A laboratory for cyber-attack generation and testing in Industrial Control Systems: Design and Simulation

Vedang Ranade

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

A laboratory for cyber-attack generation and testing in Industrial Control Systems: Design and Simulation

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Vedang Ranade

August 15, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Industrial control systems (ICSs) are used widely throughout the world for the control of large, complex industrial plants and consist of the entire setups of control system including sensors (inputs), PLCs (Programmable Logic Controllers), actuators (outputs) and communication devices. The communication between these ICS devices is performed using industrial communication protocols such as Modbus, EtherCAT, etc. With the advancements in the use of the internet, ICS are being enabled to share real-time information over the internet worldwide. While these features make the ICS more accessible for remote supervisory control, they also make them vulnerable to cyber-attacks. This makes it the need of the hour to investigate risks and impacts of cyber attacks on ICS.

Generating, injecting and testing cyber attacks on real world ICS, controlling critical infrastructure will involve several financial risks and safety issues. This gives rise to the necessity of an ICS testbed, with the ability to inject and test cyber attacks, in a safe and secure environment. A testbed provides a cheaper alternative for testing impacts of cyber attacks and also offers more flexibility to simulate multiple industrial scenarios. Together, these aspects form the core reasons behind the requirement of an ICS testbed. As the real world ICSs are often costly and specialised for industrial usage, there are not many research laboratories around the world with the availability of a testbed to study cyber attacks. Therefore, this Master of science thesis, through its primary research question, investigates if a cyber-attack testbed can be built in the NERD lab at Delft University of Technology (TU Delft), which is able to replicate a real-world ICS network to identify and test vulnerabilities of ICSs working on Modbus protocol to cyber-attacks?

To answer this question, this report studies the vulnerabilities in ICS working with Modbus protocol. A novel design for an ICS testbed for generating and testing cyber attacks at NERD Lab in TU Delft is presented during this thesis. The proposed testbed utilizes real world ICS components such as Programmable Logic Controller (PLC) and Human Machine Interface (HMI), combined with a plant simulator, which is used for simulating an industrial process. The testbed utilizes a Linux based attack PC to generate and inject various cyber attacks. A virtualization platform connects the attack PC to the ICS network, giving the flexibility of injecting attacks on the testbed, without the attacker being physically present

on the plant site. With the use of real world ICS, the testbed therefore allows to replicate a typical ICS scenario in the real world industry.

Further, a simulated version of the actual testbed, with open source softwares, mimicking the ICS systems has been developed in this report. This simulated version provides a cheaper and flexible platform to perform initial testing on the working of the testbed and checks the feasibility of the actual testbed. The testbed simulates a plant, controller and HMI in Matlab/Simulink on different physical PCs, which communicate with the Modbus protocol. An attack PC with a virtualization environment has been used to launch cyber attacks on the simulated testbed, same as that to be used in the proposed testbed at Delft University of Technology. Two main types of cyber attacks namely, Man-In-The-Middle (MITM) and Denial-of-Service (DoS) attack have been successfully implemented on this simulated testbed. To conclude this thesis, advanced versions of these attacks have also been developed and their impacts have been analysed.

Table of Contents

List of Figures

List of Tables

Preface

The idea for this Masters thesis came into existence while I was discussing my background of working in the process control industry with my professor Dr. Riccardo Ferrari. In September 2020, he suggested an idea of working on cyber-attacks on industrial control systems and it instantly grabbed my attention.

I am an instrumentation and control engineer by background. Before pursuing my Masters in systems and control at TU Delft, I had been working full time as a controls engineer at Honeywell Automation India Ltd. During my job, I had worked extensively on the industrial control systems i.e. ICSs and implemented these ICS such as PLC, HMI for control of chemical plants and refineries. Most of these plants contained some or the other hazardous or inflammable materials and ensuring their safety was always of paramount importance. The inevitable threat of cyber attacks on these ICS was discussed frequently during the job, yet, it was not studied methodically and lacked enthusiasm amongst the industry to attain conceivable results.

During my Masters, I learned in depth about control systems and their design and related this well with my industrial experience. This motivated me to combine my professional work experience with the the theoretical knowledge of control system by studying this topic of cyber security in industrial control systems. I consider myself lucky that I could work with Professor Riccardo on this subject and use his valuable experience, without which it would not have been possible to explore this uncharted territory.

I hope this thesis gives the reader, those with technical as well as non-technical background, a clear idea about the threats posed by cyber attacks to our lives and in the hope that the industry dives into this topic with deep emphasis.

Acknowledgements

This Masters thesis at Delft University of Technology proved to be a great opportunity for me to combine my professional working experience with industrial control system with the theoretical knowledge gained at TU Delft during the course of Systems and Control. I am also grateful for having a chance to meet so many wonderful people and professionals who led me throughout this thesis project.

I would specially like to thank my supervisor Dr. Riccardo Ferrari, Assistant Professor in Fault Tolerant Control, TU Delft, for his assistance, guidance and support throughout the tenure of this thesis. It would not have been possible for me to complete this thesis work in time without his support. I would also like to convey my gratitude towards Ir. Twan Keijzer, Phd candidate, TU Delft, who helped me at all steps of the thesis and had regular conversations, which enabled me to tackle day-to-day problems in my thesis. I would also like to take this opportunity to thank Dr. Peyman Mohajerin Esfahani, for accepting the request to form the committee for my thesis.

I perceive this Masters thesis as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives.

Delft, University of Technology — Vedang Ranade
August 15, 2021

Karmanye Vadhikaraste Ma Phaleshu Kadachana |
Ma Karmaphalaheturbhurma Te Sangostvakarmani ||

You have the right to perform your work, but not to expect the fruits of that work. You should not be attached to the hope of success; nor must you give up due to the failure.

- Bhagvad Geeta : Chapter 2, Verse 47

I dedicate my work throughout this thesis to my Mom(Aai), Dad(Baba), Brother(Dada) and my close friends across Europe and in India, for being with me through my thick and thin in this journey at TU Delft!

Chapter 1

Introduction

From the mid-1970s, Industrial Control Systems (ICS) are at the core of every automated industrial factory throughout the world. Even the newest advancements of industrial robots supporting huge assembly lines depend on these ICS for reliable functioning and control. An industrial control system (ICS) is a broad class of automation systems used to provide control and monitoring functionality in manufacturing and industrial facilities.[14] ICS include every component needed for the automated operation of such an industrial facility or plant. Therefore, a typical ICS comprises of sensors, controller, actuators, as well as the networking, communication, data storage and backup devices working together in the industrial setting. At the very core of these ICS is a Programmable Logic Controller (PLC).

ICS differs from the standard control systems in the sense that they are rugged in their use. ICS are essentially used when the processes to be controlled have complex logic and involve heavy, costly machinery. ICS also differ from the standard control systems as ICS are built to perform in stand-alone mode continuously for 24 hours of the day. ICS also combines various control loops such as input signal, output signals, controllers, switches and present a single unit, easy-to-modify approach to deal with complex control plants. These are the reasons that ICS are widely used in almost every control system plant, over various domains throughout the world.

As mentioned above, continuous availability and time critically are important aspects of ICS. [15] To ensure these factors in communication between ICS devices, specially devised industrial communication protocols are used. Some examples of these industrial communication protocols are Modbus, EtherCAT, Profinet. The main aim of these protocols is to provide fast and reliable communication for accurate operation of industrial plants. Some of these industrial communication protocols such as Modbus, was developed in the 1970s and is still widely used in the industry. At the time when this protocol was developed, access and usage of internet was limited and therefore, the risk of hacking such a communication between ICS components remotely was limited. Hence, the risk of cyber attacks on these ICS was trivial and the threat level was at a minimum level. Therefore, the industrial communication protocols developed in the early era of ICS did not focus much on the security part of the communication, focusing more on ensuring reliable and fast communication. As ICS are used

for control of critical infrastructure, these industrial communication protocols are developed with the focus on system availability, while confidentiality in communication within ICS devices occupies lower priority.[16] Due to these vulnerabilities in communication, ICS are facing a grave risk of cyber attacks in today's world.

To identify the reasons which increase the risk of cyber attacks in today's world, we compare the ICS scenario in the early stages of ICS development with a typical scenario observed these days. In the early days of the industrial revolution using PLCs as ICS, the sensor-controller-actuator networks would use only local networking protocols working in a master-slave configuration. However, with the advancements in the use of the internet, newer techniques like Industrial Internet-of-Things (IIoT) [1] and Big Data [2] are becoming essential aspects of ICS. The ICS are therefore no longer only used in isolated networks but are connected to LAN networks on the enterprise level and to the internet to get access to real-time data anytime, anywhere.

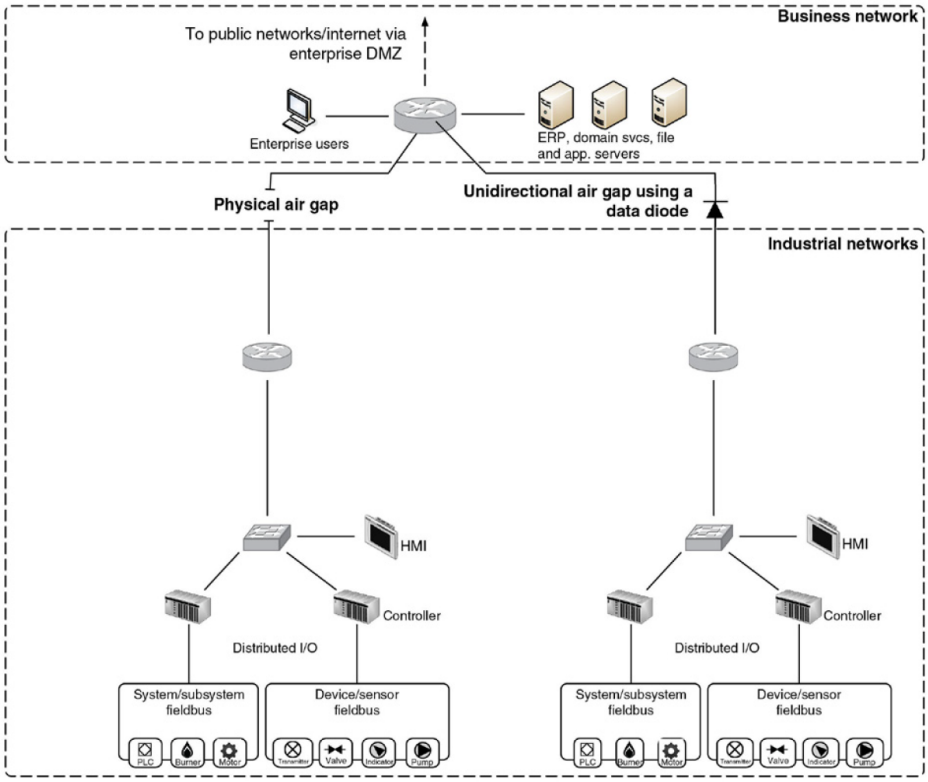

Figure 1-1: Ideal air gap separation between industrial and business networks [1]

Figure 1-1 portrays two traditional older scenarios of ICS networks. The business network

[1] *The industrial internet of things (IIoT) refers to interconnected sensors, instruments, and other devices networked together with computers' industrial applications, including manufacturing and energy management. This connectivity allows for data collection, exchange, and analysis, potentially facilitating improvements in productivity and efficiency as well as other economic benefits.* https://en.wikipedia.org/wiki/Man-in-the-middle_attack

[2] *Big data is a field that treats ways to analyse, systematically extract information from, or otherwise deal with data sets that are too large or complex to be dealt with by traditional data-processing application software.* https://en.wikipedia.org/wiki/Big_data

shown here is the internet network to which all the office computers, laptops and other communication devices are connected. It is easily accessible to any user who is logged in via any computer or laptop on that network. In this traditional scenario, the ICS devices such as PLCs do not possess the abilities to communicate via internet. Therefore, they are not connected to this business network having internet access. This creates a physical separation ie. physical gap between ICS and the enterprise network. This physical gap makes it difficult if not impossible to penetrate to the actual control system networks without physical access to the network. In the right-hand side scenario in figure 1-1, the ICS network is connected to the business network but there is only an unidirectional flow of information from the ICS to the business network. This makes it difficult to harm the control system by a cyber attack. Therefore, in this ideal case, the risk of cyber attacks is reduced to a minimum level. But, as explained earlier, this is rarely the case in today's world of IIoT and data processing.

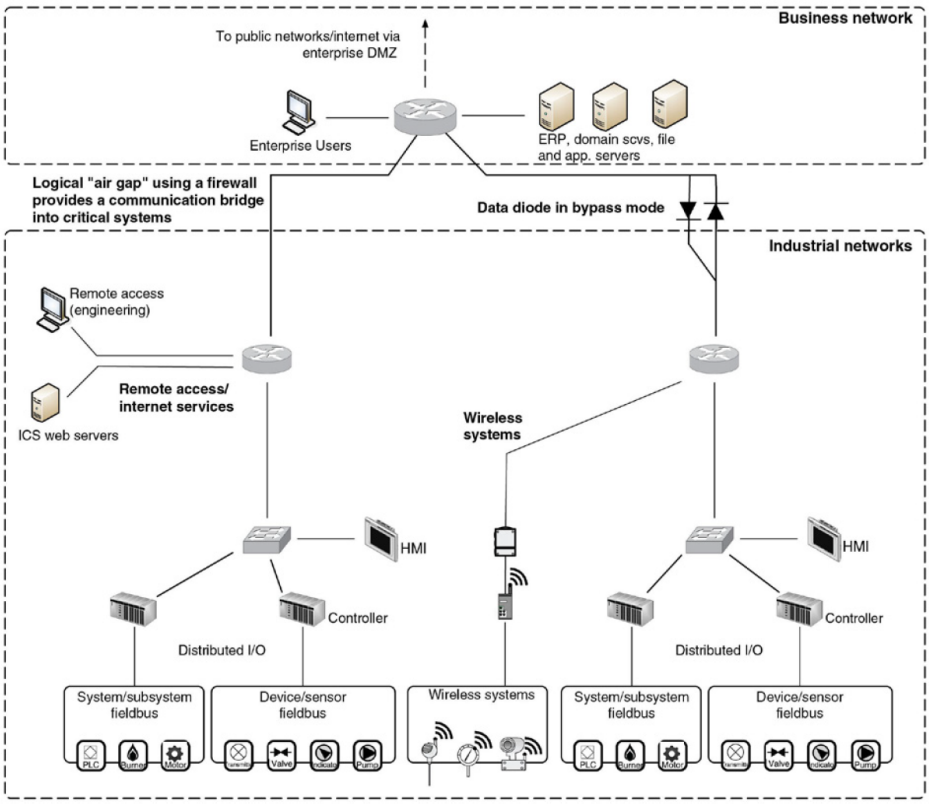

Figure 1-2: Actual scenario of interconnection between industrial and business networks [1]

This realistic case of ICS network in today's world can be observed in figure 1-2 which shows a typical ICS setting where control systems share data in a bidirectional manner with the business level network. This network is connected to a public server. This network topology makes it easier for an attacker to penetrate the actual control system of the plant remotely. The newer wireless sensor networks decrease this air gap further down as they use WiFi and wireless protocols based on the internet for communication. These developments have increased the risk of cyber-attacks on the ICS tremendously. With easy access to the plant networks, the treat vectors for cyber attacks have increased and that has increased the probability of cyber-attacks. Along with this, the cyber criminals are constantly working to make

these attacks more effective with higher penetration levels.

Cyber attacks on ICS and case study

Cyber attack is a term that has become an integral part of our vocabulary and is used widely. Different definitions of cyber attack exist and they all point in the same direction. In order to maintain uniformity in the context with this report, cyber attack is defined as "any operation, whether in offence or defence, intended to alter, delete, corrupt, or deny access to computer data or software for the purpose of (a) propaganda or deception; and/or (b) partly or totally disrupting the functioning of the targeted computer, computer system or network, and related computer-operated physical infrastructure (if any); and/or (c) producing physical damage extrinsic to the computer, computer system or network." [17]

As discussed earlier, there exist many vulnerabilities in the ICS components and industrial communication protocols, which make ICS networks vulnerable to cyber attacks. Attackers have been exploiting these vulnerabilities from the decades to launch cyber attacks on cyber-physical systems controlled by these ICS creating vast impacts. To understand the extent of these cyber attacks and their impacts, a detailed research was conducted and its output is displayed in figure 1-3.

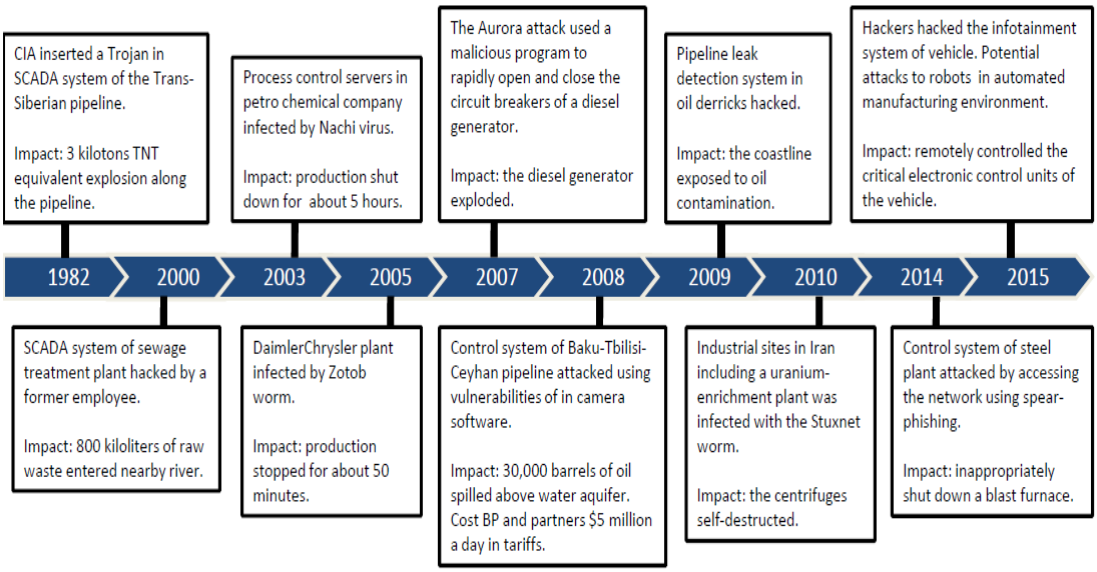

Figure 1-3: Major cyber attacks on ICS

Figure 1-3 presents major, historic cyber attacks on ICS and the hazards that they caused. A key list of these cyber attacks [18],[19],[20],[21],[22],[23],[24],[25] was created along with their impacts and presented in a timeline format in figure 1-3. These cyber attacks were injected on various ICS systems and components such as Supervisory Control and Data Acquisition (SCADA), PLC, etc. Their hazardous impacts ranged from shutting down a plant for hours to causing an explosion in oil pipelines. We will now discuss one of the most important and most impactful of these cyber attacks i.e. the Stuxnet cyber attack on Iran's Nuclear Program.

Stuxnet attack that disabled Iran's Nuclear Program :
Stuxnet was the first well-studied cyber-attack on an ICS. The attack was launched on Iran's Natanz uranium-enrichment facility in 2010. The core control of the nuclear centrifuges was dependent on Siemens PLCs network running on Siemens Step 7 software. This software executed the control logic for the control of centrifuges. The Stuxnet malware cyber-attack infected the entire control system of the nuclear plant and is reported to have damaged one-fifth of nuclear centrifuges at the facility. This attack set the nuclear program of Iran on a back foot by a couple of years.[26]

Stuxnet malware was introduced through an auto-execution removable drive and also could have been introduced through any device connected to the plant's Local Area Network (LAN). This attack exploited network protocols vulnerabilities to get access to the Siemens Step 7 PLCs. Stuxnet performed an auto scan to identify the PLC software, and once it found a precise configuration of Siemens Step 7 PLCs, it modified the entire ladder logic code running on the PLC while suppressing other alarm notifications. Once the code was loaded on the PLC, the hidden ladder program was executed, creating malfunctions in the operations of the nuclear centrifuges. The attack was so advanced that it even intercepted the communication between PLC and Human Machine Interface (HMI) running on the industrial network and altered these messages to show no alarms, warnings or faults on the HMI where the human operator was located.

This attack used the following three access points or pathways as the points of attack.

- Infected USB drive to get physical access to any system on the plant network
- LAN communication over Transmission Control Protocol (TCP)
- Infected Siemens Step 7 software file

This attack demonstrates the real threat of cyber attacks on ICS. The attack opened the eyes of authorities and cyber-security experts worldwide, showing them that ICS are not immune to cyber-attacks. Instead, even a simple cyber-attack on ICS can produce unimaginable financial and human life loss. One can only imagine the devastating effects if such an attack is ever launched on a nuclear power plant such as Chernobyl. Therefore, studying, analysing and preventing cyber attacks on ICS has gained tremendous importance within the last decades.

Global market share of ICS

Being the industrial controller in the ICS, PLC is at the core of the ICS network. According to the report of "Industry Research" [27], the global PLC market attributed to US$ 12.9 billion in 2020 and the market size is projected to grow to US$ 15.940 billion in the coming five years. Europe region indeed leads the share of this PLC market. These figures show the vast investments in these control systems and the high financial risk associated with them, as they are deployed for automated control of large plants and manufacturing units. Moreover, many of these plants use PLCs to control critical infrastructures such as boiler control, nuclear reactor control, etc. Failure in the operation of a single PLC control loop can thus incur substantial financial and infrastructural losses and, in the worst case, can

cause human causalities. Therefore, the safety of these ICS systems is a matter of paramount importance.

From the discussion till now, it is clear that massive financial losses as well as human life dangers have been caused by various cyber attacks on ICS in the past decades. This threat of cyber attacks only increases in today's world, where ICS are increasingly being connected to internet, making it easier for the attacker to target them. Before we can invent preventive algorithms against these cyber attacks, it is hence important to identify the root causes of these attacks.

As explained in the above chapter, cyber-attacks exploit the vulnerabilities in the communication network to which the ICS are connected. ICS devices such as PLCs interact with each other via industrial communication protocols such as Modbus, Profibus, Profinet and many more. Therefore, in order to gain more insights into the cyber attacks, their causes and their impacts, we need to analyse these communication protocols and identify the vulnerabilities in network communication. In order to analyse ICS with these industrial protocols, we need to plan and perform experiments on such devices. Only after such careful experimentation can we determine the extent of cyber threats, quantify the risks associated with them and propose protection strategies to against these cyber attacks.

However, conducting such experiments for injecting and testing cyber attacks on ICS implemented for control of critical infrastructure is not practical, as that will involve several financial risks and safety issues. To give an example, suppose we want to test cyber attacks on Siemens control systems controlling the nuclear plant (as in the Stuxnet example above). To analyse the risks of cyber attacks, we will need to develop, inject and test impacts of cyber attacks on these systems. But, these tests can not be performed on the live plant as that can cause irreversible changes to the plant state and endanger human lives. This gives rise to key problem in research of cyber security for ICS. This key problem is the lack of proper infrastructure to test cyber attacks on ICS in a secure and safe environment, while still producing accurate real-world results. [28], [29] This problem is addressed, by using the approach of a testbed built with ICS, for cyber attack generation and testing.

In research culture, testbed has various meanings. Testbed definitions range from a prototyping environment, to a demonstration capability, to a training facility. However, in this report, we define testbed as a controllable cyber environment that enables experimentation.[30] As mentioned in the above paragraph, the key problem in the research of cyber security of ICS is the lack of secure and safe environment. If we are able to build a testbed which utilizes the same infrastructure as that of the real world, but at the same time, ensures safety of equipments; this will lead a way in opening doors for fundamental research in cyber security. Such a testbed would provide multiple advantages over a real-world system.

1. A cyber attack testbed for ICS would provide a safe and secure environment where cyber attack can be tested without the risk of financial and human life losses.
2. If the testbed is created with using relevant real world ICS components such as PLC, HMI and using industrial communication protocols, then the testbed will be able to replicate the real-world ICS scenario, producing accurate results for cyber attacks.
3. A testbed would prove to be a cheaper alternative over real-world industrial plant, as it would cut costs for long cables, control cabinets and cost for installation of higher safety level equipments, required in an actual industrial plant.

4. Building the testbed in a laboratory provide more flexibility as it will allow us to recreate multiple ICS scenarios with different levels of complexity using the same ICS devices, which would not be possible in industrial setting. As the testbed would be located in a laboratory with all ICS devices at one place, it will be easier to reprogram the devices then on the industrial plant.

5. Using a testbed with real-world ICS devices to test cyber attacks would prevent the downtime in the actual plants, functioning over ICS.

In the discussion above, the need for following testbed approach for cyber attack analysis on ICS is clearly explained. We also discussed several key advantages of using a testbed approach for this. From this discussion, it becomes clear that developing and utilizing a testbed with ICS devices is the preferred approach for their cyber attack analysis, as cyber attack can not and must not be injected on a live plant controlled by ICS. From our research, very few of such testbeds actually exists currently in the world. Most of these testbeds [8], [9], [11], [12], [10] are developed by different universities in collaboration with the industry and are used for the internal research purposes. These testbeds are discussed in detail in section 3. The major disadvantage of most of these testbeds is that they do not use all ICS components such as PLC, HMI and instead use simulated PLC or simulated HMI software, which do not produce results as accurate as the real world counterparts. Apart from this, many of these testbeds use in-house developed components and are not open-source. This makes it difficult to recreate the testbed to develop other ICS scenarios and test cyber attacks on it. Hence, it is vitally important to develop a new testbed to overcome the disadvantages in the existing testbeds and provide a safe and secure environment for generating and testing cyber attack at Delft University of Technology (TU Delft). Therefore, this report will try to answer the following key research question.

Research question

> Can a cyber-attack testbed be built in the Networked Embedded Robotics in Delft (NERD) lab, which can replicate a real-world ICS network with the accuracy needed to identify and test vulnerabilities of ICSs working on Modbus protocol to cyber-attacks?

Further, we layout the following research sub-questions as guidelines in answering our final research question.

Research sub-questions-

1. Which of the vulnerabilities of Modbus protocol in the real world ICS network can be exploited with this testbed to launch cyber attacks on the ICS?

2. How are the main types of cyber-attacks on ICSs, namely Man-In-The-Middle (MITM) attack [1] and Denial-of-Service (DoS) attack [2] implemented in the real world?

[1] *a man-in-the-middle (MITM) attack is a cyber attack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other.* https://en.wikipedia.org/wiki/Man-in-the-middle_attack

[2] *A denial-of-service (DoS) attack occurs when legitimate users are unable to access information systems, devices, or other network resources due to the actions of a malicious cyber threat actor. [31]*

3. How can the two attacks mentioned in point 2 be introduced in the ICS network modelled in the testbed?
4. Which software can be utilised to launch the cyber attacks on the ICS network in this testbed?
5. What software can be used to monitor and capture the Modbus data packets to analyse successful implementation of the cyber attack?
6. What are the main hardware components needed to make this testbed replicate real world ICS scenario accurately?

In the following chapters, we will explore these aspects to design such a testbed and simulate this testbed with open-source softwares to inject cyber attacks. This thesis report starts by discussing case studies to show how cyber-attacks on these ICS pose high financial risks as well as fatal consequences to human life. In the next chapter, an in-depth study of Modbus communication protocol is presented, highlighting its vulnerabilities. This chapter also provides the background knowledge about the cyber attacks and their injection methodologies with reference to a typical ICS scenario. The third chapter analyses existing testbeds and compares different approaches to develop an ICS testbed. Based on this detailed analysis, the fourth chapter presents the novel design of the testbed to be built at Delft University of Technology (TU Delft). Further, the testbed is designed to the smallest details in this chapter and the components required to build this testbed are ordered after multiple rounds of discussions, ensuring satisfaction of all stakeholders involved in the process. Additionally, the plant simulator has been setup and documented in detail for future use. The fifth chapter then presents a simulated testbed where MITM and DoS attacks are actually injected and tested on the simulated ICS network. The thesis report is hence concluded with some suggestions for future improvements to the project.

The next chapter will give all the prerequisite knowledge to understand the entire work in this report. In this chapter, all the functional details about Modbus industrial communication protocol and two main types of cyber attacks are summarised. The chapter looks into the Modbus industrial communication protocol to understand its working and highlights its vulnerabilities. A typical ICS scenario is also discussed in this chapter which is used as a model throughout the report. Finally, two main cyber attacks on ICS, which are MITM attack and DoS attack, are discussed, and their working principles are studied.

