Fakultät für Elektrotechnik und Informationstechnik
Technische Universität München

Security Challenges and Building Blocks for Robust Industrial Internet of Things Systems

Matthias Niedermaier

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr.-Ing. Wolfgang Kellerer

Prüfende der Dissertation:
1. Prof. Dr.-Ing. Georg Sigl
2. Prof. Dr.-Ing. Dominik Merli,
   Hochschule Augsburg

Die Dissertation wurde am 15.01.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 23.04.2020 angenommen.

Abstract

Digitalization is an ongoing process in home automation, automotive, and industrial processes. In the home environment, it is not surprising anymore that almost everything can now be controlled remotely. However, this trend can also be observed in the industrial environment, where systems are no longer controlled and monitored on-site, but can be done almost from anywhere in the world by using tablets and smartphones. Furthermore, in the course of predictive maintenance, where servicing is performed before damage causes outages, sensors are analyzed in industrial plants and the data is transmitted over the Internet to the vendor. Consequently, industrial components are getting increasingly connected and remotely accessible. This higher connectivity also enlarges the attack surface, because now attackers have new possibilities which did not exist in the times of air-gaped industrial plants.

Overall, in the first part of this work, problems of current Industrial Control System (ICS) in terms of robustness will be shown and potential solutions are demonstrated. In the second part, further concepts and future building blocks for increasing IT security in ICS will be introduced.

What makes today's industrial plants and devices attractive to attackers is the interaction between the network side and the real world, which does not exist in this form in the standard IT landscape. This is exactly what is addressed in this work, by highlighting the problems of current control architectures in terms of robustness and thus the related dependency of the control side on the network side. To this end, a testbed is built and used, making the effects of network traffic on the electrical side and therefore the real world measurable. Fuzzing and Denial of Service attacks are possible ways of attacking control systems. The problems shown here can be solved, for example, by secure architectures. One possibility is the feedback free separation of the communication processor from the control processor analyzed in this work. Moreover, a safe way to scan an industrial network is presented. This basic security monitoring tool allows initial assessments about the network, whether they contain vulnerable devices or not.

In addition, new concepts for increasing security in industrial plants are presented in this work. New and open components are introduced that can be used for research, evaluation, and teaching. Furthermore, an Intrusion Detection System is shown, which runs on low-performance industrial edge node devices. The advantage of this is that each device can decide for itself and no additional hardware is needed. Also, a concept of an active network scanner on edge node devices is presented. This allows a simple network scan in each subnetwork having such an edge node device. Additionally, an open and low-cost testbed is being introduced to research and train industrial security, without disrupting or damaging a productive ICS. This testbed also includes a small physical process to monitor the impact of cyber-attacks in the real world.

Kurzfassung

Die Digitalisierung ist ein fortlaufender Prozess, der bereits Heimautomatisierung, Fahrzeuge und industriellen Anlagen erreicht hat. Im heimischen Bereich ist es nicht mehr verwunderlich, dass inzwischen fast alles von unterwegs gesteuert werden kann. Dieser Trend ist jedoch auch im industriellen Umfeld zu beobachten, wo Systeme nicht mehr vor Ort gesteuert und überwacht werden, sondern dies auch aus der Ferne auf Tablets und Smartphones geschieht. In Zeiten von Predictive Maintenance, bei der die Wartung vor einem Ausfall durchgeführt wird, werden Sensoren in Industrieanlagen analysiert und die Daten an den Hersteller übertragen. Dies ist nur ein Grund dafür, dass industriellen Komponenten immer mehr miteinander vernetzt werden. Diese höhere Konnektivität erhöht jedoch auch die Angriffsfläche, da Angreifer jetzt neue Möglichkeiten haben, die es zu Zeiten der klaren logischen Abtrennung von Industrieanlagen nicht gab.

Zusammenfassend werden im ersten Teil Probleme aktueller industrieller Systeme, wie die Robustheit demonstriert und mögliche Lösungen aufgezeigt. Im zweiten Teil werden weitere Konzepte zur Erhöhung der IT-Sicherheit in industriellen Systemen vorgestellt.

Was Industrieanlagen und -geräte für Angriffe so attraktiv macht, ist das Zusammenspiel von der Netzwerkseite mit der realen Welt, was es in dieser Form in der Standard-IT-Landschaft nicht gibt. Genau hier setzt diese Arbeit an, indem Probleme aktueller Steuerungsarchitekturen in Bezug auf Robustheit und Abhängigkeiten zwischen digtaler und realer Welt aufgezeigt und Lösungsvorschläge unterbreitet werden. Zu diesem Zweck wird eine Testumgebung aufgebaut und verwendet, die die Auswirkungen des Netzwerkverkehrs auf der elektrischen Seite und damit auf die reale Welt messbar macht. Fuzzing- und Denial of Service-Angriffe sind eine Möglichkeit, eine industrielle Steuerung anzugreifen. Die hier gezeigten Probleme können beispielsweise von sicheren Architekturen gelöst werden. Eine Möglichkeit ist die rückwirkungsfreie Trennung des Kommunikationsprozessors vom Steuerprozessor, die in dieser Arbeit aufgezeigt und analysiert wird. Darüber hinaus wird eine sichere Möglichkeit zum Scannen eines industriellen Netzwerks vorgestellt, durch die eine Einschätzung über das Netzwerk getroffen werden kann.

Ebenfalls werden in dieser Arbeit weitere Konzepte zur Erhöhung der Sicherheit in Industrieanlagen vorgestellt. Es werden neue und offene Komponenten gezeigt, die für Forschung, Evaluierung und Lehre genutzt werden können. Darüber hinaus wird ein Intrusion Detection System entwickelt und analysiert, welches auf Edge-Node-Geräten mit geringer Leistung genutzt werden kann. Außerdem wird ein Konzept eines aktiven Netzwerkscanners auf industriellen Edge-Node-Geräten vorgestellt, was einen einfachen Netzwerk-Scan in jedem Teilnetzwerk ermöglicht. Des Weiteren wird ein offener und kostengünstiger Demonstrator für Forschung und Lehre in Bezug auf IT-Sicherheit in industriellen Anlagen präsentiert. Dazu gehört auch ein physikalischer Prozess, um die Auswirkungen von Cyberangriffen in der realen Welt darstellen zu können.

Contents

List of Figures

List of Tables

Acknowledgements

I gratefully acknowledge the support, guidance, and advice of my supervisor Prof. Dr.-Ing. Georg Sigl, and of my second examiner and mentor Prof. Dr.-Ing. Dominik Merli.

The number of people and companies who have supported me is so large that it is unfortunately not possible to mention them all.

At the Hochschule Augsburg, University of Applied Sciences, I received a lot of technical and organizational help from Prof. Dr. Gordon Thomas Rohrmair, Prof. Dr. Alexander von Bodisco, Prof. Dr. Jürgen Scholz, Florian Fischer, Thomas Hanka, Felix Sauer, Susanne Kießling, Robert Happacher, and Peter Knauer, among many others.

Additionally, I met a lot of fantastic people at Fraunhofer AISEC, including Dr. Sven Plaga, Dr.-Ing. Johannes Obermaier, Alexander Giehl, Martin Striegel, Gerhard Hansch, and Carsten Rolfes. Besides, I was kindly integrated by Dr.-Ing. Matthias Hiller, Dr.-Ing. Michael Pehl, and Florian Wilde at the TU München.

In the last few years, my family, my sweetheart and friends could not help wondering what I was doing the whole day. Now we are at the end of the journey, and they probably still do not know what I am doing. But I would like to thank for their unwavering support and sympathy.

My PhD research was supported by the BayWISS Consortium Digitization, which I would like to acknowledge.

Finally, I would like to say **THANK YOU** to all of you for the great support!

This thesis was supported by

BayWISS
Bayerisches Wissenschaftsforum

Hochschule Augsburg
University of Applied Sciences

Chapter 1

Introduction

Contents of this chapter

It is fair to say that we are living in a digitized world. Digitization is no longer unique to classic IT companies, it happens at companies across all sectors and in the home environment. It is not unusual to control the heating or to start the cleaning robot on the way home. However, not only these areas but also industries are connected, and thus entire factories are getting interconnected. Gartner Inc. forecasts 20 billion Internet of Things (IoT) devices in 2020 [Meu17]. As can be seen in Figure 1.1, this number is not only compound out of "consumer" devices, but also devices of the "business: cross-industry" and the "business: vertical-specific" market.

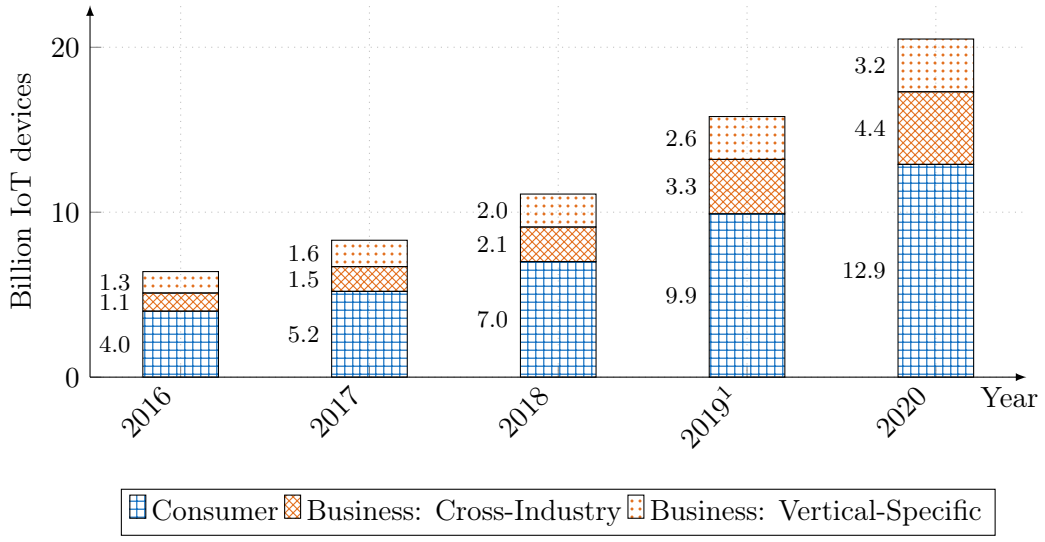


Figure 1.1: IoT units installed based on category [Meu17].

[1] 2019 figures are an interpolation of 2018 and 2020.

To be exact, Gartner Inc. says there will be 12.9 billion IoT devices in the consumer segment and 7.6 billion devices in the business area in 2020. These numbers also include connected industrial equipment. Furthermore, it can be observed that, from 2016 to 2020, there will be more than three times the number of IoT devices, both in the consumer and in the business sector.

Unfortunately, due to this high level of networking and the increased number of IoT devices, the attack surface is also expanding. Especially in an industrial environment, this allows attacks that were not possible in the past, as classic industrial plants often ran independently and were not connected to the outer world. The isolation, with no physical communication path from the industrial plant to the outer world, is therefore called "air gapped". This separation is often no longer possible due to new demands such as predictive maintenance. With predictive maintenance, sensor data of machines is analyzed in order to perform maintenance before a standstill. This exchange of sensor data is only one example of an Industry 4.0 scenario, which requires a higher connectivity and make the historical air gapped segmentation hardly possible.

For example, industrial plants can be attacked if there are misconfigurations, vulnerable jump hosts, vulnerabilities in the industrial product itself, or other exploitable opportunities for attackers. Industrial Control System Computer Emergency Response Team (ICS-CERT), which is a part of the Department Homeland Security, publishes annual reports on incidents, vulnerabilities and more. Figure 1.2 shows the numbers of vulnerability reports of the "Year in Review" report from 2012 [Ind12] 2015 [Ind15] and 2016 [Ind16]. Noticeable here is the rapid increase after 2010. This was probably due to the increasing popularity of ICS and Supervisory Control and Data Acquisition (SCADA) security in academic research, e.g. Beresford [Ber11] and attacks on plants like *Stuxnet* [Lan11; Kar11] both published in 2011. However, this graph shows that many products have vulnerabilities that attackers can exploit.

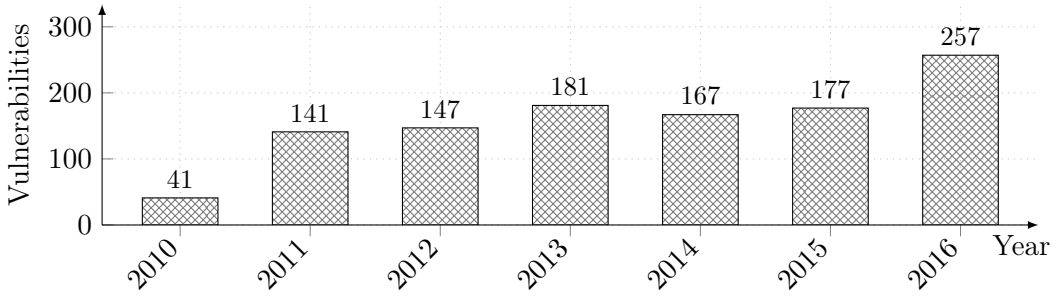


Figure 1.2: ICS related vulnerability reports – Tickets from ICS-CERT [Ind12; Ind15; Ind16].

The fact that these vulnerabilities are actively exploited in the wild is shown by security incidents in industrial plants in recent years. One of the best known of these is the *Stuxnet* worm, which was uncovered in June 2010 [Kar11] as attacking industrial control devices that controlled centrifuges for separating nuclear material. The plant was completely isolated from the outside world (air gaped). Therefore, the infection probably

took place via a USB stick or a portable programming device. Subsequently, some of the uranium centrifuges were destroyed, delaying or partially preventing uranium enrichment. In 2014 there was a cyber-attack against a *German steel mill* [Lee+14] that was initialized by spear phishing emails. As a result, a system could not shut down correctly, causing it to be damaged. *BlackEnergy 3* malware was used in a cyber-attack against an Ukrainian power grid in December 2015 [Lee+16]. The attackers used phishing emails and manipulated Microsoft Office documents to get access to the Information Technology (IT) network. The attackers fought their way "down" to the control network. A total of three electricity suppliers and around 225,000 customers were affected by this attack. Another attacking tool is the *Triton* framework, which aims for a safety control unit from Schneider Electric (Triconex™ Safety Instrumented System [Mil+19]). In attacking this system with Triton, safety-related functions were bypassed, leading to the fail state and an automatic shutdown of the industrial process. Considering that these systems control safety-relevant processes in which humans, for example, interact with machines, this poses a high risk, even to human lives.

Many attacks that have become known in recent years use spear phishing attacks to enter the company [Fed13]. In this manner, a kind of "gateway" is established on a computer in the company, where the network is scanned and other systems are getting infected. Once the attackers have reached the actual target system, they can manipulate the system or exfiltrate information. When the attackers have reached the desired goal, they mostly try to hide attack paths to remain undetected. All these examples show that especially in the Industrial Internet of Things (IIoT) environment attacks can have a big impact on the real world. This is something that distinguishes classic office IT from industrial systems. Impacts, particularly on critical infrastructures, can concern not just single companies but also complete populations. As the above examples show, the industrial devices were not attacked directly, but through a standard computer that became infected. If the computer was then infected, the industrial devices did not provide adequate protection and could be taken over by the attackers. This means that due to the usually very weak security level, the controllers are often vulnerable if a network connection exists.

Another attack vector is not using gateways and jump hosts, but the direct path. So-called jump hosts are hardened machines, which are used to access a network via the jump host from a network with a different security level. Nevertheless, this is not always used and many industrial plants are connected to the Internet for reasons of misconfiguration, remote maintenance, and other scenarios that require a remote connection.

These can be found using a specialized search engine for SCADA systems. For example, *Censys* [Dur+13] lists about 80 000 unique devices tagged "SCADA" and is partly open source. Another search engine for IoT and IIoT devices is *Shodan* [Mat09; Bod+14], which was one of the first search engines of its kind. Lesser known is *ZoomEye* [Kno16] and *Ditecting* [Dit15]. *Shodan*, *ZoomEye* and *Ditecting* are mostly closed source and less is known about the search engine mechanisms. Table 1.1 shows current scan results of these search engines on January 3, 2020. The used queries for the search engine can be found in Appendix A.1.

It is noticeable that the number of devices found varies. According to *Censys* [Cen], the different number of devices found is partly due to the fact that *Censys* is significantly more aggressive when deleting entries than *Shodan*. For example, *Censys* will delete devices that were not found in the last week. However, the *Censys* search engine uses a distributed approach and is currently less known than *Shodan*, which means that *Censys* can access more devices (distributed approach) and is also blocked by fewer users (lesser-known). *ZoomEye* is not focused on ICS and does not interpret some protocols correctly, which leads to a significantly higher number of findings only based on the open port. Tundis et al. [Tun+18] analyzed different network vulnerabilities scanning tools, including *Censys* and *Shodan*. They also have differences in the number of devices found and could not find a complete explanation, since most search engines are partially or even completely proprietary, e.g. *Ditecting*. It should also be mentioned that it is generally difficult to make a precise statement about the number of devices, since small changes in e.g. dynamic Internet Protocol (IP) ranges can cause large variations. However, it is possible in any case to get an overview, that many industrial devices are accessible from the Internet.

Table 1.1: Internet-facing ICS devices till January 3, 2020 (used queries in Appendix A.1).

Protocol	**Port**	**Shodan**	**Censys**	**ZoomEye**	**Ditecting**
Modbus	502 TCP	20 648	28 171	23 732	14 173
Siemens S7	102 TCP	22 365	4 994	31 761	2 464
DNP3	20000 TCP	436	429	61 490	367
BACnet	47808 TCP	18 225	16 882	49 913	11 750
Niagara Fox	1911 TCP	32 351	27 879	120 429	26 634
Ethernet/IP	44818 TCP	48 506	–*	12 326	1 971
Phoenix PCWorx	1962 TCP	831	–*	3 881	168
Codesys	2455 TCP	2 568	–*	8 105	2 455
	Total:	145 930	78 355	311 637	59 982

*Currently not scanned/available.

Given that some of these systems control manufacturing processes and critical infrastructures and, as previously pointed out, have vulnerabilities, there is an urgent need for action. On the one hand, industrial control system devices need to be better tested to reveal and close vulnerabilities. On the other hand, control networks must be better managed, monitored, and protected against attacks. Furthermore, open components must be developed that simplify and enhance research and teaching in this area. In this work, these points are examined and suggested solutions are presented.

1.1 Contribution

In this section, special contributions of this work are highlighted.

1.1.1 Addressed Research Questions

In the following, the research questions addressed in this thesis are introduced. These research questions represent the framework that guided this work, but do not fully reflect all the results of this work.

How can the robustness of industrial components be evaluated?
One of the first questions, that come up when dealing with the robustness of industrial control systems, is the question of how to make this measurable. A big challenge here is to measure the proprietary and different systems with a similar and comparable procedure. (addressed in Section 3.1, Section 3.2, and Section 3.3)

What are secure architectures for robust ICSs devices?
A further challenge is to provide secure architectures for devices and network scanners in order to be prepared for future requirements of secure connectivity of Programmable Logic Controllers (PLCs). (addressed in Section 3.4 and Section 3.5)

What will future building blocks for IIoT devices look like?
Another research question is how the security of IIoT devices can be increased by building blocks (features, which increase the security level) and what side effects these can cause in ICSs. What is meant here in detail, it is analyzed which attack mechanisms the IT security modules protect against and which restrictions result from them. (addressed in Section 3.4, Section 4.3, Section 4.2, and Section 4.4)

How can open source help facilitate ICS security research and education?
In order to conduct device-level research, it needs open-source components to make changes. Furthermore, testbeds are also important for teaching to understand ICSs. Here, a big challenge is that most of the purchasable components are closed-source. (addressed in Section 3.5, Section 4.1, and Section 4.4)

1.1.2 Scientific Publications

During the time working at this thesis, several scientific publications were published. This section provides a short description of publications and information about the corresponding conferences. These are the published scientific papers used in this thesis, sorted by release date, with the newest first.

"Efficient Intrusion Detection on Low-Performance Industrial IoT Edge Node Devices" – Matthias Niedermaier, Martin Striegel, Felix Sauer, Dominik Merli and Georg Sigl – arXiv preprint 2019 [Nie+19c].
Summary: A method for intrusion detection that combines distributed agents on IIoT edge devices with a centralized logging is introduced. Additionally, a Proof of Concept (PoC) implementation on an Microcontroller Unit (MCU) running *FreeRTOS* with *LwIP* is demonstrated and evaluated.

"LICSTER – A Low-cost ICS Security Testbed for Education and Research" – Felix Sauer, Matthias Niedermaier, Susanne Kießling and Dominik Merli – 6th International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR) 2019 [Sau+19].
Summary: In this paper the "LICSTER" testbed is introduced, which is an open-source low-cost ICS testbed, enabling researchers and students to get hands-on experience with industrial security for about 500 euros.

"Network Scanning and Mapping for IIoT Edge Node Device Security" – Matthias Niedermaier, Florian Fischer, Dominik Merli and Georg Sigl – Proceedings of the IEEE 24th International Conference Applied Electronics (IEEE AE) 2019 [Nie+19a].
Summary: A network scanning and mapping building block to scan directly from IIoT edge node devices is presented. This enables the detection of unwanted devices in an industrial network.

"A Secure Dual-MCU Architecture for Robust Communication of IIoT Devices" – Matthias Niedermaier, Dominik Merli and Georg Sigl – 8th Mediterranean Conference on Embedded Computing (IEEE MECO) 2019 [Nie+19b].
Summary: A setup to ensure a resilient controlling for IIoT devices like PLCs is shown. This is a possible solution for the demand of secure architectures in the IIoT. Moreover, a PoC implementation with a benchmark and a comparison with a standard PLC is presented.

"Future Proofing IoT Embedded Platforms for Cryptographic Primitives Support" – Sven Plaga, Norbert Wiedermann, Matthias Niedermaier, Alexander Giehl and Thomas Newe – 12th International Conference on Sensing Technology (IEEE ICST) - Wireless Sensor Networks 2018 [Pla+18].
Summary: The presented approach enables designers and developers of embedded systems to achieve comparable results over an extended range of algorithms and implementations for the usage cryptographic functions in IoT and IIoT devices.

"Efficient Passive ICS Device Discovery and Identification by MAC Address Correlation" – Matthias Niedermaier, Thomas Hanka, Sven Plaga, Alexander von Bodisco and Dominik Merli – 5th International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR) 2018 [Nie+18b].
Summary: In this paper, a lightweight passive network monitoring technique using an efficient Media Access Control (MAC) address-based identification of industrial devices is proposed. This is based on an incomplete set of known MAC addresses to device associations. The presented method can guess correct device and vendor information. Proving the feasibility of the method, an implementation is also introduced and evaluated regarding its efficiency.

"You Snooze, You Lose: Measuring PLC Cycle Times under Attacks" – Matthias Niedermaier, Jan-Ole Malchow, Florian Fischer, Daniel Marzin, Dominik Merli, Volker Roth and Alexander von Bodisco – 12th USENIX Workshop on Offensive Technologies (USENIX WOOT) 2018 [Nie+18c].
Summary: This work demonstrates that the electrical side of a PLC, i.e., the controlled process, can be influenced by packet flooding the network side. This differs from already known Denial of Service (DoS) attacks as the target is the actual process and not the network connectivity.

"CoRT: A Communication Robustness Testbed for Industrial Control System Components" – Matthias Niedermaier, Alexander von Bodisco and Dominik Merli – 4th International Conference on Event-Based Control, Communication and Signal Processing (IEEE EBCCSP) 2018 [Nie+18a].
Summary: In this paper, a testbed and measurement methods for communication robustness test research of ICS components is presented.

