
- A SCADA system made with ScadaLTS on a dedicated Raspberry Pi.
- The same physical process that we have in our setup, which is explained in subsection 4.1.2

The entire testbed uses Modbus/TCP to communicate, contrary to our testbed which uses NodeRED's REST APIs, with HTTPS (or HTTP in its unsecured configuration).
Numerous attacks, ranging from ICS levels 0 to 2, have been successfully carried out on their setup, including DoS and MitM attacks, showing that their testbed could be used for regular PLC training and research.
While this thesis has multiple attacks in common with the LICSTER paper, the main difference is that our remote setup simulates a cloud server communicating with the physical process, which is also suggested by the LICSTER paper as future work.

3.3 Cloud-based ICS

Over the years, many industries have moved their ICS, such as SCADA, to cloud servers, which are accessible by employees and other authorised personnel anywhere,

anytime. This change allows for better data accessibility and increased cost efficiency, flexibility, optimisation, availability and scalability. However, these benefits come at a price: a system that moves from a LAN to a cloud server is exposed to many more attacks, such as MitM, DoS and DDoS, replay attacks, etc. Therefore, they must put more effort into securing their systems. Such security systems and practices have been researched, developed and tested, such as cloud-based IDS in [29] and [30], encrypting SCADA systems' communications, more secure protocols, authentication, regular risk assessments, etc.[29][30][31][32].

3.4 Demilitarized Zone (DMZ)

A demilitarized zone, in the context of network security, is a sub-network in a larger network, separating an un-trusted network (generally Internet) of the rest of the network with firewalls, giving an extra layer of security. Only services who requires Internet are in the DMZ (Web servers, mail servers,...), reducing to only what's in the DMZ what an attacker can attack[3].
Use of one (or multiple) DMZs to secure SCADA systems and PLCs have been proven effective, although with a decreased transmission rate as a tradeoff[33].

3.5 Intrusion Detection System (IDS)

Intrusion Detection Systems are commonly used to monitor a host or network and detect, by signature or statistical anomaly, any attack or abnormal process.
In our case, network IDSs are primarly used for PLCs, since the majority are lacking most of the hardware requirements (storage, memory space,...) to run a host based IDS.
Every malicious process has a signature, a predetermined pattern, that defines them uniquely, this is what a signature based IDS works on. It detects and tracks any of these malicious signatures and kick them out of the network, although this is a most common choice for its efficiency, its main problem is that any unknown signature or new attacks will not necessarily be detected.
A prime example of this is the Stuxnet worm, a malware that spreads across a network, originally targeted only at the Iranian nuclear program to sabotage their centrifuges, which were obsolete, inefficient and prone to damage upon high variation of speed. This worm quickly spread outside of Iran and went into the wild, its signature and existence was unknown, making it invisible for many years since the U.S government launched it[34][35][36].
Another type of IDS are anomaly based IDS, which are comparing a multitude of traffic parameters to see if their values are abnormal. These parameters can differ

from port numbers, bandwidth, payloads,... Any uncommon values are directly sent to the network administrator, who takes the appropriate actions to counter the attack.
All IDS also have an Intrusion Prevention System(IPS) to reconfigure the firewall to block the traffic related to an intrusion[3][37].

3.6 Protocol Modification

Many network attacks succeed not because of an elaborate exploitation, but because the network protocols themselves lack security, such as lacking authentication, confidentiality or data integrity.
Examples include protocols without timestamps, which allow replay attacks since messages are not dated and can easily be replayed if captured by an attacker. Another example is using an outdated version of TLS, which allows an attacker to perform a downgrade attack by choosing weaker or no ciphers during the TLS handshake. Both attacks are also explained and demonstrated in section 4.3.
Using the latest protocol versions or modifying the protocols directly greatly increases the network's security.
Modifications have been made to protocols such as Modbus, which is used for communication between PLCs. These include the use of a Secure Hash Algorithm 2 (SHA-2) digest for data, as well as public and private keys for authentication, and a timestamp for each message sent[3].

3.7 Honeypot

A honeypot is a highly monitored decoy on a network, mimicking a real system, for the sole purpose of attracting potential attackers and to gain information on these attackers and their attack methods and behaviors.
None of the entities in a honeypot are, and should be, connected to the actual system the network uses, to avoid an attacker making his way out of the honeypot to the actual network the concerned entity uses. Honeypots are great at gathering information about new attacks and exploration trends, as well as giving an early warning to counter these trends.
Using honeynets, a multitude of honeypots on the same network, is recommended to improve the quality and quantity of the data. Since this network should never be used by anyone, the only users possible are potential intruders making unauthorized interactions with it, allowing us to get information on them, even after the intrusion. PLCs are not an exception to it, as fake SCADA infrastructure with fake PLCs are

used to gather as much information as possible, giving it an advantage to secure these systems early[38][39].

Chapter 4

Solution

This chapter presents our local and remote setups, the choices we made in creating them, and their uses and goals in this paper.
First, we explain the purpose of each component used and why these components were chosen for their corresponding setup, AKA what makes the setup realistic. We also explain the different applications and/or programs that come with it or were made specifically for this paper.
Secondly, we explain how each of the tested attacks can be carried out and how they work. We also explain the setups on which they can be performed and how to secure against them. Finally, we categorize the attacks by scope, difficulty, impact and detection. The attacks are ordered by the ICS levels they impact.

4.1 Local Setup

Although the 'mini-factory' model is the same, the components and configuration of the testbed are different from those of the LICSTER testbed.
The local setup is composed of:

- A fischertechnik "punching machine with conveyor belt", this is the physical process simulating a factory, hence the name "Mini-Factory"
- A Revolution Pi Connect +, that we will call Revpi for short, acting as the PLC, as well as the SCADA system
- A Revolution Pi DIO, attached to the revpi, handling the inputs and outputs between the mini-factory and the PLC.

The realism of this setup comes from directly to the technology we are using. Every single components is widely used in industries, factories, and in other industrial settings. The Fischertechnik is actively used for PLC training, by small and much

larger companies, since it mimics very well the functioning of a production line, and because it contains everything that needs to be tested in PLC training, such as sensors, actuators, motors, conveyors, etc.
The Revolution Pi Connect + and the DIO are actively used in factories for production lines, and complies with the industrial regulations needed for it. The Local setup can be seen in the Figure 4.2 and a diagram showing the ICS levels of the local setup in the Figure 4.1.