Chapter 2

ICS, Modbus and cyber attacks on ICS

2-1 Modbus industrial communication protocol

Modbus is a communication protocol launched by the Modicon brand in the late 1970s, used widely for industrial communications in large-scale industries such as the process industry and chemical industry. Modbus implements a simple mechanism of sending message request and receiving a reply between a client and a server device. In a typical Modbus network, the "client" device which requests the information is equivalent to a "master" device in the networking terms and "server" device sending the required information to the client corresponds to a "slave" in standard networking terms.[1]. To give an example, a Modbus client or server can be a PLC or an HMI used for visualising and sending commands to the PLC.

A typical example of a Modbus communication network between control systems deployed in an industrial plant is shown in figure 2-1. In this figure, various ICS components such as PLC with the input and output devices are communicating with each other over Modbus protocol. Refer to the red box in figure 2-1 to see how multiple PLCs are communicating with each other over Modbus TCP protocol. Modbus TCP protocol uses an Ethernet cable to transfer the data between devices. In the red box in this figure 2-1, an HMI is working as a Modbus client communicating with the two PLCs, which are working as Modbus servers, via Modbus TCP protocol. The figure also shows other types of Modbus protocol, such as Modbus with RS232 and RS485, which communicate via serial mode, as explained in section 2-1-1.

Modbus is an application layer protocol. Therefore, the physical layer, type of cabling and line of communication required for Modbus protocol is flexible. Modbus communication was

[1] *The Modbus Organization Board of Trustees announces it is expunging all occurrences of inappropriate language of the query and response paradigm of Modbus communications. All instances of "master-slave" in the organization's literature and on its website will be removed. The organization is using "client-server" to describe Modbus communications, characterized by communication between client device (s), which initiates communication and makes requests of server device(s), which process requests and return an appropriate response (or error message). [32]*

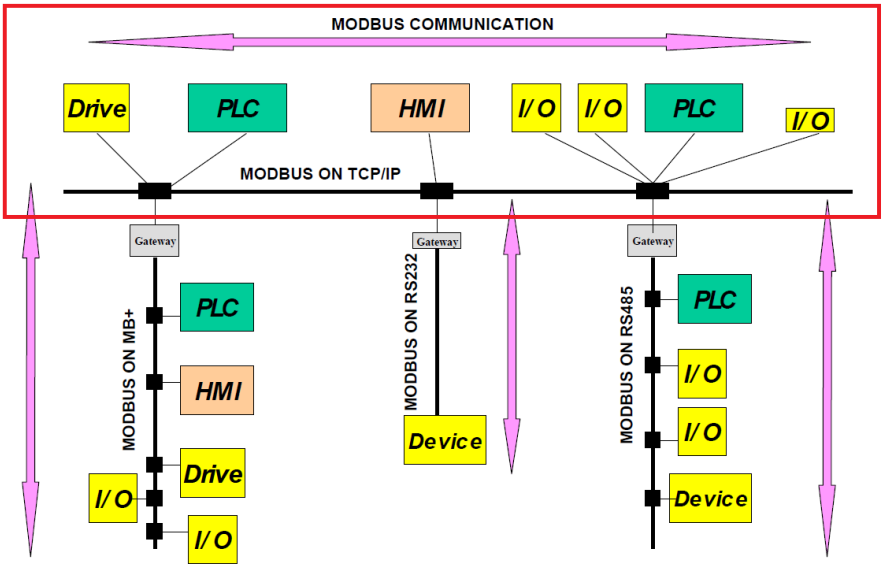

Figure 2-1: Example of Modbus network [2]

originally built to utilise a serial mode of communication and was extended to communicate over Ethernet TCP/IP mode of transfer when industrial Ethernet was developed.

2-1-1 Modbus with serial communication

Modbus is used with the serial setting in two variants. These two modes are different in their communication parameters, such as a baud rate, parity mode, etc. This serial communication takes place over RS232 or RS485 communication standards and is also referred by the standard names in the industry as "RS232 serial communication" or "RS485 serial communication". Here, the baud rate refers to the maximum number of bits that can be transferred per second between two devices using the Modbus serial protocol.

- Modbus Remote Terminal Unit (RTU) - In this Modbus setting, each of the 8-bit data in a message is sent using 4-bit hexadecimal characters. RTU mode has the edge over ASCII mode as it offers higher character density and hence better data transmission capabilities at a baud rate than that offered by the ASCII mode. There are 8 bits in every message of an RTU mode transmission that contain the actual data. Additionally, there is one start bit, one stop bit and one bit for parity completion. Therefore, each message frame contains 11 bits in RTU mode of Modbus.
- Modbus ASCII - In this mode, there are 7 bits in every message that contains the actual data, plus one start bit, one stop bit and one bit for parity completion. Therefore, each message frame contains 10 bits in ASCII mode of Modbus. ASCII mode is implemented when the physical communication capabilities do not allow transmission using RTU mode.

In both of these modes, the message structure is identical. There are three sections in the architecture of the message, as shown in figure 2-2.

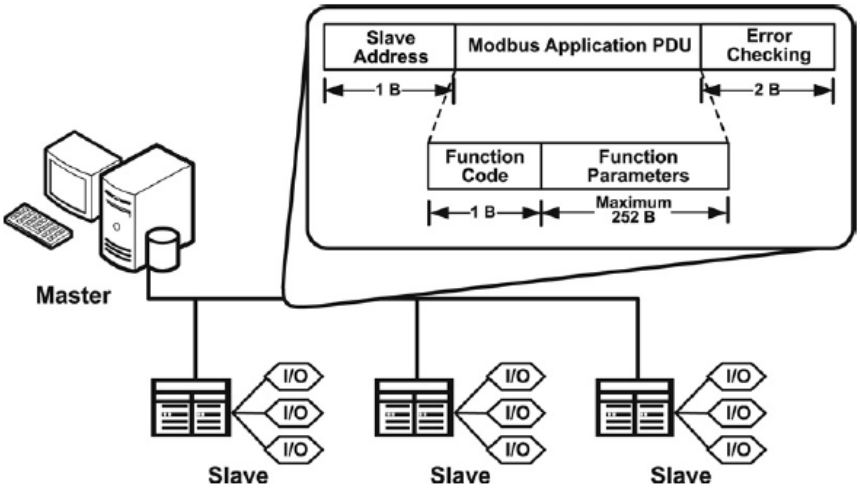

Figure 2-2: Message architecture for Modbus serial communication [3]

The server (equivalent to slave device) address in a message is used to identify a particular server device in the Modbus network, i.e. to decide the recipient. The Protocol Data Unit (PDU) section has two parts. First is the function code of 1 byte, which consists of a predefined number code that conveys the operation to be performed by the server in a request message. In the response message from the server to the client, this function code conveys the status of task completion (e.g. Error information and details, if any). The function parameter section contains the data or values or parameters of the sensor or actuator as requested by the client. Modbus function codes convey which operation to be performed to the server, i.e. read or write command to the servers, error codes or diagnostics codes. These function codes can be public codes, which are predefined by the Modbus organisation and are universally applicable, or user defined codes where users can generate and decide frequently used actions as function codes for their own use case. There are also some reserved function codes that are used to maintain the compatibility of Modbus protocol with legacy control systems. The last section of the message is error check field as shown in top right corner of figure 2-2. Modbus implements a Cyclical Redundancy Checking (CRC) or Longitudinal Redundancy Checking (LRC) algorithm to ensure if all fields of the every message are valid. If there is any discrepancy in these fields which make them in-acceptable by the Modbus frameworks, this error field is activated giving an error in the communication. [33]

2-1-2 Modbus with Transmission Control Protocol (TCP)

When industrial Ethernet became available and widely accepted, Modbus protocol was extended to be used in LAN based networks. Modbus TCP (Transport Control Protocol) makes use of LAN, which assigns Internet Protocol (IP) addresses to devices, making it easier to make multi-drop connections. Modbus TCP, designed specifically for LAN (Ethernet), hence improves capabilities of Modbus serial mode as it enables the client devices to connect with multiple servers as well as with other client devices in an IP setting. In figure 2-3, one can observe the improved capabilities of Modbus TCP. This can be understood by observing how the Modbus client is connected to other client devices on a TCP/IP network and can com-

municate bidirectionally with them as well as with multiple servers on the same TCP/IP network at the same time. All communications through the TCP/IP mode of Modbus take place through port 502 by default.

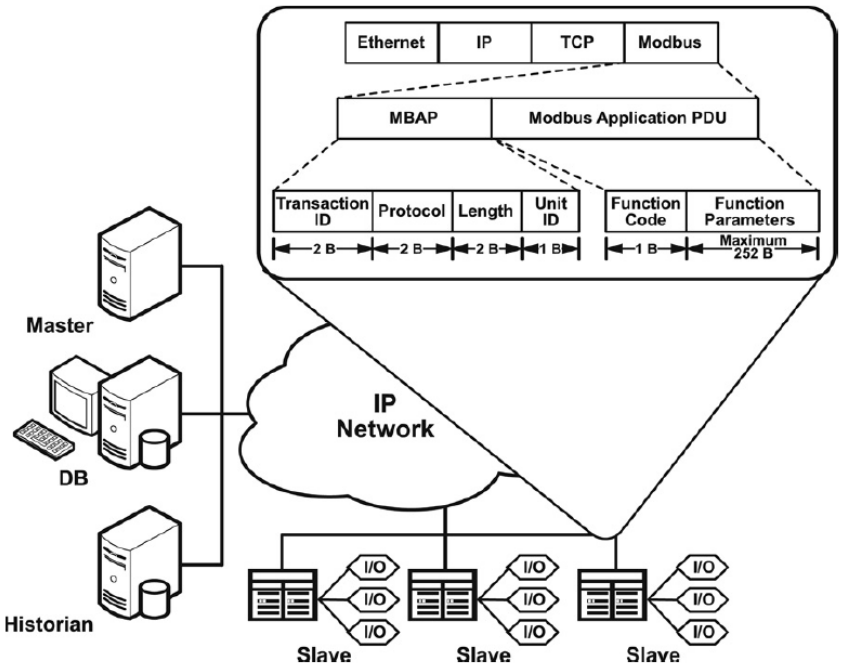

Figure 2-3: Message architecture for Modbus TCP/IP communication [3]

Modbus TCP message structure is similar in many ways to that of Modbus serial communication. The messages still follow the PDU structure, but in this case, they are encapsulated in a TCP message. The Modbus PDU, in this case, is preceded by a Modbus Application Protocol (MBAP) header. This dedicated MBAP header is 7 bytes long and is used to set and identify all parameters required for a successful TCP/IP communication. The sections of this MBAP header and their functions are listed in figure 2-4.

The details of various field of the MBAP header are given in the table in figure 2-4. The functions of these 4 fields in a MBAP header are explained below.

- Transaction identifier - These 2 bytes are utilised to pair transactions between the client and server. In the response message, the server copies the transaction identifier of the request sent by the client to pair its response correctly to the request, as shown in the first row of figure 2-4.

- Protocol identifier - In the multi-protocol system, this identifies the Modbus protocol by setting these bytes as 0.

- Length - This represents the bytes included in the successive fields which include a unit identifier and data fields. This field is shown in row 3 in the table in figure 2-4. This field tells the how many bytes are going to be present in the message after itself.

Fields	Length	Description -	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction.	Initialized by the client	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the client	Recopied by the server from the received request
Length	2 Bytes	Number of following bytes	Initialized by the client (request)	Initialized by the server (Response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the client	Recopied by the server from the received request

Figure 2-4: MBAP header composition [4]

- Unit identifier - This is used to facilitate communication to a Modbus serial slave by forming a gateway between Ethernet TCP/IP network and the Modbus Serial line bus. This is only used in legacy architectures.

The function codes used in the case of both Modbus RTU and Modbus TCP/IP are common, and we will now look into those in the next section.

2-1-3 Function codes for modbus

Function codes in a Modbus PDU message specify the action to be taken by the server. The server identifies the action to be performed by looking into the look-up table of function codes and performs the action as directed by the client. After the action is performed, the server sends a response message back to the client in which the same function code is copied to suggest that the action is performed successfully. In case of an error or inability of the server to perform the specific task, a different function code (representing a specific error) is sent in the response message function code section to the client.

The four basic actions performed by a control system through a Modbus communication are listed below. These form the basic data models of the Modbus protocol.

1. Discrete Input - This is a read-only boolean function that takes in a single bit of data. It is mainly used to take digital input data from an Input and Output (IO) system.
2. Coils - This is a read and write boolean function using a single bit of data. This type of data on coils can be altered by an application program.
3. Input Registers - Similar to discrete input, this is also a read-only function used to read data from an IO system. The difference is that, unlike discrete input, these registers utilise 16-bit data format.

4. Holding Registers - These registers are identical in nature to coils as their values can also be altered by an application program. These holding registers, however, deal in 16-bit data, i.e. word to read and write analog values.

As discussed earlier, function codes direct which of these actions is to be performed by the server and this information is passed on by the client to the server through every message request. Some common and unique function codes are predefined by the Modbus organisation for the Modbus protocol for easy use. Users can also define their own function codes, but they are not guaranteed to be unique. In figure 2-5, the most important public function codes are listed.

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	6.2
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	6.1
			Write Single Coil	**05**		**05**	6.5
			Write Multiple Coils	**15**		**0F**	6.11
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	6.3
			Write Single Register	**06**		**06**	6.6
			Write Multiple Registers	**16**		**10**	6.12
			Read/Write Multiple Registers	**23**		**17**	6.17
			Mask Write Register	**22**		**16**	6.16
			Read FIFO queue	**24**		**18**	6.18
	File record access		Read File record	**20**		**14**	6.14
			Write File record	**21**		**15**	6.15
Diagnostics			Read Exception status	**07**		**07**	6.7
			Diagnostic	**08**	**00-18,20**	**08**	6.8
			Get Com event counter	**11**		**OB**	6.9
			Get Com Event Log	**12**		**0C**	6.10
			Report Server ID	**17**		**11**	6.13
			Read device Identification	**43**	**14**	**2B**	6.21
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**	6.19
			CANopen General Reference	**43**	**13**	**2B**	6.20

Figure 2-5: Public Function codes for Modbus protocol [2]

Many of the cyber attacks on control systems communicating with Modbus protocol misuse these function codes to alter or stop the ongoing communication and bring hazardous, physical changes in states of the plant parameters. For example, consider a level control loop where the level of liquid in a tank is "read" by the level sensor and sent to a controller in every cycle. The controller then decides on the necessary action to be taken by the control valve and "writes" this value of the actuator output to the valve. All of the communications take place through the Modbus protocol.

The query generated by the client to "read" the sensor value and the response given by the sensor is shown in the following table 2-1. Notice the function code in the request and response as 04, which refers to the "read input registers" command. The command of reading the level value from the sensor is described entirely by the function code, which is embedded in the message sent by the Modbus controller. If an attacker tampers these function code values, let us say from a function code of 04 to 57, the action to be performed by the level sensor

Query generated by client	Response given by the sensor
00 14 00 00 00 06 01 04 00 C8 00 02	00 14 00 00 00 07 01 04 04 27 10 C3 50

Table 2-1: Modbus query and response message structure

changes from "read input register" to "write single-coil", as described in figure 2-4. As the sensor is an input device, it is unable to perform a write action, and hence it will either give an error response or give no response to the request sent by the controller. In either case, the communication is broken down, and the control loop can be made uncontrollable and nonoperational. All this can be simply done by changing a single bit value of the function code in a PDU of Modbus request. We have hence provided all necessary information regarding the Modbus protocol in this section.

2-2 Typical Industrial Control Systems (ICS) scenario

Figure 2-6 shows the typical scenario where ICS systems are implemented in an industrial setting. Here, the ICS system is implemented to control the large scale, complex plant, which can be any industrial process. The plant consists of endpoints of the ICS, namely sensors and actuators. Sensors are used to gather data about the process, such as reading the level of a fluid in a tank or temperature in a furnace. Whereas actuators are the final control elements that take the command from the controller to change the state of the plant. Examples of actuators include valves or motors which work on electromagnetic, pneumatic or hydraulic principles.

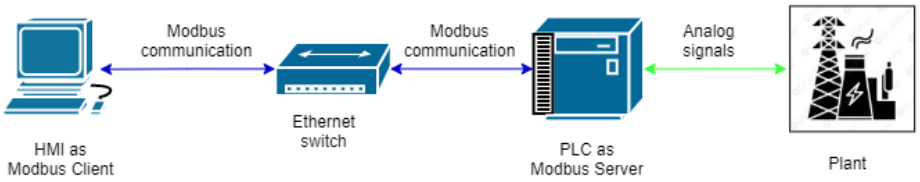

Figure 2-6: Typical ICS scenario

As described in chapter 1, PLC is at the heart of the ICS system, being the main controller of the plant. The live readings of the plant states are passed by the sensors to the PLC as analog signals. The PLC also relays the output values to be written on the actuator via analog signals. The PLC is connected to the HMI through standard Ethernet cables. All the communication between HMI and PLC goes through these Ethernet cables over the Modbus protocol. HMI is a display where the human operator is sitting in the central control room of the plant. This HMI serves two main purposes. First, it displays all the real-time input and output values in the plant, such as water level, temperatures and valve settings. Second, it allows the human operator to enter the reference values according to which the PLC logic works.

To understand the operation of this ICS setup, let us look at a simple example of a level control application in a closed tank with a single input and single output. Here, the level sensor will continuously read the water level in the tank and send it back to the PLC. Similarly,

a solenoid valve will take the real-time commands from the PLC, opening and closing in the required proportion to allow inflow of water. These values are also relayed in real-time on the HMI screen where the operator is sitting. Now, the operator will decide the level of water in the tank to be maintained at any point and enter this reference value in the HMI. The HMI will then relay this reference level value to the PLC, and the PLC will run its logic to open or close the valve as needed to bring the water level into the tank as instructed by the operator. In this way, the control of level in a tank is performed by the ICS using PLC and HMI.

Now, as explained in section 2-1, the HMI, in this case, acts as Modbus Client, requesting data from the server. The PLC acts as Modbus server, replying to the client's requests by implementing the command to read the data, i.e. level value or write the data, i.e. the reference value on it to execute the control logic. This shows the typical ICS scenario in an industrial setting and is referred throughout this thesis to mimic the real-world ICS scenario into a testbed.

In the next section, a detailed study of cyber attacks on these ICS systems is presented while referring to the Modbus client and server terminology learned in this section for ICS.

2-3 Types of cyber attacks on ICS

In this section, the most common types of cyber attack are discussed. The chapter describes the general methodology of implementing these attacks on Industrial Control Systems (ICS). The more detailed specifics of each attack injection on Modbus TCP protocol and its methodology is described in chapter 5, where results of these attacks on the testbed at TU Delft are discussed.

2-3-1 Reconnaissance attack with Address Resolution Protocol (ARP) sweep attack

This type of attack can be used by the attacker to explore network architecture to obtain information about the involved components on the network, their IP addresses and their physical Media Access Control (MAC) addresses. This attack is often used by the attackers as a primary tool to get necessary information about the network and ICS devices before launching even more dangerous attacks on the system. The level of threat of a reconnaissance attack is increased when it is performed together with other types of attacks.

To perform this type of attack, a method called ARP scan is implemented by the attacker. This method is also commonly known as ARP sweep. Before understanding the functionality of this attack, two basic terms in networking must be understood. Those are IP address and MAC address.

An IP address is a 32-bit number assigned to every device on the LAN network, which communicates through Internet Protocol, i.e. IP. This IP address is unique for every device on a network. An IP address serves the following main purposes: First, it is used for unique identification of the device on a network and second, it is used to assign an address to the device used for communication. A device is assigned an IP every time it is connected to a network, acting as a logical address of that device on that network for communication. A

typical IP address looks as follows : **192.168.100.108**. Here, each dot separates four, three-digit decimal numbers, which contain 8 bits each. Therefore, an IP address is $4 \times 8 = 32$ bits long.

On the other hand, a MAC address is a 48-bit unique identifier assigned to the networking card of each device by the manufacturer. It is therefore referred to as an Ethernet hardware address which is unique worldwide. The MAC address serves the purpose of identifying the unique, physical address of a device, irrespective of which network the device is connected to. A typical MAC address looks like following **00:3C:75:26:5D:C4**. Here, each colon separates six, two digit hexadecimal numbers, containing 8 bits each. Therefore, a MAC address is $6 \times 8 = 48$ bits long. The difference between an IP address and a MAC address is that a MAC address is unique globally, whereas an IP address is unique to the network and is not physically engraved on the device.

Now that we have understood the functions and differences between IP and MAC addresses, we now see what an ARP scanning attack is and how it is performed. ARP is a communication protocol that associates the IP address of a device on a network to the unique physical MAC address of the device. Therefore, what ARP does internally is that it translates the 32 bit IP address to a 48 bit MAC address and vice-versa to link them with each other. This procedure is hence at the base of every communication on a LAN network.

To understand the working of ARP better, consider the following example. Two computers are connected with each other via an Ethernet cable to create a LAN network. Each of these two PCs, PC A and PC B, will be assigned an IP address that will be unique to this network, and they will also have a MAC address of their own which is their actual physical address. Now, suppose PC A wants to send a message to PC B. PC A will then send a broadcast message on the network with ARP protocol asking **"who has 192.168.100.108"**, which is the IP address of the destination. PC B will identify its own IP address, and since it matches the request, it will reply to the source with ARP protocol saying **"I am 192.168.100.108"**. This message will contain its own MAC address, which is sent in response to the request of the sender. In this way, the physical MAC addresses of each device on the network are linked with their IP addresses and stored in an ARP table or ARP cache. Entire communication on this LAN is now performed in reference to this ARP table, identifying the source and destination devices for each message accurately. Figure 2-7 shows such an example of the ARP table performing on a LAN. The IP address of each device is identified and linked with its MAC address, i.e. Physical address. This can be clearly seen in table in figure 2-7.

ARP scan attack methodology

This attack is generally implemented as an initial step in launching actual cyber-attacks such as Man-In-The-Middle (MITM) and Denial-of-Service (DoS) attacks. This attack exploits the lack of authentication in ARP for identifying devices on the network and their details. The weakness in ARP, called lack of authentication, is that for every request written in ARP, the device replies without verifying if the host for that request is indeed an authenticated host on the network. Therefore, any PC on the network can send an ARP request to which the device with the correct IP address has to reply with its MAC address and other details.

Therefore, in this attack, the attacker device broadcasts ARP requests for each IP address in the range provided by the attacker. For example, if the LAN is working on a network

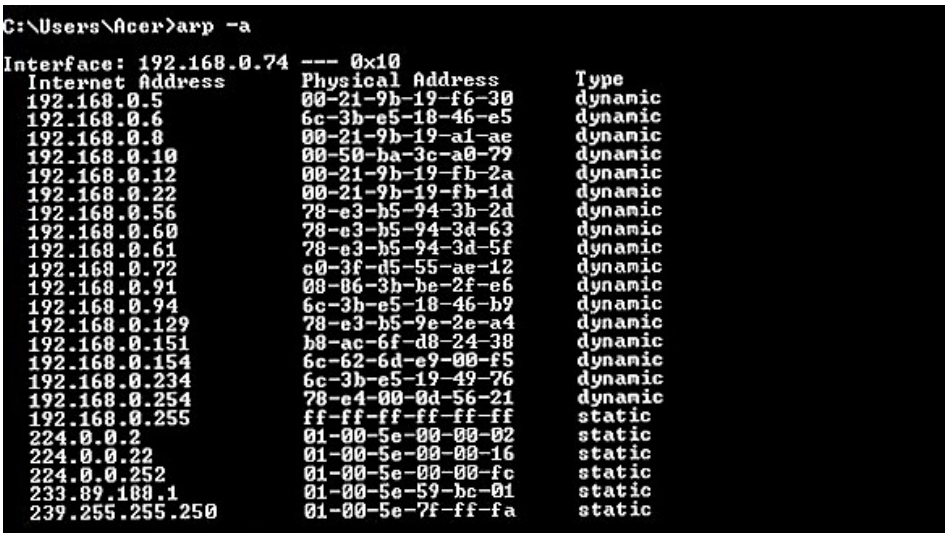

Figure 2-7: ARP Table example [5]

192.168.100.xxx with the attack PC having IP address as **192.168.100.108**. Then the attacker can send ARP requests for all addresses in the same subnet of the network. In this case, then, the attack PC will start from address **192.168.100.000** and go up to the address **192.168.100.255**, sending an ARP request of the format **"who has 192.168.100.xxx"** . Whenever there is an active device present on the network with the matching IP address, it will respond to this request with it its MAC address and details in the format **"I am 192.168.100.xxx"**. In this way, entire ARP table is therefore created on the attacker PC, having all the details of the devices working on the same network. Once this information is available to the attacker, more harmful Man-In-The-Middle (MITM) and Denial-of-Service (DoS) attacks can be launched.

To understand the threat caused by such an attack, let us consider a simple example of a water pipeline network for the city. Such an ICS system typically has standard actuator valves and sensors which provide data to RTUs, which are small PLCs placed over various locations, which in turn communicate with the central PLC. All of this lower level communication is taking place using Modbus protocol. Now, if the attacker gets access to any one node of the network, an address scan (ARP scan) attack can be launched, through which knowledge about IP addressing of Modbus clients-servers on the entire network is leaked. With this knowledge about network and device details such as IP addresses and MAC addresses, injecting other active attacks is made possible, which is discussed in the next sections.

2-3-2 MITM attack with ARP poisoning

MITM attack is an active eavesdropping attack in which the attacker device intercepts and alters conveyed data between a client and a server by impersonating one of the authenticated devices on the network. Figure 2-8 visualises the methodology of this attack where an attack PC "sits" between a client and a server device to intercept the communication in between them.

The injection of a MITM attack can be viewed in two steps. In the first step, the attack uses

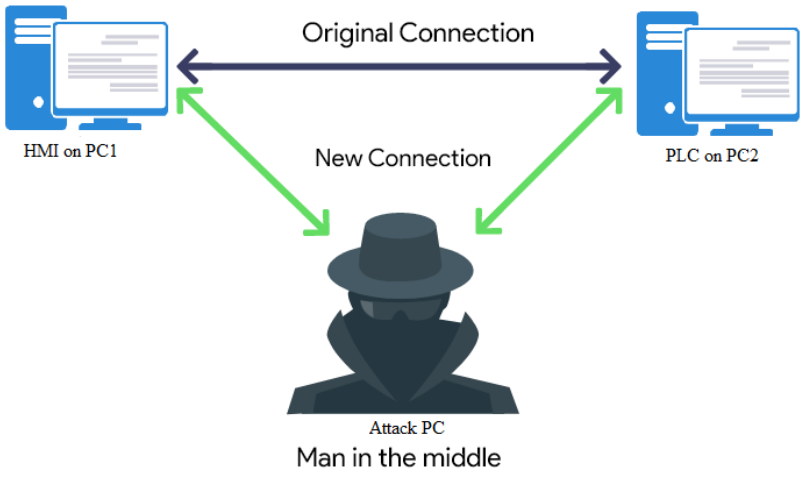

Figure 2-8: MITM attack visualization

techniques such as ARP spoofing (also known as ARP poisoning) to get itself in between the client-server network. In the second step, the attacker then captures and can modify the data being exchanged between the client and the server.

ARP poisoning

ARP poisoning utilises the technique of ARP scan attack as explained in section 2-3-1 to first gather the network information, including the IP and MAC addresses of the devices on the network. The attacker then uses ARP poisoning to link its MAC address to the IP address of an authenticated device on the network. This essentially changes the information in the ARP table. Since the devices "spoof" the network by copying fake MAC addresses, it is also known as "ARP spoofing". To give a better idea of this attack procedure, an example is given below.

Let's consider a network where a Modbus Client and server are communicating with each other and an attack PC is performing the MITM attack with ARP poisoning. Table 2-2 shows the details of the devices with their IP and MAC addresses on the network. These are the real details of the devices before an ARP poisoning attack is performed on the network.