"PropFuzz – An IT-Security Fuzzing Framework for Proprietary ICS Protocols" – Matthias Niedermaier, Florian Fischer and Alexander von Bodisco – 22nd International Conference Applied Electronics (IEEE AE) 2017 [Nie+17].
Summary: A new fuzzing framework, called PropFuzz, which can fuzz proprietary ICS protocols and monitor the behavior of the controller is introduced. Furthermore, the first results of a security assessment with the framework are presented.

1.1.3 Revealed Vulnerabilities, Assigned Common Vulnerabilities and Exposures, and Published Advisories

While working on this dissertation, several vulnerabilities in products were found. These vulnerabilities were always reported to the vendors and not made public before the 90-day responsible disclosure timeline.

Common Vulnerabilities and Exposures (CVE) is an industry standard that aims to introduce a common naming convention for security vulnerabilities and other vulnerabilities in computer systems. Multiple naming of the same threats by different companies and institutions is supplemented by a identifier (eg, CVE-2020-1337) to ensure clear identification of the vulnerability. This number is made out of the year, followed by a consecutive number, with at least four digits. The list of CVEs is managed by the Mitre Corporation [MITa] in collaboration with the CVE numbering authorities, like security professionals, educational institutions, government agencies, and security software vendors.

A security advisory is a public announcement, which informs users of a product about a security problem. This helps users to take appropriate measures, like the installation of security updates or the information of secure configuration methods.

The ICS-CERT is part of the United States Computer Emergency Readiness Team (US-CERT), which is subordinate to the Department of Homeland Security and is specifically responsible for the security of ICSs. The ICS-CERT, co-ordinates vulnerability reports and issues advisories accordingly. In Germany, the Federal Office for Informa-

tion Security (BSI) performs similar tasks. Additionally, the Cyber Emergency Response Team at the Association of Electrical Engineering, Electronics and Information Technology (CERT@VDE) is an IT security platform for the coordination of IT security reports in Germany, especially risks targeting the industrial sector.

Advisories:

- Advisory (ICSA-19-106-03) [ICS19]:
  "PLC Cycle Time Influences" – Matthias Niedermaier, Jan-Ole Malchow and Florian Fischer
- VDE-2018-013 [CER18b]:
  "WAGO 750-8xx Controller Denial of Service" – Matthias Niedermaier, Jan-Ole Malchow and Florian Fischer
- VDE-2018-012 [CER18a]:
  "PHOENIX CONTACT Inline-Controller (ILC) 1×1 ETH Denial of Service" – Matthias Niedermaier, Jan-Ole Malchow and Florian Fischer
- Advisory (ICSA-17-264-04) [ICS17]:
  "Improper Authentication issue in iniNet Solutions iniNet Webserver" – Matthias Niedermaier and Florian Fischer
- Advisory (ICSA-16-313-01) [ICS16]:
  "Phoenix Contact ILC PLC Authentication Vulnerabilities" – Matthias Niedermaier and Michael Kapfer

CVEs:

- CVE-2019-10953 [Nat19]:
  "PLC Cycle Time Influences" – Matthias Niedermaier, Jan-Ole Malchow and Florian Fischer
- CVE-2017-13995 [Nat17]:
  "Improper Authentication issue in iniNet Solutions iniNet Webserver" – Matthias Niedermaier and Florian Fischer
- CVE-2016-8366 [Nat16a]:
  "Phoenix Contact ILC PLC Password Macro Leakage" – Matthias Niedermaier and Michael Kapfer
- CVE-2016-8371 [Nat16b]:
  "Phoenix Contact ILC PLC Access without Authenticating even if the Authentication Mechanism is enabled" – Matthias Niedermaier and Michael Kapfer
- CVE-2016-8380 [Nat16c]:
  "Phoenix Contact ILC PLC Access to Read and Write PLC Variables without Authentication" – Matthias Niedermaier and Michael Kapfer

1.2 Structure of the Thesis

The thesis is structured as follows (see Figure 1.3). In Chapter 2, relevant background information is provided on ICS security, focusing on the interaction between the network and the physical world, as well as other specific requirements in this environment.

Chapter 3 focuses on the communication robustness of ICS components. Section 3.1 introduces the testbed which is used for the measurement and evaluation of proprietary components. Fuzzing related to robustness is handled in Section 3.2. In Section 3.3 discusses the effects of DoS attacks on PLCs. Thereafter, a hardware architecture is presented, which is robust against these attacks due to two separated MCUs (Section 3.4). Additionally, a secure method for scanning in fragile ICS environments is presented in Section 3.5.

Chapter 4 introduces building blocks for secure ICS devices and platforms for further research. More precisely, Section 4.1 presents open-source components and a open testbed. In Section 4.2 an IDS on IIoT edge node devices is shown. Section 4.3 introduces a network scanner for IIoT edge node devices. An open-source testbed for education and research is presented in Section 4.4.

At the end, a conclusion and a discussion of open research questions are given in Chapter 5.

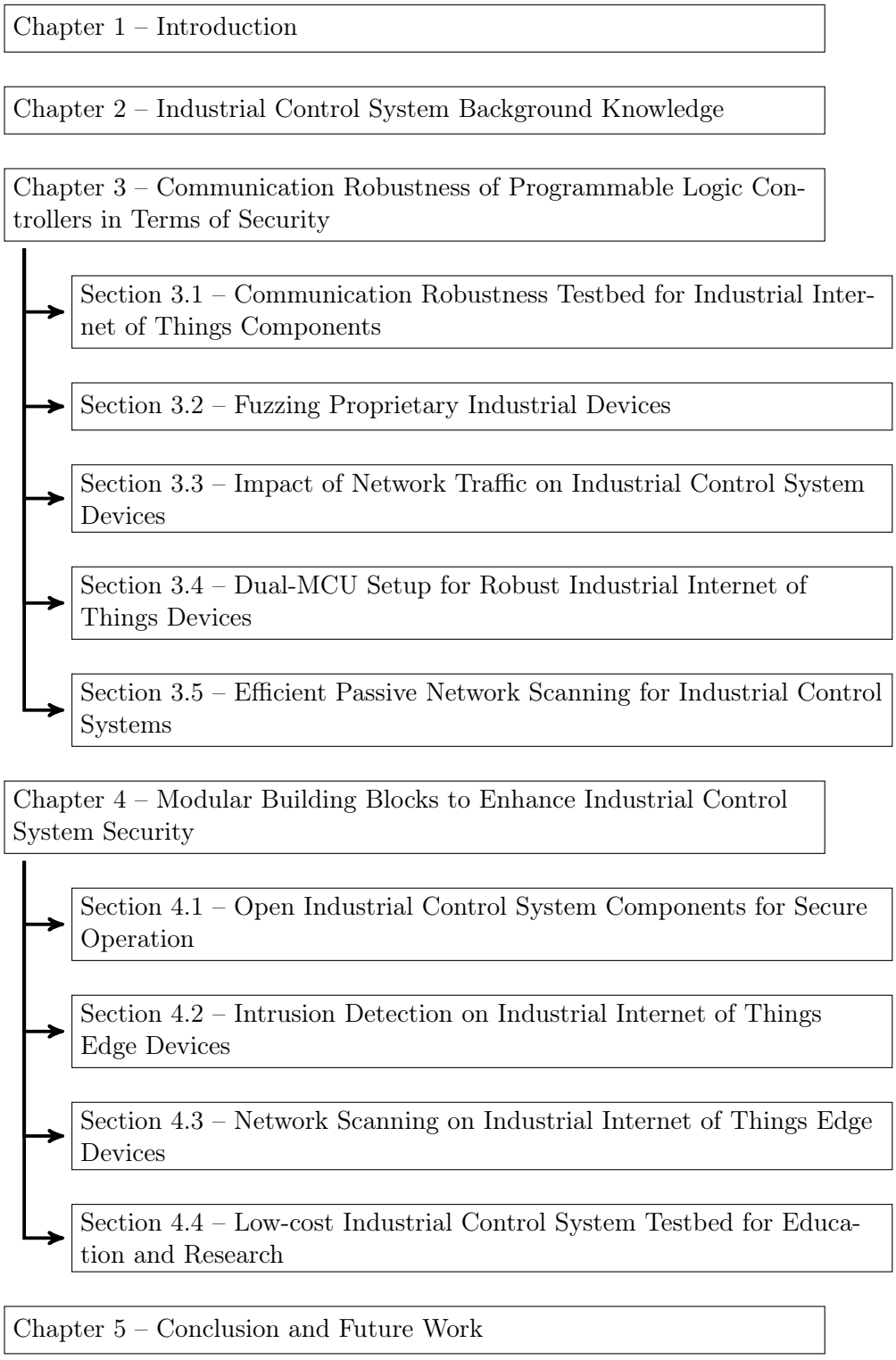


Figure 1.3: Structure and focus of this thesis.

Chapter 2

Industrial Control System Background Knowledge

Contents of this chapter

Industrial Control System (ICS) is the common umbrella term for the various devices used in industrial infrastructures. These devices generally include sensors, actuators, and embedded computers interconnected via network. The basic hierarchical scheme is standardized by IEC 62264 [Int03] and described as automation pyramid illustrated in Figure 2.1 [Kel+09]. Compared to other domains, the individual components of ICSs tend to have a long lifetime, and cycles need to be processed in real time within certain time constraints [Sad+15].

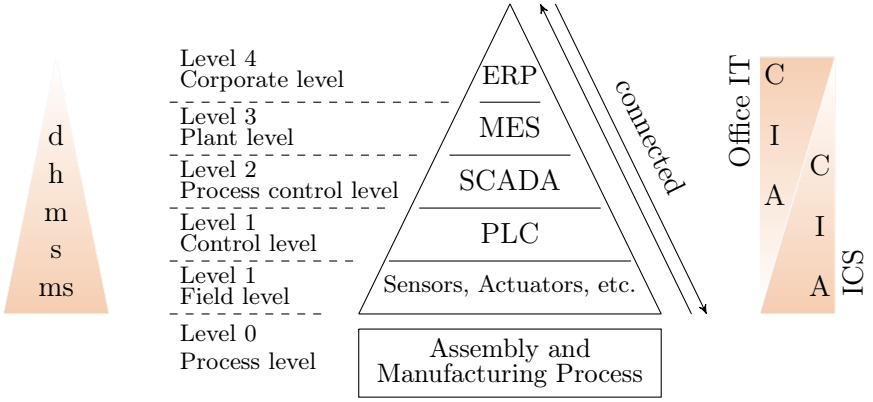


Figure 2.1: IEC 62264 industrial automation pyramid.

Enterprise Resource Planning (ERP) facilitates the entrepreneurial task of planning and controlling resources such as capital, personnel, operating resources, and materials in a timely and appropriate manner. One of the world's largest commercial providers of ERP software is SAP [Lei08]. ERP systems should reflect all business processes and should move away from isolated solutions to a holistic ERP system that can be used to manage resources company-wide. In addition, ERP systems improve the communication flow in the company and, in the sense of e-collaboration, can make cooperation in the company more efficient. A **Manufacturing Execution System (MES)** refers to a process-related level of a multi-layered manufacturing management system. A MES distinguishes itself from similarly effective systems for production planning, the so-called ERP systems, by direct connection to the distributed systems of process automation and enables the management or control of the production. **Supervisory Control and Data Acquisition (SCADA)** defines a system for monitoring and controlling technical processes through a computer system. The data is then presented in a userfriendly way, allowing control to intervene in the process. Additionally, there is often a historian server in the SCADA layer, which stores measurement values and production states over a period of time, so that, among other things, it is possible to generate statistic or find faulty batch. A **Human Machine Interface (HMI)** resides at level 1 of the industrial automation pyramid and allows an operator to interact with a machine or plant. Today, communication within SCADA systems increasingly depend on TCP-based techniques. A **Historian** is a server at level 2, where conditions, input and output information are stored for long-term analysis. A **Programmable Logic Controller (PLC)** is a device that is used to control a machine or plant and that is digitally programmed. **Remote Input/Output (IO)** devices read inputs and write outputs over a fieldbus or network connection.

Figure 2.1 shows, on the left side, the differences in timing requirements of an ICS at the corresponding levels. The devices at higher levels, such as 3 and 4, often have a data exchange time of several hours or days. For SCADA systems, data transfer time has to be in the order of seconds to minutes. This is in contrast to hard real-time processes at the control and field levels, where transmissions must complete in milliseconds. Of course, these timing requirements are heavily dependent on the underlying physical process. For example, temperatures in large tanks change very slowly, but in comparison, the throughput of a water pipe can change very quickly.

As indicated on the right side in Figure 2.1, the classic Confidentiality, Integrity and Availability (CIA) model from office IT is often reversed, so availability is the highest protection goal in ICS systems. In itself, this statement has to be differentiated, as it also depends on the underlying physical process. In the following, these ICS specific singularities will be considered separately in detail.

2.1 Timeline on Industrial Control System Attacks

For a long time, industrial plants were not in the focus of attackers or little was known about successful attacks. In general, the *Stuxnet* Malware, which focus on attacking the

Iranian uranium enrichment program is one of the best-known attacks and ushered in the age of cyber attacks on industrial systems. A survey of publicly available information about cyber attacks on industrial systems [Hem+18] shows, how the number attacks has increased over the recent years. A selection of malware affecting ICS and attacks on ICS is illustrated in Table 2.1. In the course of this, the year of the occurrence of the malware or the attack as well as the name and a brief description are listed.

Table 2.1: Selection of attacks against industrial systems and malware targeting ICS, illustrated in a timeline.

2000	Attack	Maroochy Water	Cyber-attack caused the release of untreated sewage
2010	Malware	Stuxnet	Stuxnet is one of the first and best known malware targeting PLCs
2010	Malware	Night Dragon	Attack targeting energy, oil, and petrochemical companies
2012	Malware	Shamoon	Attack on national oil companies including Saudi Aramco and RasGas
2013	Malware	Havex	With this malware it is possible to control the infected system remotely
2014	Attack	German Steel Mill	Cyber-attack with massive damage
2015	Malware	BlackEnergy	Malware targeting HMIs
2017	Attack	NotPetya	Targeting Ukraine with Ransomware, with no way to decrypt
2017	Malware	TRITON	Malware bypassing safety mechanisms

The first example is the attack on *Maroochy Water* services in Queensland, Australia [Sla+08]. In the case of this attack, a contract worker took revenge for the fact that he did not get a permanent job. The attacker controlled 150 pumping stations with

a laptop and released millions of liters of contaminated water into public water over a period of 3 months.

With *Stuxnet* [Lan11; Kar11], malware that specializes on ICSs has been programmed for the first time. Several vulnerabilities in Microsoft Windows were exploited and valid certificates were used. If an infection occurs, a scanner is installed in the process control system. This searches for and manipulates data packets of the PLCs, which are controlling motors. Attention was drawn to the suspected smuggling of *Stuxnet* into plants of the Iranian uranium enrichment plant "Natanz" in 2010. This assumption is fueled by the specific properties of *Stuxnet* to attack only one specific Siemens process control and only two specific motor controls. These specific component constellation were used at the centrifuges in "Natanz".

Night Dragon [Cyb11] is a mix of different attack techniques. The hacker groups specifically target sensitive data and information from the oil, gas and chemical industries. The focus is particularly on projects and financing plans for oil and gas fields, which, in the hands of criminals, can impact billions of dollars in business and thus affect the energy industry worldwide. Attacks of this type have been known to McAfee since 2010. In the case of these attacks, it is assumed that confidential information has been gathered from PLCs.

Shamoon [Bro+13] is a malware that is aimed exclusively against oil and energy plants in Saudi Arabia. The aim was to sabotage the plants of Aramco and RasGas. For this purpose, central computers in computer systems with *Shamoon* were used to infect other systems connected to the network. In this way it was possible to infect up to 30,000 computers. After *Shamoon* had spread through a network, the actual malicious code was triggered which consisted in deleting critical files and hard disk areas relevant for the operating system. Such damaged computers crash and can no longer be used without manual repairs. This less sensitive and open approach by *Shamoon* sets him apart from the other known incidents. Considering that some systems, SCADA, and PLCs use Microsoft Windows, which is often not up to date, it becomes clear why production was also affected here.

The *Havex* [Nel16] malware is categorized as a Remote Access Trojan (RAT), which makes it possible to bring an infected system under the attackers control and control it remotely. Another task of *Havex* is to read network traffic. Various industries were attacked by *Havex*, especially in Europe and the USA.

Attackers, targeting a *German steel mill* [Lee+14], took control of the furnace and massively damaged the plant. The hackers manage to produce a failure of the entire systems in the plant. Those responsible in the steel mill were no longer able to shut down the furnace. According to the report, the attackers first used fake emails tailored to employees to gain access to the plant's office network and then worked their way into the production networks. In this case, the BSI rates the hackers knowledge as "very advanced" and assumes that their knowledge went far beyond the knowledge of classic IT security and includes specialist knowledge of the ICSs and production processes used. The attack initially caused individual components to fail, ultimately the furnace could no longer be controlled and was in an "undefined state". The employees of the steel mill could no longer shut down the furnace and the entire furnace system was damaged.

The *BlackEnergy* [Lee+16] malware was used on December 23, 2015, which lead to the world's first extensive blackout caused by hackers in Ukraine. The exact details on the attack are not entirely clear. However, is is assumed that they have been able to inject *BlackEnergy* malware into the system of a regional energy provider using prepared office files. To this end, they are said to have sent fake emails to employees of the energy supplier, with the Ukrainian parliament as sender. The content should encourage employees to download the attachment. Anyone who opened the file was informed that the Word version was out of date. In order to read the content, the victim should allow a macro to be executed. Macros are often used to automates certain tasks in many companies. Whoever clicked "activate" then let the *Black Energy* malware into the system. The attackers then find their way down to the ICS level via this "gateway".

NotPetya [Gre19] is malware that targets on windows systems. As more and more industrial controls are also based on windows, they were also affected by *NotPetya*. For example, HMIs from different vendors were affected.

The *Triton* [Dud17] malware is targeting Triconex™ Safety Instrumented System [Mil+19] from Schneider Electric. The malware aim is to shut down the affected systems or to deactivate safety mechanisms so that the system is operated in an unsafe condition. An attack on a refinery of the Saudi Arabian company Tasnee is said to have almost resulted in an explosion after an attack with *Triton*.

All of these attacks and malware show, that the threat situation in the area of industrial IT security is high. Therefore, it is necessary to investigate the security of ICS more closely and to develop protective mechanisms. On the one hand, these must protect the network and, on the other hand, make industrial devices secure by its own.

2.2 Programmable Logic Controller Application

A PLC is an industrial digital computer designed to control physical processes. A PLC is electrically connected to sensors and actuators and mostly with a fieldbus or Ethernet-based network. A user-specific program running on the PLC controls the actuators based on the inputs read from the sensors. Since the majority of PLCs operate in a cycle-oriented fashion, we focus on this type of device. Figure 2.2 shows a simple example application where a PLC controls the filling process of a container on a conveyor belt. The sensor reports to the PLC when a container passes it. The PLC then controls the valve that opens and fills the container. This process must have the right timing, or else the liquid would not end up in the container. If the cycle time is too high, or if the processing is very slow, the opening or closing of the valve gets delayed and occurs at false container positions. A more detailed analysis of PLC cycle time requirement can be found from Electro Cam Corporation [Ele98].

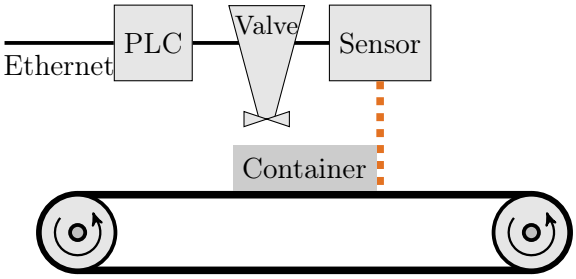


Figure 2.2: Example application where liquid/goods is filled in a container.

This example highlights the peculiarity of Cyber-Physical Systems (CPSs), as it demonstrates a physical process were delays could lead to intolerable failures of the physical process handling. Using a simple process as described here, many damage scenarios can be covered. For example, if you imagine the containers are filled with medicine and a cyber-attack may result in too little or too much active ingredient, the effects on the patient can be extreme.

2.3 Programmable Logic Controller Overview

A PLC is a device programmed on a digital basis that is used to control machines or plants. These replace the previously hardwired systems, thus enabling flexible control and reprogramming. In the simplest case, a PLC has inputs, outputs, an operating system, and an interface to upload the user program. The user program defines how the outputs have to be switched and the inputs handled. The PLC is connected to the physical world with sensors and actuators. Figure 2.3 shows a common PLC from WAGO Kontakttechnik GmbH & Co. KG."

The picture shows a controller with the base module and power supply, an input module, an output module, and an end module. The base module houses the Central Processing Unit (CPU) with the network interface and also some status Light-emitting Diodes (LEDs). All modules have an internal bus, over they talk to each other. This bus is mostly manufacturer dependent and proprietary. From the IO modules, the cables go to the corresponding sensors and actuators. The current generation of control systems usually have an Ethernet connection via which industrial protocols can be used. For example, status information can be queried over this connection to show sensor values on HMIs. There are usually two options to upload the user program to the PLC – one via Ethernet if available and the other via a local interface like an SD card or a debug interface to make settings when the network interface is not available. Additionally, updates are generally uploaded over these interfaces.

There are different PLC devices which differ mainly in the processing:

- **Cycle-oriented PLCs**: A large group of PLC devices are cycle-oriented, i.e. they work strictly according to the input-process-output model principle. The operating system of the PLC controls the cycle. At the start of the cycle, all inputs are first

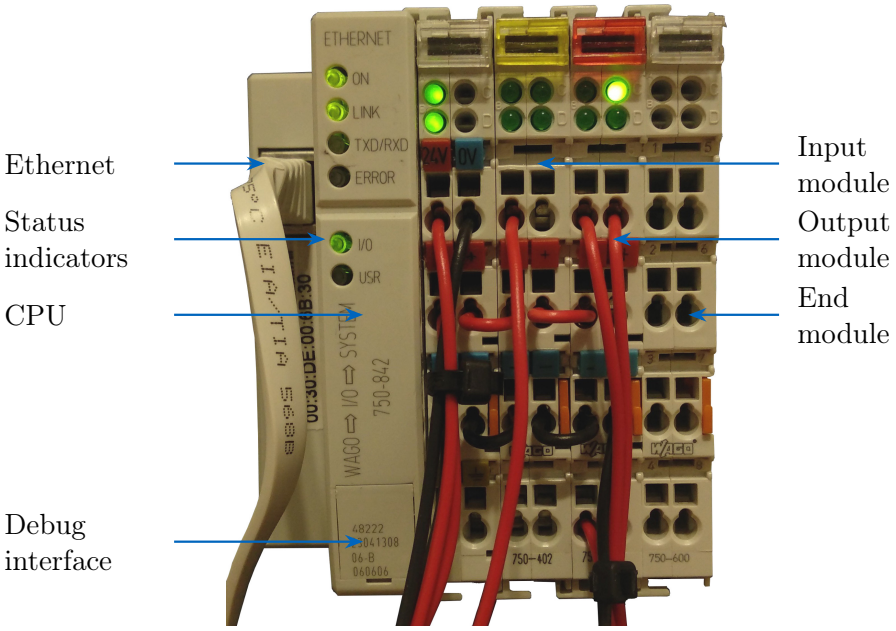


Figure 2.3: Picture of a common PLC (WAGO 750-842).

read in the system and a process image is created. The user program is then executed, and at the end of the cycle, the process image is written back.