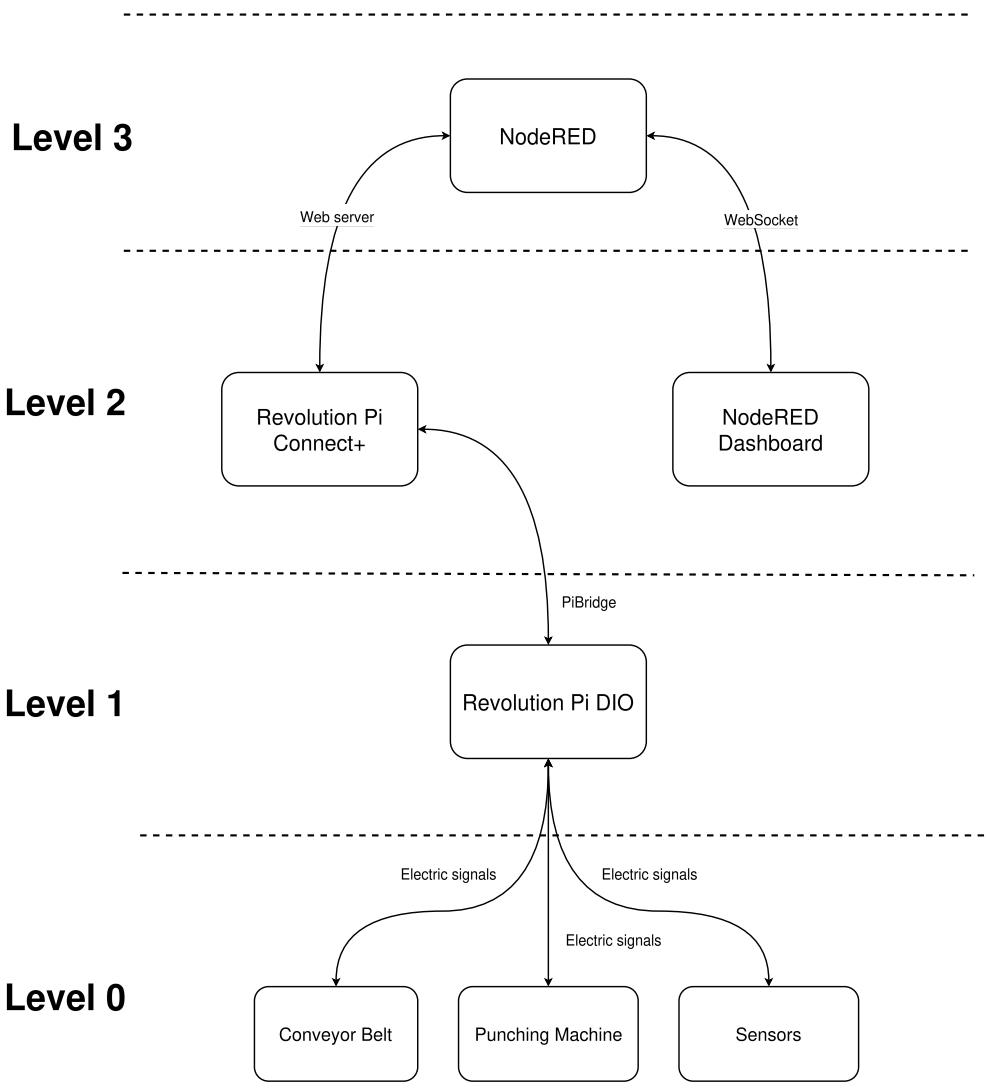

Figure 4.1: ICS representation of our local setup

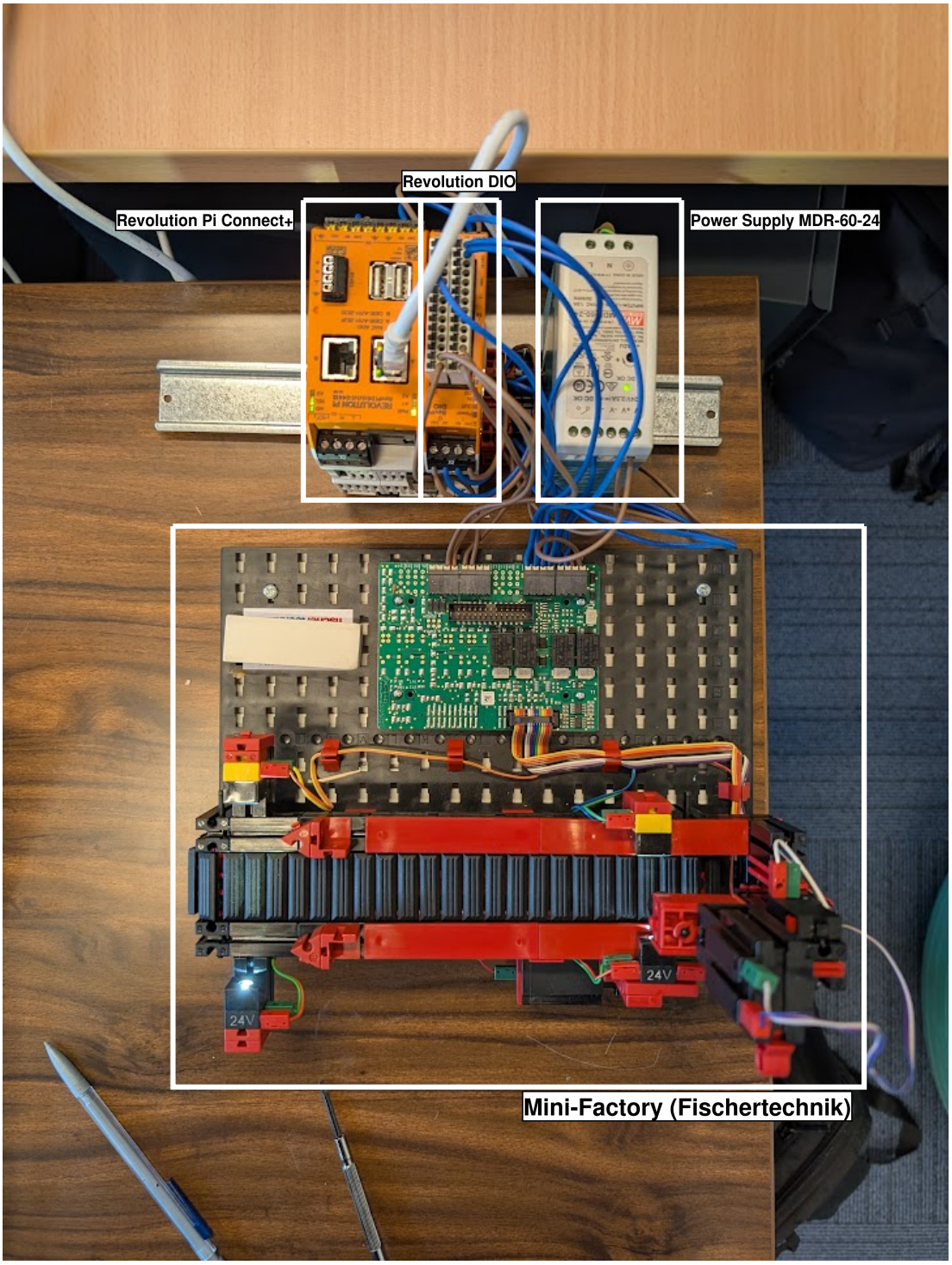

Figure 4.2: The local setup

4.1.1 Revolution PI

The PLC we are testing in this paper is a Revolution Pi Connect + from Kunbus, based on a Raspberry Pi Computes Module 3 and features a 1.2 GHz quad core processor and 1GB of RAM, with an open source operating system based on the Raspberry Pi OS.
The operating system is the Revpi Bookworm, an OS based on the Raspberry Pi OS version Bookworm Debian 12.
It is equipped with numerous tools and support many protocols for automation, IIoT, energy networks and other areas, such as:

- **Cockpit**: Used to configure the revpi, allows network configuration, user configuration, status and log views to be managed via a browser.
- **PiCtory**: Create and manage the configuration file for a Revpi system, such as adding another module, like a Revpi DIO.
- **NodeRED**: see section 2.4
- **Modbus**: Used to integrate the modbus protocol into the Revpi.

Attached to it with a PiBridge, a physical bridge between modules, allowing Ethernet and RS485 channels, is a Revolution Pi DIO, an expansion module that gives 14 digital inputs and 14 digital outputs, allowing us to control the testbed[40].

4.1.2 Mini-Factory (Fischertechnik)

The Mini-Factory is a "Punching machine with conveyor belt" from Fischertechnik, which can be seen in Figure 4.3.
Two photosensors, a type of sensor that allows a current to pass if it detects light, have been installed: one at the start of the conveyor belt and the other at the end beneath the punching machine.
There are also four different motors: two related to the conveyor belt, which make it move forwards and backwards, and two related to the punching machine, which make it move upwards and downwards.
A list of every inputs and outputs available to the machine can be seen in Table 4.1, these I/Os are transmitted directly with electrical wires plugged in the controller in front of the machine. In our setup these wires are plugged to the Revpi DIO.

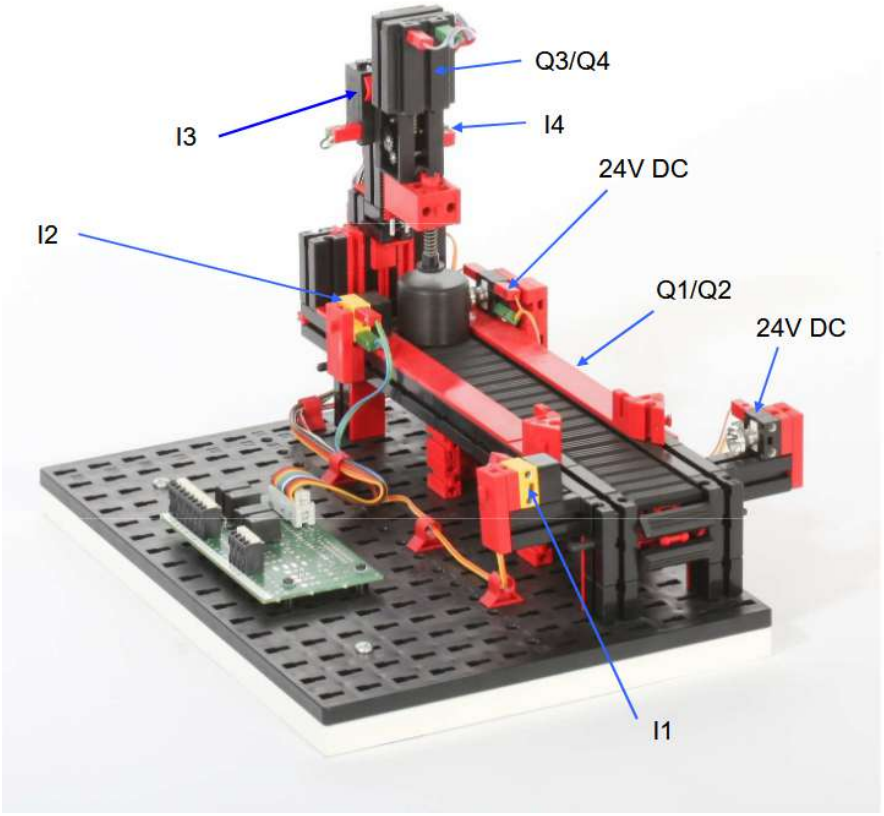

Figure 4.3: Punching Machine with Conveyor Belt, Fischertechnik[41]

Name	**I/O**	**Description**
I1	Input	Detects if any objects comes in front of the 1st photosensor
I2	Input	Detects if any objects comes in front the 2nd photosensor
I3	Input	Detects if the punching machine is at its maximum height
I4	Input	Detects if the punching machine is at its minimum height
Q1	Output	Moves the conveyor belt forward (towards the punching machine)
Q2	Output	Moves the conveyor belt backward
Q3	Output	Moves up the punching machine
Q4	Output	Moves down the punching machine

Table 4.1: The inputs and outputs of the punching machine

4.1.3 Control System

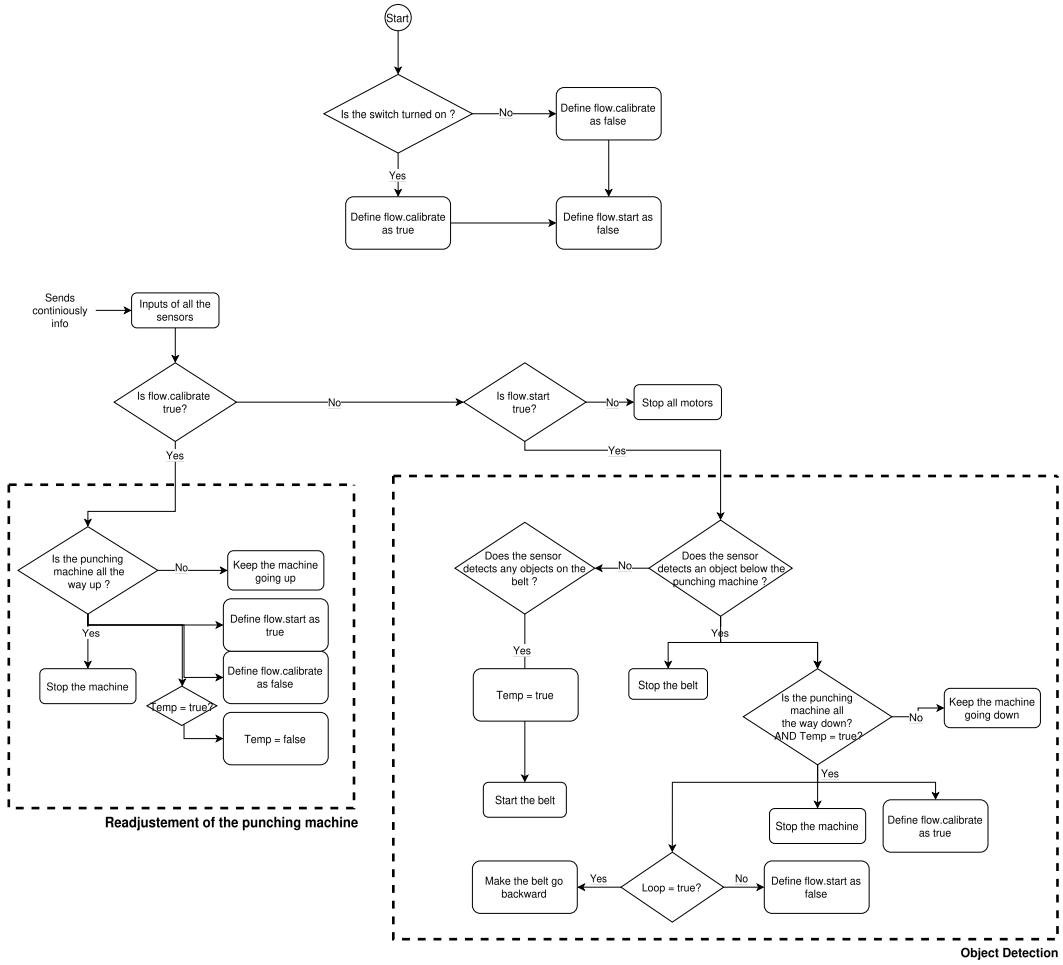

Figure 4.4: A diagram explaining the control system and its different modules

The control system is entirely coded in NodeRED and is made up of one unique flow. Multiple custom nodes were added, such as the revpi-nodes, to enable us to view and modify the state of the pins on our RevPi DIO. These nodes simply communicate with port 8000 of the RevPi's IP address to retrieve information on all the pins.
This control system allows us to fully control the physical process and every controllable aspect of it. It contains a dashboard, with different modes of operation depending on the need:

1. **Manual mode**: To move every motor manually, by the click of a button.

2. **Automatic mode**: To make the mini-factory act as Figure 4.4, and later explained below.

3. **Loop mode**: To make the Automatic mode loop on itself, making the machine run indefinitely, or until no object is detected.

The automatic mode, as seen in Figure 4.4, is meant to simulate a real factory, taking goods and processing them as follows:

1. At the start, we are checking if the "Automatic" switch is turned on, if it is, we are calibrating the punching machine, so it is at its default position.

2. When the "Readjustment of the punching machine" phase is done, the program starts (as long as flow.start is true, giving the authorization to start).

3. During the "Object Detection" phase, we first check if any objects is detected by the sensor below the punching machine, if not, we check if the sensor at the beginning of the conveyor belt detects an object, if it does, the variable "temp" is set to true, meaning that an object is currently moving on the conveyor belt towards the punching machine, and the conveyor belt starts.

4. If an object is detected under the punching machine, the conveyor belt stops, and the punching machine is activated.

5. If the "Loop" mode is activated, the belt go backward until the first sensor, and the program repeats.

4.2 Remote Setup

To simulate using the control system in the cloud, we have set up an Ubuntu 24.04 virtual machine (VM) with Node-RED installed. This VM, which we will call the "Cloud VM", is connected to the Revolution Pi Connect+ via an SSH tunnel that forwards port 8000 to the same port on the Cloud VM. A diagram and an ICS representation of the remote setup can be found in Figure 4.5 and Figure 4.6
These ports transfer the state of each RevPi DIO pin to the Cloud VM. Under normal circumstances, users would be able to access the dashboard directly via the enterprise's website. In this scenario, we simulate this by forwarding the port used by Node-RED to users via another SSH tunnel.
To simulate an attacker, we use another VM, that we will call the "Attacker VM", on the same subnet as the Cloud VM, to avoid triggering the university's firewall.