Sr. No.	Role in the network	IP Address	MAC address
1.	Modbus Client	192.168.0.101	78-2B-46-34-82-85
2.	Modbus Server	192.168.0.103	3C-F0-11-64-F3-BB
3.	Attack PC	192.168.0.113	08-00-27-1c-49-7b

Table 2-2: IP and MAC addresses

Now, the ARP poisoning attack is performed on the network by the attacker PC. During this

attack, the attacker modifies the ARP table in both client and server by linking its own IP address to the MAC address of client and server. Therefore, in the ARP table of the client, the attack PC will modify the MAC address of the server with its own MAC address, making them the same. Hence, the messages that are usually sent directly from the client to the server directly will now instead go through the attack PC, as the attack PC now has copied the MAC address of the server. The same procedure is followed while poisoning the ARP table in the server PC. Here, in the ARP table of the server, the attack PC will copy the MAC address of the client. Due to this, the messages that are going from server to client will now be re-routed through the attack PC, as the attack PC now has copied the MAC address of the client. This ARP poisoning attack can be verified by checking the ARP table of the Modbus Client and Server given below.

Figure 2-9 shows the ARP table of the client. Compare this with the initial ARP table before the attack, where the original IP and MAC addresses of the devices are shown. Here, it can be easily observed that the ARP table has been poisoned by the attack PC by changing the MAC address of the server by its own MAC address. Therefore, this ARP poisoning attack completed the first and major step of the MITM attack, by which the attack PC is already introduced in between the client-server network to intercept and modify the communication.

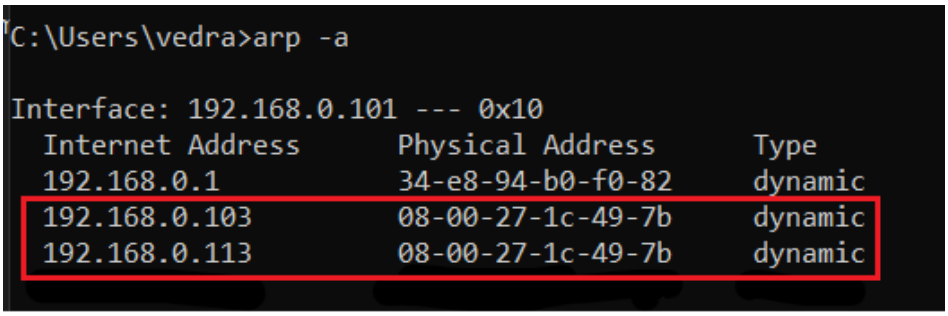

Figure 2-9: ARP poisoning attack on client side

Similarly, Figure 2-10 shows the ARP table of server. Here, it can be noticed that the ARP table has been poisoned by the attack PC by copying MAC address from the client PC.

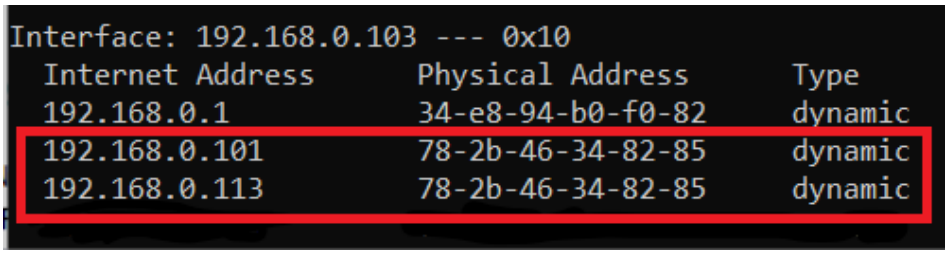

Figure 2-10: ARP poisoning attack on server side

The second step of the MITM attack is the actual data changing step. This step consists of modifying or altering the data going through the network to bring malicious changes to the network state. For doing this, various tools can be used, such as Wireshark, Ettercap, etc. These tools are capable of capturing the data packets going between client and server through the attack PC, which now sits between this client-server network, "sniffing" the data

between them. This is also hence known as "data sniffing", where an attacker passively gains the information being exchanged between the client and server and can further analyse this data to inject malicious attacks. In chapter 5, this MITM attack is implemented using the tools mentioned above to disturb the industrial process controlled by ICS communicating with Modbus protocol. This concludes the description of the MITM attack, and its results are described in detail in chapter 5.

2-3-3 Denial of Service (DoS) attack

Denial-of-Service (DoS) attacks are implemented in order to halt the normal functioning of the industrial control system. The line of action of these attacks is to overwhelm the client or the server by injecting bulk messages, which take time for processing and hence will cause a delay in reading vital information or even completely crashing of the communication between client and server.

Modbus TCP protocol is vulnerable to such DoS attacks as it is a routable protocol. This means that the Modbus TCP protocol allows third party devices to originate and inject DoS attacks on the target victim IP address. Modbus RTU and Modbus ASCII are not routable protocols as they work on the physical communication layer of RS-232 or RS-485. Yet, when the network is spread over longer distances in a star topology, the serial lines can be injected with DoS attacks.

As explained in section 2-1-2, Modbus TCP is an application layer protocol working over TCP/IP protocol. This means that in Modbus TCP, Modbus messages are sent over the layer of TCP protocol which used Ethernet cable to transfer the data. For establishing a successful communication between a client and server using TCP protocol, an initialisation procedure known as "three-way handshake" or "SYN-ACK" is followed. This can be observed in figure 2-11.

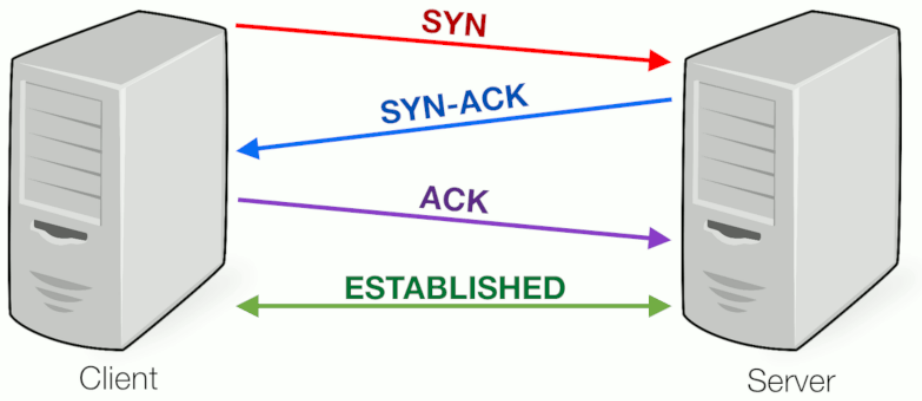

Figure 2-11: Three-way handshake for TCP communication [6]

The procedure to establish a connection when a client sends a **SYN** (Sync) request to the server. When the server gets this SYN request and is ready to respond, it responds to the client with a **SYN-ACK** (Sync-Acknowledgement). The client then responds to this with an **ACK** (Acknowledgement) message completing the three-way handshake between client and server. A successful connection is established only when this procedure is completed, and

then data can be transferred between the devices. When any of these three messages are not successfully sent, the connection is not established, and a specific amount of memory from the device remains blocked by such unresponded requests. In a DoS attack, this vulnerability of TCP protocol is exploited by the attacker. The technique used for this DoS attack is TCP SYN Flood.

TCP SYN Flood DoS attack

In this attack, the attacker utilises a vulnerability in a three-way handshake mechanism to essentially exhaust the device's memory and resources, such that it is then unavailable to respond to legitimate requests of establishing a connection.

During this attack, the attacker on the network "floods" the server with a huge number of SYN requests. The server keeps on responding to these SYN requests with the SYN-ACK messages, hoping to receive the final ACK message from the client, establishing the connection. However, the attacker does not send back the ACK response keeping the connections half-open. At some point, the resources of the server are drained, and it cannot further respond to any messages, even when they are coming from the legitimate devices on the network, producing a Denial of Service attack.

Now, as the attacker is also sending a high number of SYN requests and receiving the same number of SYN-ACK responses, the memory of the attacker device would also indeed be drained, stopping the attacker PC's functionalities as well. Therefore, to avoid this issue, the TCP SYN Flood attack continuously sends these SYN requests from randomly generated, fake IP addresses and ports. Hence, the ports of attacks PC does not get saturated with the SYN-ACK messages, preventing it from being a victim of its own DoS attack. This procedure is shown in figure 2-12, where the memory resource of the victim is diminishing due to the SYN requests from the spoofed IP addresses, while the attacker remains fully functional.

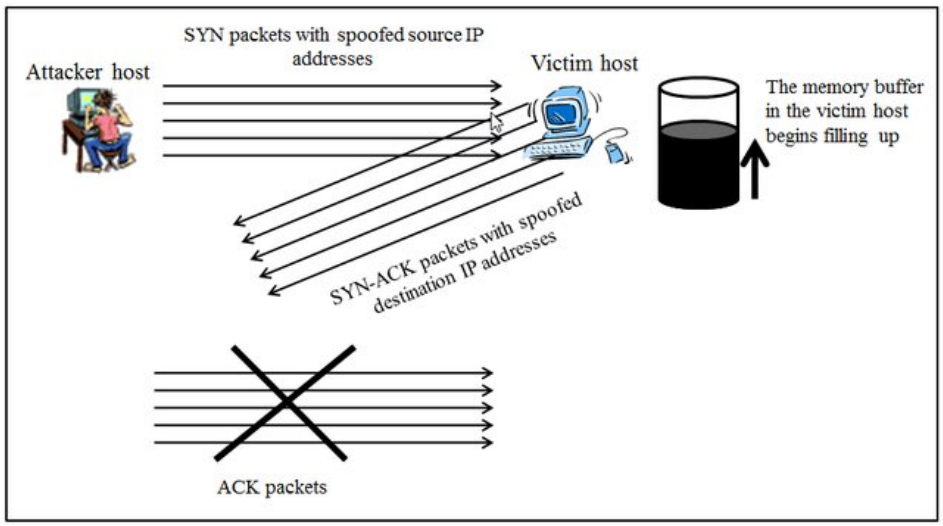

Figure 2-12: TCP SYN Flood attack with spoofed addresses [7]

In this way, a DoS attack can be implemented on a plant controlled by ICS working on Modbus protocol. In chapter 5 a DoS attack is implemented on a simulation of the NERD

lab testbed, and the resulting plant behaviour is shown.

This chapter gives all the background knowledge, which is important to understand the work performed during this thesis. In the first section of the chapter, a detailed account of Modbus protocol along with its types and function codes was studied. In the second section, a typical ICS scenario was discussed relating the roles of Modbus client and servers to HMI and PLC respectively. The third section described reconnaissance attack with ARP sweep, MITM attack with ARP poisoning and DoS attack along with their methodologies. These methodologies are referred again in chapter 5 to generate and inject cyber attacks on the simulated testbed during this thesis.
Now that this background knowledge about ICS is clear, the next chapter will analyse various existing testbed, in operation at different research laboratories, for testing cyber attacks on ICS.

Chapter 3

Analysis of existing ICS testbeds for cyber attack testing

In the previous chapter, a detailed account of Modbus industrial protocol and cyber attacks injection on Industrial Control Systems (ICS) was given. The previous chapter also described a typical ICS scenario in the real world. Now, in this chapter, these concepts are utilized in order to analyse various testbeds which are already in operation across various research institutes in the world.

The first five sections of this chapter describe architecture and functioning of five testbeds, implemented in various research institutes. Further, for each of these ICS testbeds, a description of cyber attacks that are tested with the testbed is provided. The last section of this chapter compares these testbeds in depth to draw conclusions on the approaches to build an ICS testbed at Delft University of Technology (TU Delft).

3-1 Industrial control systems security testbed at Binghamton university

The testbed [8] created at Binghamton university uses actual hardware to create the system to be controlled instead of using a simulated plant. The ICS devices used are also actual PLCs which make the testbed close the real-world ICS system. Using actual hardware of sensors and actuator makes it possible to identify vulnerabilities to a higher extent concerning the physical properties of communication and plant .

Figure 3-1 shows the architecture of the testbed with its components topology.

3-1-1 Functioning of the testbed

The plant to be controlled here is basically 2 motors coupled with each other, as seen at the bottom of figure 3-1. An AC motor driver is used to control the voltage sent to the motors

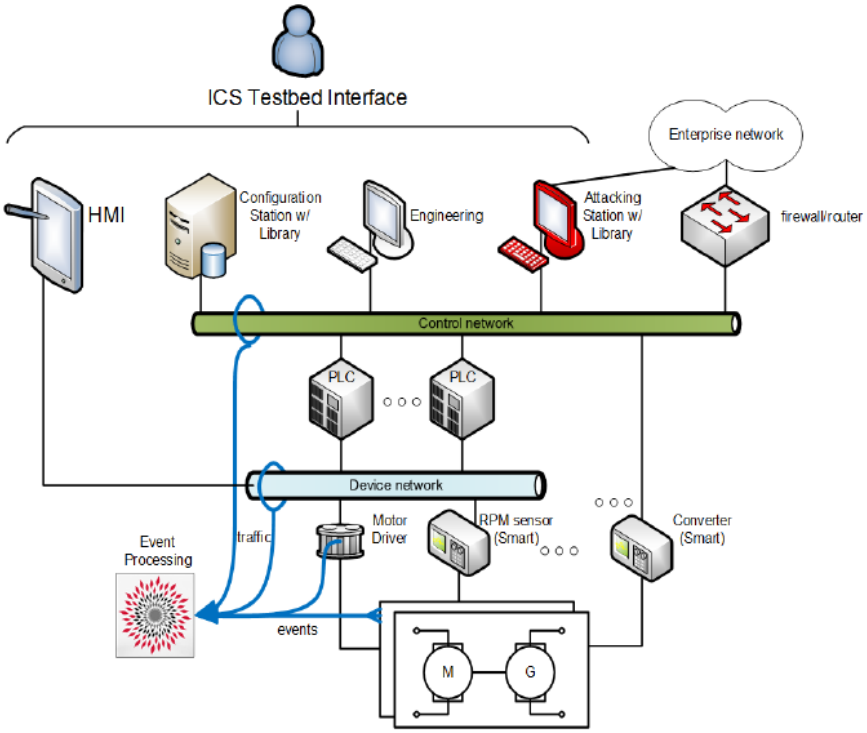

Figure 3-1: Architecture of the BU testbed [8]

to control their speed. The first PLC is connected to the RPM sensor and receives the values from the sensor on analog input cards. The second PLC is connected to the motor driver. This PLC runs the control code written in ladder logic to decide the control input and sends it back to the motors. The two PLCs are connected over the same device network and can communicate with HMI through this network. The PLCs send these values through to the Personal Computer (PC) running HMI software over modbus protocol. An operator stationed at this HMI station can hence visualize current values of the plant and can also change the parameters such as set points in real time. The PLCs can be programmed through the engineering station, which is connected through the control network to them. This situation replicates the real world scenario in a control plant running on ICS, except the fact that the control setup in this case is much smaller than the real world plant.

Communication between this entire control testbed takes place with Ethernet/IP protocol. Rockwell Ethernet/IP modules are installed on the chassis of the PLC to send the Modbus data to the HMI. RSLinx Enterprise communication server is used for sending and receiving data between controller and HMI. The Modbus data packets which are exchanged on this network are captured and analysed by using Wireshark [34], which is a real time-packet analyzer.

The attack PC is connected to the same "control network" shown by green network in figure 3-1. This PC has the BU-testbed attack software running. First, a reconnaissance type of attack (ARP scan) is performed on the ICS network. The attack is implemented on the network by capturing data packets by poisoning ARP (Address Resolution Protocol) between

PLC and HMI operator station. The static data of the plant i.e. the configuration information about components, their make, model and limits of operations can be extracted by the attacker. This is followed by injecting an altered control set point attack with MITM technique as discussed in section 2-3-2. The frequency operating limits of motors are manipulated during the attack and hence speed of the motor was increased drastically which makes the system crash with overproduction of voltage, this failure caused in this small setup of motors generating electricity demonstrates the huge failures which can be caused with such a cyber attack in the actual power generation plant.

The testbed also successfully implements DoS attack on the network which is very common in real-world complex plants having internet connectivity to cover monitoring of the wide area. A large flow of data packets carrying random information to the controller-HMI network was created by the attacker PC. The network was unable to handle such large traffic simultaneously and as a result crashed with the loss of communication.

3-1-2 Advantages and disadvantages of the testbed

The advantages and disadvantages of this approach can be stated as follows.

Advantages:
1. The testbed gives a simplified approach to analyse cyber security in ICS by reducing complexity of the plant by taking a smaller control setup instead of the large plant network.
2. As no simulator is utilized to simulate plant, the cost of the testbed setup is low as compared to the other testbeds listed ahead in this report, which utilize a plant simulator.

Disadvantages:
1. The testbed is not able to accurately replicate the real world scenario of ICS implemented in a complex plant.
2. One of the most important limitations of this testbed is that it can not perform and observe impact of cyber attacks on large scale control plants for which ICS is ideally used in the real world.
3. Since the software tool used for injecting the cyber attacks on the ICS network is an in-house developed component created at the BU under two Air Force grants, the testbed can not be replicated easily universally.

3-2 Testbed for implementing attacks on cyber-physical systems at Texas AM university

We will take a look at a testbed built for cyber attack generation for the application of smart grids at Texas AM University [9]. This particular testbed is built to implement attacks on a network communicating with Modbus TCP/IP protocol. The detailed architecture of this testbed is given in figure 3-2. All of the components used in this testbed are according to industrial standards and are universally available making it easy to verify and replicate the results of this testbed.

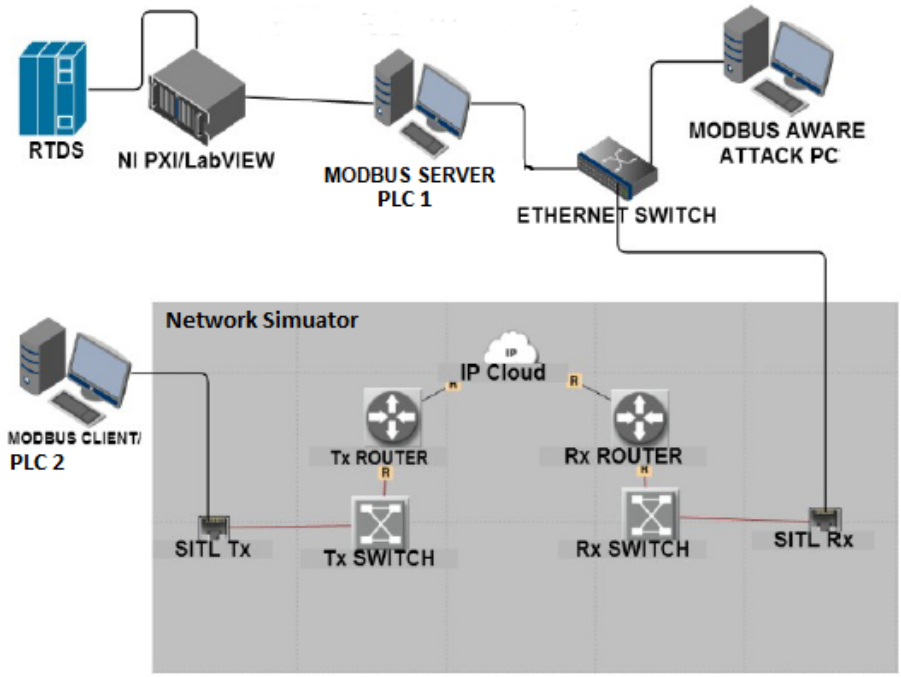

Figure 3-2: Architecture of the Texas AM testbed [9]

3-2-1 Functioning of the testbed

As shown in figure 3-2 (blue cabinet), RTDS is a real time electro-magnetic data simulator used to simulate a power plant. Real-time output data is provided as analog and digital data signals from the RTDS and is connected to a controller device. The RTDS simulates plant with input values and gives out the digital and analog signals to the PXI module on the corresponding channels. After receiving the values from RTDS i.e. the component replicating plant in the real world, the PXI module working with Labview bridges RTDS and Opnet. The PXI module in this setup works as the Modbus server i.e. PLC 1 and is programmed with LabView to communicate with Modbus protocol. This Modbus server communicates through an Ethernet switch to a Modbus client i.e. PLC 2, as shown in figure 3-2. The attack PC is also given access to the network through an Ethernet switch and it injects the attacks on the Modbus communication with Ettercap and LibModbus tools. A network simulator called Opnet simulates a TCP/IP network and enables to emulate Modbus client and server as a separate substation.

The network is analyzed with the Opnet network simulator. Modbus client generates a query containing message structure of Protocol Data Unit (PDU) and function codes as described in section 2-5, through "Tx" port. The query travels to the Modbus server and a response is received from the Modbus client back to the master.

3-2-2 Advantages and disadvantages of the testbed

Advantages:
1. The testbed integrates commercial tools such as real-time plant simulators and Opnet with open source tools in Linux to generate cyber attacks. This makes it possible to easily rebuild a similar testbed for the study of vulnerabilities in ICS.
2. The testbed used a RTDS to simulate a large, complex plant as in the real world scenario. This makes it possible for the testbed setup to accurately replicate real world ICS scenario.
3. The testbed can also simulate large network with help of network simulator, expanding the testing capabilities of the testbed.
4. This setup does not use any components that are manufactured in-house. Therefore, as all of the components are universally available, either open source or with a license, this approach is easily extendable in building new testbeds anywhere in the world.

Disadvantages:
1. The testbed does not use actual hardware for HMI workstation as used in the real world ICS plants.
2. As the testbed simulates a network with virtual components, this can introduce a lag in the communication producing a delay in message transmission.
3. The setup utilizes Opnet Modeller as the network simulator which has System-In-The-Loop (SITL) simulation capabilities. This product has been discontinued and taken over by another company called "Riverbed", which makes it highly priced, making this setup costly to replicate.

3-3 Testbed for industrial security at Fraunhofer institute

This Fraunhofer IT Security Laboratory for Industrial control systems (ICS) is developed for security research, and education of security personal.[10] The detailed architecture of the testbed is given in figure 3-3. All of the components used in this testbed are according to industrial standards and are universally available making it easy to verify and replicate the results of this testbed.

3-3-1 Functioning of the testbed

As shown in figure 3-3, the testbed is divided in two sections. First is the physical part where all the real-world ICS components are utilized to create an ICS network. The second part is developed inside a virtualisation environment using virtual machines hosted on a single physical machine. We first look at the physical part. A real-world process of disk drilling on a rotary encoder is used as a plant. This plant is controlled using ICS components i.e. PLC and HMI. The PLC used in this testbed is Siemens S7-300, which is commonly used in various small to mid-sized industrial processes. The encoder inputs and motor outputs are connected to the corresponding analog input and output cards on the PLC. This PLC is connected to a real-world HMI, again from Siemens and is communicating over Ethernet cable with S7 protocol, a proprietary industrial communication protocol from Siemens. Therefore, this testbed utilizes all ICS components in a typical ICS scenario to control a small process.

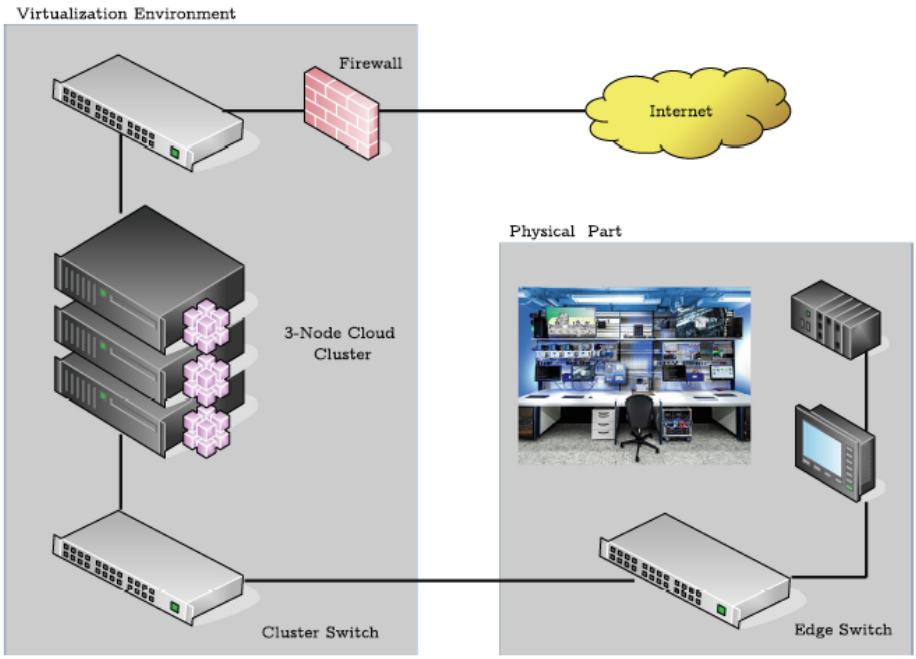

Figure 3-3: Architecture of the Fraunhofer testbed [10]

The left part of figure 3-3 shows the virtual part of the testbed. A virtualisation environment is used to host three different virtual machines on a single server. These machines are connected with the edge switch using virtual switch created with Open vSwitch. [35] The three virtual machines hosted in this virtualisation environment are collectively referred to as 3 node cloud cluster, as they are hosted on a single physical node. These three virtual machines are used to serve following three purposes : 1. Virtual machine 1 as Engineering station to program PLC and HMI. 2. Virtual machine 2 as attack PC to develop and inject attacks. 3. Virtual machine 3 for hosting attack detection tools. A firewall is part of the virtualisation environment to protect the testbed from attacks coming from outside the institute network. This completes the testbed setup.

In this testbed, it is assumed that the attacker has already gained the access to the plant communication network and is able to inject arbitrary data on to the network. The attacker also possesses the knowledge of the controlled process which enables him or her to modify and inject messages to the PLC such that physical damage to the plant can be caused. [10] The attacker therefore intercepts HMI-PLC communication using Man-In-The-Middle (MITM) attack to make the disk holding pieces to be drilled, to rotate at high pace. This causes damage to the work pieces due to random and inaccurate drilling actions. In this way, the testbed successfully generates and injects MITM attack on the ICS network.

3-3-2 Advantages and disadvantages of the testbed

Advantages:
1. The testbed used all real-world components in a typical ICS scenario to build the network. This allows the testbed to replicate the real-world ICS scenario closely enough to produce

accurate results of attack injections.
2. As the attack PC is hosted as a virtual machine, it gives more flexibility for developing attack scenarios and does not limit the attacker to be physically present at the site of plant to launch an attack.
3. No in-house developed components have been used to develop this testbed. As commonly used and globally available components are used in the testbed, this approach can be used to recreate the test results with new testbed.

Disadvantages:
1. The testbed used real industrial process as the control plant. This reduced the flexibility of the testbed to test impacts of cyber attacks on different ICS scenarios.
2. In this testbed, only a single attack i.e. basic MITM attack has been tested. This leave room for development to generate other cyber attacks and test them.
3. Use of a virtualization environment limits the possibilities of creating a large virtual network, as compared to a network simulator such as Opnet.

3-4 Supervisory Control and Data Acquisition (SCADA) cyber security testbed using Real-time Immersive Network Simulation Environment (RINSE)

This testbed is developed to simulate and analyse cyber attacks in a power generation plant. [11] Main components of this testbed and their role is described ahead. Figure 3-4 shows the main components of the testbed along with its architecture.

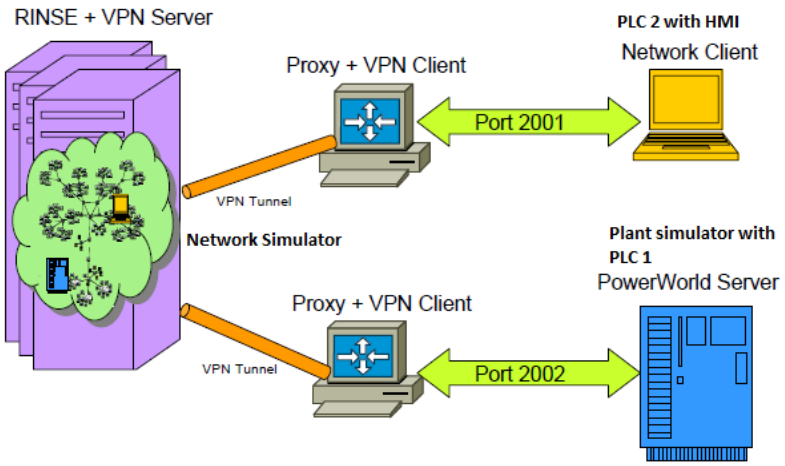

Figure 3-4: Architecture of the RINSE testbed [11]

3-4-1 Functioning of the testbed

Figure 3-4 shows the architecture of the testbed. The data flow and working can be explained as follows. A query is generated by the network client as an HMI does in the real ICS network.