- **Cyclic PLCs with interrupt processing**: Generally, the regular cycle is executed. If an interrupt occurs, e.g. due to a change of a sensor value, the interrupt is handled. After this, the regular cycle continues to run.
- **Event-driven PLCs**: Here, the operating system lists all "events" and processes the corresponding user program parts in this order.

In this work, cycle-oriented PLCs and user programs are used, as these are the most popular ones. How a PLC cycle works is explained in detail in the following.

2.4 Programmable Logic Controller (Scan) Cycle Time

The run mode of a cycle-oriented PLC consists basically of a loop of four phases, as illustrated in Figure 2.4. In the first phase, inputs such as sensors are read into the internal registers of the PLC and a process image is created. In the second stage, the program execution is performed. The third phase handles internal housekeeping, for example, diagnostic functions and communication. At the end of the scan cycle, the outputs are written back from internal registers (process image) to the electrical circuits. Typical cycle times are between one and 10 milliseconds. In more powerful models, or small programs, cycle times may be in microseconds. There are versions with either fixed or asynchronous cycles. The user program may include branches and conditional calls,

resulting in varying execution times. For these reasons, it is not possible to specify a generally valid cycle time, as this usually depends on the process and the corresponding user program.

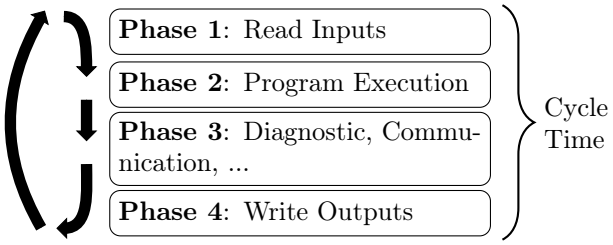


Figure 2.4: Simplified sequence of a PLC cycle.

When the generic cycle is not sufficient, there are special input cards that report an interrupt or handling the control loop itself. By using interrupts, the currently running program is interrupted, and, for this situation, the assigned program is executed and then the interrupted program continues. This allows the mastering of time-critical tasks that are in conflict with the cycle time.

2.5 Structured Text User Program

In most cases a PLC is first of all a black box which executes an operating system and software from the manufacturer. Then the so-called user program is executed on top, which comes from integrators or operators. In most cases only this user program can be modified by the end users. Structured Text (ST) is a text-based programming language for PLCs. The standard EN 61131-3 [IEC93] specifies, among other things, the language scope of ST. Listing 2.1 shows an example of a ST user program which operates cyclically. This will toggle two outputs every 100 cycles.

```
1 PROGRAM PLC_PRG
2   VAR
3     Counter:INT := 0;  (* Variable to count PLC cycles initialized with
          0 *)
4   END_VAR
5
6   IF Counter >= 200 THEN  (* Reset Counter *)
7     Counter := 0; (* Set Counter to 0 *)
8   END_IF;
9
10  IF Counter < 100 THEN (* Output1 high 0-99 cycles *)
11    Output1 := TRUE; (* Set Output1 high *)
12    Output2 := FALSE; (* Set Output2 low *)
13  ELSIF Counter < 200 THEN (* Output2 high 100-199 cycles *)
14    Output1 := FALSE; (* Set Output1 low *)
15    Output2 := TRUE; (* Set Output2 high *)
16  END_IF;
```

```
17
18   Counter := Counter + 1; (* Increase cycle counter *)
19 END_PROGRAM
```

Listing 2.1: Example of ST user program code.

It is important to note that, as illustrated in Figure 2.4, the program cycle is executed completely before changes are written to the process image. The outputs are thus updated each cycle exactly once and this at the end. The user programs, which are used in this work are programmed in ST and toggle the outputs, as illustrated in Listing 2.1, to get a correlation between the electric output signal and the cycle time. This correlation is achieved because the electrical signal is changed at every cycle, and therefore the electrical signal pulse time correlates with the cycle time. Assuming each cycle takes exactly 1 ms, an output signal with 100 ms high and 100 ms low would be generated for the code shown here.

2.6 Attack on the Programmable Logic Controller Cycle Time

The basic idea behind the PLC cycle time attack in this work is to influence the timing of a PLC by means of network traffic. This raises the research question of whether industrial systems are prone to network traffic and what impact this has on the electrical side and in real-world situations. In other words, the attacker aims to alter the timing of PLC outputs.

Wedgbury and Jones [Wed+15], as well as Cárdens [Cár+08a], predicted that extra network traffic might affect the process controlled by an ICS. However, they did not present evidence for their prediction. The experiments conducted in this work lend support to their assertion because the results of this work show that network traffic can affect user programs running on PLCs.

If an output of a PLC is changed with every cycle, a kind of square wave signal is generated. This regular behavior is represented by the blue line in Figure 2.5. A common output voltage for PLCs is 24 V, which results in flatter rising edges. If the cycle time becomes longer due to the load on the CPU, there is a shift in the output signal Δt, which is represented by the dotted orange line.

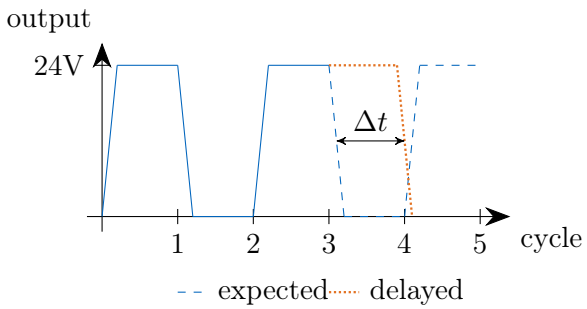


Figure 2.5: Electrical view of a PLC toggling an output.

A ST code, which changes the output every PLC cycle is shown in Listing 2.2. This ST code will generate a signal similar to the signal illustrated in Figure 2.5.

```
1 PROGRAM PLC_PRG
2   VAR
3     Output:INT := 1;  (* Variable for Output, "1" starting with high *)
4   END_VAR
5
6   IF Output = 0 THEN  (* Compare Output Variable with 0 *)
7     Output1 := FALSE; (* Set Output1 low *)
8     Output := 1; (* Set Output Variable to 1 *)
9   ELSE
10    Output1 := TRUE; (* Set Output1 high *)
11    Output := 0; (* Set Output Variable to 0 *)
12  END_IF;
13
14 END_PROGRAM
```

Listing 2.2: Example of ST user program code, to switch "Output1" with every PLC cycle.

As an attacker, it can be a goal to influence this cycle time and thus also the electrical behavior. The attack surface is a combination of device design and software implementation. More precisely, it is the implementation of the network stack, PLC-specific protocols, and PLC runtime. For example, sharing resources between system tasks and the actual control program can be problematic. If attackers are able to exhaust the resources available to system tasks, they also succeed in preventing normal operation of the control program.

2.7 Noteworthy Industrial Control System Characteristics

In the following, some noteworthy circumstances of ICS are explained with reference to the "ICS Security Compendium" [Fed13] of the BSI. This document gives an overview of the challenges in ICS, as well as security recommendations. Compared to the standard IT environment, Operational Technology (OT) has different priorities and infrastructure to protect, which are explained in the following.

2.7.1 Industrial Control System and the Confidentiality, Integrity and Availability Triad

Protection goals are defined to achieve information security and protect data of IT systems. Basically, there are three protection goals, which are represented in the Confidentiality, Integrity and Availability (CIA) [Per08] triad.

- **Confidentiality**: Data may only be read or modified by authorized users, both when accessing stored data and during data transmission.
- **Integrity**: Data may not be modified unnoticed. All changes must be traceable.

- **Availability**: Prevention of system failure: Access to data must be guaranteed within a specified timeframe.

These are often extended by two more protection goals.

- **Authenticity**: This refers to the characteristics of the authenticity, verifiability, and trustworthiness of an object.
- **Non-repudiation**: It requires that "no improper denial of actions performed" is possible. Among other things, it is important in the electronic conclusion of contracts. It can be reached, for example, by electronic signatures.

The CIA model is often presented in a triad, with the extended protection goals in a pentagon. Both of these representations are shown in Figure 2.6.

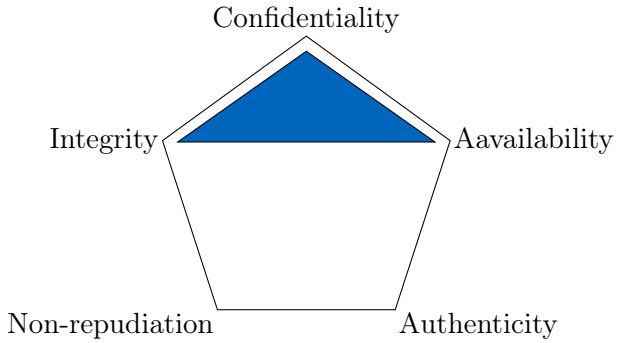


Figure 2.6: CIA Protection goal triad, with extension.

Two worlds come together in the field of industrial IT security – those coming from office IT and those from automation engineering. From the perspective of office IT, the established CIA protection goal model is often inverted in the literature [Zhu+11; Sto+11], resulting in availability as the most important asset. In automation engineering, the focus is often not on the CIA triad. One example is the Safety, Reliability and Availability (SRA) model [Eur11], which describes a classically applied structure of priorities in the industrial environment. Reliability, Availability, Maintainability and Safety (RAMS) [Int02] provides another definition of goals in the industrial environment. It is mainly used in railway applications and uses these four aspects on a similar priority level. An extension of RAMS is RAMSST, which also includes "security," the protection of the system against external attacks and "testability," the degree to which a system supports tests in a given test context.

All in all, a unified approach cannot be found here, as it depends heavily on the requirements and the physical process. Depending on the area in which the controls are used (railway, industrial automation, etc.), other directives and security models apply. Often there is a way to combine the two worlds of office IT and automation technology, to achieve the goals of both areas. Therefore, in this work, the CIA model is used, taking into account a different prioritization of the individual protection goals.

2.7.2 Lifecycle

The lifecycle of an ICS is derived from the associated production facilities. This is significantly longer than the typical time periods in office IT environments. Usually, it is 10–15 years [Sto+11] and sometimes even more than 20 years. In office IT, it is usually only three to five years.

2.7.3 Functional Safety

There are many applications in which the operation of the equipment is subject to regulatory requirements, like nuclear power plant safety. In these cases, significant changes, including software changes to the ICS, may require an approval process [Nov+08]. Owing to this test process, the possibilities to make timely security updates are limited or not present here.

2.7.4 User Program

In contrast to office IT, ICSs are operated over longer periods of time with virtually the same user program. Changes are made in the context of controller options such as controller parameters, sensor limits, and process parameters.

2.7.5 Update Management

In the field of office IT, systems are often updated as soon as possible after the detection of errors or vulnerabilities. Contrary to this, in ICS updates are rare or not unavailable [Cár+09]. In the area of ICS, software changes, including software updates, must comply with the systems, which should be patched. Thus, application-specific tests must be performed before and after the updates are used in actively used plants. As a result, updates can only be made as part of maintenance activities at longer intervals.

Chapter 3

Communication Robustness of Programmable Logic Controllers in Terms of Security

Contents of this chapter

This chapter examines the correlation between communication load and the timing behavior of cycle-oriented PLCs, which is also called reliability or robustness in the industrial context.

In computer science, software development and engineering, the term "robustness" means the ability of the correct functionally of a process, even under unexpected conditions. Additionally, robustness, also known as "fault tolerance," is one of the quality criteria for software and systems. Examples of such precautions include preventing undefined states and system crashes and, in particular, intercepting erroneous user or data inputs. Nevertheless, 100% robustness is not achievable. But for a computer program, it is often possible to bring the system in a safe state and produce an error message. An example of one of the best-known behaviors, in the case of an error, is probably the Microsoft Windows blue screen [Mic19], in which a critical error occurs and, as a precaution, the computer is shut down. Furthermore, an error message is displayed, which with the user can localize the problem. The aim of this fail-safe behavior is clearly to protect the hardware and not to maintain operations.

In contrast, in the industrial environment, the goal, when an error occurs, is usually to bring the plant respectively the process into a safe state and also not to injure any persons. However, to get back to the IT security context, in general network communication, such as DoS flooding, should not have any impact on the control process of a PLC. It is this requirement for the robustness of industrial systems that is analyzed in this chapter, with

the focus on interactions between the network side and the physical world. It discusses how the robustness of a PLC can be influenced and what options are available for a secure operation of ICSs.

The chapter is structured as follows. At first, in Section 3.1, the proprietary testbed with which the evaluations are done is introduced. After this, the possibilities of influencing the electrical output signal of a PLC are examined. This will be tested with the help of fuzzing (Section 3.2) and then extended by DoS attacks (Section 3.3). Afterwards, a new type of PLC architecture is presented in Section 3.4, which is based on a dual MCU setup, in order to achieve a separation between IO control and the network communication. Furthermore, a passive scan method based on Address Resolution Protocol (ARP) packets and vendor MAC address assignment analysis is presented in Section 3.5.

3.1 Communication Robustness Testbed for Industrial Internet of Things Components

Contents of this section

Parts of this section have already been published in the paper "CoRT: A Communication Robustness Testbed for Industrial Control System Components" at the 4th International Conference on Event-Based Control, Communication and Signal Processing 2018 (EBCCSP) [Nie+18a].

3.1.1 Introduction

One possible attack vector is the exploitation of the network communication of IIoT devices. Thus, a robust communication system is essential to ensure security. Unfortunately, the high requirement in operational availability for real-world ICSs makes it difficult to assess component security during its runtime. However, this is possible in a research testbed where tests could be done and analyzed in a safe environment. There are already many testbeds for ICS security research. Holm et al. [Hol+15] analyzed 30 ICS testbeds in 2015. However, these mostly focus on analyzing a complete ICS infrastructure, as opposed to testing single components. In this section, a testbed, focusing on the security of industrial components, especially the robustness of communication, which can influence the control behavior, is introduced. It is then used for the evaluations in this chapter. The following requirements for the testbed, with corresponding measurement methods, have been defined:

- **Network capture**: The generated network traffic during attacks and tests must be captured for further investigation.
- **Network reachability check**: It must be probed if the devices are reachable within the network during attacks.
- **Electrical monitoring**: The electrical outputs must be monitored to recognize changes in the control behavior.
- **Fast integration**: The integration of new devices into the testbed must be easy and fast.

The rest of the section is structured as follows. In Section 3.1.2, an overview of the Communication Robustness Testbed (CoRT) is given. The devices currently deployed in

the testbed are described in Section 3.1.3. Security tests which can be done with CoRT are explained in Section 3.1.4. Finally, a conclusion is provided in Section 3.1.5.

3.1.2 Communication Robustness Testbed

A communication robustness test measures the steadiness of control signals under various communication parameters and loads. Figure 3.1 shows our testbed divided into two identical racks, enabling comparison of the results.

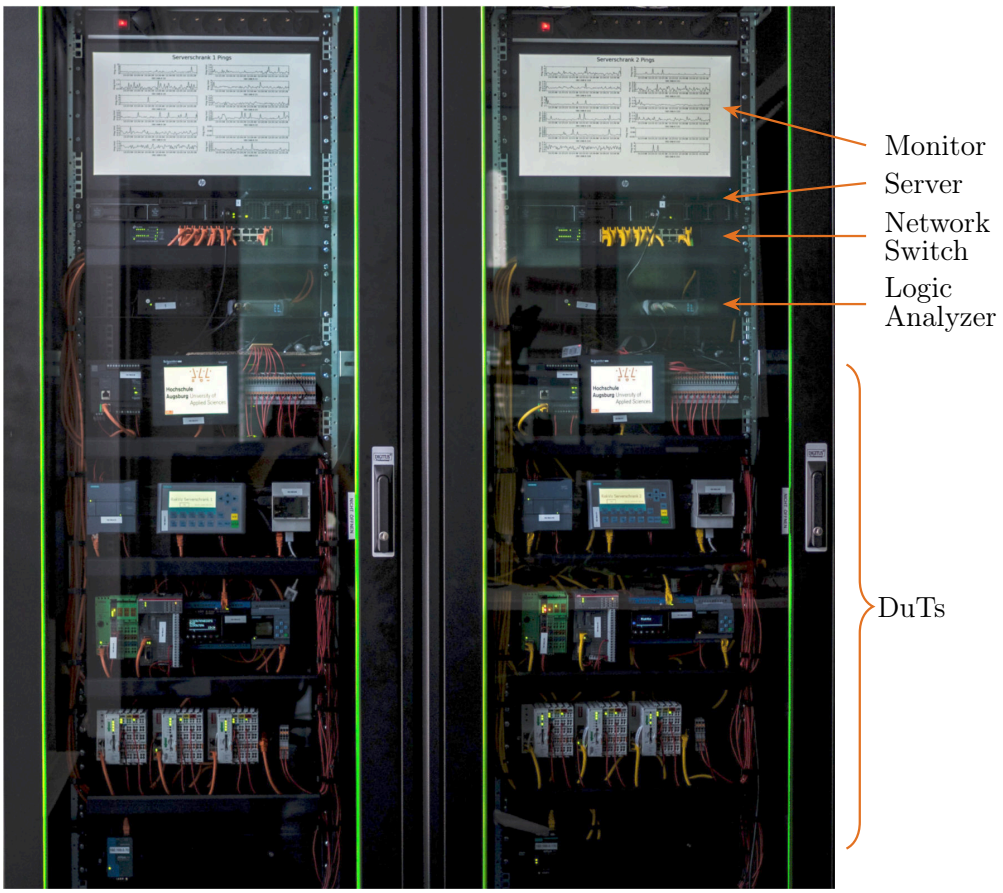


Figure 3.1: Picture of the testbed built into racks.

At the lower half the Devices under Test (DuTs), such as PLCs, HMIs, and bus couplers are placed. The electrical outputs of these are wired to a logic analyzer. All DuTs are interconnected with a network switch to a server. Besides, there is a monitor built in to keep track on the analyzed data. The DuTs are mounted on an EN 50022 rail, which is commonly used in the industrial sector. Both racks are lockable and have wheels for easy transportation.

3.1.2.1 Attacker Model Defined for the Testbed

This testbed is focused on, but not limited to, two attacker models: ❶ an attacker who has remote access to the network and ❷ an attacker who has local access to the ICS components with basic knowledge. Both are able to inject network traffic, e.g. send commands or perform a Man-in-the-Middle (MitM) attack. Attacker model ❷ is also able to locally manipulate input signals and has direct network access to the DuT. This is the case, e.g. if the attacker unplugs sensors or connects to the ICS network with a computer.

3.1.2.2 Testbed Focus on Robustness of Devices

The robustness of an IIoT device is essential, because they mostly control machines and interact with their environment. Therefore, an outage or loss of control creates a problem. With the CoRT, the research question, how network communication could influence the robustness of industrial systems can be analyzed. The communication robustness of different industrial components can be measured with e.g. fuzz testing and DoS attack frameworks. These must be specialized on proprietary protocols, which are partly used in ICSs.

3.1.2.3 Schematic Overview of the Testbed

Figure 3.2 gives a schematic overview of the testbed components. On the server, there are three Virtual Machines (VMs) for measuring, programming, and attacking the DuTs. In order to not influence the measurement, this separation is necessary, because some attack tools lead to a high system load. For this reason, the cores of the server are fixed assigned to the VMs. On the programming VM the necessary Integrated Development Environments (IDEs), to configure the PLCs and to load the user program, are installed. The attacking VM offers a platform to execute the control as well as attack tools. The measuring VM already comes preconfigured with all measurement methods, which will be explained in detail in the following sections. Furthermore, additional attack tools and further DuTs can be integrated into the testbed.

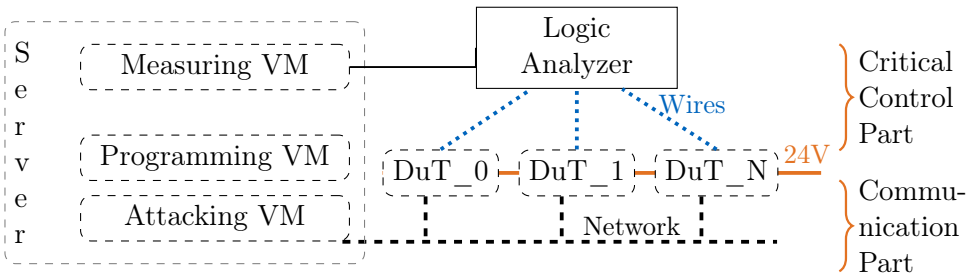


Figure 3.2: Schematic overview of the testbed.

The DuTs are powered by 24 V, which can be switched on and off remotely by the measuring VM. The logic analyzer is used to observe the critical control part, which is also the side a physical process is connected in real scenarios.

3.1.2.4 Measurement of the Electrical Behavior

To measure the robustness of the DuTs, the electrical outputs are observed with a logic analyzer. All PLCs are configured to toggle every single cycle, resulting in a frequency between 20 Hz and 20 kHz, depending on the device. These signals are measured with a logic analyzer and logged on the measuring VM. For the logic analyzer task, a BeagleBone Green running *BeagleLogic* [Abh14] with a custom Printed Circuit Board (PCB) is used. It is possible to analyze up to 14 channels in a continuous mode with a maximum of 100 Ms/s. The Ethernet interface (100 Mb/s) was used to send the data to a computer for further analysis. The capture is done on a fixed rate of 1 MHz, which allows calculating the timing without an additional timestamp. Only the state of the output of the device currently under test is of interest. Therefore, a byte per sample (8 active channels) or two byte (14 active channels) was needed to transfer data over the network, leading to a feasible data rate. Figure 3.3 shows the adapter board designed for this purpose. The schematic can be found in Appendix A.3.

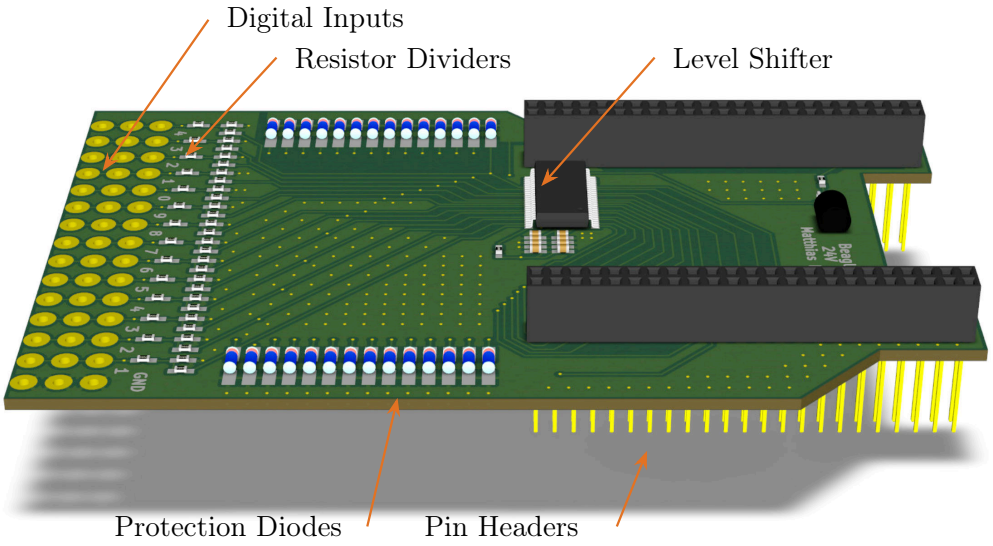


Figure 3.3: Rendered image of the PCB of the BeagleLogic adapter board.

This is necessary to measure up to 30 V and convert it to the 3.3 V input signal of the BeagleBone Green. On the PCB, there are mainly resistor dividers, protection diodes, and an SN74LVCH16245A [Tex14] 16-Bit level shifter. The left side of the PCB is connected to the outputs of the DuTs and the pin header is mounted on the BeagleBone Green. With the logic analyzer setup, influences on the control behavior of PLCs are measured and logged.