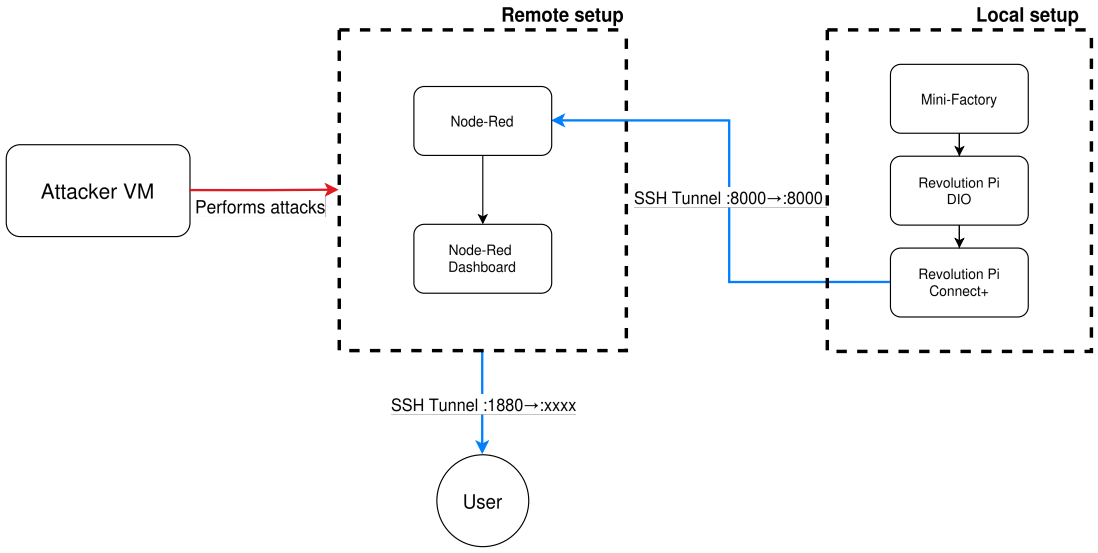

Figure 4.5: Diagram of the remote setup

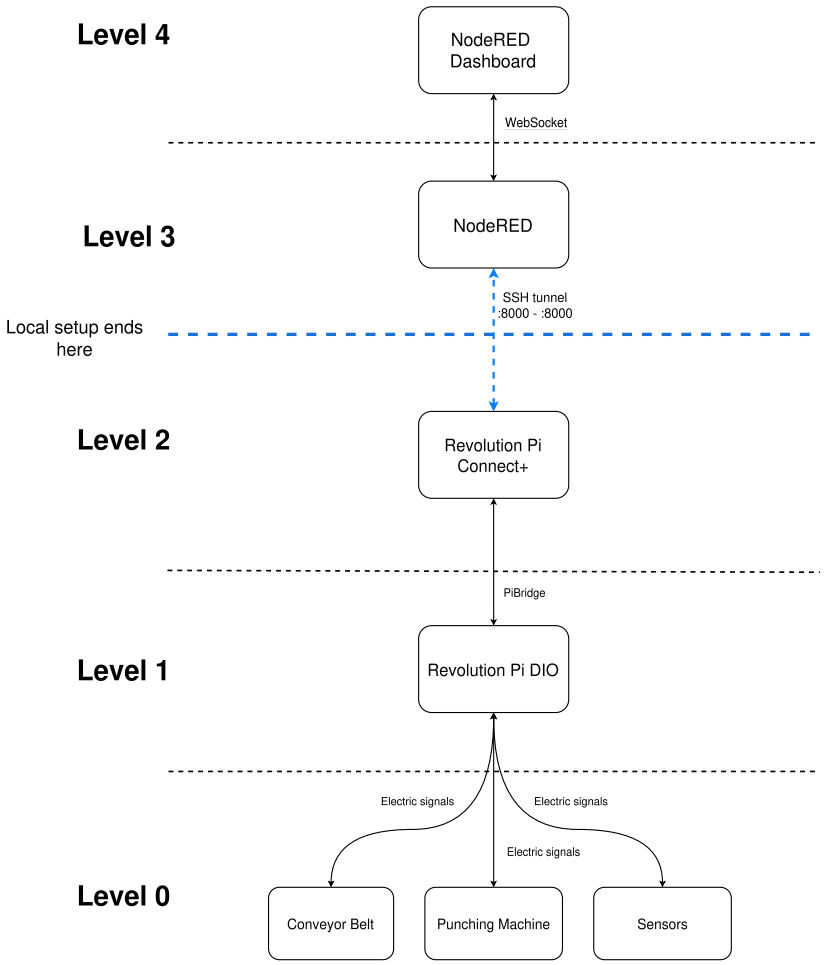

Figure 4.6: ICS representation of our remote setup

4.3 Attacks

As said in section 2.5, the different attacks demonstrated in this paper are ordered by what ICS levels they are impacting. The Table 4.2 lists every attack that has been tested, with each attack divided into multiple categories.
This table does not define the limits of what can be done in either setup; other attacks are possible at each level. However, due to time constraints, the most relevant and impactful attacks were chosen for testing and analysis.
Other attacks were not chosen due to their lack of relevance or impact, or simply because there was not enough time to analyze them without compromising the quality of testing the other attacks.

1. **ICS Levels**: Which ICS levels the attack impacts.
2. **Attacks**: The name of the attack
3. **Remote**: If this attack can be done on the remote setup.
4. **Local**: If this attack can be done on the local setup.
5. **Attacker Position**: The network scope required to perform the attack. Four values are possible: **Physical** (requires physical access to the device), **LAN** (requires access to the local network), **Internet** (can be performed remotely over the internet), or **Network** (can be performed from either a local network or the internet).
6. **Difficulty**: Measures the difficulty to perform the attack, theoretically or practically.
7. **Impact**: The level of impact of the attack AKA how important the damage done by the attack is.
8. **Detection**: How easy is it to detect the attack, by an IDS or just a network supervisor.

ICS Levels	Attacks	Remote	Local	Attacker Position	Difficulty	Impact	Detection
0	Physical Damage	x	✓	Physical	Low	High	Easy
0	Sabotage	x	✓	Physical	Medium	High	Medium
1	Disconnect wires/power	x	✓	Physical	Low	High	Easy
1	MitM spoof values PLC-DIO	x	✓	Physical	High	High	Hard
2	USB Injection	x	✓	Physical	High	High	Hard
2	ARP Poisoning	✓	✓	LAN	Low	Medium	Easy
2	DoS Revpi Connect+	✓	✓	Network	Low	High	Easy
2	Sniff Network	✓	✓	LAN	Low	Medium	Hard
2	RST Attack	✓	✓	LAN	Low	Medium	Hard
2	Replay Attack	✓	✓	LAN	Low	High	Hard
2	Downgrade Attack with MitM	✓	✓	LAN	High	High	Hard
3	NodeRED Flow Injection	✓	✓	Network	Low	High	Medium
3	Malicious NodeRED Node	✓	✓	Internet	High	High	Hard
3,4	DoS NodeRED	✓	x	Network	Low	High*	Easy
3	DoS SSH Tunnel	✓	x	Network	Low	High	Easy

*High on the local setup, but low on the remote setup

Table 4.2: The list of attacks performed/that could be performed

4.4 Level 0

Since this level only include physical process most of the attacks possible are physical. Either physical violence on the machines themselves, or physical sabotage,

by operating the machine outside its safety limits (overspeed, overpressure,...). These attacks were not tested, to continue using the setup during the writing of this paper.

4.5 Level 1

With this level getting all of the information from the physical process below, the most impactful attacks consist of tampering physically with the wires or the I/O device to read and send fake values to the upper layers by doing a MitM. This was not tested, as it enters the scope of hardware hacking and electronics, which is not our domain of expertise.

4.6 Level 2

4.6.1 Denial of Service

A denial of service, or DoS, consists of flooding a network or host with a heavy amount of traffic to make the service slow or crash, preventing users to access any service hosted on it. Every services that rely on a host or network can be affected. There are multiple ways to make this attack, one of the most common that is used in this paper is a SYN Flood, which sends multiple TCP SYN requests to connect to the server, but does not finish it, leaving the port waiting for a response and unavailable for further requests.
Distributed Denial of Service uses multiple machines to do a DoS on a same target, using a network of hijacked machines, named botnet, making it more powerful and difficult to identify and stop.
While DoS attempts are inevitable, multiple ways exist to reduce or stop completely this attack, such as DoS protection services (like Cloudflare), reduce the number of accessible devices connected to Internet (with a DMZ for example, see section 3.4), install a firewall, etc.[42].

4.6.2 ARP Poisoning

ARP Poisoning, as seen in Figure 4.7, consists of sending fake ARP (Address Resolution Protocol) replies to the gateway indicating that the attacker's MAC address belongs to the target's IP, and vice versa with the target and the gateway's IP.
The entire traffic destined to the target is re-routed to the attacker, who can see and possibly modify it before sending it to its original destinator, making a Man-in-the-Middle attack (MitM).

An attacker could also make a Denial of Service (DoS) by not sending the packets to the target, cutting him any connection, as demonstrated in subsection 4.6.3. This attack is carried out exclusively on the local area network (LAN). There are multiple ways to counter this attack, such as encrypting with public and private keys, or using a secondary cache containing entries relating to ICMP responses and deleting any IP duplicates in the cache[43][44].

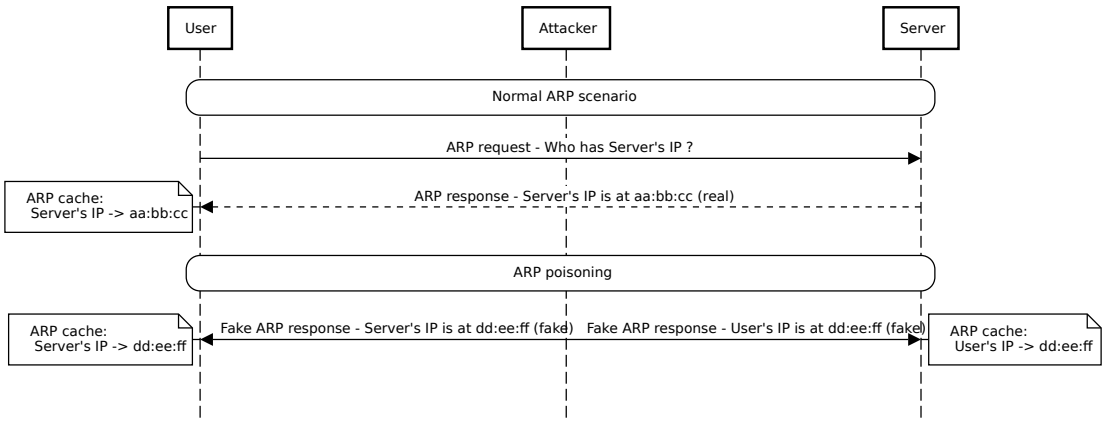

Figure 4.7: A sequence diagram of an ARP Poisoning

4.6.3 RST Attack

If a TCP connection needs to be closed suddenly, a TCP RST packet is sent to the other recipient, asking it to discard future packets and communication.
An attacker can then use a spoofing method of any choice (for example ARP Poisoning) on a target , to send a forged RST packet to the server, ending communications between the target and the server. This attack is a type of DoS attack, that can be used as a step towards a bigger attack, such as a downgrade attack.
Multiple countermeasures have since been made, such as authentication of TCP packets, with timestamps or a cryptographic hash function[45][46].