The query packets are sent via a proxy server on port 2001 visualized by upper green link in figure 3-4. These packets are sent to the RINSE simulator [1] node through a Virtual Private Network (VPN) tunnel. RINSE network simulator then processes these queries and pass them to a virtual node corresponding to the power world server. The query is transmitted to the server via a proxy server where the plant is simulated. The response from this server to the Modbus client transmits through the same way. The port numbers here are flexible in the sense that they can be changed to any other port numbers but they must be different from each other. The testbed simulates Distributed Denial of Service (DDoS) attack on this network. The attack is injected using the a virtual node (virtual PC) in the network simulated with RINSE. The attack is injected using a simple command specifying attacker and victim server as an arbitrary server in the simulated network. Effect of shielding against such an attack is also studied with the help of this testbed [11].

3-4-2 Advantages and disadvantages of the testbed

Advantage:
1. The testbed combines multiple components of the testbed such as attacker and network simulator together in the RINSE architecture,reducing the hardware requirement of building the testbed.

Disadvantages:
1. The main disadvantage of this approach is that the testbed has no physical ICS components as PLCs and neither the plant components of sensors and actuators. This makes the testbed less suitable to analyse vulnerabilities in the real world ICS plants.
2. The network simulator used in the testbed i.e. RINSE is an in-house developed component. This is not available for use to anyone outside the university making it very difficult to replicate this testbed for further research.

3-5 TASSCS - Testbed for Analyzing Security of SCADA Control Systems

This is a testbed built at the University of Arizona for generating and analyzing cyber attacks on industrial SCADA system which communicates via Modbus TCP/IP protocol [12]. The approach for building this testbed for implementing cyber attacks on ICS is explained ahead.

The testbed uses an HMI as Modbus client to receive data and send commands to the virtual PLC controller, which acts as a Modbus server. Figure 3-5 shows the architecture of the testbed with major components.

3-5-1 Functioning of the testbed

The PowerWorld simulation tool works as a virtual plant in this testbed as shown in figure 3-5. It is used to simulate an electric grid which gives input values from the sensors and

[1] *The Real-time Immersive Network Simulation Environment for Network Security Exercises (RINSE) is a tool for realistic emulation of large networks as well as network transactions, attacks, and defenses [11]*

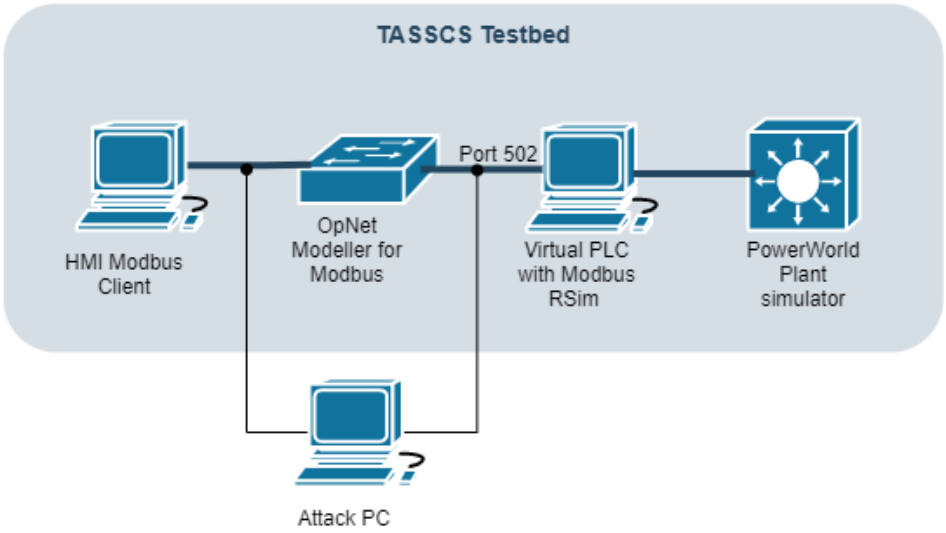

Figure 3-5: Architecture of the TASSCS testbed [12]

output values to the actuators. PowerWorld is also used to visualise and analyse the impact of cyber attacks, by observing the changed malicious values due to the attack on the HMI. The virtual PLC hosted on a PC with Modbus RSim software works as a Modbus server in this testbed. This PLC is connected to the PowerWorld plant via SimAuto tool. A large network has been simulated between HMI and PLC with OpNet network simulator, which also acts to monitor the traffic through the network. Entire communication uses Modbus TCP/IP protocol.

An attack PC is introduced in the network which has access to the network at links between the client and server. This attack PC performs a spoofing attack where sensor measurement attack is injected on the system. A MITM attack is also performed by the attacker to change the parameters of the components of a Modbus slave. Denial-of-Service (DoS) attack is launched via the NMap tool [1] by the attacker and is studied in the testbed to observe the performance of Autonomic Software Protection System (ASPS) tool [12], developed in-house by Arizona university for defending such an attack.

3-5-2 Advantages and disadvantages of the testbed

Advantages:
The approach used for building this testbed is very similar to the one in section 3-1.The main difference here is that this testbed uses a virtual PLC instead of the actual hardware. 1. As this testbed uses a virtual PLC instead of real world PLC, this decreases the cost of the testbed considerably.
2. This testbed also differs from the testbed at Binghamton as it uses a simulated, complex plant as opposed to the small physical assembly of power generating motors used in the previous testbed. This makes the testbed more accurate in replicating real world ICS scenarios as compared to the testbed at Binghamton university.

[1] Website for NMap tool : https://nmap.org/book/man.html#man-description

Disadvantages:
1. While using a virtual PLC decreased the cost of the testbed, it also makes the testbed less suitable for observing attacks in real-world ICS networks as PLC which is at the core of real world ICS network is omitted from the testbed.
2. The HMI used in this testbed is an in-house developed component making it difficult to replicate this testbed for future research globally.

3-6 Comparison between various testbeds

In this chapter, we discussed four testbeds build at various universities in the world for research purposes. Below in table 3-1 we do a tabular comparison between these testbeds which will provide insightful guidance while formulating our own approach for building a testbed.

Generic name of the component → Analogous component in	**BU Testbed in section 3-1**	**Texas AM testbed in section 3-2**	**Fraunhofer testbed in section 3-3**	**RINSE testbed in section 3-4**	**TASSCS testbed in section 3-5**
Plant simulator	No plant simulator (Motor control setup)	RTDS simulator	No plant simulator (Disk drilling setup)	PowerWorld simulator	PowerWorld simulator
Modbus server (controller)	PLC (Allen Bradley)	Labview with NI PXI module	PLC (Siemens)	Virtual PLC	Virtual PLC with RSim
Modbus client(HMI)	PC hosting virtual HMI	PC hosting virtual HMI	Siemens HMI	Virtual HMI	Virtual HMI
Network simulator	No network simulator (Wireshark as network analyzer)	Opnet *(Commercial)*	Virtual Machine (OpenvSwitch) *(Commercial)*	RINSE *(In house)*	Opnet *(Commercial)*

Table 3-1: Analogous components between testbeds

In the testbeds that we discussed, two plant simulator are widely used, which are, Real Time Digital Simulator (RTDS) and PowerWorld simulator. Both of these simulators are commercially available where the user has to buy a license for its us. Both of these simulators are able to simulate large industrial processes in real time. However, the PowerWorld simulator is specially designed to simulate power systems and electrical grid lines. It supports simulation of 60000 busses for simulating power market with simple graphical coding. As opposed to this, the RTDS offers more flexibility to simulate various plants in different domains and is not limited to power applications.

As for the Modbus servers used in different testbeds, a real world PLC has been used in two of the five testbeds while a virtual PLCs i.e. a software simulating PLC's behaviour has

been used in two of the remaining three testbeds. The Texas testbed runs with LabView software with PXI module to simulate server. Of these three options, a real world PLC makes the testbed replicate the real world ICS scenario to the greatest extent. While the virtual PLCs allow the user to program controller using IEC 61131-3, which is the standard programming language used for programming real world PLCs; it will still not incorporate physical connections of input and output signals. On the other hand, Labview will not use industrial programming language, but it offers integration with NI PXI module which allows the RTDS to be directly connected with the controller with physical Input and Output (IO) connections. Therefore, of the three approaches, which each have their own advantages and disadvantages, the approach where real world PLC is preferred as it combines the advantages of both the remaining approaches, on the software as well as hardware side.

Out of the five testbeds, only the testbed built as Fraunhofer institute uses a real world HMI, which makes it possible to setup a Modbus communication with the PLC reliably without having develop own communication functions or use third party solutions for setting up the communication. The real world HMI offers additional functions such as ruggedness, reliability and easy troubleshooting which are not offered by the software solutions. Therefore, using a real world PLC is the preferred choice while creating an ICS testbed.

As for the network simulator, two of the five testbeds use Opnet network simulator while one testbed uses the in-house developed component called RINSE for this purpose. Opnet network simulator has SITL simulation capabilities. However, this product has been discontinued and taken over by another company called "Riverbed", which makes it highly priced, making this setup costly to replicate. Otherwise, the user has to find out a comparable alternative for Opnet modeller, however, none of the open source network simulators offer SITL capability. The Fraunhofer institute uses a novel approach of using a virtualization platform instead of network simulator. Using a virtualisation platform, a virtual switch is created, making it possible for the attack PC to simulate its presence at a different geographical location than the testbed and still inject attacks. This approach is also therefore preferred while developing a testbed.

As we have looked at various approaches for building a testbed in this section, we discuss the advantages in each of these testbeds that we will try to implement into our testbed. At the same time we also summarize the disadvantages in these testbeds which we will try to avoid while building testbed at NERD Lab. This comparison is given in the following table 3-2.

With this detailed analysis of various approaches of existing testbeds, some very important observations can be made for formulating an approach to build a new testbed at TU Delft. The new testbed should try to incorporate the advantages of these existing testbeds while omitting the disadvantages discussed above. These key factors are given ahead.

1. The testbed at TU Delft should ideally try to avoid the use of any in-house developed components. Use of in-house developed components such as virtual HMIs, which are not made open source for use to fellow researchers makes the testbed unsuitable to replicate in other places worldwide. This also makes it difficult to verify the results of the testbed. Therefore, the testbed at TU Delft will try to use all components which are universally available, avoiding the use of in-house developed components.

2. The testbed at TU Delft should aim to replicate the real world ICS scenario. From the discussion about various approaches of testbeds in section 3-6, it is clear that using

Sr. No.	Name of the testbed	Advantages	Disadvantages
1.	BU Testbed in section 3-1	Simplified, low cost approach by taking a real control setup instead of the large plant network.	Unsuitable to replicate attacks on large scale control plants for which ICS is ideally used in the real world. In house components used.
2.	Texas AM testbed in section 3-2	Commercially available tools are used to simulate complex plant making it replicate the real world scenario.	Does not use real world HMI workstation.
3.	Fraunhofer testbed in section 3-3	Uses all real world ICS components for accuracy and provide virtualisation environment as network simulator alternative.	Used small scale hardware as a plant limiting flexibility of testing impacts of attacks.
4.	RINSE testbed in section 3-4	Combines multiple components together to reduce cost of the testbed.	No physical components neither PLC nor for plant hardware. In house network simulator is used making the testbed hard to replicate.
5.	TASSCS testbed in section 3-5	Use of virtual PLC reduces cost of the testbed.	No use of real world PLC Inhouse HMI is used.

Table 3-2: Advantages and disadvantages of various testbed approaches

largest possible number of real world ICS components will help in achieving this aim. This includes the the use of actual PLC and HMIs. This also means that the testbed should use a real time plant simulator for replicating a real world, complex plant making the testbed mimic the real world ICS scenario.

This concludes the discussion on various approaches of existing testbeds in this chapter. The next chapter will take forward the idea of building a new testbed Networked Embedded Robotics in Delft (NERD) Lab in the TU Delft following the approach discussed above.

Chapter 4

Design of ICS testbed at Delft University of Technology

The last chapter analysed various existing ICS testbeds and compared their relative advantages and disadvantages. Now, this chapter will utilise this knowledge to develop an ICS testbed at Delft University of Technology (TU Delft), combining all the advantages in the existing testbeds, while omitting the disadvantages in them.

To achieve this, in section 1, the functional requirements for the testbed at TU Delft have been formulated. Section 2 formulates the functional requirements for each individual component in the testbed at TU Delft. By analysing this information, a new approach has been developed and a setup involved in the testbed at TU Delft is designed in section 3. The architecture of this testbed is discussed and its detailed working is explained in section 4 of this chapter. Section 5 gives the design for the entire testbed comprising of five test setups. Section 6 goes into depths of the selection process followed for the components of this testbed. Section 7 gives the deeper electrical cabling diagram in order to build the lab. In the last section, the dSPACE plant simulator is programmed to run the plant simulation.

4-1 Functional requirements for the testbed at Delft University of Technology

1. The testbed should replicate real-world scenarios of ICS in plant control. This will be a key feature in our testbed, allowing us to explore practical vulnerabilities of the ICS setup as in the real world.

2. The testbed should be able to realise attacks on real world ICS which are typically used for control applications of complex plants.

3. The testbed must have ICS components such as PLC, HMI which are able to communicate with each other via Modbus TCP. In addition to this, the PLC must be able to

communicate with other PLCs via one more industrial control protocol such as EtherCAT, Profibus, etc. to ensure future extension of capabilities.

4. The testbed should include a component which is able to capture Modbus data packets transferred between the components and is also able to visualise the flow of packets in real time.

5. The testbed must be able to implement common cyber attacks as discussed in chapter 2.

6. In addition to these requirements, the testbed should aim to keep in-house developed components to a minimum if not zero. The testbed should aim to use all components which are either commercial and universally available or open source and free for use. This will ensure the replicability of the testbed while ensuring good customer support for the real world devices. This will also make the testbed mimic the real world ICS scenario more closely.

7. The testbed should include multiple setups with ICS components so that they can be operated simultaneously either for different applications or in connection with each other for a single application.

4-2 Functional requirements for individual components in Delft University of Technology testbed

1. Plant simulator - To replicate a large scale industrial plant, the testbed should include a component eg. a plant simulator. The plant simulator should be able to simulate a large, complex plant based on given analog input from the PLC and also should be able to send analog output signals to the PLC for control purposes. The simulator must simulate the process in real time and should be re-programmable to simulate various industrial applications.

2. Programmable Logic Controller (PLC) - The PLC used in the TU Delft testbed must be able to mimic the real world ICS scenarios accurately and should be universally available to increase replicability of the testbed. Therefore, the PLC must be provided by one of the major manufacturers in the control industry. The PLC must have the capability to receive and send analog input and output data. The PLC should have capabilities to communicate with other Modbus devices such as other PLCs, HMIs, etc. Further, the PLC should support communication with other devices through at least one more industrial communication protocol, for example, Profinet, EtherCAT, etc. This will make our testbed ready to extend its capabilities for generation and testing of cyber attacks on ICS with other industrial communication protocol.

3. HMI - An HMI used in TU Delft testbed should be able to replicate the real world ICS scenario. To achieve this, the HMI should provide reliability of Modbus communication with proprietary licence for Modbus protocol. The HMI should to re-programmable to visualise different processes accurately. The software of HMI should have the capability to be installed on a standard PC, to make it cost effective, without the need of any other advanced hardware.

4. Attack PC - The testbed must be able to implement common cyber attacks as discussed in chapter 2. An attack PC should be running on an Operating System (OS) which supports easy installation of appropriate software tools to generate cyber attacks. The attack PC should have access to the same ICS network as that of the PLC and HMI, to inject such attacks. The attack PC should have capability to simulate a scenario where the attacker can be located at different geographical location and still make him or her able to inject attacks on the testbed.

5. Network simulation - A network tool should be selected such that it is able to capture Modbus data packets transferred between the components and is also able to visualise the flow of packets in real time.

6. Network switch - The testbed should include an Ethernet switch with enough free ports to connect all the ICS components. The Ethernet switch should allow free data flow between all components of the ICS testbed, but at the same time, it should not allow any other unauthorised device to be connected to the ICS network.

4-3 Architecture of the testbed : Single setup

In figure 4-1, we show the architecture of a single setup of the proposed testbed at TU Delft NERD lab. The whole testbed consists of 5 such setups to be built at the NERD lab. This entire setup is explained in section 4-5. The main components of this testbed along with their functions are explained below.

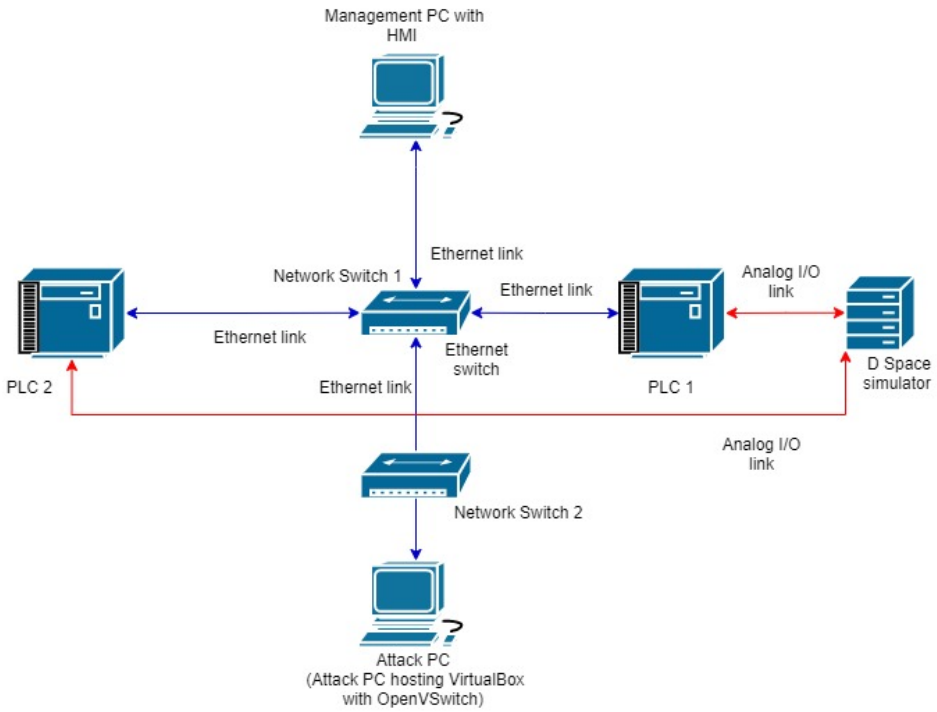

Figure 4-1: NERD Lab testbed architecture : Single setup

- dSPACE Simulator - This component represents a virtual industrial control plant that will be controlled by ICS controller i.e. PLC. A large network of sensors and actuators can be simulated in this dSPACE simulator in real time. This satisfies our criterion for the plant simulator. The dSPACE simulator will simulate and pass the values of sensors and actuators to the analog input card connected to the servers PLC 1 and PLC 2.

- PLC 2 (Modbus server 1) - A real-world PLC as PLC 2. The functions of this PLC are listed as follows.

  1. To read analog sensor data from the dSPACE simulator in the analog input card connected to it and to write the analog output control values (actuator values) to the dSPACE simulator via the analog output card connected to it.

  2. To receive the reference values sent by the HMI and to pass the plant data in real time for visualisation to the HMI.

  3. To run the control logic programmed on it with the values received from the dSPACE simulator as well as the HMI to achieve the desired output.

  This PLC, which is the first controller of testbed, plays the role of Modbus Server, responding to the queries generated by HMI in our network. In the future, the functionalities of this PLC 2 can also be extended for watermarking and fault diagnosis.

  We will select a PLC from one of the major providers in the industry such as Siemens or Beckhoff. This will make the TU Delft testbed mimic the real world ICS scenarios more accurately, while satisfying every functional requirement for PLC, as defined earlier.

- Ethernet switch 1 - A 48 port Ethernet switch will be used to connect the Modbus devices with each other.

- PLC 1 (Modbus server 2) - Another PLC with the same structure as described above will be used as our second controller, controlling another process on the dSPACE simulator. This PLC 1 will also act as the controller of the plant. This PLC will generate queries to receive data of sensor states from the dSPACE simulator, process the data and decide on the necessary control input and will write the output data to dSPACE simulator to change the actuator values as per the control strategy.

- Management Personal Computer (PC) with HMI (Modbus client) - The management PC will serve two purposes. First, it will be used to install and manage all the softwares required for programming PLC and HMI. Through this PC, the user can make changes to the programs written on the PLC. Second, this PC will be running HMI software, which will act as Modbus client visualising data from the PLCs. The management PC running an HMI will be connected to the network and will be used to visualise real time data of the process and send control set points to PLC1 and PLC2.

- Attack PC - A powerful PC will be used to host attack PC and is connected to the Ethernet switch 1 giving it access to the ICS testbed network to perform cyber attacks. The attack PC will be running on Linux OS and will have tools to generate and inject attacks on the Modbus network. The two major tools will be Hping3 and Ettercap. The Hping3 tool will help us generate spurious messages of Modbus protocol at a high speed to launch DoS attack on the network. Ettercap is another tool which will help injecting MITM attacks by capturing the data packets being transmitted between Modbus client

and server. This PC can host virtual machine on it and connect them to the network using a virtual switch called Open vSwitch. [35] This virtual switch is illustrated by the Network switch 2 in the figure 4-1. Using a virtual switch overcomes the limitation of the attacker being present at the plant site to launch the attack by simulating a scenario where attacker is connected to the network remotely.

4-4 Functioning of the testbed : Single setup

The testbed will have a dSPACE simulator which will act as a virtual plant simulating the sensor, actuator data. This data will be transmitted to the PLC 1 and PLC 2 via the analog input cards on the same chassis as the Central Processing Unit (CPU) of the PLC. PLC 1 and PLC 2 will act as the controllers of the plant, receiving and forwarding analog input and output signals from the HMI and taking the necessary control action on the plant. These PLCs will control two individual processes, as parts of a single complicated process that are performed after one another. In other words, the PLCs will control a batch process.

This data will be passed on to the HMI hosted on the management PC through a Modbus connection established via an Ethernet switch 1. The PLC 1 and PLC 2 will send this data to the HMI which will act as the Modbus client. The HMI will visualise real time data on the screen and send commands to the server PLCs. An attack PC will be connected to the network via Ethernet switch. This PC will host virtual machine as explained in here 4-3. This virtual PC will act as the attack PC connected to the network with OpenV switch and will be used to inject cyber attacks on the network. In future, the functionalities of PLC 2 can be extended in future to attack diagnosis or watermarking.

4-5 Architecture and working of the entire testbed

In the previous sections, the role and specifications of each component of the testbed was described in detail, to further elaborate the working of the single setup of the testbed. In this section, we will discuss how these individual setups come together to form the entire testbed at TU Delft NERD Lab.

Figure 4-2 shows the entire testbed at TU Delft NERD lab. This testbed consists of five of the test setups as described in section 4-3. The entire testbed, therefore contains five pairs of ICSs as follows:

- ICS Setup 1 : PLC 1 and PLC 2 with Management PC 1
- ICS Setup 2 : PLC 3 and PLC 4 with Management PC 2
- ICS Setup 3 : PLC 5 and PLC 6 with Management PC 3
- ICS Setup 4 : PLC 7 and PLC 8 with Management PC 4
- ICS Setup 5 : PLC 9 and PLC 10 with Management PC 5

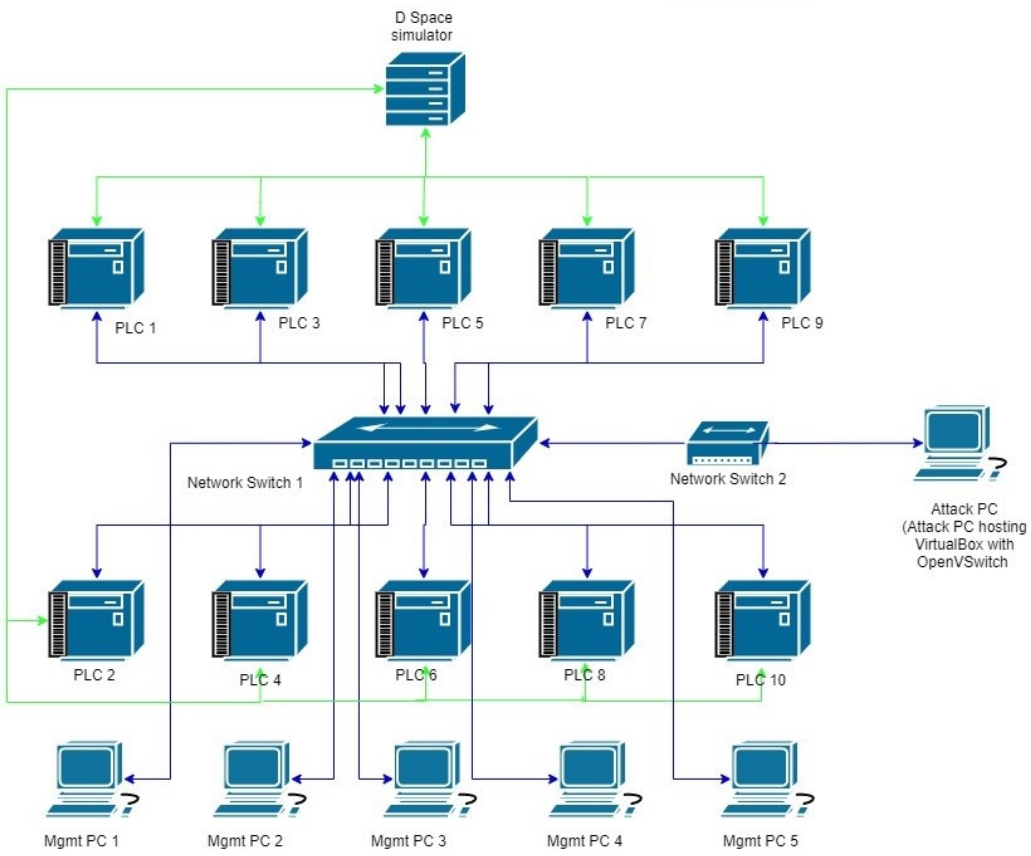

Figure 4-2: NERD Lab testbed architecture : Entire testbed

The dSPACE simulator, which plays the role of plant, is connected to all the 10 PLCs. The analog IO (Input/Output) cards on the PLC are connected to communicate these analog signals with the dSPACE simulator. The structure of each PLC is therefore exactly similar. Each pair has 1 management PC, shown by the label "Mgmt PC" in figure 4-2. This management PC is used to program PLCs in each setup and has the necessary softwares installed on it. This makes the reprogramming of the PLCs easier. Each management PC also runs the HMI for its own setup. All these equipments are connected via a 48 port Ethernet network switch and are communicating with each other on Modbus TCP protocol.

This entire setup has a common attack PC connected via a virtual switch i.e. Network switch 2, created with OpenV switch using Virtual Box and connected to Network switch 1. This attack PC runs on Kali Linux OS and has the necessary softwares installed to launch cyber attacks on this network of ICS testbed. With the use of this virtual switch, the limitation of attack PC being continuously present at the site is tackled, as in this case, attack PC is running as a virtual machine, connected to a virtual switch. Therefore, this simulates a scenario where an attack can be injected by the attacker remotely. In this ways, the ICS testbed at TU Delft is designed combining 5 individual setups, able to work simultaneously to mimic a real world ICS scenario. In future, each of these setups can also used with different dSPACE simulators to simulate different processes simultaneously, thus satisfying our functional requirements.

4-6 Selection of the components for the testbed

After finalising the approach of the testbed, a detailed market analysis for the ICS components was performed i.e. for PLC, Network simulator, HMI and a virtual machine software (to host attack PCs on a single hardware PC). The selection criteria used to compare these options and the reasoning for selecting each component is given in this section. The functional requirements for the testbed as mentioned in section 4-1 were reviewed as reference while shortlisting the ICS components. Only the options that satisfy these basic functional requirements of the testbed were compared further. The criteria used for this comparison were then features of the component, power requirements of the components, communication protocols supported by the component, open or closed source licensing of the product, cost of the product, etc.
Following such detailed comparison, a probable list of items from two providers was created. These two providers are Beckhoff and Siemens. Then quotations were requested from both these providers, thoroughly explaining them the requirements and application case of the testbed. This process was done meticulously as these ICS devices cost in thousands of euros and hence needed very careful assessment before ordering. After exchanging many versions of the quotations, a final quotation was received and approved by both sided. These final selected components along with their relevant specifications are given in the following :

1. Plant simulator(dSPACE simulator in section 4-3) - dSPACE simulator with model number 1401/1514 is used in the testbed. This simulator is selected as it simulates the process in real time, allowing to send analog input and output values to the PLC and HMI, which physical connections through its connector. Additionally, the dSPACE simulator is our preferred choice as it is able to simulate multiple different scenarios in various industrial applications and is not limited to just one application.