To ensure the validity of the tests, a function generator and a Picoscope 2208 [Pic] oscilloscope to measure the capabilities of the *BeagleLogic* is used. 1 Mega samples per second (Ms/s) on the *BeagleLogic* is configured, which is also the sample rate in the test setup. The results of this tests are summarized in Table 3.1. The deviation of the *BeagleLogic* adapter compared to the function-generator was below 0.1 %, which was sufficient for the test setup.

Table 3.1: Measurement results of the BeagleLogic validation.

Freq. (Hz)	100	1000	10000
P Mean	100.0	1000.0	10000
B Mean	99.97	999.9	9999
P Min	99.96	999.9	9999
B Min	99.97	999.0	9990
P Max	100.1	1000.0	10000
B Max	99.98	1000.0	10008
P Std	0.003621	0.0623	0.5102
B Std	0.004198	0.2185	7.252

P = Picoscope, B = BeagleLogic

3.1.2.5 Visualization

Besides the logging, the measurement results are directly visualized on monitors on the testbed. The cycle time [Mad00] of a Siemens S7-1211C during idle is illustrated in Figure 3.4. Idle means that the PLC is measured without any attack or intentional injected communication load. But this is not constant as it is influenced by communication and housekeeping during the execution. One cycle time takes from 140 μs to 300 μs during idle measurement on the S7-1211C.

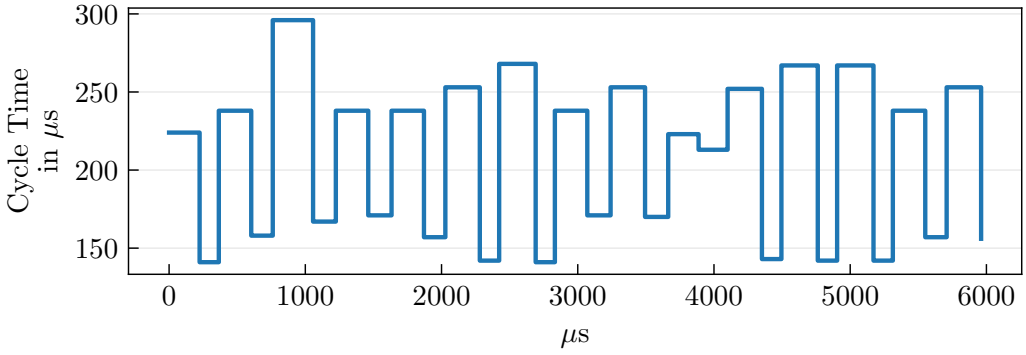


Figure 3.4: Example visualization of the cycle time of a Siemens S7-1211C PLC.

Furthermore, the ping response, current test results, and statistics of the network capture are visualized and can be optional used for analyses. This enables a quick overview at any time.

3.1.2.6 Optional Measurement Methods

Further test methods are already implemented and evaluated in the testbed, which could be important for later measurements. Currently, these are not taken into account in this chapter. However, they offer a basis for further research questions.

Optional: Ping Response of DuT To ensure network reachability during tests, the ping response of all devices is measured. This is done with a fping [Sch] script every 100 ms, which has been tested as a trade-off between measuring accuracy and not influencing the devices by flooding. If a network stack on a DuT crashes, it is detected and logged.

Optional: Network Capture To analyze tests in detail, a network capture of the test period is essential. The complete network traffic within the testbed is mirrored on one port and captured with tcpdump [Jac+89] in a log rotation. For example, if a device fails during an attack, the timestamped and corresponding captures are used for further investigation.

Optional: Virtualized Devices For the measurement of the response time of a command sent over the network to the DuT, it is necessary to virtualize components, e.g. sensors or other IIoT devices. The control command (e.g. set an output over an industrial protocol) is sent by the measurement VM, which reduces timing dependencies during the measurement. Thereafter, the electrical output is measured by the logic analyzer and is processed. In order to send a command, the industrial protocol, e.g. BACnet or Modbus, can be used and integrated into the test.

Optional: Input Signal Generation To simulate electrical input signals for the DuTs, these must be generated to be read by real hardware. This is realized with a Universal Serial Bus (USB)-to-serial converter, which is connected to the attacking VM. The RTS pin of the USB-to-serial converter is used to generate the signal and Metal-oxide-semiconductor Field-effect Transistor (MOSFET) are used to switch 24 V. Both the generated input and the output signals of the DuT are measured by the logic analyzer. The delay between them is the response time. This could cause a jitter if, for example, the DuT faces a high CPU load due to network communication.

3.1.3 Devices under Test in the Testbed

For the CoRT testbed, diverse vendors and products are chosen. Thus, it is possible to compare implementations based on their security level from a technical point of view. However, single devices can also be tested. Table 3.2 lists currently deployed devices with a selection of open ports.

Table 3.2: PLCs deployed within the testbed, with additional components regarding completeness.

No.	Vendor	Product	Vendor No.	Selection of Open Ports
1	Wago	Controller KNX IP	750-889	21, 80, 443, 502, 2455, 6626
2	Wago	Controller PFC100	750-8100	22, 80, 443, 502, 4840, 6626, 11740
3	Wago	Controller ETH.	750-880	21, 80, 443, 502, 2455, 6626, 44818
4	Wago	Controller BACnet	750-831	21, 80, 443, 502, 2455, 6626, 47808
5	Siemens	CPU 1211C	6ES7211-1AE40-0XB0	80, 102, 443
6	Siemens	Simatic S7-1212	6ES7212-1AE31-0XB0	80, 102, 443
7	Siemens	Simatic ET 200SP	6ES7155-6AU00-0AB0	-
8	Siemens	Simatic S7-314	6ES7314-6EH04-0AB0	80, 102, 443
9	Siemens	Simatic S7-1516F	6ES7516-3FN01-0AB0	80, 102, 443
10	Siemens	LOGO! 24RCE	6ED1052-1CC01-0BA8	80, 102, 502, 8080
11	Phoenix	ILC 151	2700974	21, 80, 1962, 41100
12	Phoenix	ILC 150 ETH	2985330	21, 80, 1962, 41100
13	Phoenix	ILC 171 ETH 2TX	2700975	21, 80, 443, 1962, 41100
14	ABB	PM554-T	1SAP120600R0071	21, 502, 1200, 1201
15	Crouzet	em4 B26-2GS	88981133	502, 42424
16	Schneider	TM221CE16T	TM221CE16T	502, 44818
17	Siemens	KP 300	6AV6647-0AH11-3AX0	102, 2308
18	Schneider	HMISTU855	HMISTU855	502, 6001
19	OpenPLC	Raspberry Pi 3	Commit f1a2645	22, 502, 8080, 20000
20	Moxa	NP5110	NP5110	23, 80, 443, 950, 966, 4900

3.1.3.1 Common Network Protocols

Industrial components often use common network protocols for tasks such as monitoring and visualization. On the testbed, the following common network protocols are available:

- **File Transfer Protocol (FTP) "Port: 21"** is used for file transfer. Within industrial networks, this is used for logging and updating the firmware, for example.
- **Secure Shell (SSH) "22"** refers to a network protocol that can be used to securely communicate with a remote device. It is often used to make a remote command line available locally if, for example, the PLC runs Linux.
- **Telnet "23"** is a client/server protocol based on a character-oriented data exchange over a TCP connection. PLCs could be partly configured over it.
- **Hypertext Transfer Protocol (HTTP) "80"** is used to load web pages (hypertext files) into a web browser. **Hypertext Transfer Protocol Secure (HTTPS) "443"** uses an additional transport security. These are used for the visualization of diagnostic information and sensor values.

3.1.3.2 Industrial Network Protocols

In modern plants, fieldbuses are increasingly being replaced by IP-based communication systems. Within CoRT, five common industrial protocols on real hardware are available.

- The **Modbus/TCP "502"** protocol is a communication protocol based on a master/slave architecture.
- **KNX IP "3671"** is mostly used for building automation.
- **OPC Unified Architecture "4840"** is an industrial Machine to Machine (M2M) communication protocol that works across manufacturers.
- **Ethernet/IP "44818"** is a real-time Ethernet mainly used in automation technology.
- **BACnet/IP "47808"** is chiefly used in building automation, ensuring interoperability between devices of different manufacturers, if all partners agree on certain building blocks defined by the standard.

3.1.3.3 Proprietary Network Protocols

There are other proprietary protocols, which are only partly understood.

- The **S7comm "102"** protocol is used by Siemens devices to communicate, for example, with the IDE and HMIs.
- **Phoenix Contact "1962, 41100"** use these ports for programming and debugging.
- **ABB "1200, 1201"** is the standard communication port.
- **WinCC "2308"** is mostly used for the communication with panels.
- **WAGO-Service-Protocol "2455, 6626"** used by Wago for Codesys.
- **Crouzet "42424"** is used for programming.
- **Codesys "1217, 11740"** is used by Codesys for programming.

With respect to security, these protocols are particularly interesting, because they execute privileged commands, such as setting the run mode of a device and updating the user application.

3.1.4 Experiments with Communication Robustness Testbed

With CoRT, a playground for researchers, organizations, and academic collaborators is made available. It is nearly impossible to make tests in an operating ICS, let alone change hardware or software. Even if tests can be performed, the bulk of the necessary data is not recorded and cannot be evaluated.

To evaluate different DuTs, predefined sequences of tests are used. However, it is important to say here that the CoRT only provides the basis (e.g. measurement equipment and procedure) for the tests and that different test scenarios can be implemented. One of these test sequences is illustrated in Figure 3.5. After it starts, an automated power cycle of the DuT can be done. This becomes necessary if previous tests have influenced the DuT or if it does not recover, e.g. after a successful DoS attack. Afterwards, the measurement begins, including network captures, continuous reachability check, and electrical output measurement. The measurement happens in three phases: (1) pre-idle, (2) attack, and (3) post-idle monitoring.

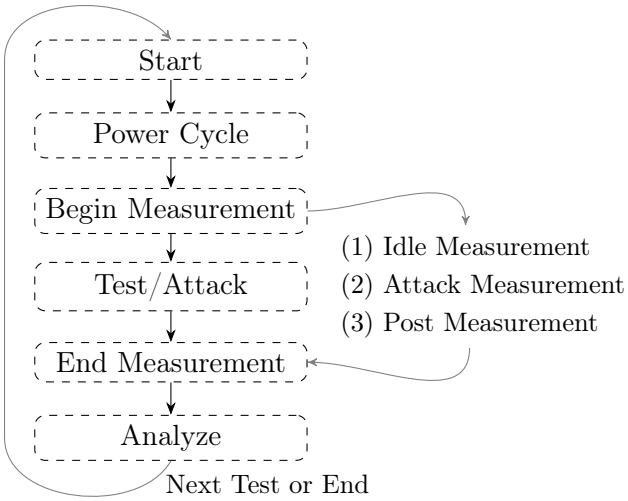


Figure 3.5: Illustration of a test sequence within the testbed.

During the test/attack, the DuT can be subjected to different kinds of network robustness tests and vulnerability scans. Finally, the results are analyzed, logged, and are visualized on the monitor. If additional tests are in the queue, the sequence starts from the beginning until all tests are finished. It is possible to perform a full test on a single device, or to apply one test on every device in the rack. This means that a specific test can be done on all devices, for example if a new test should be measured. This depends on the type of validation, such as fuzzing of a single protocol on a device, which is not supported by other DuTs.

The control of the tests is taken over by the measuring VM, which thus takes over the central control of the complete CoRT. This in turn starts the measurements and sends e.g. the attack parameters and duration to the attacking VM.

3.1.5 Conclusion

The proposed testbed combines network devices, measurement equipment, and industrial components. It allows studying ICS devices like PLCs, sensors, and HMIs in detail without influencing real-world processes. Furthermore, the testbed measurements are automatically recorded and analyzed. Also, new test scenarios can be built up, without having to worry about the measurement setup. With this testbed, a basis for communication robustness tests has been established.

3.2 Fuzzing Proprietary Industrial Devices

Contents of this section

Parts of this section have already been published in the paper "PropFuzz – An IT-Security Fuzzing Framework for Proprietary ICS Protocols" at the 22nd International Conference on Applied Electronics (AE) 2017 [Nie+17].

3.2.1 Introduction

Most PLCs offer the possibility to configure and program them via a proprietary TCP/IP connection. This simplification allows remote access to these devices, if there is no additional hardware restricting the communication. Thus, it is often possible for attackers to interact directly with the PLC and its configuration interface. Therefore, it is necessary to analyze the communication between the control system and the IDE, e.g. with fuzzing, to find security issues in proprietary industrial protocols. The main problem concerning the fuzzing of these protocols is defining the data structure to be fuzzed. Since these are proprietary and therefore usually not openly accessible for security analysis. In this section, a new fuzzing framework called PropFuzz, which is capable of fuzzing proprietary ICS protocols is introduced. Additionally, to demonstrate the feasibility a selection of components from the testbed presented in Section 3.1 is fuzzed.

The section is structured as follows. Popular fuzzing frameworks are introduced in Section 3.2.2. Section 3.2.3 describes the concept of fuzzing a proprietary PLC communication. Section 3.2.4 explains our framework architecture. Section 3.2.5 presents the first results, along with possible attacks. Finally, an outlook and a conclusion are given in Section 3.2.6.

3.2.2 Related Work and Motivation

Barton Miller discovered a program crash caused by noise as a result of a lightning strike on his network connection during a thunderstorm [Mil+90]. The bug was triggered by a random input called "fuzz-testing" or "fuzzing" in the literature. Fuzzing could only trigger bugs, if the input is not rejected by a validation function of the DuT. A fully automated fuzzing framework for ICSs includes the process steps illustrated in Figure Figure 3.6 [Kim+16].

There are two elementary categories of fuzzers, based on how they create input for fuzzing. Generation-based fuzzers create input from scratch and thus require some knowl-

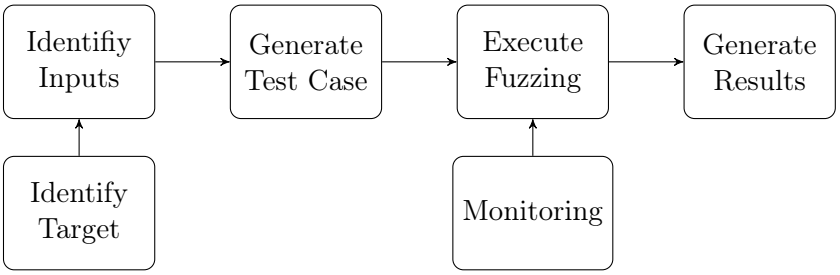


Figure 3.6: Fuzzing test process procedure.

edge of the protocol with corresponding data fields. With mutation fuzzers, samples of valid input are used to produce malformed input. A simple mutation fuzzer can modify a valid input sample and send it to the DuT.

- **Generation-based Fuzzing** applies with generation rules to fuzz input. *Boo-Fuzz* [Per17], a fork and successor to the *Sulley* [Ami+07] fuzzing framework, and *Peach* [Edd09] are block-based fuzzers. These kinds of fuzzers need a deep knowledge of the protocol structure and test case definition to generate inputs. Recent generation-based fuzzers like *VUzzer* [Raw+17] are able to automatically generate input test cases for basic communications.

- **Mutation-based Fuzzing** uses valid inputs and modifies them to create fuzzing input. Most of the frameworks analyze previously captured traffic, although there are fuzzers which allow live capturing. *Radasma* [Hel16] is an input-generation tool for basic protocols to identify field boundaries. *LZFuzz* framework [Sha+11] is an online fuzzer that intercepts traffic directly, analyzes it with the Lempel–Ziv compression algorithm, and sends the manipulated packages to the device. The compression algorithm is used, to guess boundaries between the structural units.

For fuzz-testing ICSs, these fuzzers and frameworks do not fulfill our requirements in automated input generation for proprietary protocols and electrical monitoring [Sha+11] and thus these fuzzers are barley compatible with our testbed introduced in Section 3.1.

3.2.3 Concept

Most modern PLCs are programmed with an IDE over TCP/IP. This communication is often open and not filtered. Figure 3.7 illustrates the minimal setup to interact with the ICS and the IDE. This is quite simple and consists of a standard Personal Computer (PC) with a network connection to the PLC. In the CoRT, this task is performed by the programming VM.

Because, the majority of these protocols are proprietary and have no publicly available documentation, the here presented fuzzing framework allows a direct investigation of this communication without previous configuration. To start the data transfer between an IDE and a PLC, a TCP/IP handshake is done. After that, there is often an additional proprietary handshake with a kind of challenge. The command and data transfer could

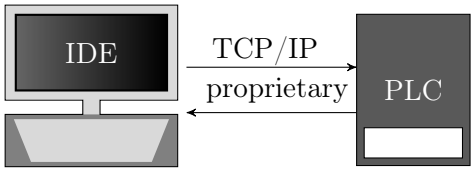


Figure 3.7: Communication between a standard PLC and the corresponding IDE.

be started after this. For a permanent connection, it may be essential to send keep-alive messages between the IDE and the PLC, which is mostly not required for single-command interaction. To make fuzzing feasible, it is necessary to perform the proprietary handshake and determine the protocol field that should be fuzzed.

3.2.4 Framework Architecture

At a high-level view, illustrated in Figure 3.8, PropFuzz is separated into three parts to fulfill the requirements of a fully integrated fuzzing framework [Pla+16]. The **analysis** part splits the protocol and filtrates the information necessary to **fuzzing** the DuT. In addition to the network response monitoring, the **monitoring** part observes the PLC electrically. For the analysis, the data can be captured live within the test environment or provided by a packet capture (pcap) respectively packet capture - next generation (pcap-ng) file. The analysis uses the *pcapy* module [COR], a Python implementation for *LibPcap* [Jac+], for further examination of the collected packages. With the analyzed data, *scapy* [Bio+11] is used to send data to the DuT. *Pico-python* [OFl] or *BeagleLogic* [Abh14] is used to monitor the DuT. These measurements can be made, on the one hand on the testbed presented in Section 3.1 (*BeagleLogic*) or on the other hand, independently with a simple setup with a PC and an oscilloscope (*Pico-python*).

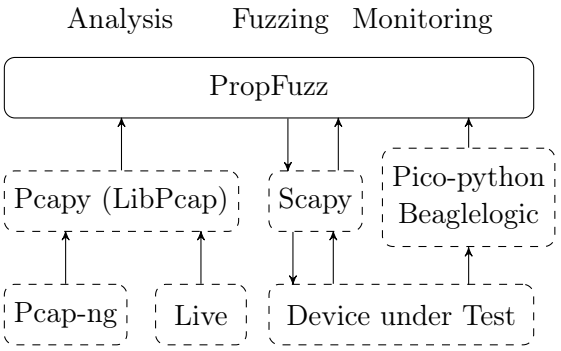


Figure 3.8: Data-flow within the PropFuzz framework.

A detailed view of the PropFuzz structure is provided in Figure 3.9, which illustrates the Python modules, classes and configuration files within PropFuzz. The implemented Python modules are modular and can be extended or used within other frameworks.

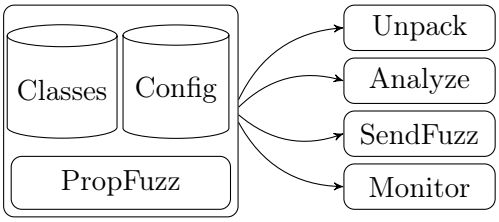


Figure 3.9: Overview of the PropFuzz framework modularity.

Unpack The PropFuzz implementation provides two possibilities for package analysis. First, a live capturing of the communication between DuT and IDE is possible. The second option is to read in existing *pcap* files. For live capturing, ARP spoofing is needed, which could be prevented by some network devices. Once data acquisition is done, the Unpack module splits up the information of the captured packages and stores them in objects.

Analyze Inside the analysis process, the messages of the proprietary protocol are interpreted. At the start, a statistical analysis with the Ratcliff/Obershelp [Rat+98] pattern recognition algorithm (see Equation 3.1) of the created package objects is done. With these similarities, the proprietary handshake between the IDE and the PLC can be determined. After the detection of a handshake in the captured communication, commands between the IDE and the PLC must be identified. The commands can be identified by comparing different captures containing a similarity match of the same command. Table 3.3 illustrates the symbols used in the formula.

Table 3.3: Symbols used in formulas.

Symbol	Description
K_m	Number of matching characters recursively
$\|S_1\|$, $\|S_2\|$	Length of each string
$0 \leq D_{ro} \leq 1$	Similarity metric

Equation 3.1 calculates the similarity metric of two strings. The value 0 means that not even one character in two strings is the same.

$$D_{ro} = \frac{2 * K_m}{|S_1| + |S_2|} \tag{3.1}$$

Table 3.4 shows an example, how to identify similar strings. In the following example two strings are used, which have the same beginning and then differ. This is also often the case for protocols that have a similar data structure for commands.

Table 3.4: Example of pattern matching .

P	r	o	p	F	u	z	z	$\lvert S_1 \rvert$:	8
P	r	o	p	R	e	z		$\lvert S_2 \rvert$:	7
✓	✓	✓	✓	✗	✗	✓	✗	K_m:	5

The similarity get a similarity metric as illustrated in Equation 3.2. It is important here that this is also calculated recursively, for example if a field is shorter or longer, but still similar.

$$D_{ro} = \frac{2 * K_m}{|S_1| + |S_2|} = \frac{2 * 5}{8 + 7} = \frac{10}{15} = 0.67 \tag{3.2}$$

There are two ways to define the input. Either one byte is used and compared, or 4 bits, which are represented in hex. This example shows how similar packets can be identified in a network stream and how similarities of individual parts of a packet can be compared with another. This information is then used for the mutation-based fuzzing.

SendFuzz The SendFuzz module is responsible for sending and receiving packets inside the PropFuzz implementation. For constructing these messages the Python module *scapy* is used. The gained information from the analysis module is used to mimic the protocol handshake by sending sniffed messages to the DuT, which is basically a reply. After a protocol handshake is successfully established, further packages containing protocol-specific commands are sent to the DuT. Parts that change in the data structure of the protocol (low similarity metric) are changed here on the basis of mutation-based fuzzing. The fuzzing part must run on the attacking VM, to not influence the measurement (see Section 3.1.2.3).

Monitor Most fuzzing frameworks only observe the network connection during the test. What is special about fuzzing ICSs is the monitoring of the process control [Yoo+16], which is possible within the CoRT. To detect the effect of fuzz-testing with our framework, an output channel is monitored by a logic analyzer or oscilloscope as described in Section 3.1.2. This measurement feedback must run on the measuring VM or dedicated thread with high priority, that it will not be influenced, by e.g. CPU load caused by generating the fuzzing packets.

3.2.5 Framework Evaluation

To test PropFuzz, three different PLCs, the “ILC 151 ETH” (11), “ILC 150 ETH”(12) and the “ILC 171 ETH”(13) from Phoenix Contact, are used. These PLCs were selected because little is known about the used proprietary protocol, and until now the PLCs of this manufacturer have not been evaluated in academia. For more information about the devices, see Table 3.2. According to the datasheet, the devices support the protocols shown in Table 3.5. In this assessment, the IDE “AUTOMATIONWORX Software Suite v1.83” from Phoenix Contacts is used.

Table 3.5: Phoenix Contact test equipment used for the evaluation.