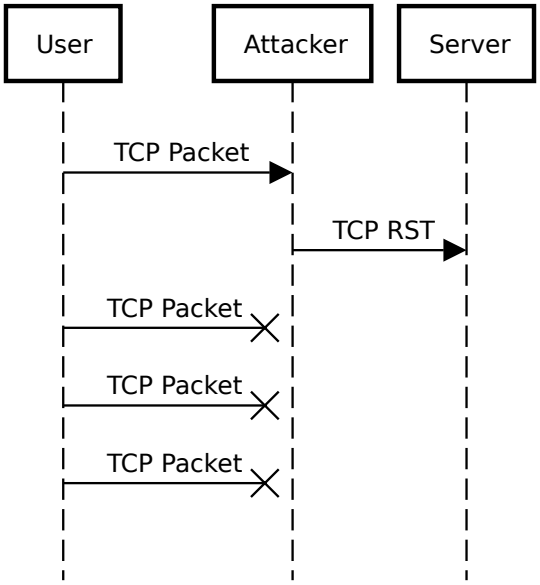

Figure 4.8: A sequence diagram of a RST attack

4.6.4 Downgrade Attack

The downgrade attack's goal, as illustrated in Figure 4.9, is to interfere between a user and a server to force them use a weaker mode of operation (e.g a weaker cipher, a weaker version of a protocol,etc) than what they would have chosen directly. This is done with a MitM attack, to intercept the packets between the user and the server. The variety of how this attack is performed depends on the mode of operation targeted, some examples would be:

1. Failing purposely the TLS handshake to force the server to accept a lower TLS version, until it uses a deprecated version of TLS. This only works if the server accepts version fallback. This was done in the BEAST attack.

2. Intercepting the ClientHello to modify its list of ciphers with weaker ciphers, so that the attacker can exploit it and read/modify the application data (done in the FREAK and Logjam attack), or no cipher and the attacker just read and modify without any exploit(done in the STARTTLS Stripping attack).

In our implementation, we used a combination of an ARP Poisoning for the MitM and an RST Attack to reset the TLS connection and restart a TLS handshake beforehand, therefore doing the second type of our examples, giving weak ciphers and NULL as a cipher.
This was done with a custom python script to do the ARP Poisoning, and then launch the RST attack and the Downgrade attack, in this order.

Our version's development begun towards the month of October 2025, but by November 2025, an update made mandatory the use of TLS 1.3, of which we were only notified in February 2026, patching this attack entirely[47][48][49].

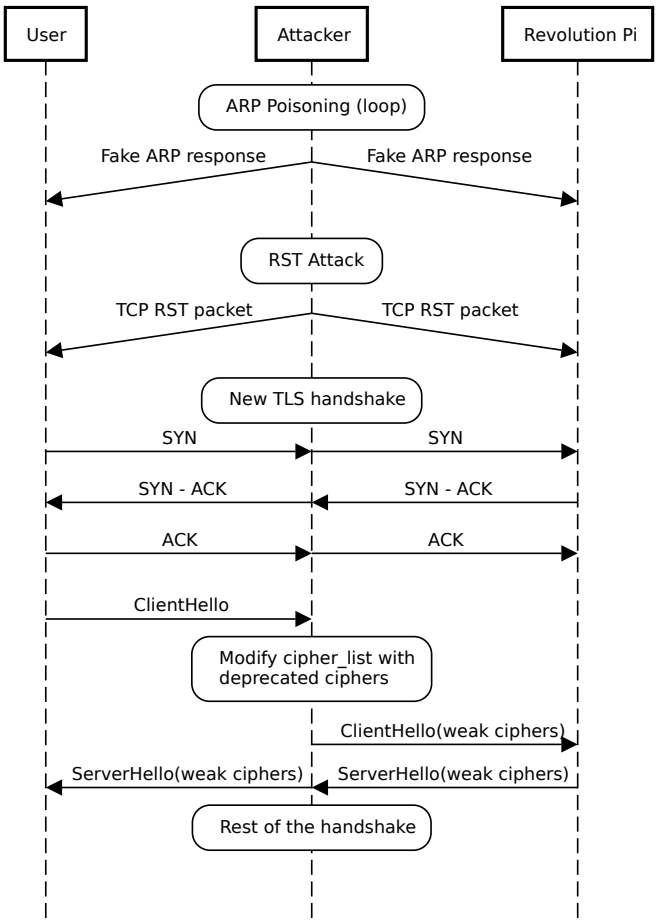

Figure 4.9: A sequence diagram of our downgrade attack on the Revolution Pi

4.6.5 Replay Attack

A replay attack consists of replaying or delaying a message, obtained by eavesdropping a network, to force the system to do what the attacker wants. This attack is particularly easy to reproduce, since it doesn't need any decryption of the message, simply to record it and play it back.
There are countermeasures to this attack, such as client-hello recording, whereby the server keeps every ClientHello message received and checks for duplicates when it receives one. Another countermeasure is the session ticket, which is associated with a 'session ticket key' that is used to decrypt the data carried in the message. The key is then deleted by the server immediately when the handshake is finished.

If someone were to replay the message, they would receive an error since the server cannot decrypt the data[50].

4.6.6 USB Injection

If an attacker has physical access to the PLC, a USB stick can be used as a deadly weapon. It can easily be used to store malware and inject it into the PLC, and it can even cause physical damage. There are many possible types of attack via a USB stick; famous examples include:

- **Firmware attacks**: A malware that alters the functioning of a firmware, AKA the real-time operating systems of an electronic device.
- **Computer worms**: Like the Stuxnet worm mentioned in section 3.5, a malware whose goal is to spread across a network.
- **Keystroke injection**: Program the USB drive to act as a keyboard, and types malicious commands; such as installing another malware, backdoor, setting up a reverse shell, etc.
- **Killer USB**: A USB thumb drive that fries any electronic device in a fraction of a second. It charges capacitor with the USB port and discharges it once it has reached 240 V[51].
- **Back door**: The USB installs a back door to the device, allowing the attacker to access and use it whenever he wants (for a DDoS attack for example).
- **Keylogger**: A malware that secretly saves every key typed on a keyboard.

This type of attack is especially difficult to detect. They often go unnoticed by network security, and many people trust USB drives more than they should. This opens the door to a new wave of potential attacks. The best ways to counter these types of attack are to:

- Physically disable the USB ports or only unused USB ports.
- Disable autorun to prevent the USB drive from automatically executing when it is plugged in.
- Only accept certified USB drives that cannot leave the premises.
- Use USB activity monitoring tools to detect and block suspicious USB activity.
- Train and raise awareness to all the employees.

However, it should be noted that these are not the only counters in existence. Depending on the machine in question, there may be a number of additional counters available [52][53][54][55][56].

4.7 Level 3

4.7.1 NodeRED Flow Injection

This attack can happen in a couple of situations:

1. NodeRED does not possess any authentication.
2. All users are granted permissions in NodeRED by default.
3. The flow editor is not blocked, either for unauthorized user or from outside of the LAN.

If NodeRED has no authentication, the REST API is going to use HTTP, rather than HTTPS, to send each requests, making these requests not encrypted and visible in plain text.
If one or multiple of these situations are met, an attacker can:

1. Read the sensitive data and libraries used of the flows, and its other flows within the global context.
2. Delete the entire flow.
3. Modify the flow, either by modifying its data or by implementing malicious code.
4. Read, modify or delete the credentials of every registered users.

This attack is, in short, a total suppression of any confidentiality of any sensitive data and used libraries. In both of our setup, this can be done directly with only the dashboard. This is easily patched by giving restricted access to default users or to only authorize authenticated users to perform actions on NodeRED, but the damage done can be very high.
The only way to protect against it is to set up authentication on Node-RED, grant permissions only to certified users, and verify that sensitive pages, such as the flow editor, are only accessible to authorized personnel[57].

4.7.2 Malicious NodeRED Node

NodeRED being an Open-Source platform built with Node.js, every node that can be installed are first added to the Node Package Manager, then automatically to the NodeRED Library. An attacker could create a malicious node, containing in the background powerful commands from Node.js like *child_process* with the function *exec()* to execute any shell commands wanted and have full control of the

server's system.
This attack has the same consequences as the NodeRED Flow Injection, which is to suppress all confidentiality and integrity of any sensitive data and libraries used in the flow, or in other flows in the global context, also allowing the intruder to read and modify the sensitive data.Contrary to a NodeRED Flow Injection, this attack is much harder to defend against.
A naive way to protect against it would be to ban the usage of such commands like *child_process*, but they are most of the time essential to the functioning of NodeRED. The general security advised would be to verify each external library used, either installed locally or installed with the Node Package Manager, but another possible solution is a monitoring framework, as explained in [57].

4.8 Level 4

The only possible attack at level four is to launch a DoS on the NodeRED dashboard, which is the equivalent to launching a DoS on NodeRED itself.

Chapter 5

Results

This chapter presents the results of the different attacks tested on both setups, detailing how they were performed in our testbed and the impact they had on both setups, providing the data to quantify this impact. These attacks range from altering or removing the user's ability to use the control system (thereby affecting their ability to control the physical process) to reading, modifying and controlling different data and components of the setup.
Most of the graphs shown in this chapter were made with Python, with the module Scapy, which allows .pcap files (network recordings) to be read and modified. These PCAP files were created using the Wireshark tool or the tcpdump command in certain cases. These recordings were made before and during the attacks, in order to compare the results in normal situations and attacks.
All of these attacks were performed under the scenario that a single attacker was launching the attacks, either remotely or from the LAN. As mentioned in section 4.4 and section 4.5, attacks at levels 0 and 1 were not performed in order to keep the testbed intact while writing this thesis.

5.1 Level 2

5.1.1 Denial of Service

On this level, The two possible denial of services possible are on the PLC, AKA the Revolution Pi Connect+, and the NodeRED Dashboard. Since the latter is also possible on the level 4 of the remote setup, we will talk about it in its section.
To perform these denial of services, we use **hping**, a network tool used to send custom TCP/IP packets, display target replies, and many other things such as testing firewalls, routers, etc.[58].
In our case, the –**flood** parameter enables this tool to send as many packets as pos-

sible without waiting for any replies. We will make a SYN Flood DoS, as explained in subsection 4.6.1, and target the port used by the PLC, which is 41443 in our setup.

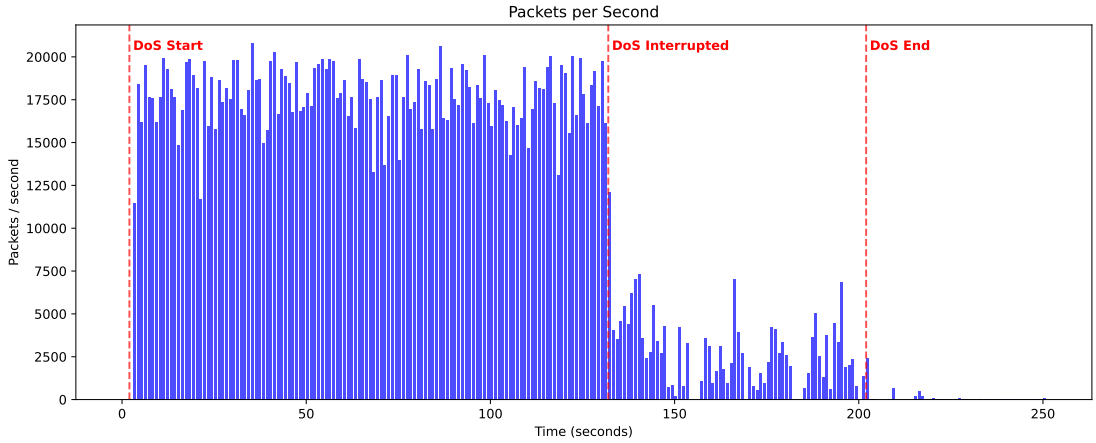

Figure 5.1: The number of packets sent to the PLC during the DoS

The Figure 5.1 is a bar chart showing the number of packets received by the PLC per second, from a network capture with Wireshark.
The beginning of the DoS attack is clearly visible in the first few seconds, as the number of packets per second rises to a maximum of 20,000. The attack continues even after we interrupt the hping command, at which point the maximum reduces to around 7,500.
This is due to the routers' buffers, made to avoid congestion, emptying. One consequence of a DoS attack is bufferbloat, which occurs when routers' buffers fill up, resulting in delays.
Another reason is TCP retransmission, whereby packets are resent due to a lack of an ACK segment or the presence of multiple duplicate ACK segments[59][60].
Other evidence of the impact of the DoS is the packet loss rate, which hping3 tells us is 156.456 for 239.583 sent, 65.3% of the packets sent never received any reply whatsoever. The attack truly ends at around the 200-second mark, when the number of packets drops below a thousand, returning to a normal ratio.
We can also prove that the application is being slowed down or inaccessible by the DoS with ICMP pings and the network jitter, that is the variation of the Round Trip Time (RTT) between each packets sent, as seen in Figure 5.2.