2. PLC - According to the functional requirements laid out by us for the testbed in section 4-2, the PLC should be supplied by one of the major providers of PLC that is used commonly worldwide. The PLC must have the ability of connecting analog I/O cards to interface with the simulator. The PLC should be able to communicate with other PLCs via Modbus TCP and one other industrial network protocol such as EtherCAT, Profibus, etc. PLC should cost around 1K euros.
   These requirements are best satisfied by **Beckhoff C6030** ultra compact industrial PC and is therefore selected for the testbed. It has mounting plate on the rear wall for the free alignment of the connecting area. This feature, combined with its small size satisfies the mounting requirements submitted by the TU Delft technicians at the NERD lab, occupying less space for mounting. It has 4 Ethernet ports on the CPU itself, which saves the need for a separate card for Ethernet connections. It runs on 24 V power supply. It has high processing power with Intel processor with windows 10 OS, which is sufficient for our test purposes at the NERD lab. Therefore, this CPU model is selected for the the ICS testbed at TU Delft.
   The CPU is combined with various cards which are powered internally to perform various function. These are the accessories of the PLC. For each PLC, following cards are ordered, which fit on a single chassis and their functions are given in list below.

   – EtherCAT coupler (EK1100) : This card is used as a communication bridge between the CPU and other IO (Input/Output) cards. This card powers the IO cards

and enables the CPU of the PLC to communicate with them.

- Digital input card (EL1008) : This is an 8 channel digital input card which allows the digital sensor values from the field to be connected to the PLC.
- Digital output card (EL2008) : This is an 8 channel digital output card which allows the digital actuator values from the PLC to be written on the actuator plant side.
- Analog input card (EL3004) : This is a four channel analog input card which allows the analog sensor values to be communicated to the CPU of the PLC.
- Analog output card (EL4004) : This is a four channel analog output card which allows the analog actuator values to be written on the actuator on the plant side.

In addition to this, licences for the PLC software i.e. TwinCAT is bought to program and operate this Industrial PC as PLC. This completes the setup of PLC installed in the ICS testbed at TU Delft.

2. Management PC - According to the functional requirement in list 4-1, The PC should be powerful enough to install and manage softwares for programming of PLCs and HMIs. For this purposes, Dell Optiplex 7040 PC with high RAM , high memory (at least 1 TB) and latest version of Windows or Linux operating system is finalised. The specifications of this PC are sufficient for the functioning of the attack PC in the testbed. Additionally, the HMI software from Beckhoff is also bought along with the Modbus TCP license to run the HMI with Modbus TCP on the management PC of the setup.
3. Ethernet switch - The network switch is used to connect all PLCs, management PCs and attack PC on the same network. For this purpose, a 48 port Ethernet switch has been selected and set up in the NERD lab, which satisfies the purpose.

In this way, after a rigorous scanning and comparison procedure, to ensure selection of best components for the testbed, the selection procedure was completed. To given even more clarity, these components satisfy the functional requirements of the testbed discussed in the list 4-1 as follows :

1. All these ICS are exactly the same equipments that are used in real world control scenarios in the industry, this makes the TU Delft testbed mimic the real world ICS scenario accurately, satisfying the functional requirement number 1.
2. The testbed used dSPACE simulator, which is able to simulate a large, complex industrial process. This satisfies second functional requirement of having real world process for the control purpose.
3. The PLC supports Modbus TCP and is can also be extended for the use with EtherCAT industrial communication protocol. This satisfies our function requirement number 1 given in list 4-1, of PLC being able to support multiple communication protocols.
4. The attack PC of the testbed has various network analysers such as Wireshark installed on it, enabling it to capture and visualise Modbus TCP data packets going through the network.

5. The attack PC is installed with Kali Linux, which is a specialised version of Linux for testing cyber attacks. This ensures that common types of cyber attacks can be launched with this testbed on the ICS, satisfying the functional requirement number 5 as given in the list 4-1.

6. The requirement for replicability is satisfied as no in house developed components have been used to construct this testbed and all the softwares (except the softwares bought from Beckhoff for the PLC) are open source and free for use. This satisfies the last functional requirement for the testbed as using these components ensures good replicability of the testbed.

7. The last functional requirement is satisfied as five setups of the ICS components have been designed to operate simultaneously in the testbed at Delft University of Technology. These setups can be used for different applications or in connection with each other for a single or multiple application. This testbed can, for this purpose, be also extended to connect multiple dSPACE simulators to various setups.

This sections therefore presented a through discussion about selection procedure followed for designing the testbed and gives clear reasoning at every step to help reader understand the choices made by us in this design process. The next section further gives the electrical cabling diagrams for these components to create the testbed.

4-7 Electrical cabling diagram for the testbed

In the above section, the detailed architecture of ICS testbed at TU Delft NERD lab is discussed along with the process for selection of components. Once the components for the testbed are finalised, it is important to ensure the correct electrical wiring for all the components. This is because each ICS equipment has different power requirements and it not handy to change the wiring scheme of these devices frequently. Therefore, it was very important at this stage of the thesis to construct a detailed electrical wiring diagram for the testbed, so that it becomes easier for the laboratory technicians at TU Delft to plan accordingly.

Figure 4-3 gives a layout for electrical connection diagram for a single ICS setup of the testbed, consisting of PLC1, PLC 2 and Management PC 1. This same electrical layout is repeated for each ICS setup in the testbed as described in the list of pairs in section 4-5.

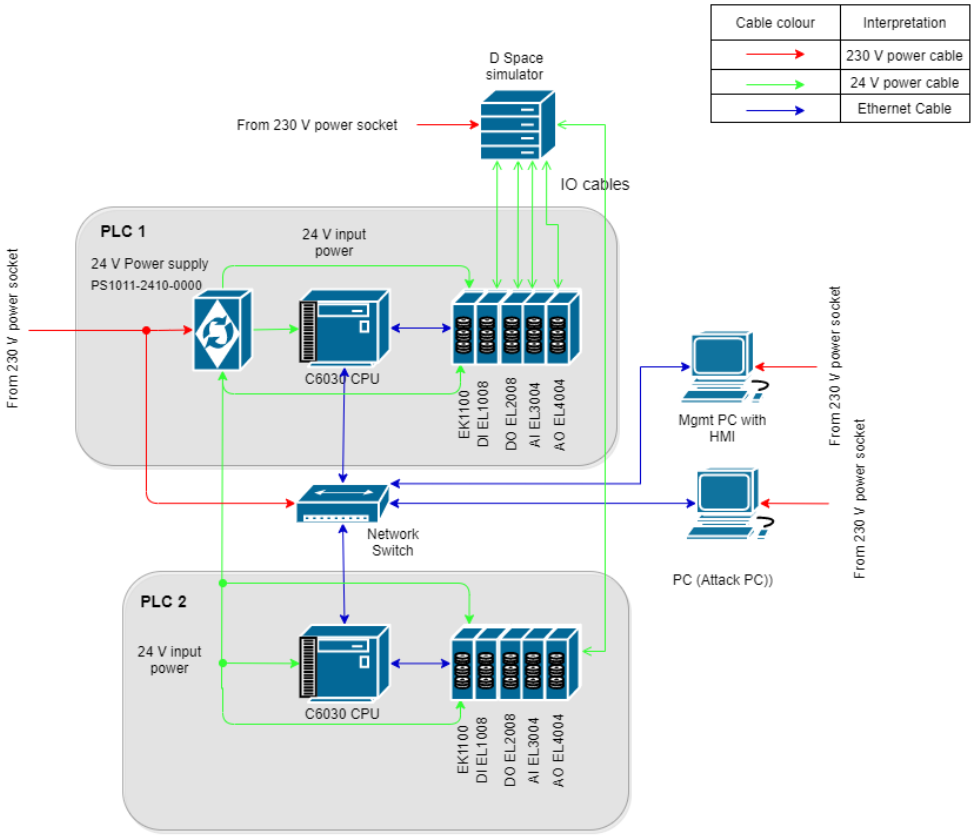

Figure 4-3: NERD Lab testbed electrical diagram

As shown in figure 4-3, each PLC consists of following components :

- CPU : C6030
- EtherCAT coupler : EK1100
- Digital input card : EL1008
- Digital output card : EL2008
- Analog input card : EL3004
- Analog output card : EL4004

The details of these equipments have already been discussed in section 4-6. In this section, the focus of description is therefore on the electrical cabling aspect for these components.

Each of the above mentioned components is mounted on a single chassis from Beckhoff. The CPU as well as all of the cards have mounting slots on their side planes, through which the power and communication signals are passed on.

The CPU of this PLC i.e. C6030 runs on 24 V power supply. Therefore, to provide this input power, a 24 V power supply is used in the setup as shown in figure 4-3. This power supply draws a current from input 230 V from the standard switchboard connection in the

lab and supplies a constant output of 24 V and 10 A current. As these voltage and current requirements are more than double of that of the PLC, one power supply per setup of 2 PLCs is utilised in this design of the testbed.

This 24 V power supply is then connected to the input terminals of the CPU as well as the EtherCAT coupler. These are the only 2 external connections that have to be made for the power supply of the PLC. The remaining cards of the PLC i.e. analog input/output cards and digital input/output cards are powered internally thought the bus connectors on the side panes of each of the card on the chassis. This gives a huge advantage in designing as it reduced the requirement for external cabling, making it simpler to implement, while also reducing the cabling cost. This also eliminates the risk of having loose end of wires and short circuits.

On the other hand, other devices of the ICS testbed, namely, dSPACE simulator, management PC, attack PC and network switch work directly on 230 V power supply and are hence connected directly to the standard wall socket in the lab. The 230 V connections in this diagram are shown in colour red while the 24 V connections are shown with colour green. All Ethernet cables are shown with colour Blue.

Once this electrical connections are complete, the connections for data communication are made. For that, the analog and digital Input and Output (IO) cards are connected with standard 24 V cables with to the respective ports of the dSPACE simulator. The actual signal values coming in from the plant and going into the plant from the PLC are communicated via theses cables. The remaining communication between PLCs, management PC and attack PC is done using the Ethernet cable connected through the network switch.

In this way, a detailed layout plan for electrical cabling for the ICS testbed was discussed in this section.

4-8 Setting up the dSPACE simulator for testbed

After discussing this detailed design for testbed, the dSPACE simulator at TU Delft NERD lab is set up in this section to complete the testbed. As shown in figure 4-4, the dSPACE MicroAutobox II is a real time plant simulator which simulates any given model and produces the analog and digital input and output signals through its pin outs. In figure 4-4, the left side figure shows the back place of the MicroAutobox where the input power connection and host PC connections are provided. On the front side, pin outs are given to which connections can be made to extract real time analog and digital signals from the plant. Therefore, in the testbed, the corresponding pins from this connector will be connected to the corresponding channels on the analog IO cards of the PLCs.

The MicroAutoBox II works with 230 V input power supply and has Ethernet connectors to be connected to the host PC to program the device. It runs on the IBM PPC processor with 16 MB primary memory for storage of programs and data files. It has total 32 channels which can be used to connect analog inputs and 8 channels which can be used as analog outputs.

For the ICS testbed at NERD lab, this MicroAutoBox II is working as a plant, as described earlier. Therefore, a typical industrial process which is, a two-tank level control process has been modelled in MicroAutoBox II for this case. This process finds its application in process control industry such chemical, food and oil and gas industry. This two tank level control plant

Figure 4-4: dSPACE MicroAutoBox II [13]

is shown in figure 4-5. The process has 2 inputs as inflow 1 and inflow 2, going into the two tanks and 2 outputs, which are the levels in the two tanks i.e. level 1 and level 2. Therefore, the input flows are mapped on two of the analog input channels on the MicroAutoBox and these signals will be controlled by the PLC. On the other hand, the level sensor outputs are mapped to the two analog output channels on the MicroAutoBox and these readings will be provided to the PLC via the analog input card on it.

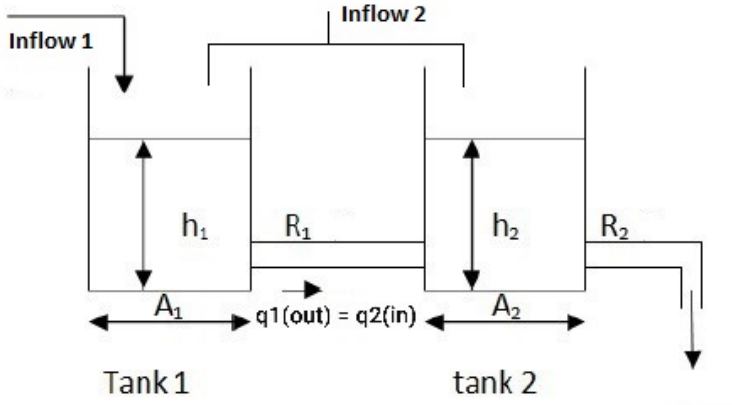

Figure 4-5: Two tank level control process

The focus of this section is to program and set up the dSPACE simulator. As this process of setting up a plant is unique and not well documented at NERD lab, this sections gives a detailed procedure to set up the dSPACE simulator, which can be used as a documentation for further work with dSPACE simulator.

To model this plant in MicroAutoBox, the plant model first is created and implemented in Simulink as a state-space model. To run this model on MicroAutoBox, it first needs to be converted into a C code. To achieve this, a library called "Real-Time Interface" (RTI) is provided by dSPACE and needs to be installed in Matlab. This library is activated when the license key for dSPACE is inserted. Figure 4-6 shows the successful installation of RTI on Matlab.

```
================================================================================
Configuring dSPACE Software for MATLAB(R) 9.5.0.1586782 (R2018b) Update 8 ...

RTI              Real-Time Interface to Simulink (RTI1401)  7.11       02-Nov-2018 okay
RTI-MP           RTI for Multiprocessor Systems             7.11       02-Nov-2018 okay
RTIGLBS          RTI Gigalink Blockset                      2.4        02-Nov-2018 okay
RTICAN           RTI CAN Blockset                           3.4.7      02-Nov-2018 okay
RTICANMM         RTI CAN MultiMessage Blockset              5.1        02-Nov-2018 okay
RTILINMM         RTI LIN MultiMessage Blockset              3.1        02-Nov-2018 okay
RTIRPCU          RTI RPCU Blockset                          2.2.3      02-Nov-2018 okay
RTIBYPASS        RTI Bypass Blockset                        3.11       02-Nov-2018 okay
RTIFRCONF        RTI FlexRay(TM) Configuration Blockset     4.2        02-Nov-2018 okay
RTIETHXCP        RTI XCP on Ethernet Blockset               1.2.10     02-Nov-2018 okay
RTIETHERNETUDP   RTI Ethernet (UDP) Blockset                1.4.3      02-Nov-2018 okay
RTIFPGA          RTI FPGA Programming Blockset              3.6        02-Nov-2018 okay
MDBS             MotionDesk Blockset                        2.5.4      02-Nov-2018 okay
RTIEMC           RTI Electric Motor Control Blockset        1.4.1      02-Nov-2018 okay
RTIETHERNET      RTI Ethernet Blockset                      1.2.3      02-Nov-2018 okay
RTISTBM          RTI Synchronized Time Base Manager Bloc... 1.1        02-Nov-2018 okay
RTIUSBFlightRec  RTI USB Flight Recorder Blockset           1.2.2      02-Nov-2018 okay
RTIWATCHDOG      RTI Watchdog Blockset                      2.1.1      02-Nov-2018 okay
DSMLCON24        dSPACE MATLAB Connection 2.4 (win64)       2.4        01-Jun-2012 okay
================================================================================

*** RTI Platform Support RTI1401 activated.
```

Figure 4-6: Real-Time Interface for MicroAutoBox successfully installed on Matlab

Once this RTI is properly installed, it is ready to be used in Simulink and contains predefined blocks to read and write analog and digital signals from the dSPACE simulator. There are various blocks such as Analog IOs, Digital IOs and can then directly be used in Simulink with the plant model that we developed previously.

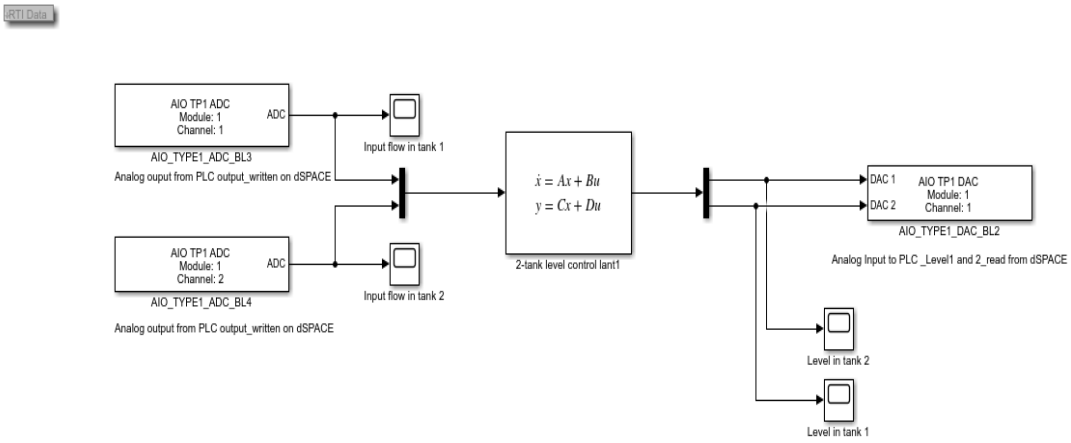

Figure 4-7: Plant model on dSPACE MicroAutoBox

Figure 4-7 shows this plant model developed in Simulink and integrated with RTI to be run on dSPACE simulator. In this figure, the output of the state space model i.e. the level 1 and level 2 in two tanks are connected to analog output channel numbers 1 and 2 respectively, using "AIO DAC" block in RTI. The "DAC" in this block represents "Digital to Analog Converter". Therefore, this block converts the digital value from the state-space output, converts it into analog value and writes it on pins 1 and 2 on the MicroAutoBox. These plant outputs are therefore connected to the analog input cards of the PLC to read the real time level values as described already. Similarly, the control inputs coming from the PLC are connected to Analog input pins 1 and 2 on the MicroAutoBox. These analog values are then read and converted into digital values using the "AIO ADC" blocks, as shown in the left side of the state space block in figure 4-7. In this way, the plant model is developed with in Simulink incorporating the RTI library to run on dSPACE simulator.

Before this model is run on MicroAutoBox, it needs to be built and converted into a ".sdf" file, which is then installed on the MicroAutoBox. Therefore, after developing this model, it is built and RTI library translates this Simulink model into the C code and builds necessary files to run this model on MicroAutoBox. While building this model, following configuration parameters are to be set to appropriate options as shown in table 4-1.

Configuration parameter	Option to be selected
Solver	Fixed step with appropriate sample size
Hardware implementation	Code generation target file should be selected as "grt.tlc"
RTI simulation options →Initial simulation state	RUN
RTI load options →Load application after build	Uncheck

Table 4-1: Configuration setting to build dSPACE model

All the files necessary for running the model on dSPACE simulator are created automatically by implementing the build command. The ".sdf" file from these files is significant to run this model on dSPACE. The C code for this is provided in Appendix B. In order to run this built model on dSPACE simulator an application called "Control Desk", developed by dSPACE is used. The application is opened and relevant model number of MicroAutoBox is selected. After that, the ".sdf" file corresponding to the plant model that we developed is selected and this downloads the model on dSPACE MicroAutoBox. Once this model runs successfully on dSPACE independently, the setup of dSPACE simulator is completed.

With the setup of dSPACE plant simulator, this chapter is concluded. This chapter presented the design for the new testbed at NERD lab in TU Delft. Further, it discussed the testbed in depth including its electrical cabling and documented the setup procedure for plant simulation on dSPACE simulator. The next chapter will describe the simulated testbed approach followed after completing the design of this testbed and setup of dSPACE plant simulator.

Chapter 5

Simulation of the testbed

The previous chapter described the design for the Industrial Control Systems (ICS) testbed for cyber attack generation at TU Delft. After completing the design process of the testbed, the procurement procedure was scheduled and the order was placed with Beckhoff. However, due to the ongoing Covid-19 crisis and time delays in the delivery of the components, we decided to present a simulated version of this testbed. The primary goal while creating the simulated testbed is that it should accurately replicate the actual testbed designed for Delft University of Technology (TU Delft), which, in turn, replicates the behavior of ICS controlling an industrial plant. This goal is achieved by designing a simulated version that mimics the real world ICS scenario for controlling a two tank level control process, which is a common application in industrial sector. By particularly selecting open source softwares for cyber attack generation, it is ensured that the same attack methodology can be directly utilized to implement cyber attacks on the actual testbed, once created at the TU Delft laboratory.

This chapter first describes the simulated testbed with its functional details. Further in section two and three of this chapter, the Man-In-The-Middle (MITM) and Denial-of-Service (DoS) attacks are injected on the testbed, following the techniques described in section 2-3-2 and section 2-3-3 respectively. Further, the results of these cyber attacks are discussed in detail giving the reader a clear picture of the hazards that can be created with these attacks.

5-1 Simulated testbed

Figure 5-1 shows the architecture of the simulated testbed used for implementing and testing cyber attacks. The testbed consists of two hardware PCs, namely PC1 and PC2, on which respective softwares are run to create the ICS components such as Programmable Logic Controller (PLC), Human Machine Interface (HMI) and attack Personal Computer (PC). A third PC is run on hardware PC1, with the help of Virtual Box, which is a virtualization platform. The next two subsections will provide the details of the hardware and software components of this stimulated testbed respectively.

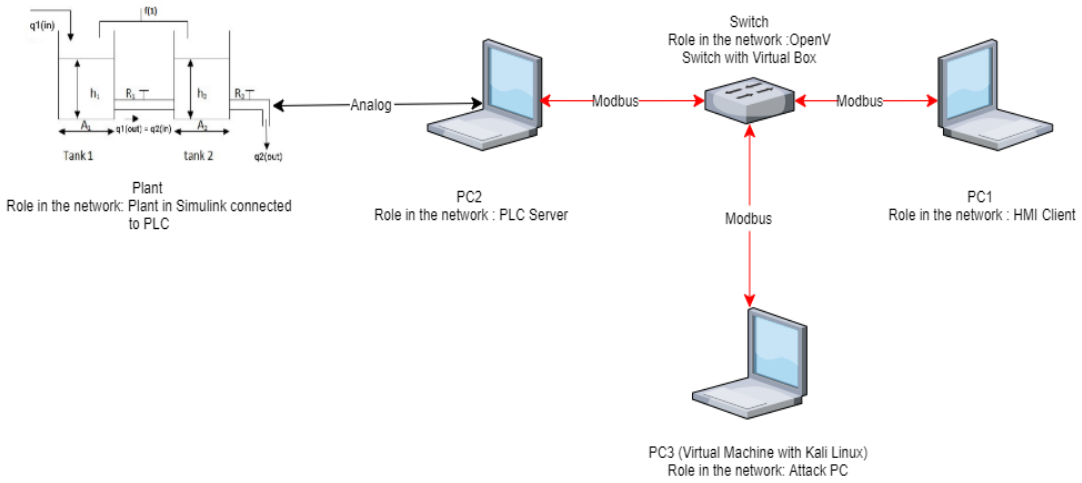

Figure 5-1: Architecture of simulated testbed

5-1-1 Hardware configuration of simulated testbed

The details of the hardware parts of the testbed and the roles that these PCs perform in correlation with the proposed testbed at TU Delft are described ahead.

- PC1 - PC1 is the hardware PC that plays the role of HMI in the testbed. To correlate with the actual testbed, this PC1 corresponds to the management PC on which HMI is running. Therefore, PC1 acts as the **Modbus Client** in the simulated testbed, similar to the HMI acting as the Modbus Client in the actual testbed. To reiterate, HMI is the screen that displays the key real time values from the plant to the operator and allows the operator to input the reference values to be sent to the controller. This is where the plant operator is located on a plant site.
- PC2 - PC2 is the hardware PC that plays the role of PLC and plant simulator in the testbed. To correlate this with the actual testbed, this PC2 corresponds to the PLC in the actual testbed. Therefore, PC2 acts as the **Modbus Server** in the simulated testbed, same as that in the actual testbed. On this PC2, a model of level control in two interconnected tanks is created with Simulink. This makes the simulated industrial process for our testbed.
- Attack PC - A virtual PC running on hardware PC 1 is used as an attack PC and is added to the setup to generate and launch attacks into the network of PLC and HMI. To reduce the hardware, this attack PC has been implemented as a Virtual PC using a Virtualization software. A virtualization platform called "VirtualBox" is used to realize this.

5-1-2 Software configuration of simulated testbed

Now that the hardware of this simulated PC is discussed, the software part of the this simulated testbed is discussed below. We use various softwares installed on various PCs to simulate

plant, PLC (Modbus server), HMI (Modbus client) and attack tools to generate cyber attacks.

HMI simulation on PC 1

On PC1, an HMI screen is created using the "App-designer" application with MATLAB. Figure 5-2 shows this HMI screen running on PC1. Here, two scales with heights 0 to 10 feet are present, through which the reference heights of the tanks are input by the operator. Matlab functions have been developed to pass the values from this HMI to the PLC on PC2 with Modbus TCP protocol. A Modbus communication toolbox provided by Matlab have been used to establish a successful Modbus communication between HMI and the PLC. On the right-hand side, two gauges are given, which display the real time values of heights in the two tanks. A master switch is given to turn the Modbus communication ON and OFF between the HMI and the PLC. This status of connection is shown with the indicator on the HMI. Additionally, an indicator is provided for each tank as an alarm, which is triggered if the liquid in the tank overflows.