DuT	ILC 151	ILC 150	ILC 171
No.	11	12	13
Man.number	2700974	2985330	2700975
Profinet	✓		✓
Modbus	✓		✓
Proprietary	✓	✓	✓
FTP	✓	✓	✓
HTTP	✓	✓	✓
HTTPS	✓		✓
SNTP	✓	✓	✓
SNMP	✓	✓	✓
SMTP	✓	✓	✓
SQL	✓	✓	✓
MySQL	✓	✓	✓

Three ports are open, if the factory default settings are applied. Table 3.6 shows the results of an *nmap* scan of the PLCs. A vulnerability in one of these protocols leads to high risks due to the remote exploitability.

Table 3.6: Factory default port scan of the Phoenix ILC150 PLC.

Port	Protocol	State	Service
21	TCP	open	FTP
1962	TCP	open	unknown
41100	TCP	open	unknown

For the intended purposes, the most interesting ports are the undocumented ones, which are used by the IDE to communicate with the PLC. Most commands are exchanged via port 1962. The connection establishment of this protocol is illustrated in Figure 3.10. The IDE sends a synchronize (SYN) request to the PLC, which should respond with a SYN acknowledgment (ACK). Consequentially, the IDE completes the Transmission Control Protocol (TCP) handshake with an ACK.

After the TCP handshake with the PLC, a proprietary initializing (PROP) sequence between the IDE and the Phoenix Contacts PLC is necessary. This starts with a request from the IDE to the controller. The request is always the same compared with different captures of the initialization, resulting in a similarity metric of 1.

```
0000   01 01 00 1a 00 00 00 80      ........
0008   64 15 00 03 00 0c 49 42      d.....IB
0010   45 54 48 30 31 4e 30 5f      ETH01N0_
0018   4d 00                        M.
```

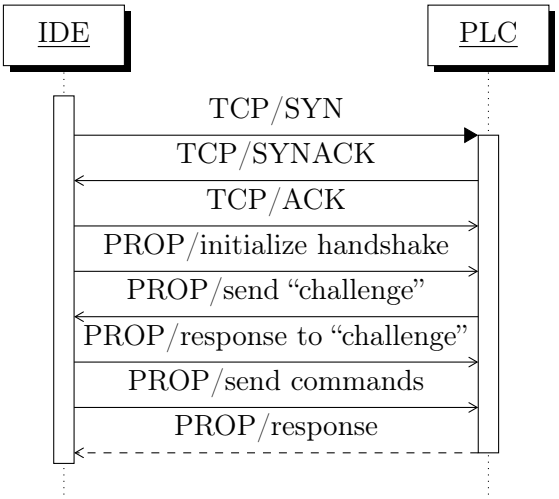


Figure 3.10: Handshake between Phoenix PLC and IDE.

This request is answered from the PLC, with an identifier (in this case 0x48):

```
0000   81 01 00 14 00 00 00 01      ........
0008   00 00 00 00 00 02 00 00      ........
0010   00 48 00 00                  .H..
```

This value must be sent back from the IDE to the PLC. It could be seen as a simple device key that does not change on a controller:

```
0000   01 05 00 16 00 01 00 00      ........
0008   e8 e9 00 48 00 00 00 1c      ...H....
0010   00 04 02 95 00 00            ......
```

For the proprietary protocol from Phoenix Contact, the same handshake for a specific DuT could be sent, because it is a constant value for each device. Thus, the handshake is a simple replay for the Phoenix PLCs and PropFuzz can detect the handshake by statistically comparing the start sequences of live captures or pcap files. After the proprietary handshake, commands could be sent to the controller. A reset command is shown below, where the PLC performs a complete reboot. It is noticeable here that the similarity to the previous packet is high, since many fields remain unchanged.

```
0000   01 05 00 16 00 10 00 00      ........
0008   e8 c8 00 48 00 00 00 00      ...H....
0010   00 04 0a ba 00 00            ......
```

At this point, *scapy* is used to fuzz different fields of the command. With the PropFuzz framework, it was possible to identify different vulnerabilities in the session management

and command handling of the PLCs. The vulnerabilities found in the Phoenix Contact products were reported to the manufacturer and customers were informed with an advisory (ICSA-16-313-01 [ICS16]).

The identified security problems within the protocol make several remote attacks possible. Considering the usage of these controllers in critical infrastructures, the severity of potential attacks is classified as high.

- A **replay attack** CVE-2016-8366 [Nat16a] is a network attack in which a valid data transmission is repeated. The attacker needs little knowledge and can simply replay previous captures containing PLC commands.
- By **manipulating variables** CVE-2016-8380 [Nat16c], it is possible to change the sequence or process of a program on the PLC. This requires knowledge about the setup of the control system.
- By **changing the software or firmware** of an ICS, complete control of it can be achieved, e.g. to create a BotNet.

These attacks demonstrate the severity of the identified vulnerabilities with our framework. It is possible to remotely exploit affected PLCs without having physical access to it.

3.2.6 Conclusion

In this section, a stable and extensible fuzzing framework for proprietary ICS protocols is presented. Compared to the available software, PropFuzz can automatically analyze the communication between the IDE and the PLC and fuzz the DuT. Additionally, it is able to monitor the output and detect suspicious behavior with the setup introduced in Section 3.1 or with a standalone oscilloscope.

Furthermore, the abilities of the PropFuzz framework have been demonstrated by fuzzing three PLCs within the CoRT, and three critical vulnerabilities were detected, which could be exploited remotely by attackers (Advisory ICSA-16-313-01 [ICS16]). Based on these findings, a close cooperation with Phoenix Contact to find solutions and fixes was initiated.

3.3 Impact of Network Traffic on Industrial Control System Devices

Contents of this section

Parts of this section have already been published in the paper "You Snooze, You Lose: Measuring PLC Cycle Times under Attacks" at the 12th USENIX Workshop on Offensive Technologies (WOOT 18) 2018 [Nie+18c].

3.3.1 Introduction

In this section, it is shown that the electrical side of a PLC, that is, the controlled process, can be influenced by packet flooding. This differs from already known DoS attacks as the target is the actual process and not network connectivity. Experiments with 16 devices from six vendors conducted in the CoRT (see Section 3.1), giving a good overview of the current market. This research question is also relevant when scanning the Internet for benign purposes, which is currently a trend in academic research. Additionally, this gains importance for active asset management within IIoT environments. If scans potentially affect controlled processes, then enhanced precautions are required to assure the safety and security of (largely unknown) scan targets.

The rest of the section is organized as follows. At the start, a description of related work in Section 3.3.2 is given. Background information of the PLC certification in Section 3.3.3 is provided and the corresponding attacker model in Section 3.3.4. In Section 3.3.5, the experimental methods and materials are summarized. At the end, the results of the experiments are given in Section 3.3.6 and a conclusion is provided in Section 3.3.7.

3.3.2 Related Work

DoS attacks on SCADA/PLC/ICS systems and devices have been a topic in academic research since at least 2005 [Bow+05; Lon+05]. However, most studies only outline the potential of attacks and do not present evidence derived from experimentation or simulation. In the following, the discussion is limited to the literature that provides at least partial evidence for possible DoS attacks.

Teixeira et al. [Tei+12] describe a variety of attacks on control systems. They focus on the disruption of communication between sensors/actuators and a PLC but overlook

the effects on the electrical side. The authors of [Ami+09] present a formalization of DoS attacks on control systems and derive an "optimal" attack plan. However, they do not evaluate their attack plan against an actual PLCs. Kalluri et al. [Kal+16] conducted flooding experiments on an unspecified Remote Terminal Unit (RTU) based on IP, SYN, and 104APCI packets. In all cases, they measured an impact on the response time of the RTU. However, their report lacks clarity with respect to what exactly caused the effects they measured. The reasons for this may range from RTU resource depletion to the saturation of other components in their test network. The authors of [Mar+13] simulated User Datagram Protocol (UDP) flooding attacks in a SCADA network model. They concluded that CPU utilization, packet drop, and traffic delays increased. In [Lon+05], the impact of DoSs attacks on network-based control is simulated and two countermeasures are proposed. The authors focus on the communication without analyzing the behaviors of the devices. A method of testing the communication robustness of industrial devices is introduced in [Til11]. However, their article mentions no concrete results. Sayegh et al. [Say+13] set up a testbed with an *Omron PLC CJ1M-CPU11-ETN* and demonstrated DoS attacks on the network interface of the device based on TCP/IP SYNs, UDP, and HTTP traffic. They did not measure effects on the electrical side, nor did they test different PLCs systematically as in the experiments in this work.

3.3.3 Certification Programs

There are three certification programs for IIoT components. In the following, three such programs which were previously discussed by Schierholz and McGareth [Sch+10], and Xie et al. [Xie+14], are mentioned. Certificates of these three programs communicate an acceptable level of stack robustness. Schierholz and McGareth argue that security-related certificates may send incorrect signals regarding security. This is primarily because not all threat vectors may be covered by a certification program. This work supports this argument as nearly all PLCs which are tested in this work are vulnerable to network flooding attacks. A short overview of the mentioned programs with respect to network robustness is provided below.

1. *Achilles Certification* [GE 17] – Initially developed by Wurdtech Security Technologies, the Achilles Program was later bought by General Electric. The program relies on a proprietary test device called the "Achilles Satellite". Applied tests include protocol fuzzing and packet storms. Of special interest to this work is the packet storm sub-test. While the Satellite is proprietary, the requirements for a certification are publicly documented. For level 2 certification of Achilles, the PLC is configured with a period cycle output of 1000 ms (500 ms high output and 500 ms low output) with an acceptable tolerance of 4 %.

2. *ISASecure EDSA Certification* [ISA19] – The EDSA includes *CRT Test Requirements for Protocols* for Ethernet, ARP, IPv4, ICMPv4, UDP, and TCP. With the exception of Ethernet, the requirements state that the device under test maintains its essential services under high load but can reduce or cease network communi-

cation during periods of high load. In all cases, the high load period (maximum supported data rate) must be long enough to allow saturation effects to manifest.

3. *Mu Dynamics MUSIC Certification* [Spi] – Mu Dynamics Inc. was acquired by Spirent Communication Inc. in 2012. The current status of the certification program is unknown. According to Xie et al. [Xie+14], MUSIC operated similarly to Achilles.

3.3.4 Attacker Model

In this work, it is assumed that the attacker is able to send network packets to the target PLC at the maximum rate supported by the device. This may be possible because the device is connected to the internet, or another device on the same network is compromised by the adversary. The compromised device may well be another PLC [Kli+15]. With regard to the attack types, it is considered that the adversary does not have or need specific knowledge about the actual process controlled by the PLC or the program running on the PLC.

3.3.5 Materials and Methods

The basic idea is to measure the changes to the signal captured on the electrical (digital) outputs of PLCs (see Section 2.6). Three sets of experiments are conducted. In the first set (Section 3.3.6.1), the focus is on the reaction of devices to different loads of SYN packets. In the second set (Section 3.3.6.2), the reaction to different protocols including device-specific control protocols is measured. In the final set of experiments (Section 3.3.6.3), the impact of scanning tools is assessed. In the remainder of this section, an overview of the methods and materials is given.

Regarding the electrical side, the PLCs under test are configured to run on their maximum performance (shortest possible cycle time). This means that an output is switched at the maximum rate. Depending on the actual device, this leads to a more or less periodic reference signal. If an attack is successful, the reference signal will be shifted as described in Section 2.6.

It is expected that the impact of attacks on the cycle time of a PLC differs from device to device. This is due to differences in the system design, quality of implementation, and possible safety mechanisms. For example, some manufacturers indirectly tie cycle time to the cost-efficiency of their devices, since the manufacturing process can be operated at a higher speed if the cycle time is shorter. An extreme example is provided by Schneider Electric [Bov17], where a reduction in the cycle time from 30 ms to 6 ms resulted in the gain of two million dollars per year.

In the experiments, the device under test is flooded with packets for a specific protocol and measure the cycle time of the device. The used protocols are depicted in Table 3.7. The measurements were made in the previously presented Communication Robustness Testbed (see Section 3.1), with the difference in the measurement procedures. In the following, the components are briefly explained and what is required for the tests in this section. The testbed comprising a capture device, an attacking VM, and a measuring

VM (see Figure 3.11). The capture device can digitize the outputs of the PLCs. The attacking VM generates traffic for the respective protocol under test. The controller software implemented on the measuring VM starts and stops the attack traffic, and stores the data sent by the capture device. It has the option to power on and off the Devices under Test (DuTs).

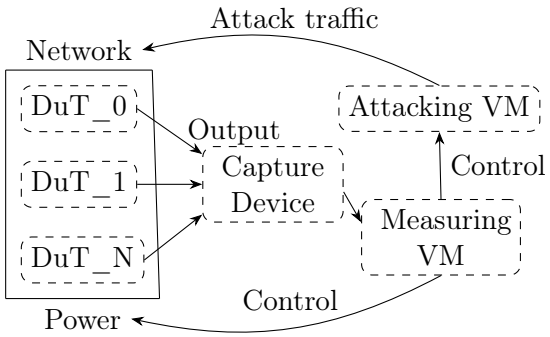


Figure 3.11: Test setup for the measurement.

In the following, the measurement setup and test cases, which are setup in CoRT, are discussed in detail.

3.3.5.1 Measuring VM

The measuring VM is a standard VM with two assigned network interfaces. One interface is connected to the capture device and the other to the attacking VM (virtual Network Interface Card (NIC)). The measuring VM runs a custom experimental control server written in Go. The server reads the definition of an experiment defined as a JavaScript Object Notation (JSON) file. An experiment defines the tool to use specific parameters, the target to measure, the channel to capture, and the runtime of the experiment. Based on this definition, the controller configures the capture device and attack server. In addition, the measuring VM stores the data produced by the capture device.

3.3.5.2 Attacking VM

The attacking VM is also a standard VM with two Gb/s Ethernet network interfaces assigned. One interface is connected to the DuTs, while the other is connected to the measuring VM (virtual NIC).

The attacking VM runs a custom experimental client that connects to the corresponding experimental control server implementation on the measuring VM. The purpose of the client is to start and stop the actual load-generating program. The tools used for load generation are listed in Table 3.7.

Zgrab [Bas+16], *Vegeta* [Sen14], and *hping3* [San99] are used without any changes. The *syn_spam*, *arp_spam*, *gre_spam*, and *snmp_spam* are custom implementations, which are explained in detail in Section 3.3.5.4.

Table 3.7: Overview of programs used, corresponding protocols, and respective parameters.

Program	Protocols	Parameters
ZGrab	S7comm / HTTP(S) / Modbus/TCP / Ethernet/IP / DNP3 / Bacnet/IP	`-s7 -port 102 / -port 80 -http=""/ -port 443 -tls -http=""/ -modbus -port 502 / -dnp3 -port 20000 / -dnp3 -port 20000 / -enip -port 44818`
Vegata	HTTP	`attack`
hping3	SYN/UDP	`-c 1 -1 -C 17 / -S -P -U -flood`
syn_spam	SYN	`-worker 20`
arp_spam	ARP	`-worker 20`
gre_spam	GRE	`-worker 20`
snmp_spam	SNMP	`-worker 20`

Table 3.8: Deployed devices in CoRT for these tests.

No.	Vendor	Manufacturer number	Name	Firmware
1	Wago	750-889	Controller KNX IP	01.07.13(10)
2	Wago	750-8100	Controller PFC100	02.05.23(08)
3	Wago	750-880	Controller ETH.	01.07.03(10)
4	Wago	750-831	Controller BACnet/IP	01.02.29(09)
5	Siemens	6ES7211-1AE40-0XB0	Simatic S7-1211*	V4.2.0
6	Siemens	6ES7212-1AE31-0XB0	Simatic S7-1212	V 3.0.2
7	Siemens	6ES7155-6AU00-0AB0	Simatic ET 200SP	V 3.3.0
8	Siemens	6ES7314-6EH04-0AB0	Simatic S7-314*	V 3.3.0
9	Siemens	6ES7516-3FN01-0AB0	Simatic S7-1516F*	V 2.0.5
10	Siemens	6ED1052-1CC01-0BA8	Logo! 8*	1.81.01
11	Phoenix	2700974	ILC 151 ETH	V.4.42.04
12	Phoenix	2985330	ILC 150 ETH	V.3.94.03
13	Phoenix	2700975	ILC 171 ETH 2TX	V.4.42.04
14	ABB	1SAP120600R0071	PM554-TP-ETH	2.5.4.15626
15	Crouzet	88981133	em4 Ethernet	1.2.75/1.0.27
16	Schneider	TM221CE16T	Modicon M221	1.5.1.0

* Achilles Level 2 Certified

3.3.5.3 Devices under Test (DuTs)

The DuTs are PLCs from different vendors as introduced in Table 3.2 in Section 3.1. This is a variety of devices, which represents a sample of the current market. A summary of evaluated PLCs with the corresponding firmware version is given in Table 3.8.

The aim is to identify and measure a worst-case scenario. Hence, each PLC was configured to switch a digital output at the maximum rate. This was configured in a cyclic task and only changed if necessary (e.g. freewheeling task). This called for device-specific configurations, especially setting the cycle time to the device-specific minimum, if applicable.

The default settings for all controllers are set, wherever possible. Of special interest are parameters for communication overhead. For the used Siemens devices, the default at 20 % is kept. Wago allows setting a data rate limit; however, this setting was disabled

by default (see Section 3.3.6.4 for effects of this setting). The used user control program was simple; it only switched the value of an output from 0 to 1, and vice versa.

The default configuration of the PLCs was changed only when necessary for the experiments. This included IP configuration and cycle time. Furthermore, there were no dependencies in the application code of the PLC, which required communication.

3.3.5.4 Protocol Implementations

In Table 3.7, the used protocols are summarized. For most of the protocols, off-the-shelf tools are used. If no standard tool was available, an implementation for the necessary tests was done. With the off-the-shelf tools, there was not much control over the sent packets. As a result, custom implementations for some protocols were used. All custom tools were implemented in *Go* and were capable of saturating the outgoing Gb/s Ethernet link of the attacking VM.

syn_spam – This implementation uses hard-coded SYN packets with no additional TCP options set.

arp_spam – RFC 826 defines multiple variants for ARP requests. The standard uses the following abbreviations: Sender Protocol Address (SPA), Sender Hardware Address (SHA), Target Protocol Address (TPA), and Target Hardware Address (THA). The following four ARP request variants were implemented:

1. Who has (ARP 1)
2. Probe (ARP 2)
3. Gratuitous ARP Request *TPA=SPA*, *THA=0* (ARP 3)
4. Gratuitous ARP Reply *TPA=SPA*, *THA=SHA* (ARP 4)

gre_spam – This implementation uses Generic Routing Encapsulation (GRE) SYN packets with no additional TCP options. In this work, GRE packets are tested as modern DoS attacks sometimes use such packets [Sea16].

snmp_spam – The implementation uses SNMPv1 with a hard-coded community string:

```
302902010004067075626c6963a01c0204036a5f430
20100020100300e300c06082b060102010101000500
```

3.3.5.5 Methods

Although the actual procedure differed across the three sets of experiments, the basic procedure remained the same. Prior to each experiment, the DuTs were powered off and on so as to start with a clean system state. To make the experiments more convenient, the execution of individual experiments was automatized. To this end, the individual experiments were combined in a single large experiment definition for the experimental server. The gathered data was stored on the control server in a single file per phase and experiment. After the execution of all the experiments, the resulting files were downloaded for analysis.

3.3.6 Experiments, Results, and Discussion

In this section, the three series of experiments which were conducted are discussed. In the first series, the measured reaction of devices under different loads of SYN packets is described. In the second series, the measured reaction to different protocols is illustrated. In the final series, the impact of scanning tools is assessed.

3.3.6.1 Increasing SYN Loads

As a baseline for the communication robustness of the tested devices, a series of tests (*hping3* SYN flood) with increasing inter-packet delay is performed. Every *hping3* attack lasted 60 s, followed by a 30 s idle phase. The delay between the flooding was created by the wait parameter of *hping3* (`hping3 -i u<wait for x microseconds> <IP>`). Through this, after each packet, *hping3* waited x microseconds until the next packet was sent. The mathematical symbols used in the formulas can be found in Table 3.9.

Table 3.9: Symbols used in formulas.

Symbol	Description
n	Number ob measurements of one segment/test
t	Time in microseconds of one PLC cycle
$\bar{t}$	Mean cycle time
$\bar{t}_{idle}$	Mean idle cycle time without network load
Δt	Factor, by which the cycle time is shifted compared to idle

The resulting mean cycle time for comparison is used. The mean cycle time of each segment, where one network load is tested, was calculated as shown in Equation 3.3.

$$\bar{t} = \frac{1}{n} \cdot \sum_{i=1}^{n} t_i \tag{3.3}$$

For better comparability, the results of dividing them by the mean idle time are normalized (see Equation 3.4). This results in Δ, which is a factor in deviation between attack and idle.

$$\Delta t = \frac{\bar{t}}{\bar{t}_{idle}} \tag{3.4}$$

The mean cycle times without network load ($\bar{t}_{idle}$) of the different PLCs are listed in Table A.5. An overview of the results is given in Figure 3.12.

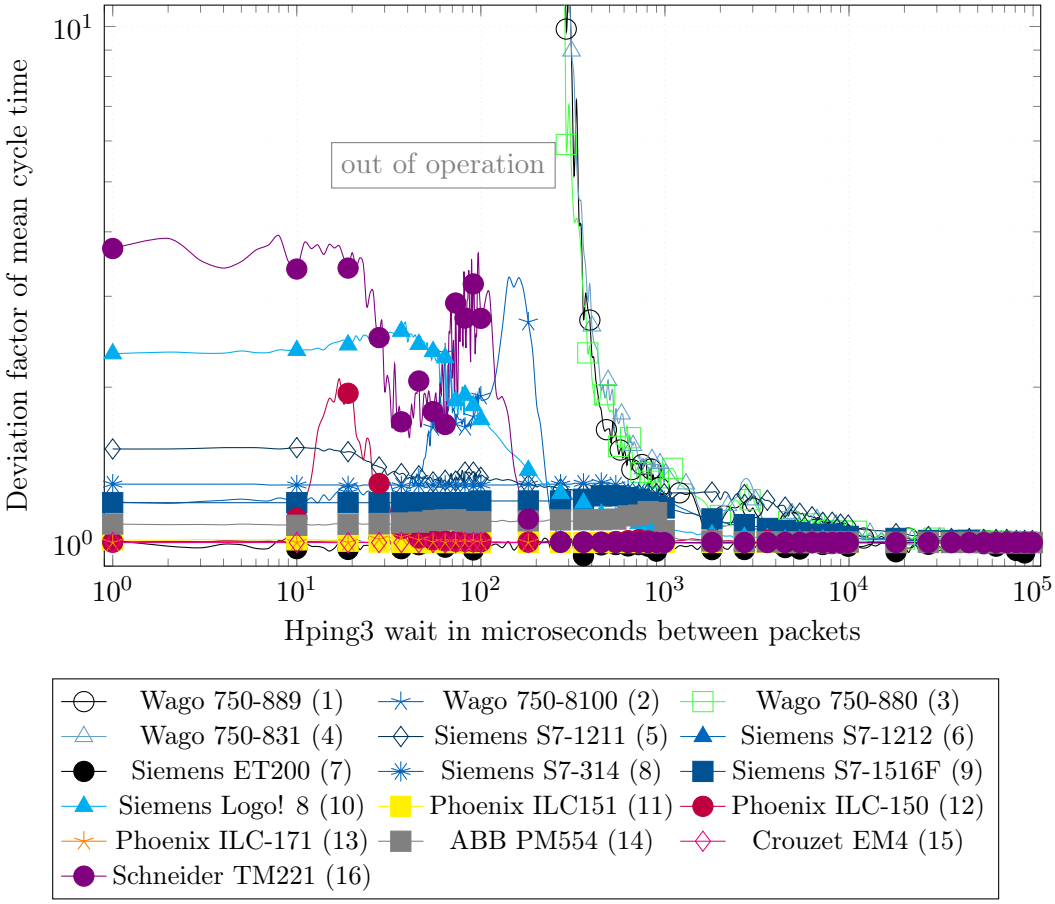


Figure 3.12: Overview of a controlled attack on PLCs with delays during packets, to achieve different network loads and measure the cycle time deviation.