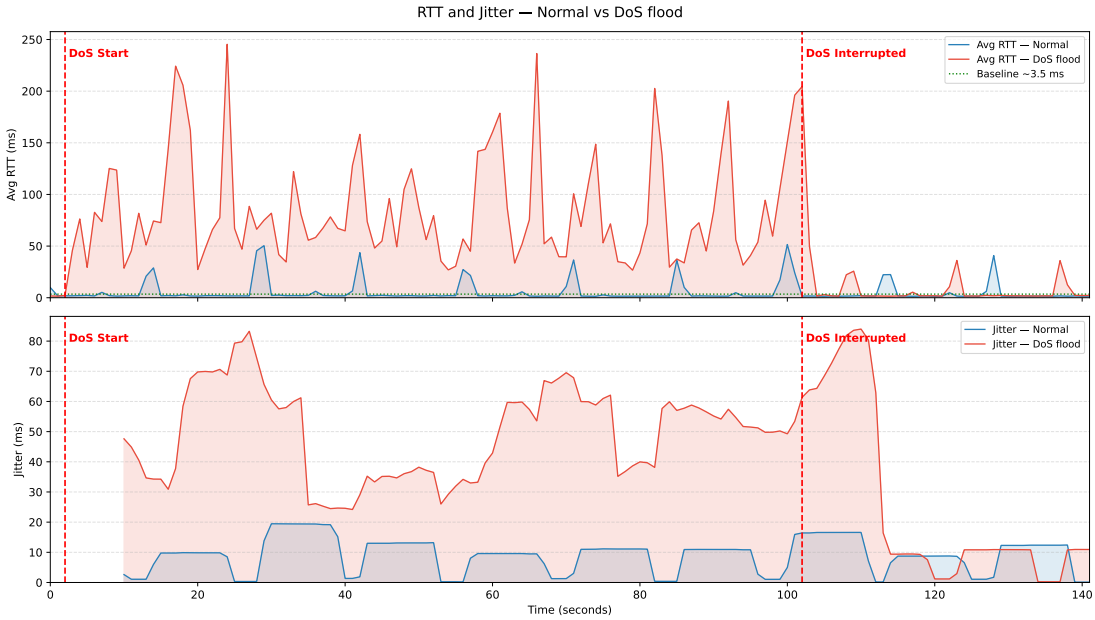

Figure 5.2: The average RTT and Jitter before and during the DoS

We can observe that the average RTT, AKA the time it takes for a packet to make a round trip from its source [61], during a DoS, the average RTT during a DoS is 11 times as the average RTT in a normal situation.
Network jitter can also be seen increasing, with an average increase from 8.95 ms to 42.16 ms, an increase of 4.71 times.

5.1.2 ARP Poisoning

In our configuration, we are doing an ARP poisoning between the Revpi and another computer connected on the network.
We can use **arpspoof** to easily do it, or in our case use a python script with Scapy, which was originally made to do the downgrade attack, mentioned in subsection 4.6.4. We can easily check if the ARP is working by either sniffing the network and looking up the ARP packets, checking the ARP tables of the Revpi or doing a ping to the gateway from the revpi: First, we can see in Figure 5.3 that, by sniffing the network, our script is indeed sending multiple ARP packets, telling the gateway that the Revpi is at our MAC address and vice versa, the packets are also signaling a duplicate use of the mac address' IP. We can even observe a normal ARP packet pointing to the correct MAC address being overlapped by our attack.

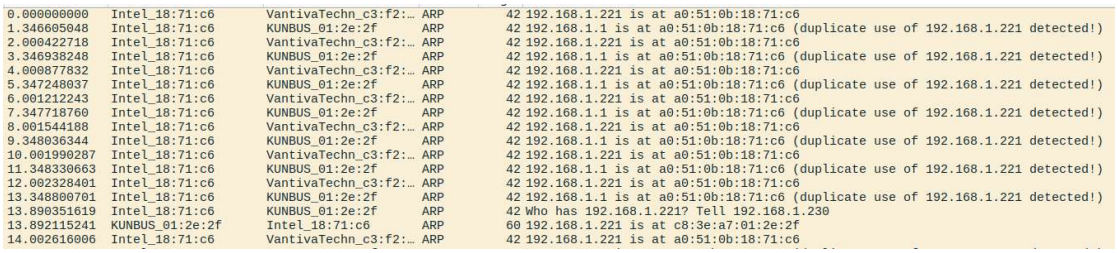

Figure 5.3: A Wireshark capture of the ARP Poisoning

Second, by doing a ping from the revpi, we can see that this ping is redirected to our machine, then resent to the gateway, as seen in Figure 5.4 and Figure 5.5.

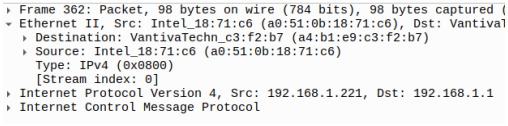

Figure 5.4: The ICMP request

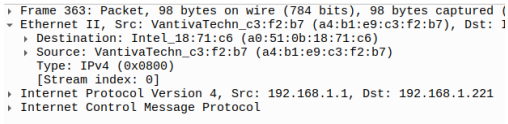

figure itself

Figure 5.5: The ICMP reply

Lastly, by checking the ARP tables, we can see that the Revpi possess our MAC address 2 times, one for the gateway and for the actual attacker, as seen in Figure 5.6.

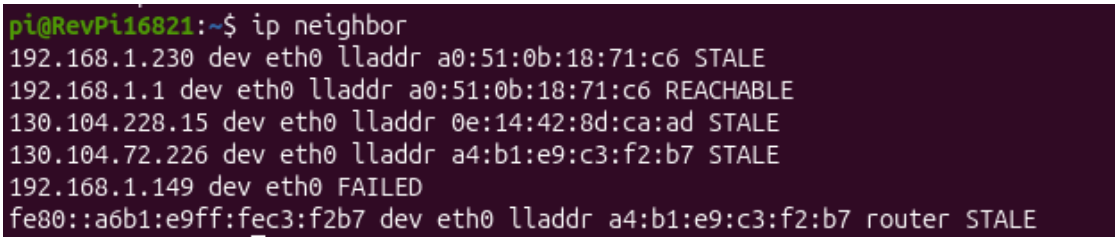

Figure 5.6: The ARP table of the Revpi

5.1.3 Sniff Network

If on the network, sniffing it is pretty simple. Such tools like Tcpdump or wireshark can be used directly to obtain information on it, either with a mirror port or a network tap, this is the passive approach. An active approach is to directly do a MitM to intercept the communications between two parties.

5.1.4 RST Attack

The attack is combined with ARP poisoning, which allows us to intercept every communication between the PLC and the gateway.
Our RST attack has different "states" which dictates what the code will do:

1. "Armed": Ready to attack, if the program see a TLS packet with application data, it sends a RST packet to both the source and destination, and change its state to "fired"

2. "Fired": When this state is set, a cooldown period begins during which no RST packets are sent, even if an application data is received from another source. Once this cooldown is finished, the state is set to "Armed" again.

Figure 5.7 shows a Wireshark recording of the network, demonstrating the RST attack and its various stages. A filter has been applied to show only packets coming from or to the PLC, as well as ARP packets.

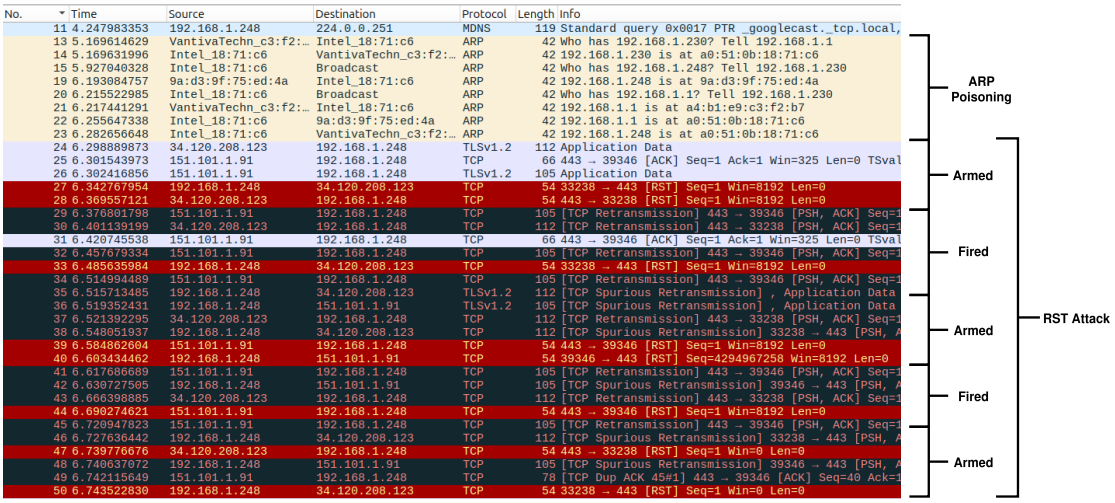

Figure 5.7: A recording made with Wireshark, showing the RST attack and its effects

As we can see, the recording begins with the ARP poisoning, which correctly misinforms the gateway and revpi by providing our MAC address and therefore doing a MitM. The next packet is a TLSv1.2 application data, which the program spots in "Armed" state. The program then sends 2 RST packets and transitions to the "Fired" state. After the cooldown, the program returns to the "Armed" state, spots the TCP retransmission packet, containing an application data, and the cycle repeats.

5.1.5 Replay Attack

In our test, we sniffed the network multiples times and saved these recordings as pcap files, which were played back with *tcpreplay*.
Many of the protocols used by the Revpi, such as TLS1.3, are protected against replay attacks, which makes this attack useless. This attack still works with the

NodeRED's API requests, which are unsecured if NodeRED doesn't possess any authentication. Replaying PUT requests could result in an older version of NodeRED's flows being replayed, which would delete any updates made since then, and GET and POST requests could be used in a MitM, to trick the target into viewing a version chosen by the attacker[62].
Figure 5.8 shows the captured GET and POST requests, with their sequence numbers, at the top of the image. Below these are the same requests, replayed later with the same sequence numbers, proving that these requests are the same and are being replayed.

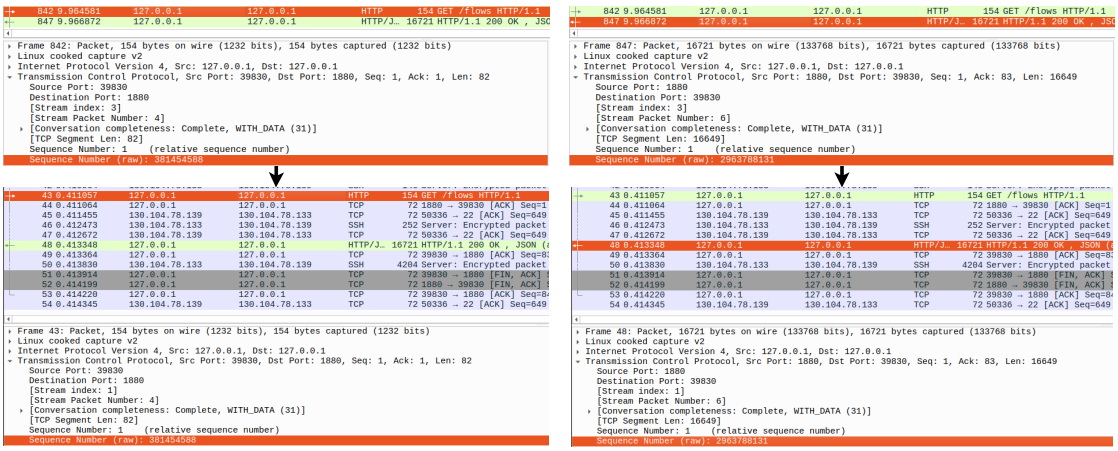

Figure 5.8: The GET and POST requests and their replays below

5.2 Level 3

5.2.1 Denial of Service

On this level, the possibles targets for a denial of service are NodeRED and the SSH tunnel in our remote setup.