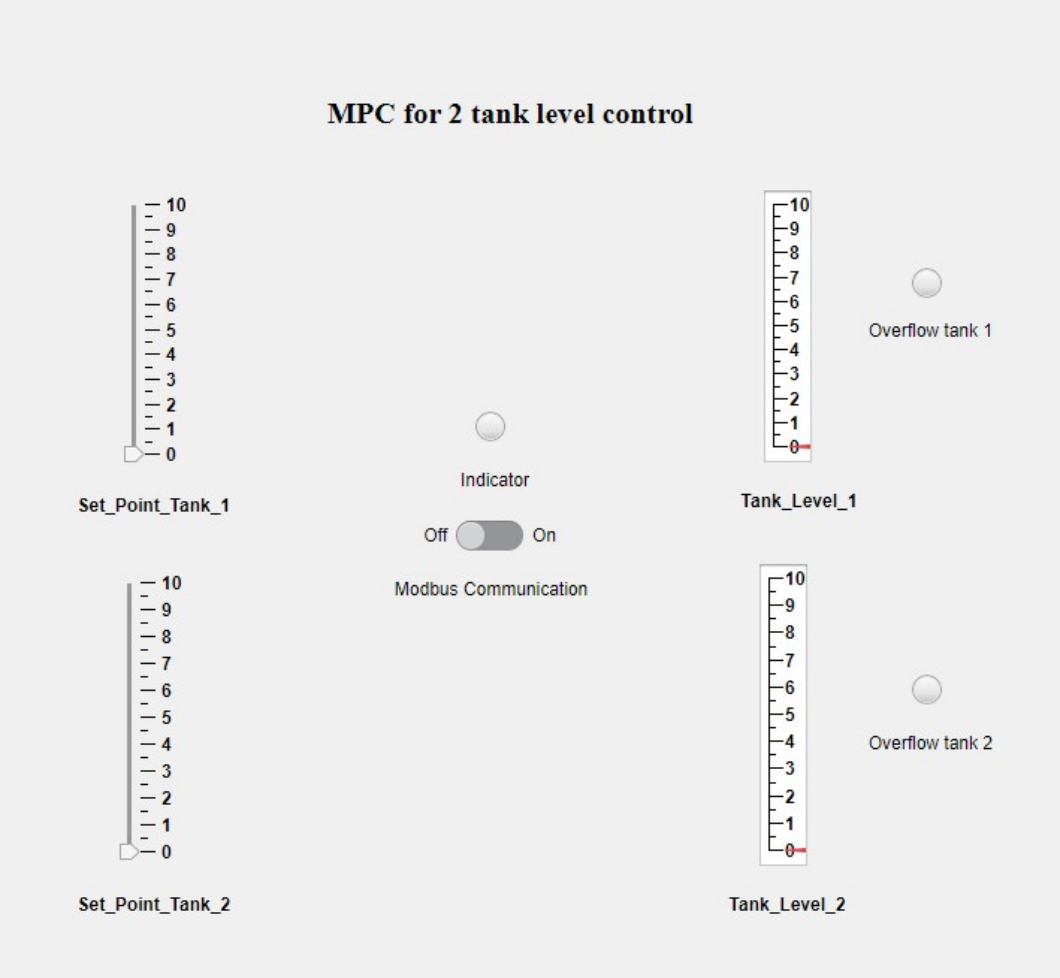

Figure 5-2: HMI display

Plant and PLC simulation on PC2

The PLC i.e. controller and plant for the testbed are simulated using Matlab and Simulink softwares on hardware PC 1. A two tank level control process is simulated in this testbed using Simulink. This process is selected particularly as it is one of the commonly found processes in many of the large scale industrial plants, for which ICS are deployed. Undertaking this process for injecting cyber attacks will enable the testbed to accurately mimic the real world scenario. Figure 5-3 shows this process under consideration where the goal is to maintain the

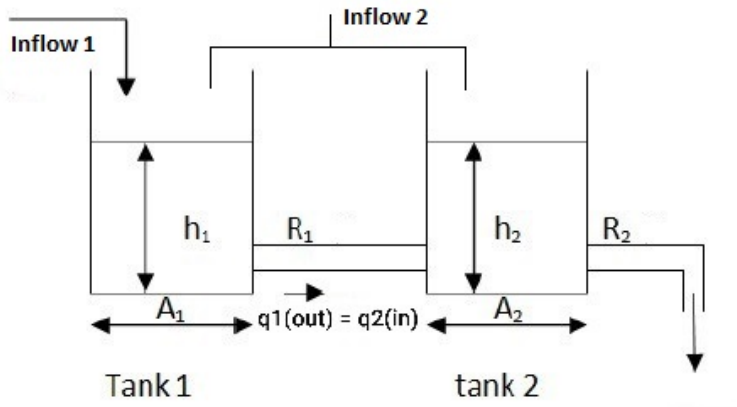

Figure 5-3: Two tank level control process

level in the two interconnected tanks according to the set points given by the operator. The process has two inputs i.e. manipulated variables as inflow 1 and inflow 2. There are two controlled variables which are level 1 and level 2. This process has been modelled in Simulink already in section 4-8. While creating this simulated version of the testbed, the same process has been selected intentionally, to keep uniformity in both testbeds. By simulating the same industrial process in the simulated testbed, it is ensured that the simulated testbed mimics the original testbed as well as the real world ICS scenario.

Figure 5-4 shows a combined model created in Simulink for the two tank level control process. This model combines the model simulated on dSPACE simulator with the simulated testbed, by adding the MPC controller running in Simulink in place of actual PLC. This model gives user an option to switch between which controller (either simulated or PLC) does the user wants to use. In the PLC mode, the model takes inputs and outputs from the dSPACE simulator and runs the logic from PLC. This is the mode to be used for actual testbed designed with real hardware. In the second mode, the model uses a MPC controller designed in Simulink (explained in the next paragraph) and uses the Modbus toolbox in Matlab to take commands from HMI. This communication is made possible by creating two Matlab functions which read and write data values through Modbus Transmission Control Protocol (TCP) to and from the HMI. In this way, the model is carefully designed so that it can be used on the the actual testbed and the simulated testbed, at users will.

Once this process is modelled in Simulink, a Model Predictive Controller (MPC) is designed and deployed for the control of this process. MPC is one of the most famous techniques

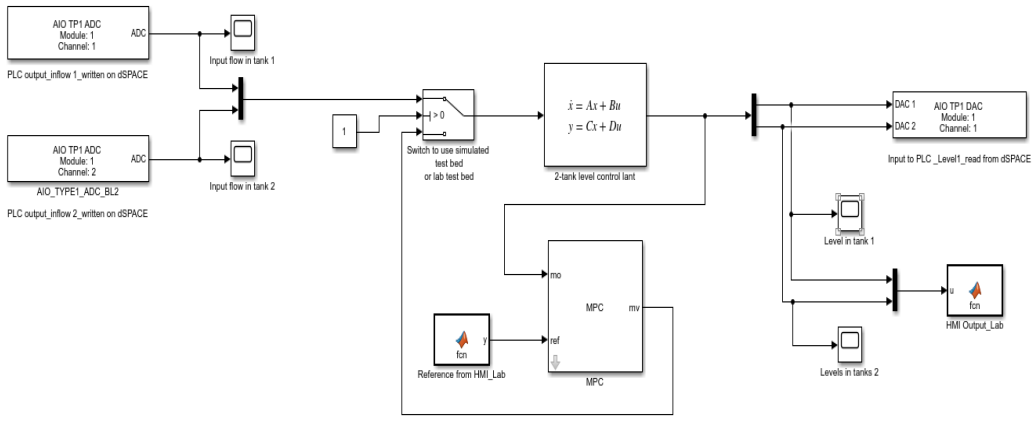

Figure 5-4: Combined model for two tank level control process

used in the process industry for control applications. Therefore, it provides an advantage that it already has ready-made blocks written in IEC61131-3 programming language, which will be used for programming the actual PLC in the testbed. Additionally, MPC provides better constraint handling to achieve optimal control of the process, which is required in this application. Therefore, an MPC is chosen as the control strategy for this application.

Following parameters are selected for the MPC to achieve the optimal result for reference tracking application.

- Prediction horizon = 30 seconds
- Control horizon = 5 seconds
- Input constraints : 0.5 $m^3/sec \leq u(k) \leq$7.5 m^3/sec. A lower bound is put on the input to avoid the dry running of the system which might create low pressure in the pipelines.
- Output weight = $\begin{bmatrix} 10^3 \\ 10^3 \end{bmatrix}$
- Input weight = $\begin{bmatrix} 0.01 \\ 0.01 \end{bmatrix}$
- The high output weight and low input weight ensures better tracking of the levels even at the cost of high input flow, which is the aim of this control application.

The performance of this MPC is shown in figure 5-5. The plant is initialized with level of 3 feet in both tanks. The set points provided by the operator are 5 feet and 2 feet respectively for both tanks. It is clearly observed that the controller tracks the given set points with rise time of 3.86 seconds. This completes the description of the PC2 in simulated setup, corresponding to PLC with the dSPACE simulator in actual setup. Further, to make this MPC code ready for implementation on the actual testbed, the toolbox "Simulink PLC Coder" has been utilized.[36] With help of this toolbox, the MPC code generated and modelled in Simulink is converted into a structured text code, ready to be run on TwinCAT software for programming Beckhoff PLC.

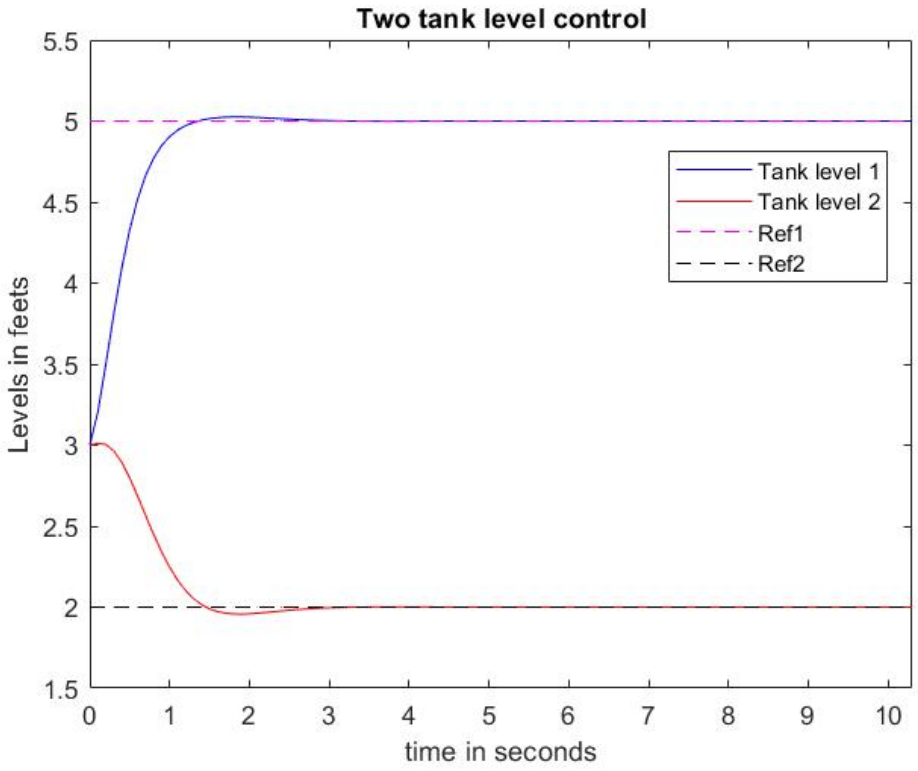

Figure 5-5: Nominal performance of MPC

Attack PC simulation on PC1

As explained earlier, the attack PC is run on hardware PC 1 as virtual PC. A tool called Virtual Box has been used to create and host this attack PC. In Virtual Box environment, a PC with its own Internet Protocol (IP) address and Media Access Control (MAC) address is created. This PC is connected to the network through a virtual switch which works on OpenV switch. Using this "virtual" switch gives the attack PC flexibility to be located anywhere in the world and not be limited to being physically present at the site of the testbed to inject the attacks. Virtual Box is a virtualization software that supports use of OpenV switch to connect the virtual PC to the network. This justifies the use of Virtual Box as virtualization platform in this testbed and was an important criterion for its selection.

This attack PC runs on Kali Linux which is a version of Linux Operating System (OS), designed specially for cyber security applications. The advantage of using Kali Linux is that the OS has many of the tools which are used for launching cyber attacks, pre-installed in it. To generate and launch cyber attacks in this thesis, three such tools are utilized. They are namely; Ettercap, Hping3 and Wireshark. These tools are pre-installed in the Kali Linux, which justifies the use of Kali Linux as the OS for the attack PC. However, these tools are also available to be installed on any version of Linux OS. This concludes the design and implementation of the simulated testbed which mimics the actual TU Delft testbed accurately combining all the hardware available with open source tools.

5-1-3 Comparison of simulated testbed with the actual testbed at Delft University of Technology

Before discussing the attacks on this testbed, it is important to understand the differences as well as pros and cons of using this simulated testbed with respect to the real testbed. This simulated testbed differs from the actual testbed in use of hardware as well as some of the softwares used for simulation. Firstly, the simulated testbed uses a plant model simulated in Simulink, whereas in the actual testbed, we are going to use the dSPACE simulator as the plant. The dSPACE simulator is able to simulate complex industrial processes with greater accuracy than Simulink. The dSPACE simulator will also provide the electric signals of these Input and Output (IO)s through physical cables to the PLC. Such physical connections between plant and controller can not be testbed with Simulink, in this simulated testbed. This behavior makes the actual testbed replicate the real world ICS scenario more closely, allowing us to test the vulnerabilities as well as impacts of cyber attacks on complex plants.

This case is also true about the controller used in the testbeds. The simulated version of testbed implements the MPC with Matlab and Simulink against the real world PLC used in the actual testbed. The real world PLC offers significant advantages over a controller designed in Matlab such as high processing speed, reliability, ruggedness in operation, safety, etc. These are the key parameters that make a PLC suitable for control of industrial plants. Additionally, the programming language used in the PLC is IEC 61131-3 (also known as structured text). We can also use different blocks which are predefined in PLC software of Beckhoff. This programming might also present some unexpected challenges in the real world scenario such as data format mismatch, connectivity issues, etc. These thing make a PLC unique from a controller in Matlab but they can not be tested using just the simulated testbed.

As for the attack PC, the simulated version follows the approach as that of the real testbed. This gives an advantage that the cyber attacks developed and tested on the simulated testbed can be directly implemented on the actual testbed, once the hardware with PLC and dSPACE simulator is setup.

Another difference between the two testbeds which is worth noting is the Modbus communication setup. In the simulated testbed, we have defined and used Matlab functions based on Modbus toolbox of Matlab. We need to recall these functions every time to ensure a successful communication. This consumes a lot of time, depleting the processing power and reducing the speed of the communication. In the actual testbed, we use a run-time proprietary license from Beckhoff to facilitate Modbus communication. Use of proprietary license will offer more speed in data transfer, increasing the capabilities of actual testbed as compared to the simulated version.

From the above discussion, it is clear that the simulated testbed, although replicates the real world ICS scenario closely enough to test cyber attacks, it still does not enable us to exact test behaviour of real world ICS under cyber attacks. The impact of these cyber attack on a large and complex plant also can not be observed with just the simulated testbed. Therefore, we conclude this section with the key takeaway that the while the simulated testbed allows us to perform initial testing of cyber-attack injection using appropriate substitutes for ICS, we still need to build the actual testbed with ICS components to achieve our aim of studying impacts of cyber attacks on real-world ICS.

The next sections will give the results of theMITM attacks and DoS attacks injected on the simulated testbed.

5-2 MITM attack injection and results

In previous section, a simulated testbed with HMI-PLC network controlling a two tank level control process is prepared. The testbed also has an attack PC running with Kali Linux having necessary applications for injecting cyber attacks. In this section, MITM cyber attack is injected on this network following the methodologies in section 2-3-2 and section 2-3-1.

5-2-1 Altered control set point attack using MITM attack

During the normal operation of the ICS testbed, the operator sitting at the HMI i.e. Modbus client, decides and inputs the set point values that are levels of liquid in the two tanks. The PLC receives these set point values as the analog values contained in Modbus data packets. Being analog values, these are then stored in the holding registers of the Modbus server i.e. PLC. The MPC logic programmed in the PLC runs to achieve these set point values and sends back the real time values of levels in the two tanks back to to the HMI. These values sent via the Modbus communication are then displayed on the HMI with the Tank level gauges. This process continues in real time to complete the ICS network under normal conditions.

Now, this attack aims to alter or modify the set points provided by the operator via input set point scale on the HMI. The PLC controller therefore receives set point values that are not intended by the operator, without the knowledge of the operator. This is achieved by the means of MITM attack as explained in section 2-3-2

To launch a MITM attack, the attacker needs to have information about the network such as IP addresses of the PLC and HMI and devices. To achieve this, a reconnaissance attack using ARP poisoning technique is performed on the network. This procedure is explained in detail in section 2-3-1 and section 2-3-2. To launch this attack, an open source tool called "Ettercap" is utilized. This attack forms the first step of the MITM attack by providing the attacker with IP and MAC addresses of the HMI i.e. PC1 and PLC i.e. PC2. The process of ARP poisoning is followed to modify the ARP tables to link the MAC address of the attack PC i.e. PC3, with the PC1 and PC2. Therefore, as shown in figure 2-8, the attack PC sits in between the HMI and PLC, ready to intercept communication between them. The attack PC now has the ability to intercept and visualize the Modbus packets being sent between HMI and PLC and vice-versa. Therefore, the attacker captures these packets and alters the significant bits in these modbus packets that correspond to the necessary information before passing them on to the server i.e. PLC. To understand this, the structure of every Modbus message is already explained in 2-1.

In order to perform the altered control set point attack, the set point value i.e. reference value which is sent by the operator from HMI to the PLC needs to be changed by the attacker. This is realized by developing and implementing "Ettercap Filters", which are able to process and alter the data on the wire in real-time.

Whenever the operator inputs a value from the HMI with the scale, the HMI being Modbus client, sends a "write holding register" command to the PLC i.e. the server. This query to

write the data is shown in table 5-1. In this Modbus packet, the underlined 4 bits are the data fields and hence contain the set point value as the data. In this attack, we have developed a filter to alter the set point from a value of 2 feet to 9 feet, which is the overflow warning value for the tank. Now, as the attack PC is intercepting these messages, it is able to modify the specific bits in this message before passing it to the PLC. Therefore, the changed data bits from 2 to 9 can be observed in the underlined bits in the Modbus packets in table 5-1. In this way the MITM attack is implemented on this testbed to alter the set point which brings the system at a position of overflowing.

Modbus write command sent by HMI and captured by attack PC	Altered Modbus command passed on by the attack PC pretending to be HMI
00 14 00 00 00 06 01 06 00 00 00 02	00 14 00 00 00 06 01 06 00 00 00 09

Table 5-1: Modbus data alteration

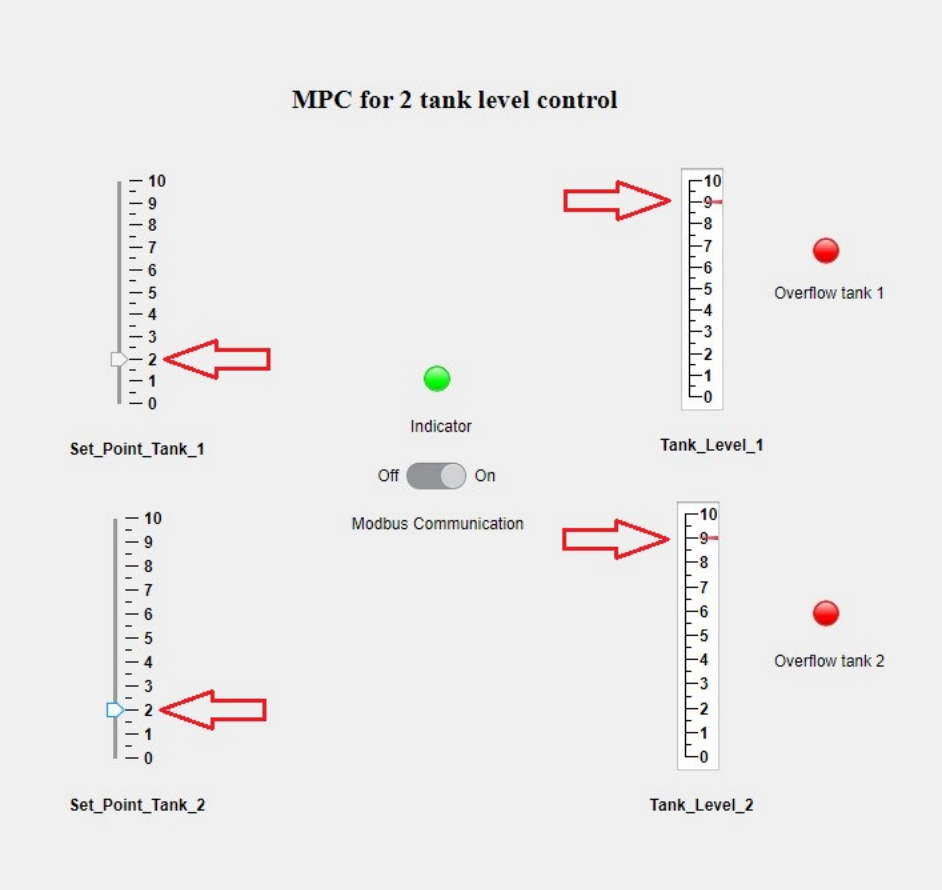

Figure 5-6: Altered control set point attack with MITM on HMI

Figure 5-6 shows this attack under operation on the HMI screen. Here, as described above, the MITM attack is changing the set point value from 2 to 9 feet. This is clearly seen in figure 5-6 where the user input value (2 feet) is seen on the left hand side input scales. However, the actual value of set point that is altered and passed on to the PLC is 9 feet, which is observed

in the gauges on the right-hand side. Due to this attack, the system's overflow alarms are also triggered, all without the intention of the operator.

Similarly, figure 5-7 shows the analysis of this attack on the PLC side. Here, we can see that initially when the attack was not injected, the controller was following set point levels of 2 feet in both tanks accurately. MITM attack was launched at 4.1 seconds, at which point the set points in both the tanks suddenly go to 9 feet, whereas the HMI is still providing the value of 2 feet as the set point. The controller then takes action on this new set point and the levels in both tanks jump to overflow value of 9 feet, against the orders of operator.

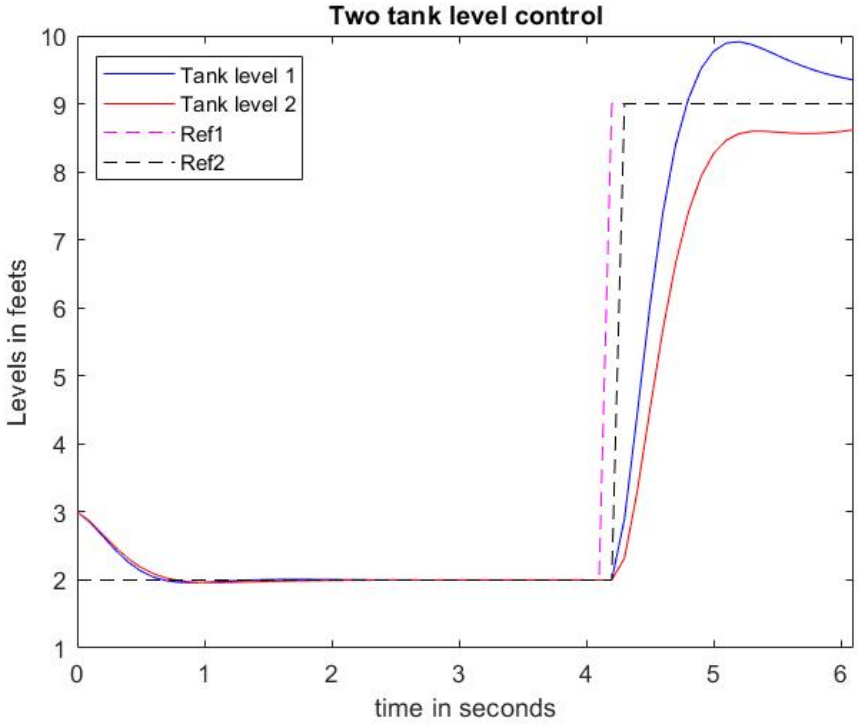

Figure 5-7: Altered control set point attack with MITM on PLC

In this way, a MITM attack was implemented on the simulated testbed to overflow the tanks, against the orders of the operator. This attack can therefore produce such hazardous situations on a real world ICS plant as shown in this section. This completes the description of altered control set point MITM attack with the results given in this section. In the next section, MITM attack is utilized to entirely kill and hence stop the modbus communication between the PLC and HMI.

5-2-2 Stop Modbus communication using MITM attack

In this section, the capabilities of MITM attack are explored further to stop the Modbus communication between the HMI and PLC. To perform this attack, first ARP poising is performed by the attacker on the network to obtain the IP addresses of the the PLC and HMI. Then an Ettercap filter is developed where, these IP addresses are specified as the source and destinations. This filter therefore, identifies and modifies the Modbus packets

exchanged only between these two devices, without hampering the communication in rest of the network.

Every time these Modbus packets are identified and captured by the attack PC, the filter activates and drops these packets going to the destination. Similarly, it kills the Modbus communication going on between the PLC and HMI. This attack therefore not only stops the entire Modbus communication, but also stops the transfer of packets between the client and server while doing so.

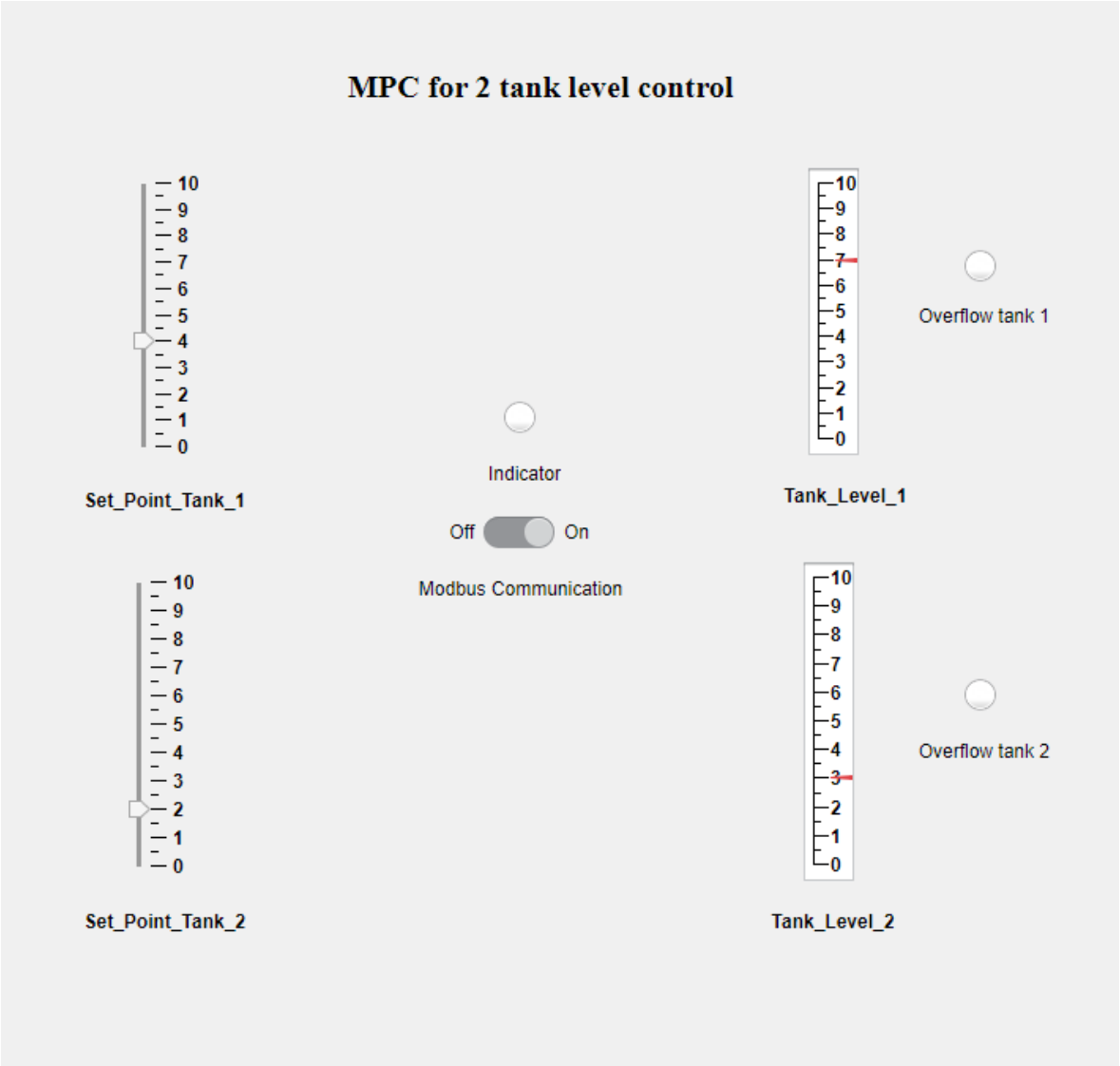

Figure 5-8: Stop Modbus communication attack with MITM on HMI side

Figure 5-8 shows this attack injected on the testbed on the HMI side. The Modbus communication between HMI and PLC is killed which can be observed here as the Modbus communication switch on HMI is ON but the indicator that shown status of communication is OFF. Therefore, operator is unable to send any set point values to the PLC as well as the HMI gets stagnated to the display the residual values of levels. Figure 5-9 shows the

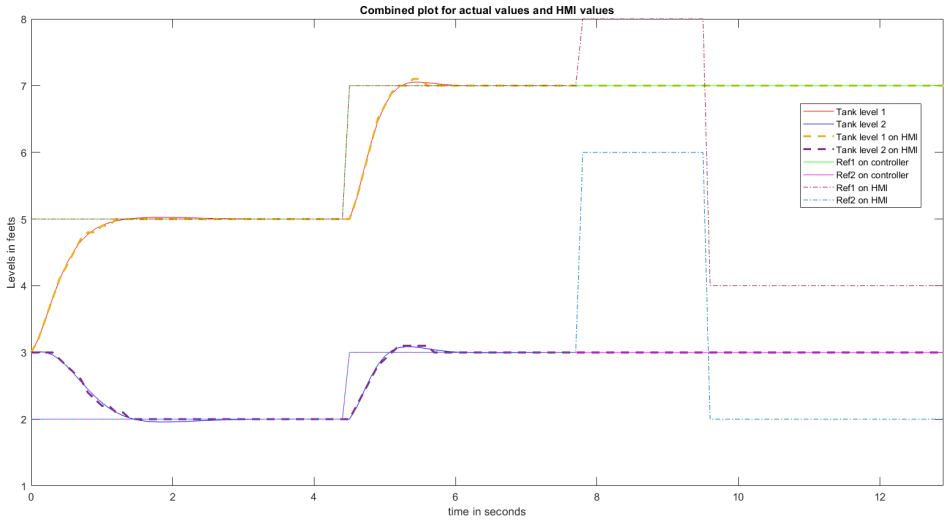

Figure 5-9: Stop Modbus communication attack with MITM on plant side

performance of the plant when under the attack. In this figure, the healthy i.e. nominal operation of the plant is observed initially until 7.8 seconds, as both the tank levels follow the reference given by HMI. But, after this point, the attack kills the Modbus communication and therefore the reference values changed by the operator after this point do not reach the PLC, as can be seen by the dotted maroon and dotted blue lines in figure 5-9. Figure 5-10 shows the error that "a communication link can not be created with the server i.e. PLC" on the HMI.