The measurement yield that for some PLCs (5, 8, 9, 10, 14, and 16) a higher network load led to higher cycle times.

For some controllers (1, 3, and 4), an "out-of-operation" state under specific data rates is observed. We defined a device as out of operation if its cycle time was increased by a factor 10 or more.

Some PLCs (2 and 12) were not influenced at the maximum packet flooding but at lower rates. This shows that it is not always useful to execute a DoS attack at the maximum available data rate.

In the detailed view in Figure 3.13 and Figure 3.13, it can be observed, that only the Crouzet EM4 (15) performs very well, during the increasing network load test. Furthermore, three devices (6, 11 and 13) in the testbed were only little influenced by the *hping3* flooding attacks.

During the *hping3* measurement, the mean cycle time of the Siemens ET200 (7) somewhat decreased, meaning that the device ran faster at different packet rates.

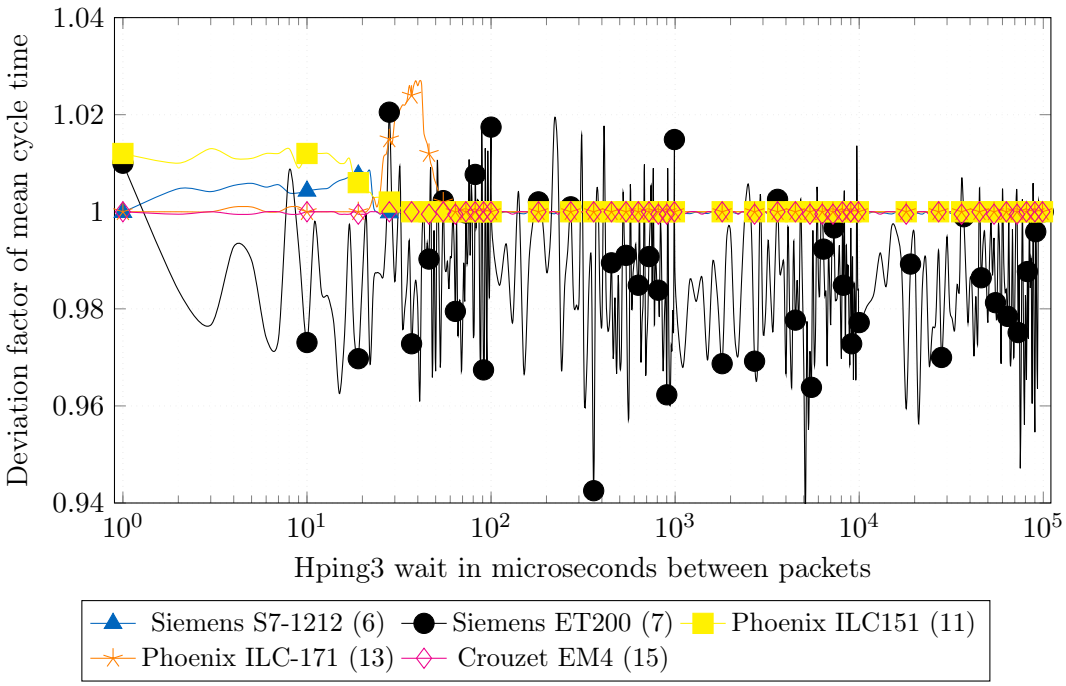


Figure 3.13: Detailed view of PLCs, which are less influenced by packet flooding.

However, most of the PLCs were affected, and further analysis showed that only the Crouzet em4 (15) was not influenced at all by the tests.

Conducting all the experiments summarized in Figure 3.12 took about a month. These experiments show that most devices can be influenced by sending SYN packets at a defined rate. Since SYN packets already have an influence on devices, it can be expected that higher-level protocols such as HTTP, Simple Network Management Protocol (SNMP), and ICS-specific protocols will be even more effective. This is due to additional resource consumption at higher levels of the network stack. In the following, a more detailed analysis of this phenomenon is presented.

3.3.6.2 Detailed Analysis of Protocols

Each experiment in this series had four phases. First, the device to test was powered off and on to guarantee a clean system state. The actual attack phase was flanked by two idle phases. The idle phase prior to the attack served as a reference to determine the impact of the attack. The post-attack idle phase was intended to observe any possible long-term effects of the attack. Each phase lasted for 600 seconds. There was a 60-second break between successive experiments.

Different impacts on the PLCs cycle time during the attacks could be observed. Owing to space constraints, the impact is categorized into six different effect classes defined in Table 3.10. For each class, only the worst-case scenario observed is presented. The results are detailed in the Appendix A.2.

Table 3.10: Cycle time classes, which are observable during the attacks.

Class	Description
Class 1: PLC "Stops"	No changes to the output during attack
Class 2: High Deviation	Over a factor of 100 slower during attack compared to idle
Class 3: Medium Deviation	Up to a factor of 100 slower during attack compared to idle
Class 4: Increased Variance of Cycle Times	High variance of cycle time during attack compared to idle
Class 5: Faster Cycle Time	Cycle time is getting faster during attack compared to idle
Class 6: No Measurable Influence	Cycle time during attack and in idle has no measurable difference

The results of the measurements are shown in a boxplot with calculated arithmetic mean (▼) and median (–). The quantiles are 25 % and 75 %, with whiskers up to factor 1.5 of the box.

Class 1: PLC "Stops" An extreme behavior is that the PLC "stops" during the attack. This means that the outputs are not updated during packet injection. Figure 3.14 shows this behavior during an ARP flood attack.

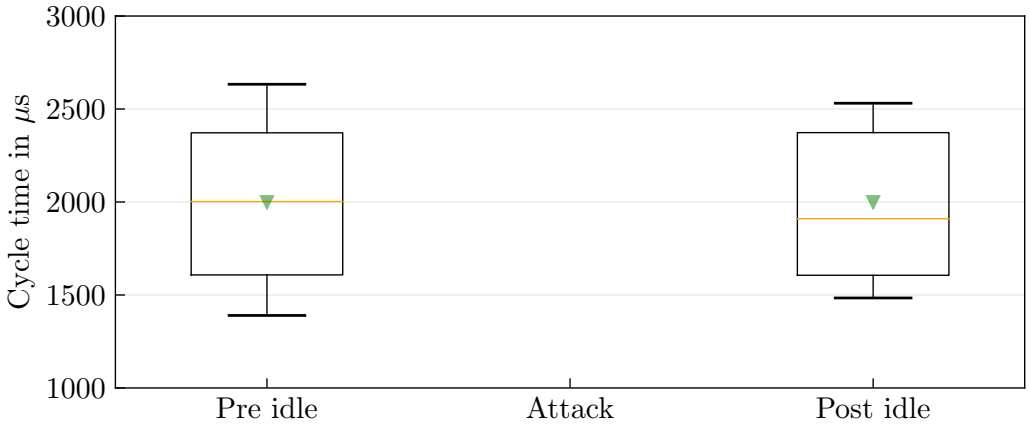


Figure 3.14: Boxplot of a Wago 750-831 (4), where the PLC stops during ARP 3 flooding.

It is interesting to note that an ARP flooding attack can be sent to the whole broadcast domain. Therefore, all devices that can be influenced in a broadcast domain can be affected by this type of attack. However, ARP requests do not cross subnet boundaries, and as such only local adversaries can apply ARP flooding attacks.

In the example given in Section 2.2, the valve remains open if it is opened when the attack has started. Thus, the material will not be filled into the container but next to it. This can obviously lead to all sorts of trouble.

Devices in this class clearly exceed the requirements for a certification as described in Section 3.3.3.

Class 2: High Deviation During a flooding attack, the cycle time of some controllers increases by several seconds. In the measurement illustrated in Figure 3.15, the cycle time increases up to 5 seconds. In the example, this influence is achieved through UDP flooding. During pre- and post-idle phases, the PLC functions as expected and toggles about 2 ms.

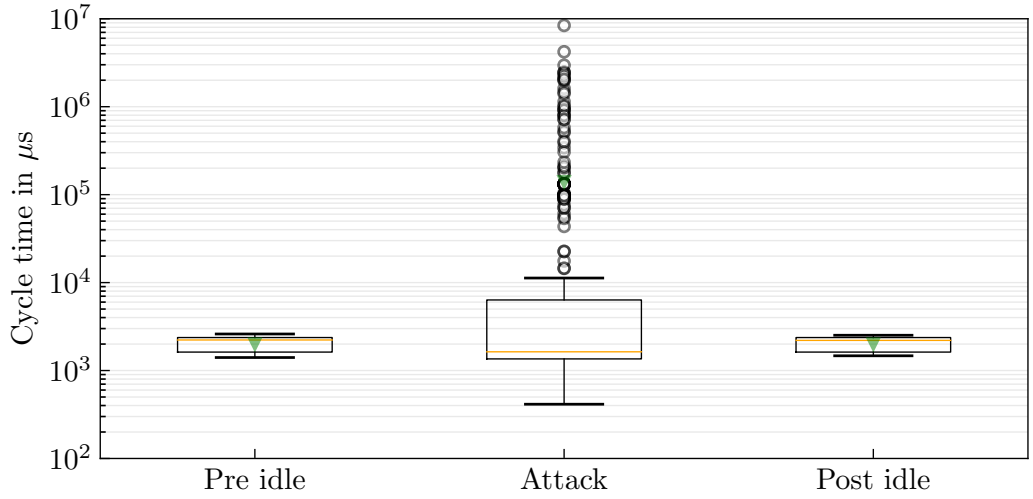


Figure 3.15: Boxplot of UDP flooding attack on a Wago 750-889 (1), resulting in a high deviation of the cycle time.

Considering the example in Section 2.2, the PLC will nearly stop reacting. More precisely, the outputs will remain at the current level (on or off), only being updated every few seconds. This means that, if the valve is opened at this moment, it will remain opened for several seconds, and the convey belt will still move forward, resulting in a similar effect to the one described before. Devices in this class break the requirements for certification as described in Section 3.3.3. Neither do the devices maintain essential services, nor is the deviation smaller than 4 %.

Class 3: Medium Deviation Another effect that can be observed is a "medium" deviation of the cycle times. Devices in this class show increased cycle times below one second. Figure 3.16 shows an example. The device toggles in idle with about 2 ms. During UDP flooding, the cycle time is up by a factor of about 40.

Owing to this factor, the controller processes everything at a slower rate. It is possible that a process is still running correctly, but at a much slower pace or imprecisely.

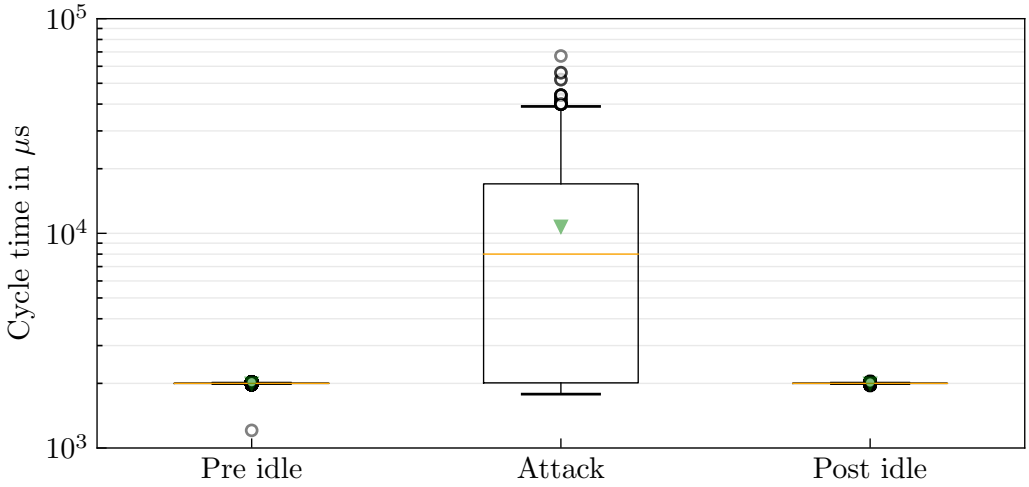


Figure 3.16: Boxplot with medium deviation during UDP flooding with hping3 of the Schneider TM221CE16T (16).

Considering the example in Section 2.2, the container may have already passed the valve when the sensor input is processed. Therefore, the loading could miss the container.

As for classes 1 and 2, the criteria for certification would not be met. Additional to the impacts described in the previous effects, this could lead to a slower production, resulting in a loss of profit. In the example given in Section 3.3.5 from Schneider Electric, the goods will not be filled into the container.

Class 4: Increased Variance of Cycle Times With regard to the results in Figure 3.17, the cycle time is only minimally affected by packet flooding attacks. The boxplot as well as the mean value shows a delay of about 25 %. However, the variance is still large under the attack. On some controllers, the boxplots and mean value representations are misleading. In fact, there may be effects which are only viewable in other representations.

Figure 3.18 shows the kernel density estimation in a histogram plot. The number of bins is set to 1,000 in order to get a good resolution of the distribution. In this, the cycle time is plotted against their probability (density). With this representation, the influenced cycles are clearly visible.

The density plot in Figure 3.18 of the cycle time shows two peeks in idle, for low and high electrical signals. It can be noticed that the low and high signals do not have the same length. In fact, the high signal is longer than the low signal. During the attack, the cycle time increases and new peeks are formed. The two peeks are shifted by a factor of about 2, which is not obvious in the boxplot but is visible in the density representation. This, in turn, means that some cycle times are twice as slow. Regarding the example (Section 2.2), the result would be variable filling quantities.

For devices in this class, it is not entirely clear if they would fulfill the requirements of the certifications. This is mainly due to the relatively broad definition of the used

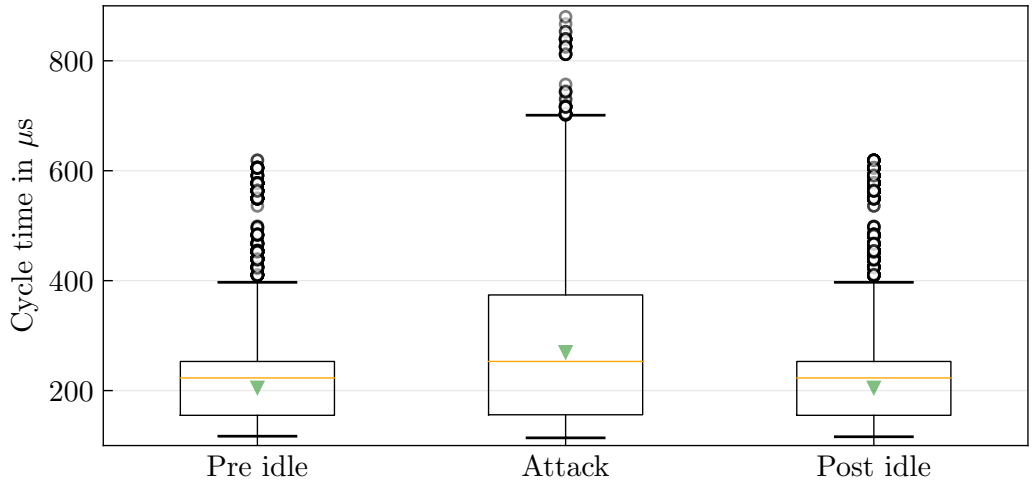


Figure 3.17: Boxplot, while an attack on a Siemens S7-314 (8) is generating a high network load with the S7Com implementation of zgrab.

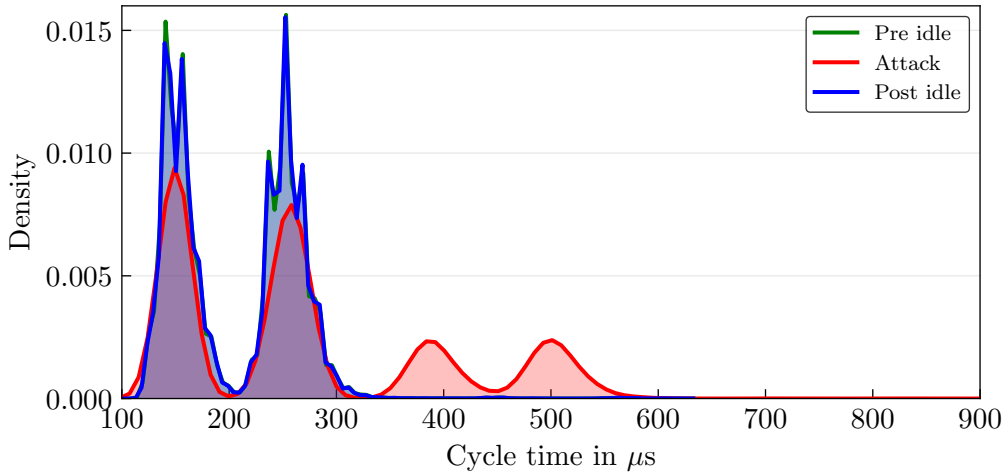


Figure 3.18: Probability Density Function to view the distribution during the S7Com flooding of a Siemens S7-314 (8) with zgrab.

classes. However, for the selected device, the answer is still clear. For the Siemens S7-314 (8) under test in the study, the maximum communication load was set to 20 %. As such, the assurance of the device was exceeded. In addition, the Siemens S7-314 (8) is Achilles level 2-certified, but the findings indicate that the device is still susceptible to network-based attacks on the electrical side of the device.

Class 5: Faster Cycle Time By considering only the mean cycle time of the PLC, no changes can be determined. However, on a closer look, the cycle time appears to be more spread and some cycles become even faster during an attack. An example under a UDP flooding attack is given in Figure 3.19.

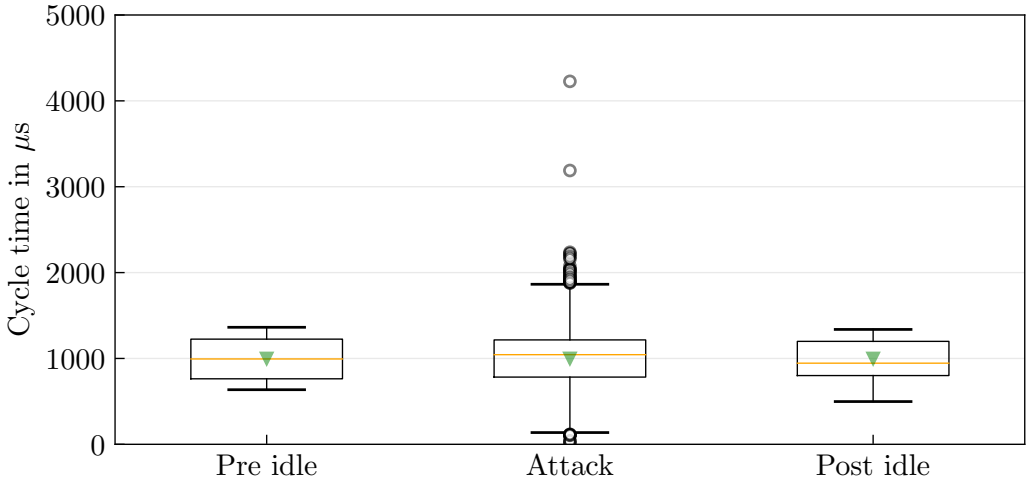


Figure 3.19: A boxplot representing a shorter cycle time of a Phoenix ILC151 (11) during Modbus/TCP flooding with zgrab.

This effect could be caused by a kind of buffer overflow of the network stack and result in a packet drop. Furthermore, maybe this is achieved by blocking or crashing the network stack, thereby allowing the CPU to process the control process faster. In a real-world example, this could make the process unpredictable if it gets faster than usual. In the context of the used example (Section 2.2), the container could not be positioned correctly, or the valve could close earlier than expected, leading to insufficient filling.

Considering the current knowledge, the certification programs listed in Section 3.3.3 do not take into account that PLCs could work faster. As such, devices in this class would meet the requirements while still being prone to attacks.

Class 6: No Measurable Influence Some tests indicated no measurable influence. Figure 3.20 shows an example where the three phases are similar.

This PLCs has a dual MCU setup, where one controller handles the communication and the other one runs the control program. The communication controller is still struggling with the network traffic and is unresponsive to pings. Nevertheless, the controller

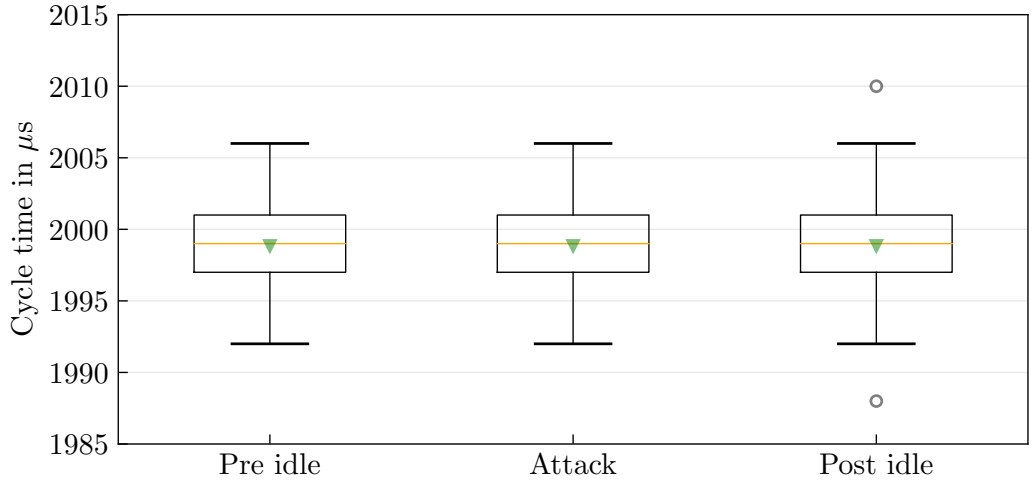


Figure 3.20: Example of a boxplot with no measurable influence on the Crouzet em4 (15).

responsible for the physical process is working as expected. However, the architecture was probably created as a necessity and is not designed for security. In further analyzes, some vulnerabilities in the Crouzet em4 (15) were found, but these are not yet disclosed at the current time. Therefore, there is still room for improvement, even if this PLC performs well in these tests.

CPU Load During Attacks In the used testbed, most devices are based on Real-Time Operating Systems (RTOSs) and the CPU usage cannot be supervised. However, the Wago 750-8100 is based on Linux (with root access), which allows the measurement of CPU utilization during attacks. The device has a single-core 600 MHz ARM processor with 256 MB of RAM. The flooding attack started after 10 ticks and stopped after 20 ticks. Figure 3.21 illustrates the CPU usage during the experiment. In regular operation, the software Interrupt Request (IRQ) has a CPU usage of up to 5 %.

During the attack, the software IRQ, which, among other things, handles the network traffic, increases to nearly 100 %. In case of an interrupt, the regular software execution is halted and the interrupt is handled. A high interrupt load seems to affect the control software of the PLC, influencing the continuous execution and resulting in asynchronous cycle times.