DoS on NodeRED

Since attacking NodeRED on the local setup results in also attacking the Revpi, since both of them shares the same IP address, this attack is made on the remote setup, where another instance of NodeRED is installed.

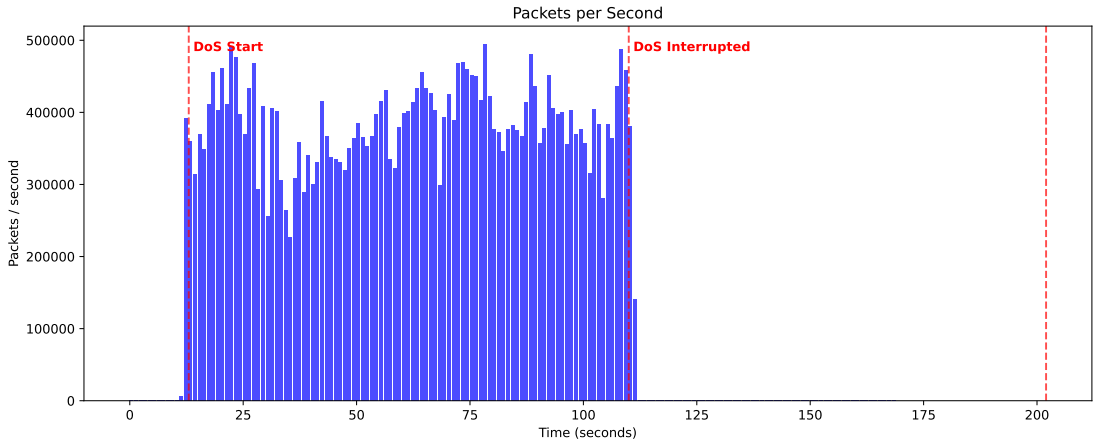

Figure 5.9: The number of packets received per seconds

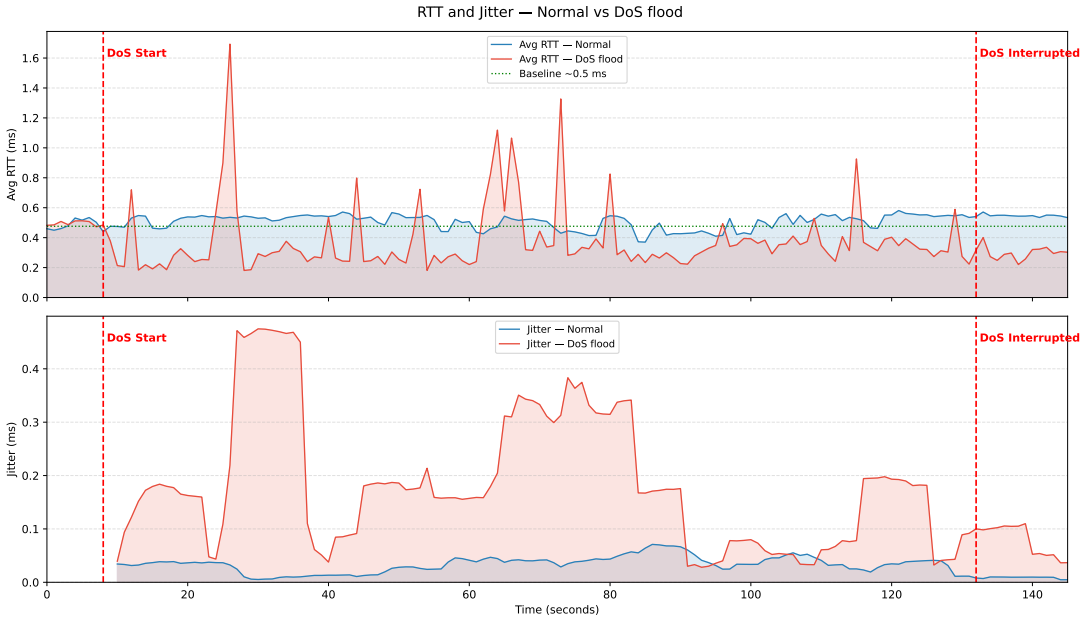

Figure 5.10: The average RTT time per second

The Figure 5.9 is a bar chart showing the number of packets received by the PLC per second, from a network capture with tcpdump.
The beginning of the DoS attack is clearly visible in the first few seconds, as the number of packets per second rises to a maximum of 500000.
The reason our maximum of packets goes from 20k in the local setup to 500k in the remote setup and that there is no residue of the attack, contrary to the attack on the Revpi, is the same reason the Figure 5.10 shows an average RTT smaller during

a DoS than during a normal situation: The remote setup is a virtual machine on the university's server, it is more capable of handling heavy traffic, with a much bigger buffer, and is more powerful in terms of hardware.
However, the DoS shows that the average RTT is way more unstable, as a lot of spikes, going up to 1.6ms, are present during the attack, as well as the network jitter, which goes from an average of 0.03ms to 0.17ms, an increase of 5.6 times. While this increase is pretty insignificant on this setup, it demonstrates that the DoS does an effect on the server. This instability and these spikes can be seen more clearly if we do not perform an average and show every RTT of every packets, as seen in Figure 5.11

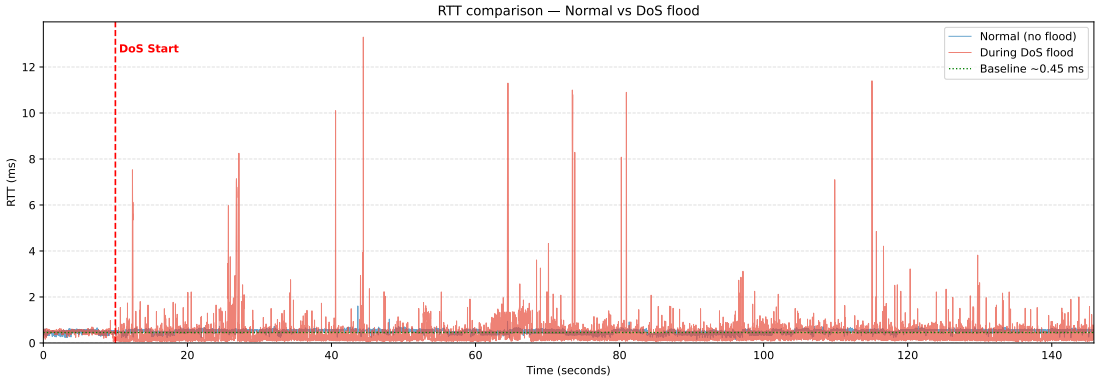

Figure 5.11: Every packet's RTT time

This proves that, while a DDoS would be much more efficient to realistically slow down the server, a DoS is still able to impact it.

DoS on the SSH Tunnel

In this DoS, we target specifically the port SSH uses, port 22.

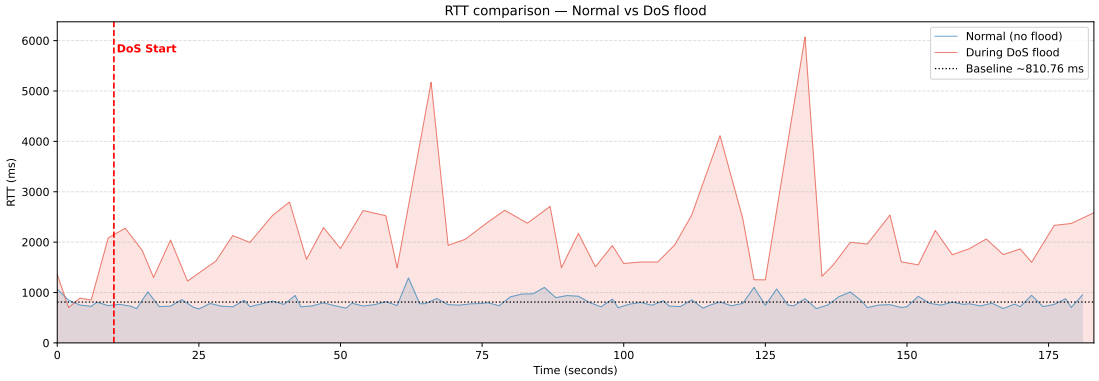

Figure 5.12: Response time of the port by seconds

Before talking about the effect, one might look at the graph and detect an unusual RTT during a normal situation for SSH. The baseline is around 810.76 ms, which is very long. This is because the pings were launched from our personal machine, which had to pass through a ProxyJump between the SSH and the remote setup, which made the RTT much longer. By comparison, pings made from the VM used for the attack, which is on the same subnet, take around 142 ms, while pings made from the local setup take around 650 ms.
Here an effect can be clearly seen before and during the attack, with spikes of the average RTT going up to 6 entire seconds. This results demonstrates that with only a DoS, not a DDoS, a huge impact on the response time can be made.

5.2.2 NodeRED Flow Injection

NodeRED uses a REST API for its own internal operation, when no authentication or security is put on NodeRED, this REST API is unsecured and free to access by everyone. With a simple curl request, we can access every flows and their information in a JSON, as seen partly in Figure 5.13:

Figure 5.13: A screenshot of the curl request

In this JSON, we can find multiple useful information:

1. All the users and passwords associated: as seen in this line:

```
{"id" :"ec6875b9ba2f1ffe",
"type":"revpi-server",
"host":"localhost",
"port":8000,
"user":"pi",
```

```
"password":"yyp2gm",
"rejectUnauthorized":false,
"ca":""}
```

2. Content of the different nodes in the flows and their libraries, for example here is a function node with its code:

```
{"id":"4f2b92833c3e558e",
"type":"function",
"z":"ee74a44bf0206b66",
"name":"Move",
"func":
"if (!flow.get(\"start\") && !flow.get(\"calibrate\"
   )) {\n
if (msg.payload === \"forward\") {\n
    return [{payload: true}, {payload: false}];\n
          }
else if (msg.payload === \"backward\") {\n
return [{payload: false}, {payload: true}];\n    }
else if (msg.payload === \"stop\") {\n
return [{payload: false}, {payload: false}];\n    }\
   n}",
"outputs":2,
"timeout":0,
"noerr":0,
"initialize":"",
"finalize":"",
"libs":[],
...
}
```

and here is the node in Figure 5.14:

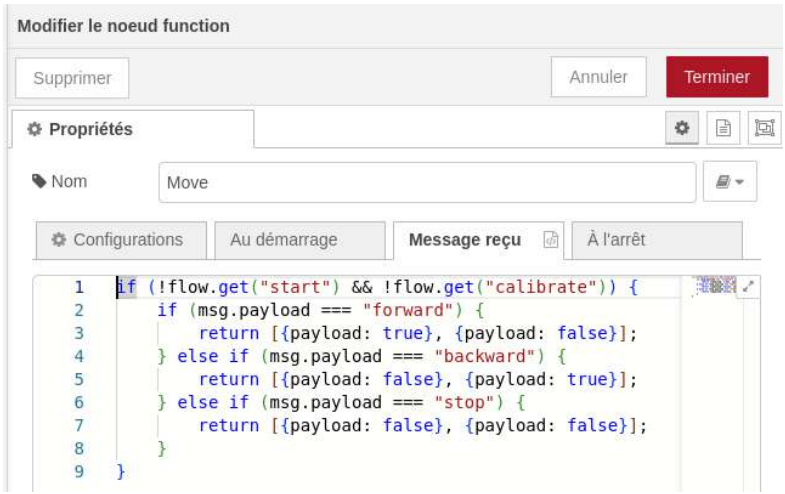

Figure 5.14: The function node found

3. Which wire/input represents which functions of the physical process:

```
{"id":"b3f1a90858592619",
"type":"revpi-single-input",
"z":"ee74a44bf0206b66",
"server":"ec6875b9ba2f1ffe",
"inputpin":"I_2",
"topic":"machine",
...
}
```