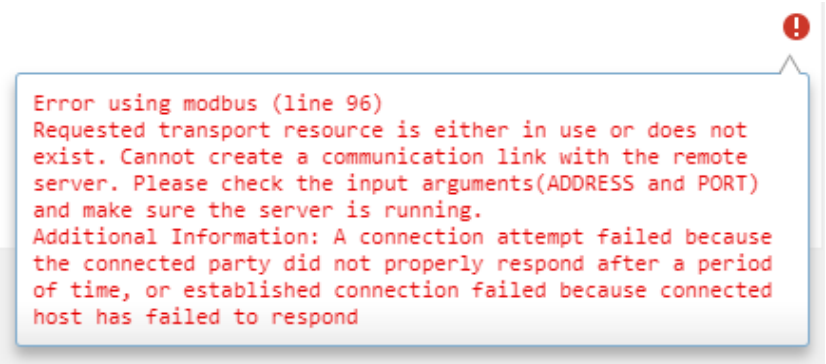

Figure 5-10: Error on HMI

Next section described another MITM attack which exploits weakness in Modbus protocol to exclude the Modbus packets reaching to the destination hampering communication in the network.

5-2-3 Invalid function code attack using MITM attack

This section describes how invalid function code attack is injected using MITM technique. This technique follows the same basic methodology as described in section 5-2-1, only difference being that during this attack, the attacker alters the bits in the Modbus message that correspond to the function code value, instead of changing the data fields.

As explained earlier, each Modbus query and response message between a client and a server contains 2 bits known as function code. This function code tells the Modbus server what action it is supposed to perform. These function codes and the corresponding actions are given in table 2-5. Under normal situation, the function codes in query and response match each other. If these codes are not supported by the destination device or do not match with each other, the message is not transferred successfully. This called invalid function code attack.

During this attack, the attacker therefore modifies the bits corresponding to the function code fields. Hence, when the client sends query to write the holding register which corresponds to function code of "06" (in Hexadecimal), the attacker captures and changes this bit to a value of "2D" (in Hexadecimal), which is a function code not supported by the server. Therefore, the message is not transferred and an error in Modbus communication is produced.

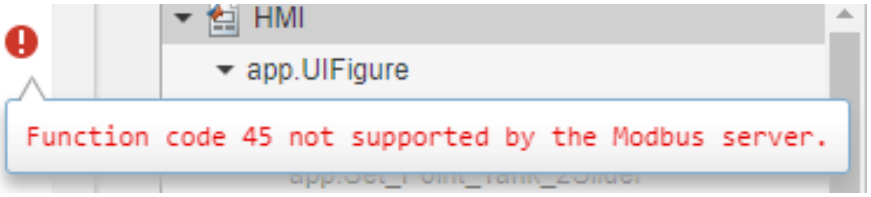

Figure 5-11: Error for invalid function code attack on HMI

Figure 5-11 shows the error produced due to this invalid function code attack on the HMI side. Further, figure 5-12 demonstrates the HMI screen, when under this attack, where it can be observed the that although the communication between Modbus devices is ON, the actual data transfer of set point values is stopped. This is observed in the figure 5-12 as the indicator for Modbus communicator is still ON unlike in figure 5-8, but the new values of control set points are not passed to the PLC.

Lastly, figure 5-13 shows the performance of the plant when under the attack. Here, the healthy operation of the plant is observed initially until 5.3 seconds, as both the tank levels follow the reference given by HMI. After this point, the HMI produces an error and the newly changed values of references after this point by the operator are not passed to the HMI.

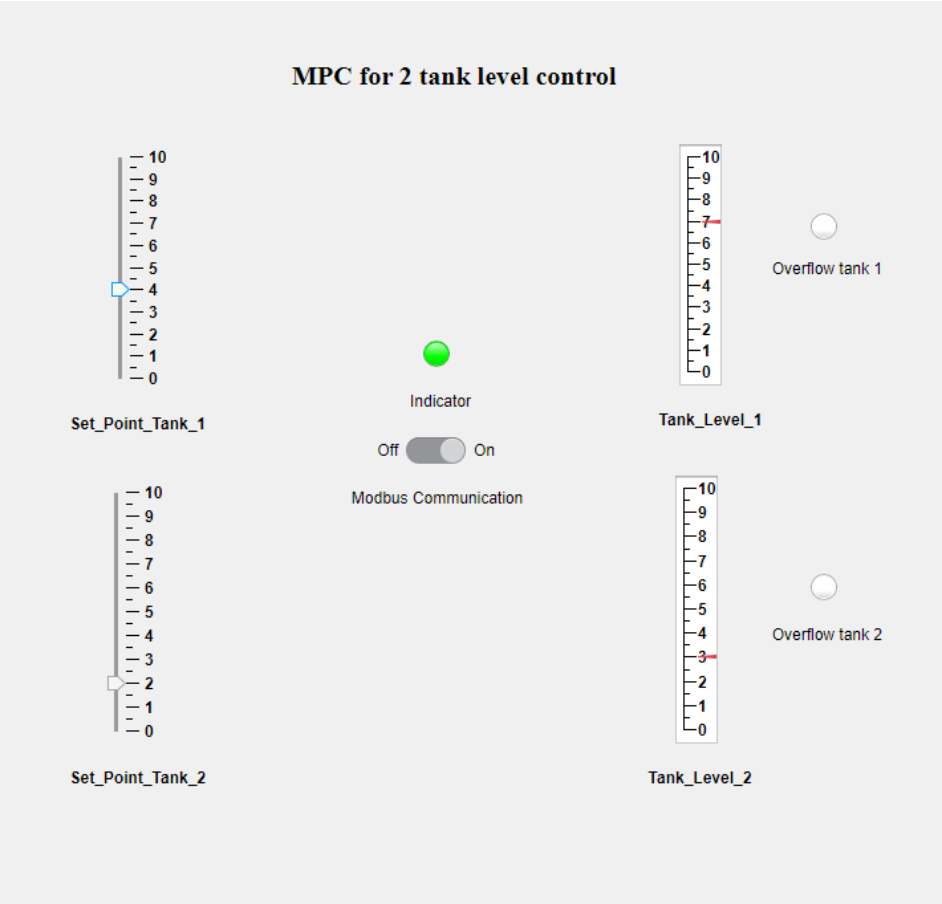

Figure 5-12: Invalid function code attack on HMI

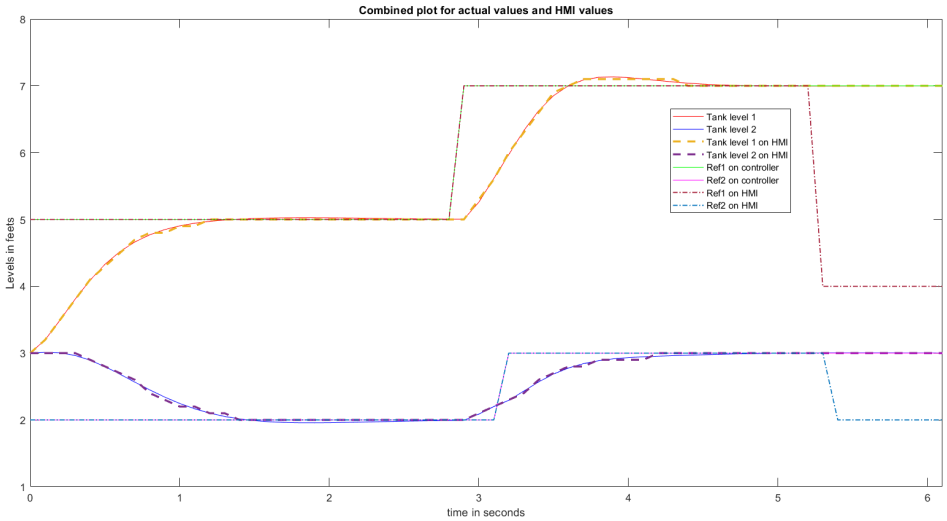

Figure 5-13: Invalid function code attack on plant side

In this way various instances of MITM attack were created, injected and tested in section 5-2-1, section 5-2-2 and section 5-2-3. The only weakness in these attacks is that, the operator sitting at the HMI screen can see that something is wrong with the control system. This can be observed by the operator from the difference in the set point provided by him and the actual level in the tank that is achieved by the controller as seen in figure 5-6. This can also be observed by the errors generated on the HMI in case of the stop Modbus communication and invalid function code MITM attacks. Therefore, as an extended measure, we tried and tested a case where undetectable MITM attack is developed and injected by the attacker.

The words undetectable and stealthy are often used with the same meaning in daily conversations. However, for their use in scientific literature and to avoid any confusion of the reader, we will define these terms here, to be referred to in the rest of this paper. There exist various definitions of an undetectable cyber attack in the literature. Fabio Pasqualetti, Florian Dörfler and Francesco Bullo, in their paper "attack detection and identification in cyber-physical systems" define an undetectable attack as follows. "An attack is undetectable (respectively, unidentifiable) if no monitor detects (respectively, identifies) the attack." [37] Here a monitor is an algorithm with access to continuous-time measurements and knowledge of the system dynamics.

Similarly, a stealthy attack also has been defined in multiple ways. One of the generic definitions of stealthy is as follows. "The attack where skilled attackers conceals their manipulation of the system and bypass intrusion detection by following the expected behavior of the system closely, while still injecting enough false information into the system after a long period of time to achieve their attack goals. Attacks of this kind are referred to as stealthy attacks." [38] However, this definition of stealthy, depends to a great extent, on the type of detection mechanisms used in the system. Therefore, to present our results with clarity, for the purpose of this report, we will refer to the definition of undetectable attacks given by Fabio Pasqualetti, Florian Dörfler and Francesco Bullo as mentioned earlier. During these attacks, the plant operator is "deceived" into believing that the system is performing perfectly, while in reality, it is not. This is the reason that the attacks go undetected. In the next section, such a case of undetectable MITM attack is developed and injected on the testbed.

5-2-4 Undetectable MITM attack

The attacks developed in the previous section are effective, yet there is a room for improvement for the attacker to develop a more effective attack. In this section, this weakness in the previously discussed attacks is tackled by developing an undetectable MITM attack. To develop and launch this advanced attack, we make the assumption that the attacker has deeper understanding of the process under operation, making him or her, the "skilled" attacker. Unlike the previously seen attacks, this undetectable MITM attack modifies the control input values to disturb the process, while still replicating the response of the system as under healthy conditions on the HMI screen. As the set point level given by operator is achieved by the indicator on the HMI screen, the operator sitting at the HMI does not notice any suspicious activity and is deceived into believing that the plant is functioning normally. The attack also does not produce any error on the HMI side and hence is undetectable in operation. As the false sequence injected by the attacker follows the healthy system dynamics, this attack cannot be detected by a monitor, as mentioned in the definition of the undetectable attack.

During this attack, the attacker uses a similar technique as used while injecting altered control set point attack in section 5-2-1, but extends it to make the attack undetectable from the HMI screen.

To implement this attack, two different Ettercap filters have been developed and are used simultaneously. The attack starts by changing the data bits in the query sent by the Modbus client to the server PLC to write a control set point value. This alteration by the attacker is shown in table 5-2. In this case, the operator is inputting a control set point of 9 feet. But the attacker is changing this value from 9 to 5 and hence the PLC will still keep on receiving the value of 5 feet as its set point.

Modbus write command sent by HMI and captured by attack PC	Altered Modbus command passed on by the attack PC pretending to be HMI
00 14 00 00 00 06 01 06 00 00 00 09	00 14 00 00 00 06 01 06 00 00 00 05

Table 5-2: Modbus data alteration between HMI to PLC communication in undetectable attack

In the second step, the attack again changes the data bits, but this time in the response coming from the PLC to the HMI, to the query generated by the HMI to read level value. A second Ettercap filter is used to gradually alter and replace the actual data bits with the false data bits. This gradual and continuous insertion of data into Modbus packets makes the level indicator increase values exactly as it would do in healthy operation (not under attack). This ensures that the level indicator does not directly jump to the set point value, but instead follows the healthy (when not under attack) response of the system to make the attack undetectable for the plant monitor. This satisfies the definition of the undetectable attack as fined the above section, making this a undetectable MIT attack.

Now, the PLC sends the value of 5 feet as its true value to the HMI. But, the attacker changes this data value from 5 to 9 feet. Therefore, the value on HMI gauge for level 1 will now reflect the value of 9 feet, which is indeed what the operator expects from the controller, as it follows the set point given by him. This is shown in figure 5-14.

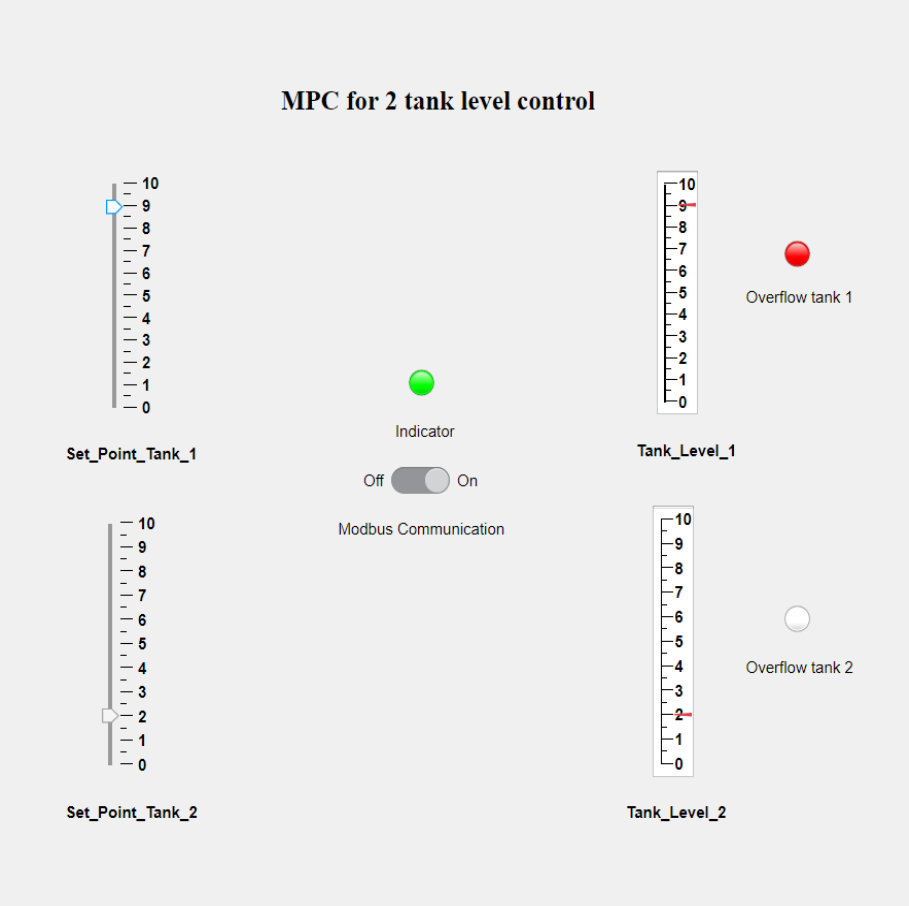

Figure 5-14: Level followed on the HMI as per the set point during undetectable attack

However, this is not the actual value in the tank, as the data has been modified by the man-in-the-middle attacker. The actual level in tank one is still only at 5 feet and yet, the HMI gives a overflow warning. This can be verified by seeing the graph in figure 5-15. In this graph, tank levels are plotted as the values received by the HMI and the actual tank level in the process. In figure 5-15, the blue lines show the real tank level one in the tank, which remains at 5 feet throughout the attack. The green line shows the tank level one, as shown on the HMI. Due to the gradual injection of false values here, the response on the HMI follows the expected response of the process closely. The level of 9 feet is achieved on the HMI as expected by the operator, which completed the undetectable MITM attack.

Therefore, during this undetectable MITM attack, the operator sends control set point of 9 feet to level 1. The HMI also shows this value of 9 feet on the gauge for the tank 1. Hence, the operator does not notice and suspect any malicious activity. But in reality, the level of in tank one still remains to 5 feet.

Such an attack can produce really devastating effects on the ICS plant. The operator is bound to take wrong decisions by seeing such false values of input and output and by thinking they are real. With this demonstration, we have taken the MITM attack to a next level to create undetectable MITM attacks.

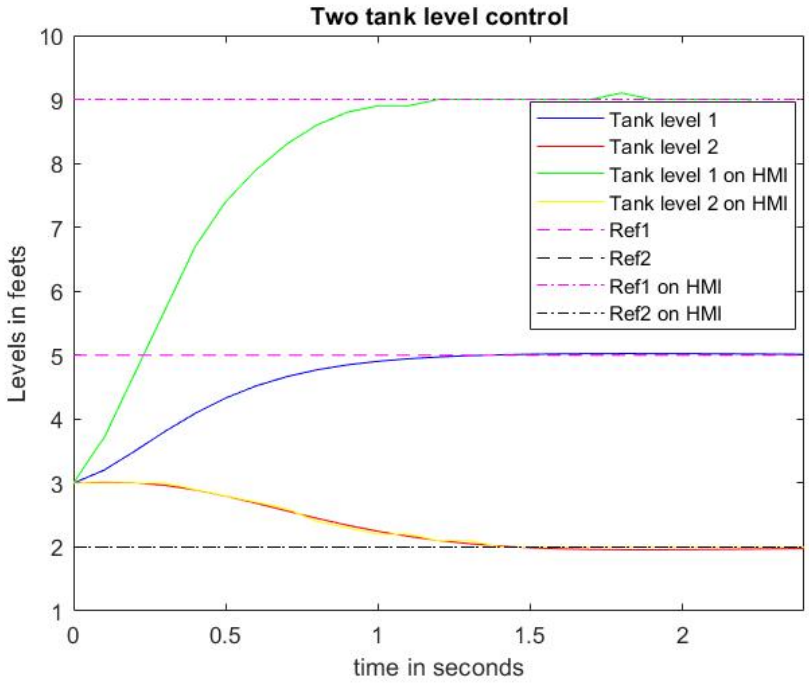

Figure 5-15: Difference in the actual tank level and level as observed on the HMI

With this detailed analysis of these MITM attacks, we move to the next section, where the results of Denial of Service (DoS) attacks are discussed.

5-3 DoS attack injection and results

The last section described the MITM attack injection and discussed its results. This section injects a different type of attack called DoS (Denial-of-Service) attack as explained in section 2-3-3.

DoS attack works on the principle of bombarding the PLC i.e. the server device with Sync (SYN) connection requests at a very high rate and not complete the three-way handshake by denying to send final Acknowledgement (ACK) (Acknowledgement) message, which is necessary to establish a successful connection. This drains the resources of server to a point where it cannot accept and respond to new connection requests from any other legitimate devices and "denies to offer necessary service" to the client. Hence, called Denial of Service attack.

In this section, a tool called "Hping3" [1] has been used to launch DoS attack on the simulated testbed. This is a tool which generates and sends TCP/IP packets to the specified server device with a specified time interval. This tool is pre-installed on Kali Linux and is also supported on various other OS for Windows, Linux, Mac and OpenBSD.

[1] *Documentation for TCP Flooding using Hping - Active Network Security Tool :* http://wiki.hping.org/109

To visualize the data traffic being injected during the attack to its results, another tool called "Wireshark" [34] is utilized. Wireshark is a pack capture and analyzer tool. It is freely available on Windows as well as on Linux OS. Using Wireshark, the real time packets being sent on the TCP network can be visualized. The tool has been used to visualize the injection of malicious TCP packets sent by the attacker using Hping3, which causes the Denial of service, as explained above.

5-3-1 DoS attack using TCP SYN flood attack

In this section, a DoS attack with TCP SYN Flood technique was injected on the testbed. In this attack, TCP SYN packets were generated and flooded on the Modbus TCP server i.e. the PLC. In this attack, the IP address of the attacker is kept constant and SYN requests are sent using multiple different ports on the same IP address. The results of this attack are shown below.

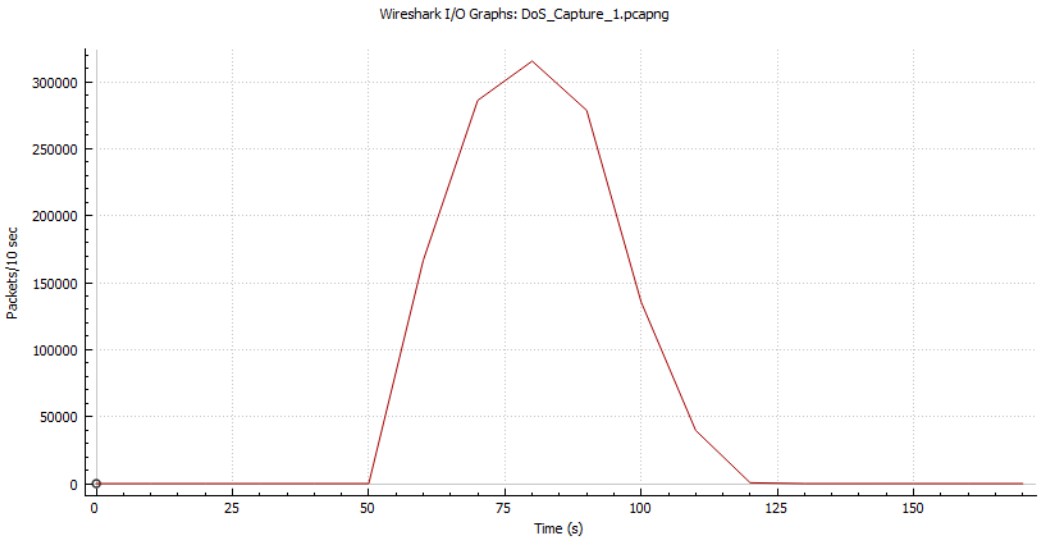

Figure 5-16: Packet capture of SYN requests injected the attacker during the attack

Figure 5-16 shows the SYN Flood attack injected on the system. In this graph number of SYN errors i.e. errors due to incomplete handshake are shown against time. The network is running without any SYN errors till 50 seconds, where the DoS attack is launched by flooding the PLC with SYN requests. During our attack, we sent the packets at the highest rate of over 300000 SYN requests / 10 seconds, using different ports on the attack PC. This depleted the memory resources of the PLC and the device became in-responsive at 76 seconds giving the errors on the HMI, as shown in figure 5-17. This decreased the rate of SYN requests and DoS attack was finally stopped at 120 seconds. This can be clearly observed in figure 5-16, as the SYN errors fall to 0 after 120 seconds.

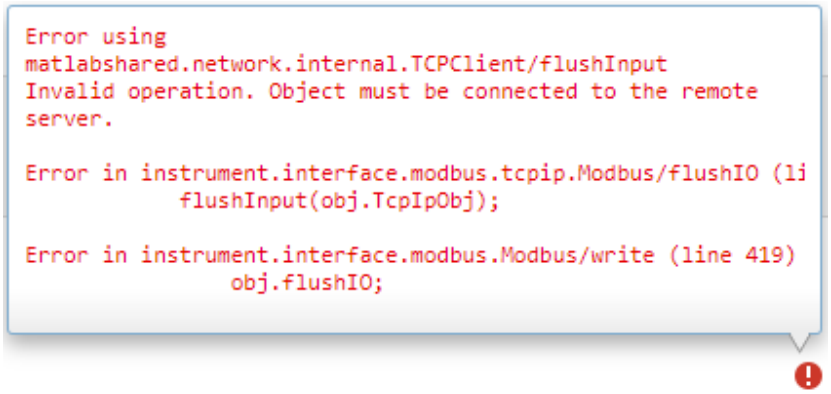

Figure 5-17: Error due to DoS attack using multiple ports and same IP address

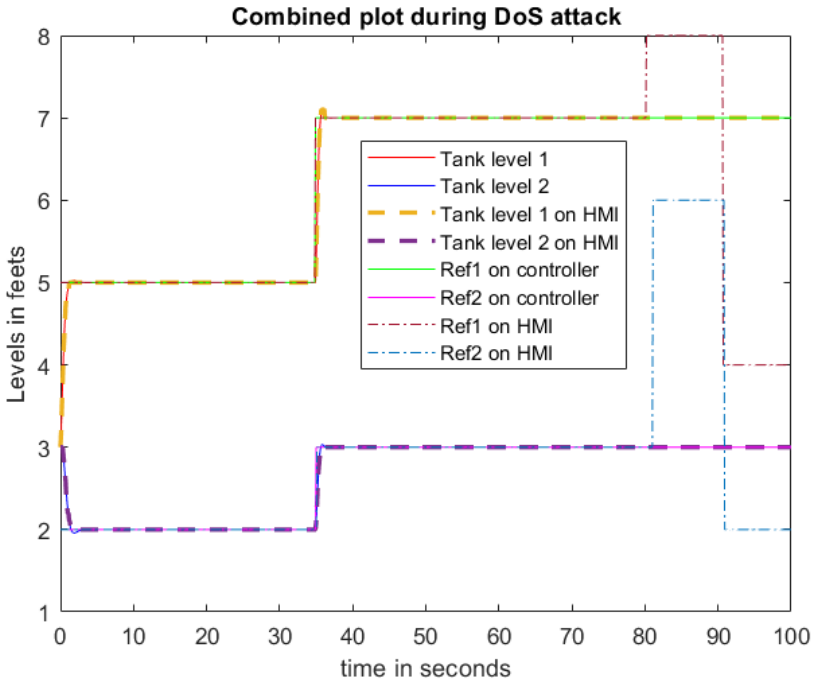

Figure 5-18: Performance of the system before and during the DoS attack

The performance of the plant when under this DoS attack is observed in figure 5-18. In this figure, the red and blue lines (actual tank levels) show that the plant is performing normally, tracking reference levels input from the HMI till 80 seconds. The reference provided by the HMI (dashed-dotted maroon and sky blue lines) also match the reference values that are received by the controller (green and magenta lines), indicating that that attack is not yet effective. But, at 76 seconds the DoS attack becomes effective, stopping the communication between HMI and controller. As a result new set point values that are input by the operator at 80 second and 91 seconds i.e. after 76 seconds, can not reach the controller. This can be observed in figure 5-18 by the difference in the HMI levels as sent by the HMI and actually received by the controller after 80 seconds. This DoS attack, therefore, crashed our HMI-

PLC communication by flooding SYN requests, making it unresponsive to new legitimate input values, disabling the operator to change state of the plant.

This attack however is comparatively easier to block, as the attacker does not mask its IP address and only uses different ports on its own IP address to send SYN requests. Therefore, to mitigate this limitation and to make this attack more sophisticated and untraceable, a second instance of DoS attack was designed and injected on the system, as explained in next section.

5-3-2 Sophisticated DoS attack combining IP spoofing and TCP SYN flood techniques

In this sections, in order to mitigate the limitation of above mentioned DoS attack, a sophisticated DoS attack is implemented on the testbed. To achieve this, Hping3 tool has been used in a different setting. In this case, the Hping3 tool still floods the PLC with TCP SYN requests without completing the handshake same as earlier, but now it creates random "fake" or "spoofed" IP addresses to send these requests. Due to this, the true identity of the attacker is masked making it impossible for the victim to trace the source of attack i.e. attacker. In this way, this sophisticated DoS attack makes it immensely difficult for the victim of the attack to block all these IP addresses, altogether.