3.3.6.3 Effects of Active Scanning

In the literature listed in Section 3.3.2, it is stated that active scans should be avoided. However, this claim is not backed by empirical evidence. Using the testbed, it is possible to precisely assess the influences of an active scan. For this comparison, a selection of active scanners *Nmap* 7.60 [Lyo09], *PLCScan* version 0.1 [Efa12], and *RiskViz Search Engine* [Ris15], which uses *ZGrab* [Dur+13] for application scanning) to analyze the behavior of ICS components under an active scan is used. To conduct this measurement,

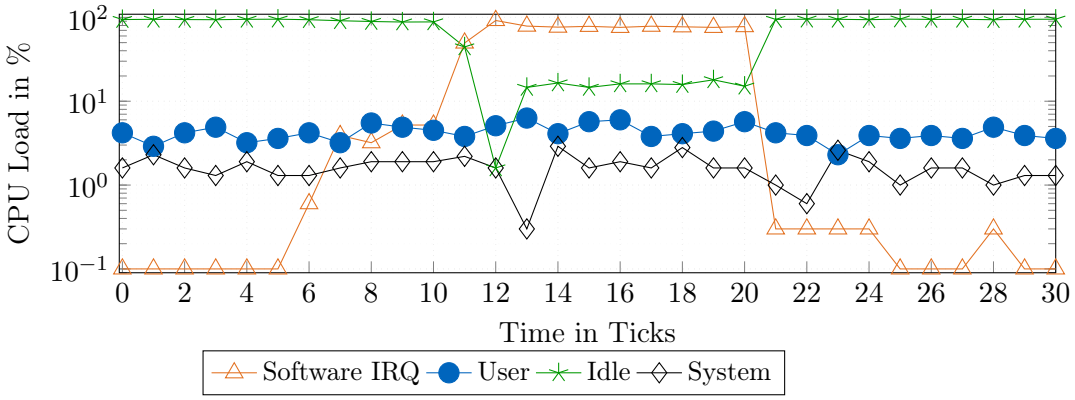


Figure 3.21: CPU load during SYN flooding attacks of a Wago 750-8100 (2) with hping3.

the default configuration of the scanners was used (IP as parameter). For this analysis, a control system (Wago 750-880 (3)), which is already known as influenceable, is selected. All network scanners require less then 30 seconds to analyze a single device. Figure 3.22 summarizes the measured effects of these three scanners compared to the idle cycle time.

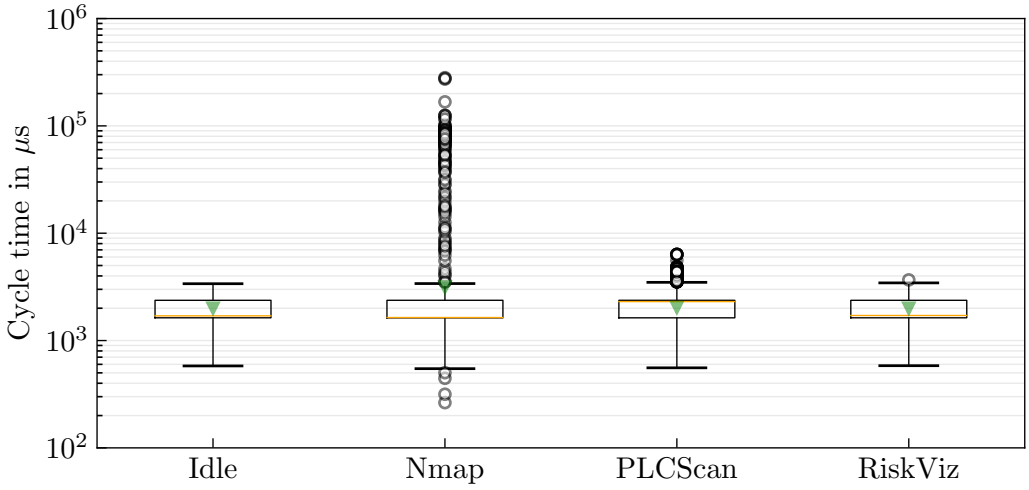


Figure 3.22: Influences of active scanners on a Wago 750-880 (3).

In idle, the PLC has a cycle time between 500 µs and 4000 µs with a mean of about 2000 µs. During a *nmap* scan, which scans in default the most 1000 common ports, the cycle time increases up to 300 ms. With *PLCScan*, which scans only two protocols (Modbus/TCP and S7comm), a cycle time up to 7 ms can be measured. The *RiskViz Search Engine* scans at low rates with about 800 kb/s and effects the cycle time slightly. Figure 3.23 illustrates the influences of the cycle time of the three network scanners over the 30-second scan time. The data used for the plots in Figure 3.22 and Figure 3.23 are

derived from the same scan. Above all, it can be seen that *nmap* and *PLCScan* influence the cycle time over a longer period of time.

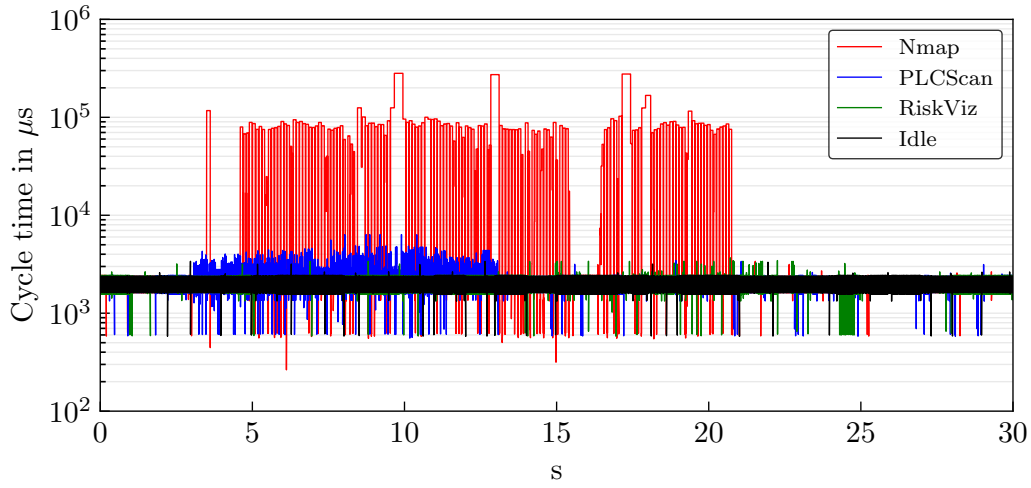


Figure 3.23: Influences of different network scanners on a Wago 750-880 (3) during network scanning.

The presented analysis of active scanning in ICS networks shows that there are measurable influences for some devices. Therefore, scanning of ICS networks presents a chicken or egg problem. Specific devices should not be scanned. On the other hand, it is not known which devices are in a network prior to a scan. The only trivial option, if a scan cannot be avoided, is to keep the data rate as low as possible.

3.3.6.4 Mitigation and Future Work

In order to secure assets, systems, machines, and networks against cyber threats, it is necessary to implement and maintain a state-of-the-art industrial security concept [Sto+11]. This includes validation of the communication robustness of single components, for example, with flooding tools. The results with these testing tools have shown that there is a lack of secure ICS component architectures. Furthermore, existing tests are not vendor-independent or transparent to the public.

Data rate limitations on the network provide a possible software solution. This feature is already implemented by controllers from Wago (1,2,3,4). The measurements show that this option can be an efficient mitigation (see Appendix A.2). However, the Wago 750-8100 is not prone to flooding attacks for data rates of 16 Mb/s and below. The effect of flooding is drastically reduced for the remaining devices for data rates of 1 Mb/s and below. Only the longest measured cycle time is increased. There is no change in the mean cycle times. For data rates of 8 Mb/s and above, the effects measured without the feature are still evident. This possibility of rate-limiting indicates that there are other configuration options which could prevent cycle time influences.

Another software-based solution would be RTOSs with hard real-time scheduling like *FreeRTOS* [Ina+11]. Such schedulers guarantee a certain task tick time. If mapped to PLCs cycle times, the expected characteristics on the electrical side could be guaranteed.

Besides software solutions, specific hardware configurations provide another option [Nao+17]. A possible configuration could be a multi-controller setup, for example, two dedicated controllers, or a System-on-a-Chip (SoC), where one controller processes the real-time task and the other controls communication. A challenge in this scenario is to prevent feedback effects between the controllers. A hardware solution is obviously only possible for new products, but it would increase production and integration costs.

3.3.7 Conclusion

In this section, the communication robustness of PLCs under network flooding attacks was tested. The results show that the electrical side of PLCs is prone to network flooding attacks. Variances in the runtime of control programs can have disastrous effects. This differs from well-known DoS attacks, as in this case physical processes are involved. A successful exploitation of found weaknesses can lead to major environmental and safety impacts, as these devices usually control physical processes in critical infrastructures.

The presented analysis shows that most of the PLCs are affected, irrespective of manufacturers. With the exception of one device (Crouzet em4 (15)), all the devices in the CoRT showed measurable changes during network flooding attacks. Some of the controllers even "stopped" operating and did not update their outputs for the duration of the attack. Additionally, it is shown that active network scans have a detectable effect on the electrical side of PLCs. These results are relevant as active network scans are a current trend in academic research. Network scans with high data rates may influence Internet-facing PLCs accidentally. Taking this possibility into account is recommended when assessing the risk of a planned project.

Apart from casualties, network-based Distributed Denial of Service (DDoS) attacks are another current trend [Sea16]. This is mainly because network flooding attacks are technically simple. In the presented scenario, an attacker can influence an actual physical process. This increases the thread imposed by DDoS attacks.

To summarize the research in this section, it can be said that a secure system configuration is of great importance. Furthermore, it is good to see that Wago offers at least a partial function mitigation feature. However, operators need to learn about and use configuration features to enable a secure operation. In the future, manufacturers should launch products which are secure by design/default.

All affected vendors have been informed about the findings using an adapted responsible disclosure.

3.4 Dual-MCU Setup for Robust Industrial Internet of Things Devices

Contents of this section

Parts of this section have already been published in the paper "A Secure Dual-MCU Architecture for Robust Communication of IIoT Devices" at the 8th Mediterranean Conference on Embedded Computing (MECO) 2019 [Nie+19b].

3.4.1 Introduction

Attacks are particularly dangerous for IIoT devices such as PLCs, as they interact with the physical world, resulting in serious damage or injury. Haddadin et al. showed how strong robots that interact with humans can injure them [Had+07]. This cannot only happen because of a bug in the program, but also due to DoS attacks, as introduced in the previews section (Section 3.3). Therefore, the communication part of PLCs should be designed in such a way that it does not influence the control part.

In this section, a dual MCU setup to ensure a resilient controlling for IIoT devices like PLCs is introduced, which is a possible solution for the demand of secure architectures in the IIoT. Moreover, a PoC implementation with a benchmark and a comparison with a standard PLC under DoS attack is provided.

The section is structured as follows. Section 3.4.2 explains the methodology behind a dual controller setup for secure controlling and gives the necessary background. In Section 3.4.3, the PoC implementation is illustrated. Section 3.4.4 compares the robustness against Denial of Service attacks of the secure architecture and a commercial PLC. Finally, a conclusion is given in Section 3.4.5.

3.4.2 Concept and Background

There are secure architectures for complete chips with patents available on the market [Fru+05]. This concept requires deep knowledge and no standard MCU can be used, which may make the end product expensive. Alves et al. introduced an open-source Linux-based PLC and implemented an Intrusion Detection System (IDS) as a DoS protection [Alv+18]. However, the used method only partially protects against DoS attacks, implementation errors, and zero-day vulnerabilities. Moreover, it is currently only possible for Linux-based systems. They also show that the program execution time on

current PLCs varies during their tests. As a result, secure architectures for IIoT devices are necessary [Cár+08a; Cár+08b].

The IIoT architecture presented in this section comprising two MCUs, a network MCU (NW-MCU), and an IO-MCU handling the connection to the sensors and actuators, offers the following advantages compared to most existing solutions:

- Well-controlled communication between the two MCUs reduces intentional and unintentional influencing of the physical process.
- Compared to a software solution, e.g. based on a single MCU and a RTOS [Ngu+15], a vulnerability in the hardware or software of the network MCU will not directly influence the IO MCU.
- The reduced code size on the IO MCU reduces testing effort, e.g. for safety certifications, because the critical code size is smaller.
- By using a unidirectional connection [Zil+10] from the IO MCU to the network MCU, it is possible to monitor without influencing.

Figure 3.24 shows the principle of the presented secure architecture introduced in this section with a dedicated network and IO MCU. This architecture differs from the Crouzet em4 (15), in that a network controller was not added afterwards, but instead focusing on security and a strong separation. The network MCU handles the communication over Ethernet and Modbus/TCP. Configuration information and control data are submitted without influencing the IO MCU. For this to happen, the IO control must be handled in a predefined time slice by the IO MCU to ensure a certain response time. The uncritical part is responsible for the network communication and the critical control part, for the physical process. Influences on this critical part affect the real world.

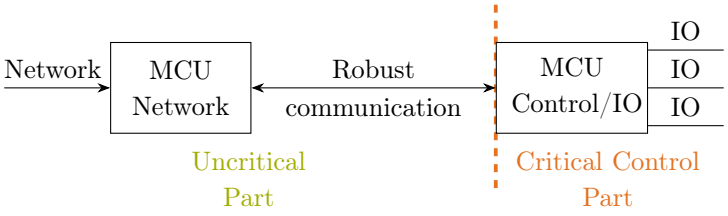


Figure 3.24: Example architecture of a dual MCU setup for robust controlling.

The cycle time of a PLC is the time for the execution of a complete program cycle including the communication (see Section 2.4). It depends on the processing time of the program, which is determined by the number of instructions. Higher prioritized tasks interrupt the cycle, thereby delaying the actual cycle. The mathematical symbols used in the following formulas can be found in Table 3.11.

This cycle time (t_{cycle}) is the sum of the phases that are processed at each pass. In a simplified representation with one task, there are four phases. First, the inputs get read in (t_{read_in}); this step has a constant processing time. It is independent of input changes

Table 3.11: Symbols used in formulas.

Symbol	Description
t_{cycle}	Summary of the time for all phases
t_{read_in}	Time to read in the inputs and create the process image
t_{comm}	Time for communication and housekeeping
t_{calc}	Time, which is necessary for the calculation (user program)
t_{write_out}	Time to write back the process image
t_{delay}	Delay, which is necessary to get a fixed cycle time

for cyclic tasks. Thereafter, the communication is handled (t_{comm}). The communication depends on external participants and can have different runtimes. For example, if the bus speed is slow or the data size is big, the communication part takes longer. At the end, the necessary calculation (t_{calc}) is done and the outputs are written back (t_{write_out}). In this case (Equation 3.5), the cycle time (t_{cycle}) is free-running and varies in time.

$$t_{cycle} = t_{read\_in} + t_{comm} + t_{calc} + t_{write\_out} \tag{3.5}$$

For the introduced approach, the IO MCU must have a constant runtime independent of the network MCU, which results in the requirement of a constant cycle time (t_{cycle}). This could be achieved by setting a timeout to the communication between the network MCU and the IO MCU. To get a constant cycle time (t_{cycle}) of the IO MCU, a delay (t_{delay}) is inserted to equalize time fluctuations. The calculation of the delay is shown in Equation 3.6.

$$t_{delay} = t_{cycle} - (t_{read\_in} + t_{comm} + t_{calc} + t_{write\_out}) \geq 0 \tag{3.6}$$

It must be ensured that the cycle time (t_{cycle}) is higher than the maximum time, which can pass through the four phases in Equation 3.5. Therefore, the maximum time of each phase must be limited depending on the desired cycle time. The behavior, which is illustrated in Equation 3.6, must be represented by the IO MCU and runs independent of the NW MCU.

3.4.3 Proof of Concept Implementation

To prove the feasibility of the introduced concept, a PoC implementation is necessary. The focus is on the robust communication between the two MCUs and the real-time behavior of the IO control during flooding attacks.

3.4.3.1 Proof of Concept Hardware

The hardware of the secure architecture consists of the network board and the IO shield, which are connected. Table 3.12 shows the specification of the hardware used. The MCU on the network board is faster but more expensive than the IO MCU.

Table 3.12: Specification of the used hardware for the PoC implementation.

Hardware	Network Board	IO Shield
Board design	STMicroelectronics	custom
MCU	STM32F767ZIT6	STM32F030F4P6
Core	ARM®Cortex®-M7	Arm®Cortex®-M0
Clock	up to 216 MHz	up to 48 MHz
RAM	512 kB	4 kB
Flash	2 MB	16 kB
MCU price	∼ 10 €	∼ 1 €

For PLCs, which often cost several hundred dollars, an additional IO MCU would not render the final product much more expensive. Furthermore, the proposed concept is possible with different MCUs, depending on the later demands.

Network Board Hardware For the network MCU, a development board (STM NUCLEO-F767ZI [STMa]) is used. This MCU was chosen because, on the one hand, this series is relatively energy-efficient, which is also used in standard PLCs, and on the other, offers enough performance for further evaluations. The board provides an RJ45 Ethernet connector and an Arudino™ Uno V3 connector. Additionally, an ST-Link programmer with Serial Wire Debug (SWD) and serial communication is attached to the MCU for programming and debugging output.

IO Shield Hardware The IO shield is a custom design, as there is no suitable shield for this purpose. Figure 3.25 shows the shield, which is designed to be compatible with the Arduino™ Uno V3 header. This makes the shield usable with many other boards. The schematic can be found in Appendix A.4

3.4.3.2 PoC Software

As explained in Section 3.4.2, the communication part between the two MCUs does not use a constant time during processing. This requires calculation and compensation to achieve a uniform runtime, resulting in a constant cycle time. The software that runs on the two MCUs is fundamentally different in terms of Random-Access Memory (RAM) and Read-only Memory (ROM) usage. Furthermore, in contrast to the IO MCU, the network MCU has no "hard" real-time requirements. Of course, this only applies if the network communication does not have real-time requirements.

Network MCU The network board has a much higher computing power than the IO board. It runs an operating system (*FreeRTOS* [Bar+08]) to handle the different tasks in a pre-emptive multi-tasking single-core implementation. As a result, the network communication with multiple subscribers can be handled through different RTOS tasks. For the network communication, the *Lightweight IP (LwIP)* [Dun01] stack is used. Figure 3.26

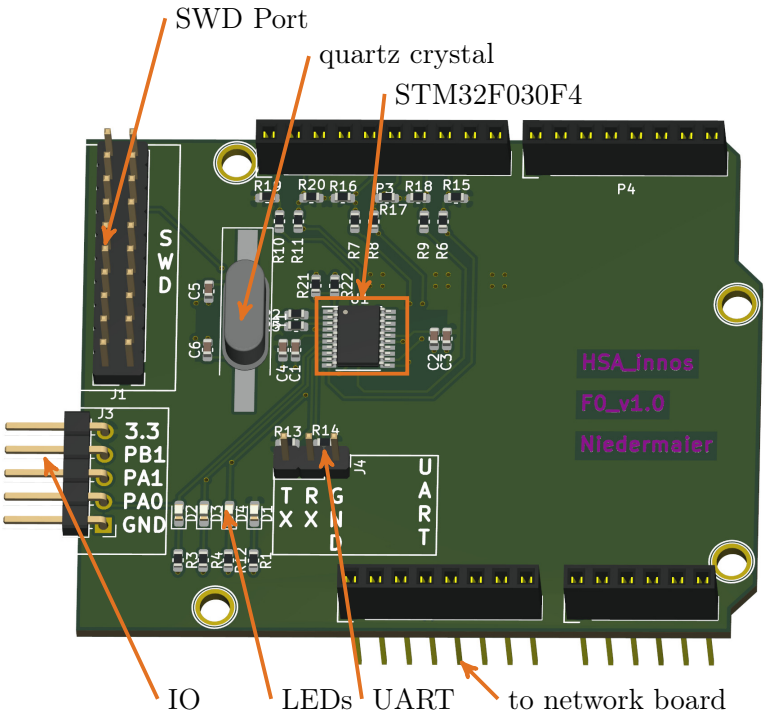


Figure 3.25: Rendered controller shield that is placed on top of the network MCU board.

shows the configuration web server running on the network board. This shows actual information, such as the uptime and the current state of the outputs. Furthermore, the cycle time of the outputs on the IO MCU can be configured.

For communication between the two boards, an Serial Peripheral Interface (SPI) with a speed of 13.5 Mb/s is used. The network MCU is the master and continuously transmits the information to the IO MCU. If the IO MCU does not respond within a certain time, the transmission is tried again after a delay.

IO MCU The IO MCU runs a bare metal system, with the usage of the STMicroelectronics (STM) Hardware Abstraction Layer (HAL). This makes later changes to the MCU easier if, for example, more performance is necessary. Within this HAL, the SysTick is set to 100 μs, which is also the resolution of all time-based HAL functions, such as the timeouts of the communication functions. The sequence of the program on the IO MCU is illustrated in Figure 3.27. The start represents the initial powering of the MCU, whereon the initialization of this is done.

After the initialization, a continuous cycle is executed. At the start of the cycle, the timer to measure the delay is reset. Thereafter, the inputs are read with the HAL functions and the SPI with a timeout of 500 μs is executed. The time is chosen so that there is enough time to transfer the necessary data and a cycle time of 1 ms is possible. After this, the calculation of the new output states is done by comparing the current

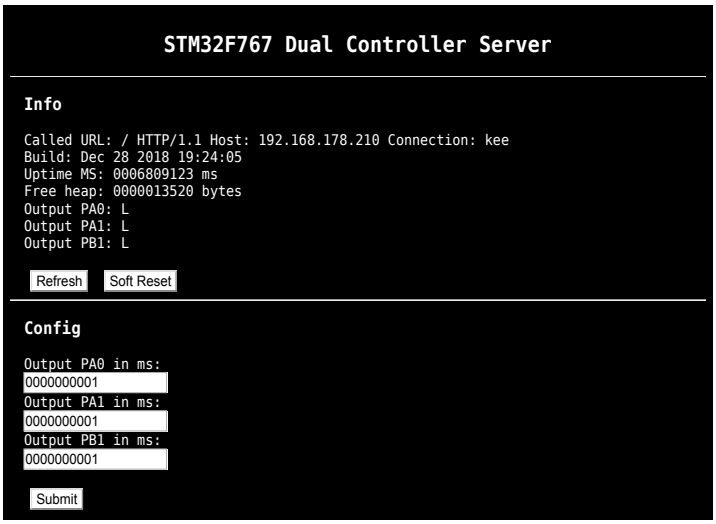


Figure 3.26: Website, running on the network MCU, showing some information and allowing configuration of the network and the IO MCU.

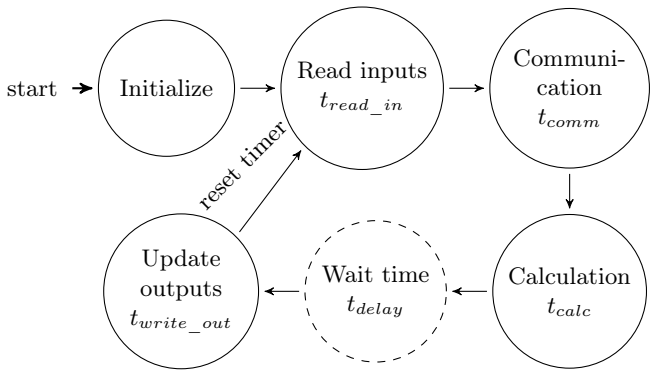


Figure 3.27: Program sequence to achieve a defined time behavior of the IO MCU. The dashed circle "Wait time" is the additional task compared to a standard PLC cycle.

cycle count with the configured cycle time. In these cycles, the varying timing, which is measured with the timer, must be compensated (see Equation 3.6) to get a constant cycle time. This is done in the wait state by holding it there until the desired cycle time is reached and then writing the previous calculated outputs back. In the PoC implementation, the cycle time is set to 1 ms. This is a common minimum cycle time for commercial PLC solutions. As a result, multiples of 1 ms can be used as the IO response time. This is common for current commercial PLCs.