If the user gets write permissions by default, we can easily modify this JSON and add new nodes (also explained in subsection 5.2.3), function or code, such as a shell command:

```
const stolen_require = global.get('process') || this.
   constructor.constructor('return process')();
const exec = stolen_require.mainModule.require('
   child_process').exec;
exec('id > /tmp/pwned.txt', (err, stdout) => {
    node.warn(err ? "failed: " + err : "success");
});
node.send({ payload: "called" });
```

Which the results can be found in our /tmp/pwned.txt file:

```
root@TFE-Barbason:/tmp# cat pwned.txt
uid=0(root) gid=0(root) groups=0(root)
```

5.2.3 Malicious NodeRED Node

For obvious ethical and legal reasons, we are not going to publish a real malicious NodeRED node in the Node Package Manager, but we can simulate the process of downloading one, or even to simulate an intruder planting this node in the system's files, for example with the NodeRED Flow Injection explained before.
All nodes are stored in a folder in NodeRED's working directory, one can find it with the *which* command, simply create a new folder, and add the requisites for a node in it, which are at least:

1. package.json: used to describe the content of the node and for the runtime to know which file is used.

2. A JavaScript script: The node's programming

3. A HTML file: The appearance of the node, the edit menu and the help text.

We can input our reverse shell in the javascript's script:

```
const { exec } = require('child_process');
module.exports = function(RED) {
    function TestNode(config) {
        RED.nodes.createNode(this, config);
        exec('id > /tmp/pwned.txt', (err, stdout) => {});
    }
    RED.nodes.registerType('test-node', TestNode);
};
```

The code will only execute when the malicious node is inserted into the flow, either willingly by the user, unknowing that this is a malicious node, or we can insert it directly by using the same method as subsection 5.2.2, to see the same results.

Chapter 6

Improvements

Some improvements can definitely be made in this paper, in order to have a bigger range of attacks to perform and to make this mini-factory more realistic, here is a non-exhaustive list of what could be made:

1. Diving into the subject of electronic's hacking, to monitor and spoof values between the Revpi DIO and the Revpi Connect + would be a great addition, since this attack breaks any confidentiality and integrity of the data to all the upper layers.

2. Having multiples VMs to make a DDoS attack on the remote setup would show a more realistic impact on the system, and what would be its consequences, since DDoS is much more powerful than DoS.

3. Using an industrial communication standard in the local setup, such as Modbus or MQTT, would be of great value, considering these standards are widely used. It would open the way to new multiple scenarios and attacks.

4. Using another type of PLCs, for example an older model that is still widely used in the industry.

5. Making a local setup that separates the PLC and the SCADA system, contrary to what we have, would make way to new scenarios and attacks.

6. Making the control system with a different programming, such as with CODESYS or Python, to discover new breaches in their systems, the consequences of not securing them enough, and how to exploit them.

7. Implement a full cloud architecture for the remote setup.

8. Configure an USB drive to test different types of USB injection.

Chapter 7

Conclusion

This thesis's goal was to demonstrate multiple cyber attacks on a low-cost physical testbed with two different setups, a local setup with a Revolution Pi Connect +, a Fischertechnik machine and with NodeRED as a control system, and a remote setup, simulating a cloud server with a virtual machine, connected to the local setup with a SSH tunnel. The goal of having these 2 different setups was to have a realistic situation, to get more realistic results.
While most of the attacks in the level 0 and 1 were not tested, in order to keep intact the setup during the writing of this thesis, the results are more focused towards the level 2, 3 and 4.
Beginning with level 2, the ARP poisoning successfully worked between the RevPi and another machine, with forged ARP replies spotted while sniffing the network, ICMP pings going through the attacker and ARP tables mapping the attacker's MAC address for 2 different IPs.
The DoS on the RevPi, which sent a maximum of 20000 packets per seconds, increased the average RTT 11 times during the attack, with a 65.3% packet loss and with a jitter increase of 4.71 times higher, paralysing the network even after the attack stopped, leaving residual traffic for 72 seconds after the attack.
The RST attack coupled with ARP poisoning has proven successful to disrupt communications between the client and the server, as there is little to no protection against it on TLSv1.2.
The replay attack proved ineffective on TLSv1.3 but successful on NodeRED HTTP API requests, when NodeRED has no authentication whatsoever, allowing us to replay older POST and PUT requests, deleting any previous work who has been done due to the lack of security in HTTP.
Continuing with level 3, the NodeRED Flow Injection allows us to get every credentials, functions, IOs information, flows, global variable and libraries used if no authentication is set. It also allow us to modify the code and inject a malicious function, such as a reverse shell.

These attacks demonstrate multiple things, first that an unprotected and unmonitored network leaves room for a significant number of attacks. Countermeasures at any layer must be implemented, such as 802.1X authentication on the network layer, which denies entry to any unauthorized devices on a network, whether wired or wireless, all attack requiring at least access to a LAN would be impossible to perform (unless the attacker compromises an authorized device), another countermeasure on the data layer would be to implement a dynamic ARP inspection, which would secure the network against every attack relying on this method of MitM.
Secondly, NodeRED is not protected by default, which means confidentiality and integrity of data can't be guaranteed, data can and will be altered, especially if NodeRED is put online. Securing NodeRED is more than crucial, and there are many ways provided directly by NodeRED to do it, such as authentication, enabling HTTPS, etc.
However, this thesis has limitations that should be acknowledged. Firstly, all of our attacks rely on only one single attacker, which makes some attacks, such as the DoS on NodeRED, rather weak. A DDoS with multiple machines acting as "zombies" would provide much more realistic data, and show us more clearly the effects of such an attack.
Secondly, most of the level 0 and 1 attacks were not tested, due to leaving our area of expertise and to preserve the setup for this thesis, a stress test by sabotaging the hardware would be very useful to test the limits of the hardware.
To conclude, the results of this thesis prove not only that a low-cost testbed is sufficient and efficient to reproduce cyber attacks, get realistic results and accurately measure the impact of these attacks on ICS systems. More importantly the results shows that most vulnerabilities are due to a lack of security, authentication and encryption, either because of default configuration or simply a lack of interest in these issues. As ICS systems become increasingly hosted or accessible via cloud servers, these attacks will become easier and easier to execute if nothing is done to make these systems more secure.

Bibliography

[1] Mavis. *Understanding PLC Cybersecurity: The Definitive Guide*. en. Apr. 2024. URL: https://www.txone.com/blog/ultimate-guide-to-plc-cybersecurity/ (visited on 12/22/2025).

[2] Hannes Holm et al. "A Survey of Industrial Control System Testbeds". In: *Secure IT Systems*. Ed. by Sonja Buchegger and Mads Dam. Vol. 9417. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 11–26. ISBN: 978-3-319-26501-8 978-3-319-26502-5. DOI: 10.1007/978-3-319-26502-5_2. URL: http://link.springer.com/10.1007/978-3-319-26502-5_2 (visited on 05/20/2026).

[3] G. P. H. Sandaruwan, P. S. Ranaweera, and Vladimir A. Oleshchuk. "PLC security and critical infrastructure protection". In: *2013 IEEE 8th International Conference on Industrial and Information Systems*. ISSN: 2164-7011. Dec. 2013, pp. 81–85. DOI: 10.1109/ICIInfS.2013.6731959. URL: https://ieeexplore.ieee.org/abstract/document/6731959 (visited on 04/20/2026).

[4] Felix Sauer et al. "LICSTER – A Low-cost ICS Security Testbed for Education and Research". In: arXiv: 1910.00303 [cs]. 2019. DOI: 10.14236/ewic/icscsr19.1. URL: http://arxiv.org/abs/1910.00303 (visited on 10/02/2025).

[5] *https://www.insecurity.be/verboten/seminaries/ICS_archs_and_sec_essentials/ICS_Overview* URL: https://www.insecurity.be/verboten/seminaries/ICS_archs_and_sec_essentials/ICS_Overview.pdf (visited on 05/19/2026).

[6] Xabier Etxezarreta et al. "Software-Defined Networking approaches for intrusion response in Industrial Control Systems: A survey". en. In: *International Journal of Critical Infrastructure Protection* 42 (Sept. 2023), p. 100615. ISSN: 18745482. DOI: 10.1016/j.ijcip.2023.100615. URL: https://linkinghub.elsevier.com/retrieve/pii/S1874548223000288 (visited on 05/19/2026).

[7] *What is PLC and Its Use Cases?* en-GB. URL: https://www.patrion.net/en/blog/what-is-plc-and-its-use-cases (visited on 12/22/2025).

[8] *What's the Role of PLCs in Automation? \textbar RS*. URL: https://uk.rs-online.com/web/content/discovery/ideas-and-advice/plc-automation-guide (visited on 12/22/2025).

[9] Geeta Yadav and Kolin Paul. "Architecture and security of SCADA systems: A review". In: *International Journal of Critical Infrastructure Protection* 34 (Sept. 2021), p. 100433. ISSN: 1874-5482. DOI: 10.1016/j.ijcip.2021.100433. URL: https://www.sciencedirect.com/science/article/pii/S1874548221000251.

[10] *Flow-Based Programming*. URL: https://www.jpaulmorrison.com/fbp/fbp2.htm (visited on 03/16/2026).

[11] *Low-code programming for event-driven applications : Node-RED*. URL: https://nodered.org/ (visited on 04/21/2026).

[12] *ANSI/ISA-95.00.01-2025 (IEC 62264-1 Mod), Enterprise-Control System Integration Part 1: Models and Terminology*. en. URL: https://www.isa.org/products/ansi-isa-95-00-01-2025-iec-62264-1-mod-enterprise (visited on 04/19/2026).

[13] *ISA-95 Standard: Enterprise-Control System Integration*. en. URL: https://www.isa.org/standards-and-publications/isa-standards/isa-95-standard (visited on 04/23/2026).

[14] Fu-Hau Hsu et al. "TRAP: A Three-Way Handshake Server for TCP Connection Establishment". en. In: *Applied Sciences* 6.11 (Nov. 2016), p. 358. ISSN: 2076-3417. DOI: 10.3390/app6110358. URL: https://www.mdpi.com/2076-3417/6/11/358 (visited on 05/10/2026).

[15] Wesley Eddy. *Transmission Control Protocol (TCP)*. Request for Comments RFC 9293. Num Pages: 98. Internet Engineering Task Force, Aug. 2022. DOI: 10.17487/RFC9293. URL: https://datatracker.ietf.org/doc/rfc9293 (visited on 05/10/2026).

[16] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Request for Comments RFC 8446. Num Pages: 160. Internet Engineering Task Force, Aug. 2018. DOI: 10.17487/RFC8446. URL: https://datatracker.ietf.org/doc/rfc8446 (visited on 05/18/2026).

[17] *An overview of the SSL/TLS handshake - IBM Documentation*. URL: https://www.ibm.com/docs/en/ibm-mq/9.4.x?topic=tls-overview-ssltls-handshake (visited on 05/18/2026).

[18] *What is Transport Layer Security? | TLS protocol*. en. URL: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/ (visited on 05/18/2026).

[19] *What Is ARP (Address Resolution Protocol)? How Does It Work?* en. URL: https://www.fortinet.com/resources/cyberglossary/what-is-arp (visited on 05/18/2026).