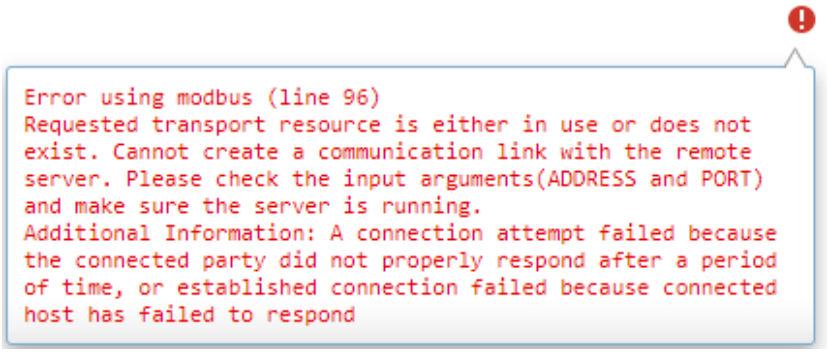

Figure 5-19: Error due to DoS attack due to DoS attack with multiple IP addresses

Figure 5-19 shows the error produced on the HMI side due to this DoS attack. The functioning of the HMI is stopped in the same way as in the DoS attack below. The difference lies in the IP addresses used to launch this attack that makes it impossible to locate and pinpoint the attacker. This use of multiple "spoofed" IP addresses while launching the attack can be observed in figure 5-20.

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Source IPv4 Addresses	6869				0.1723	100%	39.3100	39.783
192.168.0.103	1434				0.0360	20.88%	0.9400	2.934
192.168.0.101	1332				0.0334	19.39%	0.2200	13.379
52.222.139.39	6				0.0002	0.09%	0.0400	0.000
52.114.92.89	1				0.0000	0.01%	0.0100	2.373
142.250.102.188	3				0.0001	0.04%	0.0200	15.379
62.138.16.230	25				0.0006	0.36%	0.0700	18.247
52.159.49.199	2				0.0001	0.03%	0.0200	6.568
142.250.102.189	21				0.0005	0.31%	0.0100	11.496
34.120.186.93	3				0.0001	0.04%	0.0200	19.624
20.190.159.134	1				0.0000	0.01%	0.0100	12.561
172.217.19.197	1				0.0000	0.01%	0.0100	13.035
192.168.0.1	5				0.0001	0.07%	0.0200	13.363
52.109.28.63	29				0.0007	0.42%	0.1000	13.717
13.107.18.11	17				0.0004	0.25%	0.1700	13.449
20.190.160.8	9				0.0002	0.13%	0.0600	14.253
34.107.195.226	1				0.0000	0.01%	0.0100	15.009
192.168.0.40	6				0.0002	0.09%	0.0100	16.272
84.53.185.144	15				0.0004	0.22%	0.0600	17.901
172.217.17.131	7				0.0002	0.10%	0.0700	18.334
137.135.225.146	1				0.0000	0.01%	0.0100	20.359
52.113.205.7	1				0.0000	0.01%	0.0100	20.586
172.67.74.43	2				0.0001	0.03%	0.0100	20.946
142.251.36.14	11				0.0003	0.16%	0.0700	21.382
52.112.231.38	2				0.0001	0.03%	0.0100	29.366
198.252.206.25	1				0.0000	0.01%	0.0100	34.418
68.232.34.200	1				0.0000	0.01%	0.0100	37.641
40.127.110.237	1				0.0000	0.01%	0.0100	37.693
236.254.40.229	1				0.0000	0.01%	0.0100	39.783
190.178.231.238	1				0.0000	0.01%	0.0100	39.783
178.36.73.67	1				0.0000	0.01%	0.0100	39.783
204.156.194.123	1				0.0000	0.01%	0.0100	39.783
89.130.249.36	1				0.0000	0.01%	0.0100	39.783
89.194.172.184	1				0.0000	0.01%	0.0100	39.783
145.97.45.104	1				0.0000	0.01%	0.0100	39.783
58.123.204.251	1				0.0000	0.01%	0.0100	39.783
145.251.45.172	1				0.0000	0.01%	0.0100	39.783
201.179.131.131	1				0.0000	0.01%	0.0100	39.783
253.197.101.30	1				0.0000	0.01%	0.0100	39.783
239.110.242.164	1				0.0000	0.01%	0.0100	39.783
203.129.156.202	1				0.0000	0.01%	0.0100	39.783
90.110.103.104	1				0.0000	0.01%	0.0100	39.783
106.24.64.110	1				0.0000	0.01%	0.0100	39.783
118.108.84.237	1				0.0000	0.01%	0.0100	39.783
233.84.105.104	1				0.0000	0.01%	0.0100	39.783
225.137.106.138	1				0.0000	0.01%	0.0100	39.783
163.60.188.197	1				0.0000	0.01%	0.0100	39.783
50.212.174.154	1				0.0000	0.01%	0.0100	39.783
171.158.157.24	1				0.0000	0.01%	0.0100	39.783
202.136.246.45	1				0.0000	0.01%	0.0100	39.783
93.54.172.33	1				0.0000	0.01%	0.0100	39.783
221.226.108.245	1				0.0000	0.01%	0.0100	39.783

Figure 5-20: Sophisticated DoS attack with multiple fake IP addresses

Here, it can be observed how various random IP addresses are used by the attacker to insert the SYN queries in the network. Compare this with figure 5-21, where only a single IP address to send all they SYN requests is used, making it easily detectable to cautious victim. Whereas, as these randomly generated IP addresses do not even exists on the network in reality, they never complete the three-way handshake, keeping all the connection requests open, draining

the resources of the PLC and eventually crashing it.

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
192.168.0.106	2776				0.0386	13.21%	14.5800	71.629
TCP	2776				0.0386	100.00%	14.5800	71.629
3994	1				0.0000	0.04%	0.0100	71.822
3993	1				0.0000	0.04%	0.0100	71.822
3992	1				0.0000	0.04%	0.0100	71.822
3991	1				0.0000	0.04%	0.0100	71.822
3990	1				0.0000	0.04%	0.0100	71.821
3989	1				0.0000	0.04%	0.0100	71.821
3988	1				0.0000	0.04%	0.0100	71.821
3987	1				0.0000	0.04%	0.0100	71.821
3986	1				0.0000	0.04%	0.0100	71.821
3985	1				0.0000	0.04%	0.0100	71.821
3984	1				0.0000	0.04%	0.0100	71.821
3983	1				0.0000	0.04%	0.0100	71.821
3982	1				0.0000	0.04%	0.0100	71.817
3981	1				0.0000	0.04%	0.0100	71.817
3980	1				0.0000	0.04%	0.0100	71.817
3979	1				0.0000	0.04%	0.0100	71.817
3978	1				0.0000	0.04%	0.0100	71.817
3977	1				0.0000	0.04%	0.0100	71.817
3976	1				0.0000	0.04%	0.0100	71.817
3975	1				0.0000	0.04%	0.0100	71.817
3974	1				0.0000	0.04%	0.0100	71.817
3973	1				0.0000	0.04%	0.0100	71.817
3972	1				0.0000	0.04%	0.0100	71.817
3971	1				0.0000	0.04%	0.0100	71.812
3970	1				0.0000	0.04%	0.0100	71.812
3969	1				0.0000	0.04%	0.0100	71.812
3968	1				0.0000	0.04%	0.0100	71.812
3967	1				0.0000	0.04%	0.0100	71.812
3966	1				0.0000	0.04%	0.0100	71.812
3965	1				0.0000	0.04%	0.0100	71.812
3964	1				0.0000	0.04%	0.0100	71.812
3963	1				0.0000	0.04%	0.0100	71.812
3962	1				0.0000	0.04%	0.0100	71.812
3961	1				0.0000	0.04%	0.0100	71.812
3960	1				0.0000	0.04%	0.0100	71.812
3959	1				0.0000	0.04%	0.0100	71.812
3958	1				0.0000	0.04%	0.0100	71.812
3957	1				0.0000	0.04%	0.0100	71.812
3956	1				0.0000	0.04%	0.0100	71.812
3955	1				0.0000	0.04%	0.0100	71.812
3954	1				0.0000	0.04%	0.0100	71.812
3953	1				0.0000	0.04%	0.0100	71.812
3952	1				0.0000	0.04%	0.0100	71.812
3951	1				0.0000	0.04%	0.0100	71.812
3950	1				0.0000	0.04%	0.0100	71.812
3949	1				0.0000	0.04%	0.0100	71.812
3948	1				0.0000	0.04%	0.0100	71.812

Figure 5-21: Sophisticated DoS attack with multiple ports on same IP address

This detailed analysis of MITM and DoS attacks along with the results of this attacks on the simulated testbed completes this chapter. This chapter described the detail procedure for developing and injecting different types of MITM and DoS cyber attacks. Using MITM techniques, Altered control set point attack, Stop modbus communication attack and Invalid function code attack were injected on the testbed. Whereas, using DoS techniques, TCP syn flood attack was performed. Further, these both major types of attacks were extended in their

capabilities. A more effective MITM attack was implemented by developing a undetectable MITM cyber attack. Whereas, a sophisticated DoS attack was developed, which enabled the attacker to hide its true identity, making it difficult for the victim to block this attack. This concludes the detailed analysis of simulated ICS testbed for cyber attack generation developed during this thesis.

Chapter 6

Conclusion

ICS are widely used worldwide for control of industrial plants and they include all necessary instrumentation such as sensors, Programmable Logic Controller (PLC), Human Machine Interface (HMI), actuators and all required communication infrastructure. With the advancements in internet and increased need of automation due to Covid-19, these ICS are rapidly being connected to the internet and are therefore facing increasing cyber threats. Through our research, we identified this increasing risk of cyber attacks and recognized the need for an ICS testbed to generate and test cyber attack on ICS at TU Delft. In this conclusion, we have summarized our answers to the research questions formulated in section 1.

Q1. Which of the vulnerabilities of Modbus protocol in the real world ICS network can be exploited with this testbed to launch cyber attacks on the ICS?

Ans. Modbus protocol employs a standard request-response messaging structure between the client and the server. Modbus protocol is prone to cyber attacks due to its weaknesses such as having no authentication of the client while accepting the message and absence of encryption in communication. These vulnerabilities in Modbus protocol are exploited to launch cyber attacks on this ICS testbed. In this testbed, MITM attack has been injected on the testbed which captures the communication between client-server by exploiting the weakness of lack of authentication and absence of encryption in Modbus protocol. Whereas, DoS attack exploits the weakness of lack of authentication while accepting the client requests in Modbus protocol.

Q2. and Q3. How are the main types of cyber-attacks on ICSs, namely Man-In-The-Middle (MITM) attack and Denial-of-Service (DoS) attack implemented in real world? How can the two attacks mentioned in point 2 be introduced in the ICS network modeled in the testbed?

Ans. A Detailed explanation about the methodologies for injecting these attacks on ICS is elaborated during this report. We have used the same methodology to inject attacks on the testbed that is used in the real world, making the testbed replicate the real world ICS scenario accurately. MITM cyber attacks utilizes ARP poisoning technique to link the attacker PCs IP address to the MAC address of client and server. Due to this, the

attacker can now intercept all communication between client (HMI) and server (PLC) of the ICS network of the testbed. In DoS attack, the attacker sends requests to makes connection to the PLC at extremely high rate, without completing this connection. This depletes the resources of PLC and makes it unavailable to service legitimate requests from HMI, failing the ICS communication. In this way, MITM and DoS attacks are introduced in the testbed, following the same method as done in the real world.

Q4. Which software can be utilized to launch the cyber attacks on the ICS network in this testbed?

Ans. In this testbed, the attack PC runs on Kali Linux Operating System (OS), which is a version of Linux OS, specially created for cyber attack analysis by pre-installing tools for cyber attack generation on it. In this thesis, "Ettercap" tool has been utilized to launch MITM cyber attack on the testbed. Whereas, "Hping3" tool has been used to inject DoS cyber attacks. An analysis of data collected with the help of "Wireshark" tool was also utilized to make cyber attacks effective.

Q5. What software can be used to monitor and capture the Modbus data packets to analyze successful implementation of the cyber attack?

Ans. In this testbed, a tool called "Wireshark" has been used to monitor and analyse Modbus data packets being transferred on the ICS network. This tool is also utilized to visualize and plot graphs of packet transfer rates, which help us test cyber attacks injected on the network.

Q6. What are the main hardware components needed to make this testbed replicate real world ICS scenario accurately?

Ans. To replicate the real world ICS scenario accurately, the testbed needs to have real world ICS components. A careful analysis of various approaches to design a testbed has been carried out during the design process of the testbed. Further, the feasibility of these various approaches has been checked. After finalizing the design of the testbed, a detailed comparison between different hardware components, provided by different manufacturers has been performed. Multiple rounds of negotiations and discussions with the vendors have been carried out during this thesis, to ensure that all testbed requirements i.e functional as well as economic requirements of Delft University of Technology, are satisfied. Further, these components have been selected, and have been ordered from Beckhoff company during this thesis. These components are Beckhoff PLC (Industrial PC), Beckhoff HMI, Beckhoff analog and digital input and output cards for the PLC, the EtherCAT coupler and a 24 V power supply as other accessories.

Based on the answers to these sub-questions, we are finally able to answer our main research question as follows.

Q. Can a cyber-attack testbed be built in the NERDlab, which is able to replicate a real-world ICS network with the accuracy needed to identify and test vulnerabilities of ICSs working on Modbus protocol to cyber-attacks?

Ans. Yes. A testbed using real world Industrial Control Systems (ICS) components from brand Beckhoff has been designed during this thesis. This testbed uses Beckhoff PLC and HMI to control a complex, real world industrial process simulated using dSPACE simulator. The dSPACE simulator has been modelled. The testbed includes an attack PC with Kali Linux OS which generates and injects MITM and DoS attacks on the PLC-HMI network, using tools such as Ettercap and Hping3. A simulated version of this testbed has also been created and cyber attacks have been injected on this testbed.

In this way, all the research questions formulated before starting this research have been answered in this Master's thesis and summarized here. This Master's thesis is completed by designing a testbed for cyber attack generation and testing at NERD Lab at Delft University of Technology (TU Delft). Further, a simulated version of this testbed has been developed and cyber attack have been generated, injected and tested on this testbed. During this thesis, two major types of cyber attacks were injected on the simulated testbed, namely, MITM and DoS attack. Using the MITM attack strategy, four attacks were tested. First, altered control set point attack which altered the set point value sent by the HMI to the controller. As a result, the plant achieved incorrect tank levels, against the order of the operator and triggered the tank overflow alarms. Secondly, a stop Modbus communication attack was injected using MITM strategy, which halted the entire communication between HMI and controller. This disabled the operator from inputting new set point values and produced an error on the HMI screen making it unresponsive. The third MITM attack i.e. invalid function code attack also disabled the operator from inputting new set point values while, the Modbus communication on the HMI screen was still ON. This made it even more difficult for the operator to identify the exact nature of fault in the system. Further, a more sophisticated and undetectable MITM attack was also injected on the testbed. This attack altered the set point values as well as real time tank level values, as received by the HMI. This attack therefore caused the tank levels reach to different values than intended by the operator, while still showing tank level on the HMI as expected by the operator. This made the attack undetectable by the plant operator and is therefore the most dangerous attack developed during this thesis.

Three types of DoS attacks were also performed on the testbed which different in the degree of detectability of the attacker. In all DoS attacks, the controller of the plant became unresponsive, resulting in broken controller-HMI communication. In the standard DoS attack using TCP SYN flooding, the attacker Internet Protocol (IP) address was not hidden and hence could be easily detected by the operator. This attack therefore can be easily stopped by the plant operator by blocking the IP address of the attacker. However, in the unidentifiable DoS attacks, we launched the DoS attack using various fake IP addresses and port addresses which allowed the attacker to hide its identity. By using multiple random IP addresses, the attack was made more difficult to shield against. This concluded our cyber attack generation and testing on the simulated testbed for this thesis.

We also observed some future developments that can be implemented in this project. Firstly, the ICS testbed that has been designed during this thesis needs to be implemented as soon as the required components are delivered to the TU Delft NERD lab. Secondly, the simulated testbed can be further improved if software PLCs are used to in place of controller developed in Matlab. Using such industrial softwares will make the testbed replicate the real-world ICS scenario even more closely than the present arrangement. During the process of creating the simulated testbed for this thesis, we aimed to utilize only the open source and free softwares

that would not require us to buy additional licences, as our main goal was to create a testbed using actual hardware PLCs with Beckhoff. Additionally, the testbed can in future be utilized to launch cyber attacks on more complex and more complicated plants. Further, an encryption mechanism can be implemented in the testbed to study protection aspects from such cyber attacks.

In future, the testbed can be extended to generate and test cyber attacks on ICS working on other industrial communication protocols such as EtherCAT, Profinet, etc. To achieve this, we have already made a provision while designing the testbed to add EtherCAT communication to the Beckhoff PLC. Therefore, in future, this testbed can also be used to test and generate attacks against other communication protocols. Lastly, in future, this testbed can be extended in its capabilities to detect and protect against these cyber attacks. One of the two PLCs that we have in each setup can be then used for this detection and prevention from cyber attack purpose. This concludes our Master's thesis report.

Appendix A

Data sheets for testbed components

On the links below, the data sheets for all components from Beckhoff including the CPU, analog and digital input and output cards, power supply and EtherCAT coupler are given.

All component data sheets

Beckhoff EL1008 digital input card

Beckhoff EL2008 digital output card

Beckhoff EL3004 analog input card

Beckhoff EL4004 analog output card

Beckhoff EK1100 coupler

Beckhoff power supply

Appendix B

Converted code for model on dSPACE MicroAutoBox

On the link below, the C code converted from Simulink model necessary to run the plant model on dSPACE MicroAutoBox is given.

Converted model for dSPACE simulator

Bibliography

[1] Eric D. Knapp and Joel Thomas Langill. Chapter 3 - industrial cyber security history and trends. In *Industrial Network Security*, pages 41 – 57. Syngress, Boston, second edition, 2015. http://www.sciencedirect.com/science/article/pii/B9780124201149000034; accessed 17 January 2021.

[2] Modbus Organization Inc. Modbus application protocol specification v1.1b3. 04 2012. https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf; accessed 18 December 2020.

[3] Peter Huitsing, Rodrigo Chandia, Mauricio Papa, and Sujeet Shenoi. Attack taxonomies for the modbus protocols. *International Journal of Critical Infrastructure Protection*, 1:37–44, 12 2008.

[4] Modbus Organization Inc. Modbus messaging on tcp/ip implementation guide. V1.0b, 10 2006. https://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf; accessed 18 December 2020.

[5] Rajwinder Kaur, Gurjot Singh, and Suman Khurana. A security approach to prevent arp poisoning and defensive tools. *International Journal of Computer and Communication System Engineering 2312- 7694*, Volume 2:Pages 431–437, July 2015.

[6] Catalin Tudose. Three way handshake:building java client/server applications with tcp. https://www.luxoft-training.com/news/building-java-client-server-applications-with-tcp/, July 2021. Online; accessed 7 July 2021.

[7] Zouheir Trabelsi and Walid Ibrahim. A hands-on approach for teaching denial of service attacks: A case study. *Journal of Information Technology Education: Innovations in Practice*, Volume 12:Pages 299–319, January 2013.

[8] Emrah Korkmaz, Andrey Dolgikh, Matthew Davis, and Victor Skormin. Industrial control systems security testbed. June 2016. in MILCOM-2016 IEEE Military Communications Conference.

[9] B. Chen, N. Pattanaik, A. Goulart, K. L. Butler-purry, and D. Kundur. 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, 2015.

[10] Steffen Pfrang, Jörg Kippe, David Meier, and Christian Haas. Design and architecture of an industrial it security lab. In Song Guo, Guiyi Wei, Yang Xiang, Xiaodong Lin, and Pascal Lorenz, editors, *Testbeds and Research Infrastructures for the Development of Networks and Communities*, pages 114–123, Cham, 2017. Springer International Publishing.

[11] C. M. Davis, J. E. Tate, H. Okhravi, C. Grier, T. J. Overbye, and D. Nicol. Scada cyber security testbed development. In *2006 38th North American Power Symposium*, pages 483–488, 2006.

[12] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga, and S. Hariri. A testbed for analyzing security of scada control systems (tasscs). In *ISGT 2011*, pages 1–7, 2011.

[13] dSPACE simulator - "1401/1513/1514 Universal high performance I/O". MicroAutoBox II . https://www.dspace.com/en/inc/home/products/hw/micautob/microautobox2.cfm#175_23714. Online; accessed 16 June 2021.

[14] Eric D. Knapp and Joel Thomas Langill. Chapter 3 - industrial cyber security history and trends. In *Industrial Network Security*, pages 13–14. Syngress, Boston, second edition, 2015. http://www.sciencedirect.com/science/article/pii/B9780124201149000034.

[15] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. A taxonomy of cyber attacks on scada systems. In *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pages 380–388, 2011.

[16] Richard Candell, Keith Stouffer, and Dhananjay Anand. A cybersecurity testbed for industrial control systems in conference: 2014 process control and safety symposium, international society of automation at: Houston, tx. pages Pages 2–4, October 2014.

[17] Marco Roscini. Cyber attack definition. In *Cyber Operations and the Use of Force in International Law : Chapter 1*, page 17. Oxford University Press, Oxford, UK, first edition, 2014.

[18] Jill Slay and Michael Miller. Lessons learned from the maroochy water breach. volume 253, pages 73–82, March 2007. in International Federation for Information Processing Digital Library; Critical Infrastructure Protection.

[19] Andrew Ginter. The top 20 cyberattacks on industrial control systems. volume 1:Pages 3–26, May 2018. publisher : Waterfall Security Solutions.

[20] Mustafa Balat. The case of baku-tbilisi-ceyhan oil pipeline system: A review. *Journal : Energy Sources, Part B: Economics, Planning, and Policy*, volume 1(2):pages 117–126, 2006.

[21] Alessandro Di Pinto, Younes Dragoni, and Andrea Carcano. Triton: The first ics cyber attack on safety instrument systems. In *Proc. Black Hat USA*, volume 1, pages 1–26, 2018.

[22] Robert M Lee, Michael J Assante, and Tim Conway. German steel mill cyber attack. *Journal : Industrial Control Systems*, Volume 30:Pages 62–63, 2014.

[23] Thomas M. Chen. Stuxnet, the real start of cyber warfare? [editor's note]. *Journal : IEEE Network*, Volume 24(6):Pages 2–3, 2010.

[24] Roberto Setola, Luca Faramondi, Ernesto Salzano, and Valerio Cozzani. An overview of cyber attack to industrial control system. *Journal : Chemical Engineering Transactions*, Volume 77:Pages 907–912, 2019.

[25] Katalin Ferencz, József Domokos, and Levente Kovacs. Review of industry 4.0 security challenges. In *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 245–248, 2021.

[26] Kevin E. Hemsley and E. Fisher. A history of cyber incidents and threats involving industrial control systems. part of conference: Staggs j., shenoi s. (eds) critical infrastructure protection xii. iccip 2018. ifip advances in information and communication technology, vol 542. springer, cham. 542, 12 2018.

[27] Global programmable logic controller (plc) sales market report 2020. https://www.industryresearch.co/global-programmable-logic-controller-plc-sales-market-16678805. Online; accessed 8 December 2020.

[28] Carlos Queiroz, Abdun Mahmood, Jiankun Hu, Zahir Tari, and Xinghuo Yu. Building a scada security testbed. pages 357–364, January 2009. Journal : NSS 2009 - Network and System Security ; doi = 10.1109/NSS.2009.82.

[29] Haihui Gao, Yong Peng, Kebin Jia, Zhonghua Dai, and Ting Wang. The design of ics testbed based on emulation, physical, and simulation (eps-ics testbed). In *2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 420–423, 2013.

[30] Thomas W. Edgar and David O. Manz. Chapter 13 - instrumentation. In Thomas W. Edgar and David O. Manz, editors, *Research Methods for Cyber Security*, pages 321–344. Syngress, 2017.

[31] CISA : Official website of US Government". Understanding Denial-of-Service Attacks | CISA, November 2019. https://us-cert.cisa.gov/ncas/tips/ST04-015; accessed 30 July 2021.

[32] Modbus Organization Inc. Modbus organization replaces master-slave with client-server : Press release. page 1, July 2020. https://modbus.org/docs/Client-ServerPR-07-2020-final.docx.pdf; accessed 06 August 2021.

[33] Modbus Organization Inc. Modbus over serial line specification and implementation guide. V1.02, December 2006. https://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf; accessed 18 December 2020.

[34] Wireshark : Capturing live data packets. https://www.wireshark.org/docs/wsug_html/#ChapterCapture. Online; accessed 11 January 2021.

[35] The Linux Foundation. Open vSwitch documentation — Open vSwitch 2.16.90 documentation. https://docs.openvswitch.org/en/latest/intro/install/documentation/. Online; accessed 1 August 2021.

[36] The MathWorks, Inc. IEC 61131-3 Code Generation - MATLAB Simulink. https://www.mathworks.com/help/plccoder/code-generation-1.html; accessed 2 August 2021.

[37] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control*, 58(11):2715–2729, 2013.

[38] Yan Hu, Hong Li, Tom H Luan, An Yang, Limin Sun, Zhiliang Wang, and Rui Wang. Detecting stealthy attacks on industrial control systems using a permutation entropy-based method. *Journal : Future Generation Computer Systems*, Volume 108:Pages 1230–1240, 2020.

[39] Michael Liljenstam, Jason Liu, David M. Nicol, Yougu Yuan, Guanhua Yan, and Chris Grier. RINSE: the real-time immersive network simulation environment for network security exercises (extended version). *SAGE Journals*, Volume 82(Issue-1):Pages 43–59, 2006.

[40] B. Chen, K. L. Butler-Purry, A. Goulart, and D. Kundur. Implementing a real-time cyber-physical system test bed in rtds and opnet. In *2014 North American Power Symposium (NAPS)*, pages 1–6, 2014.

[41] Roxanne Brooks. "virtual ics test bed" (2018). Master's thesis, Graduate Theses and Dissertations. 16553, IOWA State University, 2018.

[42] Emrah Korkmaz, Andrey Dolgikh, Matthew Davis, and Victor Skormin. Ics security testbed with delay attack case study in milcom 2016 - 2016 ieee military communications conference (milcom). pages 283–288, November 2016. in MILCOM-2016 IEEE Military Communications Conference.

[43] Ettercap official website. https://www.ettercap-project.org/about.html. Online; accessed 30 July 2021.

[44] Ettercap explanation for arp spoorfing and mitm attacks. http://openmaniak.com/ettercap.php. Online; accessed 11 November 2020.

[45] Libmodbus – v2.1+ licensed modbus tcp implementation. http://libmodbus.org/documentation/. Online; accessed 25 November 2020.

[46] Andreas Paul, Franka Schuster, and Hartmut König. Towards the protection of industrial control systems – conclusions of a vulnerability analysis of profinet io. In Konrad Rieck, Patrick Stewin, and Jean-Pierre Seifert, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 160–176, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[47] Hping - Active Network Security Tool : TCP Flooding. http://wiki.hping.org/109. Online; accessed 28 May 2021.

[48] Communicate Between a TCP/IP Client and Server in MATLAB - MATLAB Simulink. https://www.mathworks.com/help/instrument/communicate-between-a-tcpip-client-and-server-in-matlab.html. Online; accessed 16 December 2021.

[49] M. Changela and Ankit Kumar. Designing a controller for two tank interacting system. *International Journal of Science and Research (IJSR)*, Volume 4 Issue 5:Pages 589–593, May 2015.

Glossary

List of Acronyms

TU Delft	Delft University of Technology
ICS	Industrial Control Systems
PLC	Programmable Logic Controller
LAN	Local Area Network
HMI	Human Machine Interface
TCP	Transmission Control Protocol
IIoT	Industrial Internet-of-Things
NERD	Networked Embedded Robotics in Delft
MITM	Man-In-The-Middle
DoS	Denial-of-Service
RTU	Remote Terminal Unit
IP	Internet Protocol
MBAP	Modbus Application Protocol
IO	Input and Output
ARP	Address Resolution Protocol
MAC	Media Access Control
PDU	Protocol Data Unit
SITL	System-In-The-Loop
DDoS	Distributed Denial of Service
VPN	Virtual Private Network
SCADA	Supervisory Control and Data Acquisition
PC	Personal Computer
CPU	Central Processing Unit

OS	Operating System
MPC	Model Predictive Controller
SYN	Sync
ACK	Acknowledgement
LRC	Longitudinal Redundancy Checking
CRC	Cyclical Redundancy Checking
RINSE	Real-time Immersive Network Simulation Environment
ASPS	Autonomic Software Protection System
RTDS	Real Time Digital Simulator