[20] Chris M. Lonvick and Tatu Ylonen. *The Secure Shell (SSH) Connection Protocol*. Request for Comments RFC 4254. Num Pages: 24. Internet Engineering Task Force, Jan. 2006. DOI: 10.17487/RFC4254. URL: https://datatracker.ietf.org/doc/rfc4254 (visited on 05/18/2026).

[21] *What is SSH? | Secure Shell (SSH) protocol*. en-us. URL: https://www.cloudflare.com/learning/access-management/what-is-ssh/ (visited on 05/18/2026).

[22] *Internet Control Message Protocol*. Request for Comments RFC 792. Num Pages: 21. Internet Engineering Task Force, Sept. 1981. DOI: 10.17487/RFC0792. URL: https://datatracker.ietf.org/doc/rfc792 (visited on 05/18/2026).

[23] *What is ICMP? | Internet Control Message Protocol*. en. URL: https://www.cloudflare.com/learning/ddos/glossary/internet-control-message-protocol-icmp/ (visited on 05/18/2026).

[24] *What is a REST API?* en. URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api (visited on 05/21/2026).

[25] *What Is a REST API (RESTful API)? | IBM*. en. Apr. 2025. URL: https://www.ibm.com/think/topics/rest-apis (visited on 05/21/2026).

[26] Mauro Conti, Denis Donadel, and Federico Turrin. "A Survey on Industrial Control System Testbeds and Datasets for Security Research". In: *IEEE Communications Surveys & Tutorials* 23.4 (2021), pp. 2248–2294. ISSN: 1553-877X. DOI: 10.1109/COMST.2021.3094360. URL: https://ieeexplore.ieee.org/document/9471765/ (visited on 05/20/2026).

[27] Alireza Dehlaghi-Ghadim et al. "ICSSIM — A framework for building industrial control systems security testbeds". In: *Computers in Industry* 148 (June 2023), p. 103906. ISSN: 0166-3615. DOI: 10.1016/j.compind.2023.103906. URL: https://www.sciencedirect.com/science/article/pii/S0166361523000568.

[28] Janis Judvaitis et al. "Testbed Facilities for IoT and Wireless Sensor Networks: A Systematic Review". en. In: *Journal of Sensor and Actuator Networks* 12.3 (June 2023), p. 48. ISSN: 2224-2708. DOI: 10.3390/jsan12030048. URL: https://www.mdpi.com/2224-2708/12/3/48 (visited on 05/20/2026).

[29] Jonathon Brugman et al. "Cloud Based Intrusion Detection and Prevention System for Industrial Control Systems Using Software Defined Networking". In: *2019 Resilience Week (RWS)*. Vol. 1. Nov. 2019, pp. 98–104. DOI: 10.1109/RWS47064.2019.8971825. URL: https://ieeexplore.ieee.org/document/8971825/ (visited on 05/19/2026).

[30] Shamsul Huda et al. "A malicious threat detection model for cloud assisted internet of things (CoT) based industrial control system (ICS) networks using deep belief network". In: *Journal of Parallel and Distributed Computing* 120 (2018), pp. 23–31. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2018.04.005. URL: https://www.sciencedirect.com/science/article/pii/S0743731518302442.

[31] Kyle Wilhoit. "Who's really attacking your ICS equipment?" In: *Trend Micro* 10 (2013).

[32] Anam Sajid, Haider Abbas, and Kashif Saleem. "Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges". In: *IEEE Access* 4 (2016), pp. 1375–1384. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2016.2549047. URL: https://ieeexplore.ieee.org/document/7445139/ (visited on 05/19/2026).

[33] Mustafa Kara and Murat Furat. "Security threats for industrial control systems and a solution with DMZ". In: *1 st International Mediterranean Science and Engineering Congress*. 2016, pp. 26–28.

[34] Marie Baezner and Patrice Robin. *Stuxnet*. en. Tech. rep. Artwork Size: 16 p. Medium: application/pdf. ETH Zurich, Oct. 2017, 16 p. DOI: 10.3929/ETHZ-B-000200661. URL: http://hdl.handle.net/20.500.11850/200661 (visited on 04/22/2026).

[35] *PLC code vulnerabilities through SCADA systems - ProQuest*. en. URL: https://www.proquest.com/openview/cd9bf9c536c1d8278159656bbdad2017/1.pdf?pq-origsite=gscholar&cbl=18750&diss=y (visited on 04/22/2026).

[36] Nicholas Weaver et al. "A taxonomy of computer worms". en. In: *Proceedings of the 2003 ACM workshop on Rapid malcode*. Washington DC USA: ACM, Oct. 2003, pp. 11–18. ISBN: 978-1-58113-785-9. DOI: 10.1145/948187.948190. URL: https://dl.acm.org/doi/10.1145/948187.948190 (visited on 05/10/2026).

[37] *On-SCADA-PLC-and-Fieldbus-Cyber-Security*. URL: https://www.researchgate.net/profile/Cordell-Davidson-2/publication/323725974_On_SCADA_PLC_and_Fieldbus_Cyber-Security/links/5aa775384585152d7665ce5c/On-SCADA-PLC-and-Fieldbus-Cyber-Security.pdf (visited on 04/22/2026).

[38] Francesco Aurelio Pironti et al. *ICSLure: A Very High Interaction Honeynet for PLC-based Industrial Control Systems*. en. Version Number: 1. 2025. DOI: 10.48550/ARXIV.2509.04080. URL: https://arxiv.org/abs/2509.04080 (visited on 04/23/2026).

[39] Iyatiti Mokube and Michele Adams. "Honeypots: concepts, approaches, and challenges". en. In: *Proceedings of the 45th annual southeast regional conference*. Winston-Salem North Carolina: ACM, Mar. 2007, pp. 321–326. ISBN: 978-1-59593-629-5. DOI: 10.1145/1233341.1233399. URL: https://dl.acm.org/doi/10.1145/1233341.1233399 (visited on 04/23/2026).

[40] *The Industrial Raspberry Pi for IIoT & Automation - Revolution Pi*. en. URL: https://revolutionpi.com/en/ (visited on 01/15/2026).

[41] *PDF*. URL: https://www.studica.com/Images/uploaded/Resources/FT/96785_51663_punching_machine_24v_9v.pdf (visited on 01/07/2026).

[42] *Understanding Denial-of-Service Attacks | CISA*. en. Feb. 2021. URL: https://www.cisa.gov/news-events/news/understanding-denial-service-attacks (visited on 04/25/2026).

[43] Corey Nachreiner. "Anatomy of an ARP Poisoning Attack | WatchGuard". en. In: ().

[44] Aayush Majumdar, Shruti Raj, and T. Subbulakshmi. "ARP Poisoning Detection and Prevention using Scapy". en. In: *Journal of Physics: Conference Series* 1911.1 (May 2021), p. 012022. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/1911/1/012022. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1911/1/012022 (visited on 04/24/2026).

[45] Joseph D. Touch. *Defending TCP Against Spoofing Attacks*. Request for Comments RFC 4953. Num Pages: 28. Internet Engineering Task Force, July 2007. DOI: 10.17487/RFC4953. URL: https://datatracker.ietf.org/doc/rfc4953 (visited on 05/17/2026).

[46] *A1: Injection Vulnerability - Top 10 OWASP 2022*. en. URL: https://www.wallarm.com/what/a1-injection-2017-owasp (visited on 04/25/2026).

[47] Eman Alashwali and Cas Cremers. "On Downgrade Attacks in the TLS Protocol". en. In: ().

[48] Sangtae Lee, Youngjoo Shin, and Junbeom Hur. "Return of version downgrade attack in the era of TLS 1.3". en. In: *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*. Barcelona Spain: ACM, Nov. 2020, pp. 157–168. ISBN: 978-1-4503-7948-9. DOI: 10.1145/3386367.3431310. URL: https://dl.acm.org/doi/10.1145/3386367.3431310 (visited on 04/27/2026).

[49] Eman Salem Alashwali and Kasper Rasmussen. *What's in a Downgrade? A Taxonomy of Downgrade Attacks in the TLS Protocol and Application Protocols Using TLS*. en. arXiv:1809.05681 [cs]. Jan. 2019. DOI: 10.48550/arXiv.1809.05681. URL: http://arxiv.org/abs/1809.05681 (visited on 04/27/2026).

[50] Reda El Abbadi and Hicham Jamouli. "Takagi–Sugeno Fuzzy Control for a Nonlinear Networked System Exposed to a Replay Attack". en. In: *Mathematical Problems in Engineering* 2021.1 (2021). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1 p. 6618105. ISSN: 1563-5147. DOI: 10.1155/2021/6618105. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/6618105 (visited on 05/13/2026).

[51] Olga Angelopoulou et al. "Killing your device via your usb port". In: The Centre for Security, Communications and Network Research (CSCAN), 2019, pp. 61–72. ISBN: 0-244-19096-8.

[52] Stéphanie Blanchet. *BadUSB, the threat hidden in ordinary objects*. Tech. rep. Technical report, Bertin Technologies, 2018.

[53] Abraham Serhane et al. "Programmable logic controllers based systems (PLC-BS): vulnerabilities and threats". en. In: *SN Applied Sciences* 1.8 (Aug. 2019), p. 924. ISSN: 2523-3963, 2523-3971. DOI: 10.1007/s42452-019-0860-2. URL: http://link.springer.com/10.1007/s42452-019-0860-2 (visited on 05/17/2026).

[54] *USB risks in industrial and OT environments - Secolve*. en-AU. Section: Blog. July 2025. URL: https://secolve.com/usb-risks-in-industrial-and-ot-environments/ (visited on 05/17/2026).

[55] *The Danger of a USB Device and Keystroke Injection Attack*. en-US. May 2024. URL: https://www.opswat.com/blog/the-danger-of-a-usb-device-and-keystroke-injection-attack (visited on 05/17/2026).

[56] *What Is a BadUSB? Understand the Threat and How to Prevent It*. en-us. July 2025. URL: https://www.ivanti.com/blog/what-is-badusb (visited on 05/17/2026).

[57] Mohammad M. Ahmadpanah et al. "Securing Node-RED Applications". In: *Protocols, Strands, and Logic: Essays Dedicated to Joshua Guttman on the Occasion of his 66.66th Birthday*. Ed. by Daniel Dougherty et al. Cham: Springer International Publishing, 2021, pp. 1–21. ISBN: 978-3-030-91631-2. DOI: 10.1007/978-3-030-91631-2_1. URL: https://doi.org/10.1007/978-3-030-91631-2_1.

[58] Salvatore Sanfilippo. *antirez/hping*. original-date: 2012-06-13T17:41:54Z. Apr. 2026. URL: https://github.com/antirez/hping (visited on 05/06/2026).

[59] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. "Sizing Router Buffers". en. In: ().

[60] Jiancheng Ye, Ka-Cheong Leung, and Steven H. Low. "Combating Bufferbloat in Multi-Bottleneck Networks: Theory and Algorithms". en. In: *IEEE/ACM Transactions on Networking* 29.4 (Aug. 2021), pp. 1477–1493. ISSN: 1063-6692, 1558-2566. DOI: 10.1109/TNET.2021.3066505. URL: https://ieeexplore.ieee.org/document/9399631/ (visited on 05/06/2026).

[61] *Network Jitter - Common Causes and Best Solutions | IR*. en. URL: https://www.ir.com/guides/what-is-network-jitter (visited on 05/11/2026).

[62] M.E Abdelhafez, Sureswaran Ramadass, and Mohammed S. M. Gismallab. "Replay Attack in TLS 1.3 0-RTT Handshake: Countermeasure Techniques". In: *2023 IEEE 6th International Conference on Electrical, Electronics and System Engineering (ICEESE)*. ISSN: 2770-9787. Aug. 2023, pp. 44–49. DOI: 10.1109/ICEESE56169.2023.10278190. URL: https://ieeexplore.ieee.org/document/10278190/ (visited on 05/15/2026).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**