To ensure robust communication over SPI, the receive and transmit functions on the IO are implemented in a blocking mode with a timeout. Interrupts and Direct Memory Access (DMA) are not used to prevent blocking and timing problems through many interrupts and memory overflows by overwriting buffer boundaries. Thus, the IO MCU could miss a transmission from the NW MCU to fulfill the real-time requirements of

the IO control. Furthermore, the send and receive buffers have fixed sizes and defined data structures, which avoid overflows and memory leakage of dynamically allocated variables. For the PoC implementation, the STM32F7 is configured as SPI master and the STM32F0 as SPI slave. This has the advantage that the STM32F7 as master can tolerate a higher timeout when sending and receiving, because this does not have to meet hard timings for IO control. In the PoC implementation the transmission from the STM32F7 to the STM32F0 has fixed 30 bytes, to configure the IO logic. 3 bytes are read back from the STM32F0 to transfer the status of the IOs. For the transmission, CRC32 can be used with the STM HAL (see Listing 3.1).

```
1 uint32_t HAL_CRC_Calculate(CRC_HandleTypeDef *hcrc,
2   uint32_t  pBuffer[],
3   uint32_t  BufferLength
4 )
```

Listing 3.1: CRC check of the transmition with STM HAL.

This was only used for the PoC implementation and should be solved cryptographically if used in a product, whereby the low performance of the STM32F0, especially the few RAM of 4 kB, must always be taken into account here.

3.4.4 Benchmarking

Figure 3.28 shows the PoC setup with the network board and the attached IO shield, which is used for the benchmark. For a secure operation, the interferences on the SPI bus must be considered so that the IO MCU is not influenced. These include flooding by the network MCU, invalid data, and shortcuts. The current PoC is not protected by cryptographic mechanisms. Nevertheless, the SPI communication cannot influence the IO MCU.

Additionally, to show the stability of the concept during network flooding attacks, the cycle time is measured during a flooding attack. For the measurements, a PicoScope 2208B USB oscilloscope is used. With this, the measured data can be exported and analyzed. Figure 3.29 shows the cycle time of the introduced implementation over time during pre-idle, *hping3* [San99] flooding attack, and post-idle. The jitter is only about 10 μs, which is equivalent to 1 % deviation. The cycle time is similar in all phases and is not influenced by the attack.

Furthermore, the introduced robust implementation is compared with a standard PLC. As a reference, the Wago PLC (HW:750-8100 SW:02.05.23(08)) number 2 from Table 3.2 is used, which is a current PLC from this vendor. This should not be regarded as an opinion about this product, but as a reference for comparison.

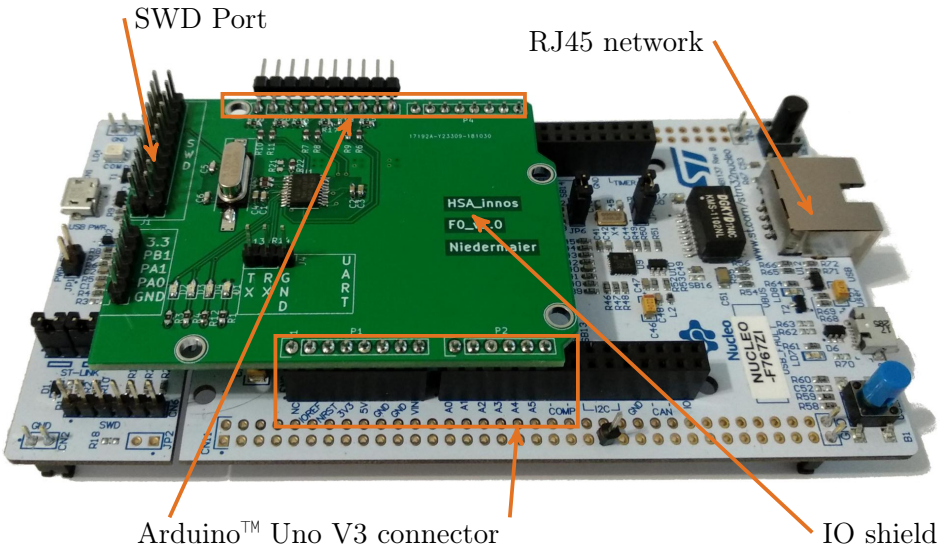


Figure 3.28: Image showing the complete setup with the network MCU board and the IO shield.

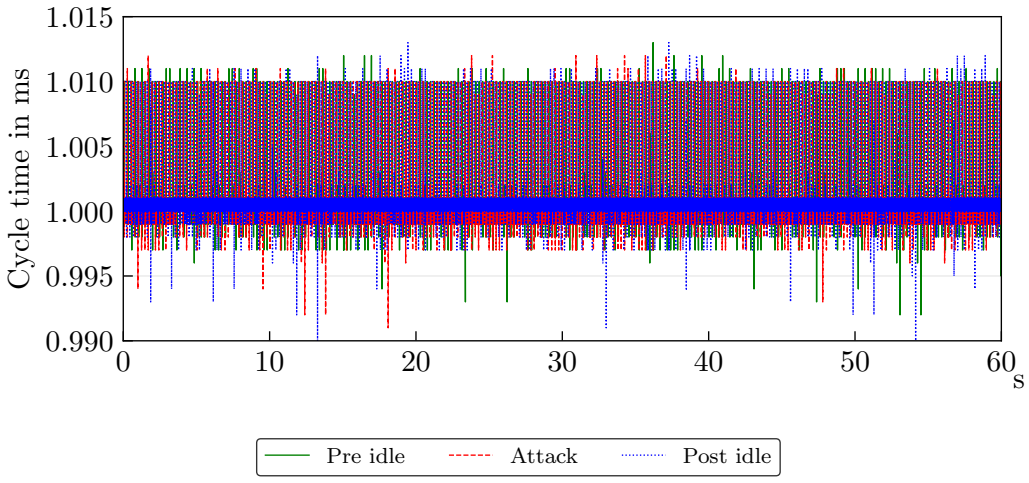


Figure 3.29: Time plot of the 1ms cycle time during pre-idle, attack, and post-idle of the introduced secure implementation. The jitter is about 10 μs, which is equal to a deviation of 1 %.

Figure 3.30 shows the boxplot of the introduced secure PoC implementation during pre-idle, attack, and post-idle. The measurement duration of each phase is 60 s. The PoC introduced in this work has a fixed cycle time of 1 ms (see Figure 3.29) and the PLC from Wago has a default of 10 ms. For this reason, the introduced implementation toggles the output every 10 cycles to allow a direct comparison. The maximum jitter during idle is less than 1 % for the secure architecture and about 300 % for the Wago PLC. The DoS attack is done with *hping3* in the flooding mode. No difference was observed in the

cycle time on the introduced PoC implementation in this work during pre-idle, attack, and post idle. On the common PLC (Figure 3.31), the attack slows down the cycle time noticeably, as shown in Section 3.3. The same scale was chosen for the y-axis (0 ms - 100 ms) in order to be able to compare the two box plots directly.

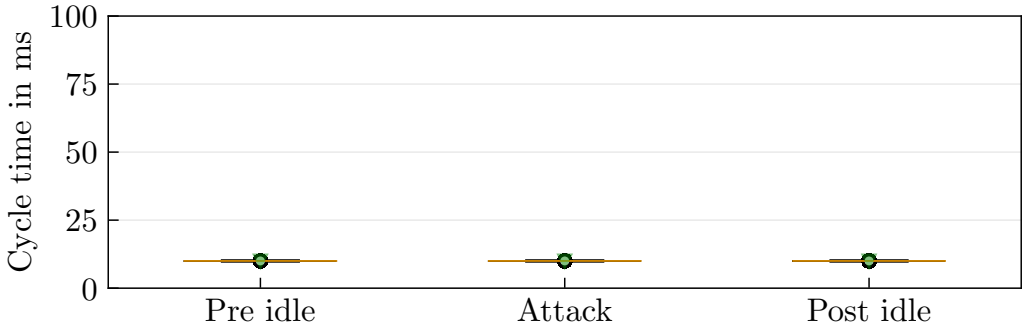


Figure 3.30: Boxplot of the cycle time of the introduced approach during hping3 attack. A constant cycle time is set to 10 ms during all phases.

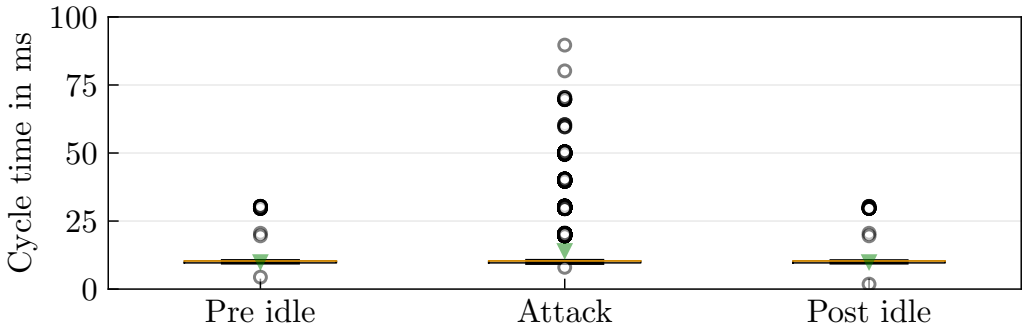


Figure 3.31: Boxplot of cycle time of the Wago PLC during hping3 attack, with variances during idle and influences during attack.

The density representation (Figure 3.32) shows that the introduced PoC implementation is stable and only varies in a small range. In contrast, Figure 3.33 shows that on a common controller a flooding attack could influence the cycle time. In this case, it ranges up to a cycle time of 100 ms (factor 10 slower), where the outputs of the PLC are not updated. The same scale was chosen for the x-axis (0 ms - 20 ms) in order to be able to compare these two plots directly. The comparison between the introduced secure architecture presented here and a current PLC shows that the proposed solution is feasible and stable.

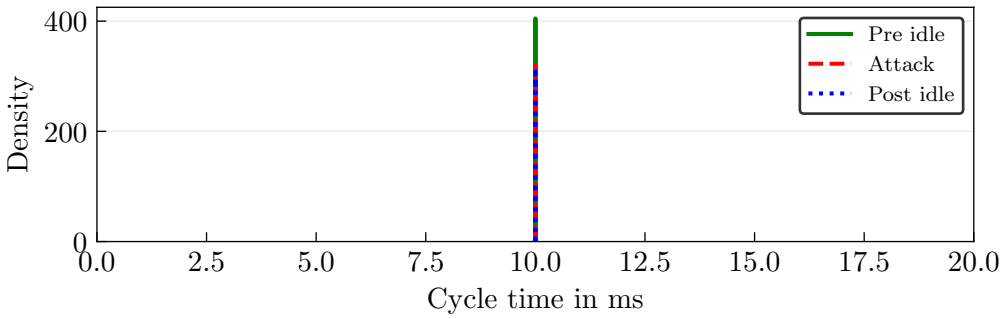


Figure 3.32: Density plot of the 10 ms cycle time of the introduced implementation during hping3 attack.

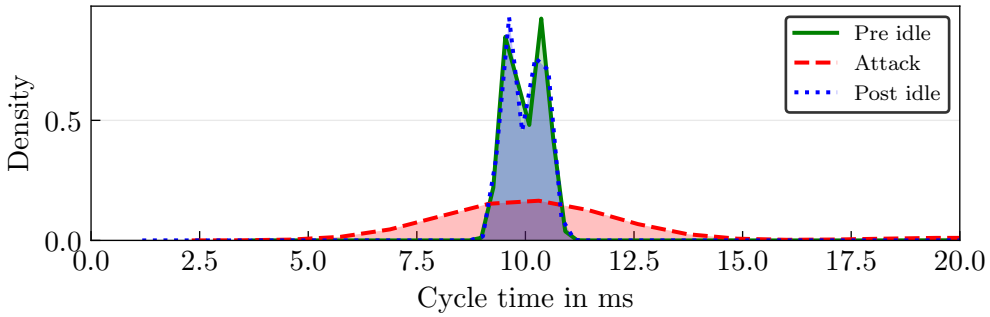


Figure 3.33: Density plot of the cycle time of the Wago PLC during hping3 attack influenced during the attack.

3.4.5 Conclusion

The presented architecture allows a secure and robust operation of an IIoT device in a network environment. This is achieved by a dual MCU architecture where one takes over the hard timing requirements and the second controller handles the network communication. This ensures that even with weak points in software implementation, e.g. vulnerabilities in the network stack or the operating system, the physical process is not affected. This enables a feedback-free process control. For future devices, it is also possible to separate the power supply and galvanically isolate the communication to reduce the possibilities of hardware attacks and failures.

In this section, the feasibility of the introduced robust architecture is demonstrated by a PoC implementation on a Cortex®-M7 MCU for the network tasks, combined with a Cortex®-M0 MCU for the time-critical IO handling. The network MCU runs *FreeRTOS* and the IO MCU runs a bare metal system. They communicate over SPI with each other in such a way that the timing behavior is predictable. Benchmark experiments have shown that the physical control process can be influenced by a deviation maximum of the

cycle time of under one percent. These experiments were performed during a simulated DoS flooding attack. The results show that the dual MCU approach introduced here is a feasible solution against these kinds of network attacks on IIoT devices such as PLCs.

3.5 Efficient Passive Network Scanning for Industrial Control Systems

Contents of this section

Parts of this section have already been published in the paper "Efficient Passive ICS Device Discovery and Identification by MAC Address Correlation" at the 5th International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR) 2018 [Nie+18b].

3.5.1 Introduction

Owing to a growing number of attacks, the assessment of ICSs has gained in importance. An integral part of an assessment is the creation of a detailed inventory of all connected devices, enabling vulnerability evaluations. For this purpose, scans of networks are crucial. In the present section, a lightweight passive network monitoring technique using an efficient MAC address-based identification of industrial devices is proposed. Based on an incomplete set of known MAC address to device associations, the presented method can guess correct device and vendor information. To prove the feasibility of the method, an implementation is also introduced and evaluated regarding its efficiency. The feasibility of predicting a specific device/vendor combination is demonstrated by having similar devices in the database.

The baseline for these actions is the discovery (determining the presence of a device) and identification (know what kind of device it is) of all devices present in the network. Technically, this is implemented either by active or passive scanning.

Active scanning broadcasts additional packets into a network environment and monitors the resulting traffic. Depending on strict timing constraints, the injection of additional network traffic might lead to an unexpected behavior of the connected devices within industrial infrastructures. A standard network scan could lead to a DoS, or result in defective devices or an incorrect behavior of the processes [Wed+15]. As availability is considered the most important IT security protection goal, active scanning methods should be generally avoided or only executed with a reduced scan rate, like introduced in Section 3.3.

Passive scanning involves observing and capturing the packets transmitted within a certain period. This scanning scheme requires a device connected to the network

to capture the network traffic. The capturing devices can be configured, limiting the captured traffic to certain packets of interest. Compared to active scans, passive scanning schemes still have to deal with a large amount of data. Hence, it requires a higher effort in extracting relevant information, thereby increasing processing time and the demands for computational power. Considering the IT security protection goal availability, these additional efforts are compensated, as passive scanning schemes can also be integrated in fragile infrastructures without impairing regular communication.

The section is organized in the following manner: State-of-the-art network scanners with a focus on passive fingerprinting are summarized in Section 3.5.2. Section 3.5.3 introduces the fundamentals of the MAC addressing scheme which are crucial for further comprehension. In Section 3.5.4, the challenges of MAC-based device discovery and identification are introduced. Section 3.5.5 presents the developed methodology. In Section 3.5.6, a PoC implementation and an evaluation in the CoRT are presented. Finally, a conclusion and a research outlook are provided in Section 3.5.7.

3.5.2 Passive Network Scanning

This section provides a survey of currently available passive network scanners. Since the topic of device identification plays an important role, publicly accessible data sources are also listed.

NetworkMiner [Hje08] is one of the most commonly used tools. Basically, it is an application-analyzing network scanner to identify hosts. Using the combination of different fingerprinting methods and tools, *NetworkMiner* can determine the Operating System (OS), which runs on a host, enabling vulnerability detection. Since the protocol stack implementations are different for each OS, the respective protocol header construction and length also differ. The SYN/ACK packet-based identification takes advantage of different initial Time to Live (TTL) values for IP and varying TCP window sizes for TCP.

For these values, *NetworkMiner* uses the database of the *p0f* [Zal18] tool. Since different OSs also employ different implementations for Dynamic Host Configuration Protocol (DHCP), identification is also possible by inspecting these packets. Here, the device fingerprints from the *FingerBank* [Bil+13] project are utilized.

SinFP [Auf10] is a tool that supports active and passive OS fingerprinting. The implementation of the two concepts increases the accuracy in situations when packets are altered by mechanisms such as packet normalization or stateful packet inspection. In these cases, *SinFP* sends several TCP probe frames to trigger different responses. After all responses are collected, the content of each packet is analyzed using an approach similar to *p0f*. Additionally, *SinFP* supports a pure passive fingerprinting mode. In this configuration, only captured packets are analyzed which are obtained either from the network or a file. Like the other tools, a database containing signatures and patterns is used to identify the detected devices. After the capture of the input data for further analysis, different capture techniques can be used.

A common approach involves using a mirror port provided by a switch or a network Terminal Access Point (TAP). With port mirroring, the entire traffic arriving at a port is

forwarded to a mirror port of the switch. In an ICS environment, however, port mirroring is not as effective as it would be in a conventional environment. This effect is caused by a huge number of small groups of devices interconnected by simple switches [Wed+15].

Compared to the investigated solutions, the proposed method has some major advantages. First, a high identification rate of industrial components compared to existing tools is reached. Furthermore, an unused network port is sufficient for this scheme, because no special network switch feature or monitoring port is necessary. Moreover, no additional network traffic is generated in fragile ICS networks. Finally, a low effort in database maintenance compared to deep packet inspection is sufficient.

3.5.3 Media Access Control Addressing

This section introduces the fundamentals of MAC network device addressing, which is implemented in the data-link layer of the ISO/OSI reference model. These are utilized for device identification by the proposed method, which will be further described in the subsequent sections.

3.5.3.1 MAC Address Structure

The MAC address is a unique hardware address assigned to each network adapter. Nowadays, all known access methods with a MAC layer (IEEE 802.1), such as Ethernet, Wi-Fi, and Bluetooth, use the same MAC address format with a 48-bit MAC address as shown in Figure 3.34.

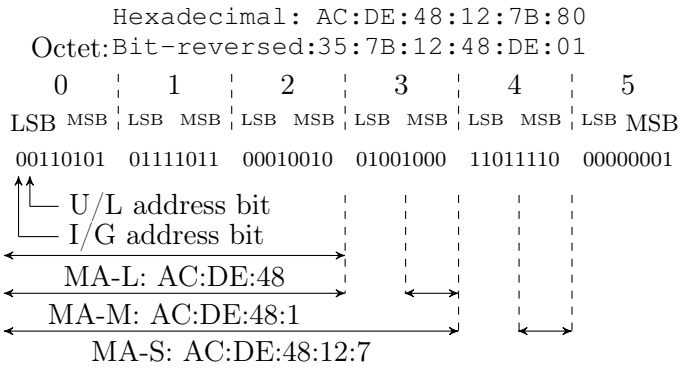


Figure 3.34: Structure of an EUI-48 [Ins14] MAC address.

The two least significant bits of the MAC address determine the type of the address: The *Individual/Group address* **I/G bit** indicates whether the frame is transmitted as unicast (0) or multicast (1). While a unicast frame is sent to one specific device, a multicast frame is forwarded to a group of devices. An address comprising 48 ones (FF:FF:FF:FF:FF:FF) is a broadcast address where all devices are addressed. The *Universally/Locally Administered* **U/L bit** is used to tell if the MAC address was taken from a fixed configuration (0) or dynamically chosen by the OS (1).

The following address types can be observed when captured network traffic is examined. If **I/G** is **0**, then it is an individual address (Unicast Address) for a network adapter. In contrast, if **I/G** is **1**, then the destination address is for a group of stations (Group address/Multicast address). A universal, globally unique, and unchangeable MAC address is used if **U/L** is set to **0**, otherwise (**U/L** = **1**), it is locally changeable.

The assignment of MAC address blocks is allocated and tracked by the Institute of Electrical and Electronics Engineers (IEEE) Registration Authority (RA). These are available in three different sizes, namely fixed length, MA-L (large), MA-M (medium), and MA-S (small), to meet the needs of vendors. The relation of the fixed portion of a MAC address to the corresponding number of possible MAC addresses is provided in Table 3.13.

Table 3.13: IEEE MAC assignment structure [Ins14].

IEEE RA assignment	**IEEE assigned**	**EUI-48 block**	**Comp./organ. identifier**
Company ID (CID)	24	0	yes (CID)
Large (MA-L)	24	2^{24}	yes (OUI)
Medium (MA-M)	28	2^{20}	no
Small (MA-S)	36	2^{12}	yes (OUI-36 only)

Information related to the current allocations, including the names of the respective vendors, can be obtained directly from IEEE in different file formats [Ins]. These files containing the currently allocated address blocks are associated with the contact information of their holders. This allows the mapping of an unknown MAC address prefix of a network device to a manufacturer. Unfortunately, this approach does not allow any correlation with devices, since the vendor is allowed to freely assign addresses within the allocated space.

3.5.3.2 Broadcast Messages

In the data-link layer, broadcasting is the transmission of certain packets to all devices within a broadcast domain. A broadcast domain is comprised of hubs, switches, and bridges, which can be divided by Virtual Local Area Networks (VLANs) or routers operating on layer 3. Broadcasting is used to accomplish different tasks of different protocols including:

- **ARP**, which is used if a network peer wants to communicate with another local device on a higher layer protocol. To learn the MAC address of the communication peer, an ARP request is sent to the Ethernet broadcast address that is forwarded by all network switches. The results containing the respective MAC addresses are cached in ARP tables of the participating devices (RFC 826 [Plu82]). Since these requests are essential for successful communication, all active MAC addresses are cyclically propagated on the network.

- **DHCP**, which allows the network configuration to be assigned to clients by a server (RFC 2131 [Dro97]).
- **Routing protocols**, which optimize the selection of proper routes for router communication.

3.5.4 Device Identification Challenges

For the proposed MAC-based security assessment, two challenges were identified. First, the approach has to determine devices from their MAC addresses even if the vendor's assignment scheme is unknown. Since passive scanning could be a time-consuming task, time estimation for a complete network device discovery was found to be important.

3.5.4.1 MAC Address Vendor Assignment

Each of the devices is programmed with a unique MAC address taken from the IEEE-assigned address blocks. Parts of the available blocks are often used sequentially for various products from a company. For example, regarding MA-L ranges, Figure 3.35 illustrates such a possible block-wise assignment of different product types. Product types X, Y, and Z are separated in a sequential and continuous block, which are often spread across the full range.

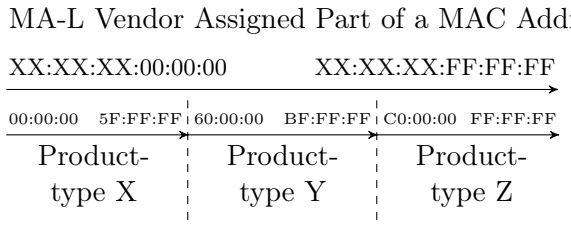


Figure 3.35: MAC address vendor assignment process.

Since this information is unknown, MAC addresses have to be collected. A correlation of yet unknown devices has to be determined from that source either by direct lookup or a proper approximation scheme.

3.5.4.2 Interarrival Time of Packets

The mathematical symbols used in the formulas can be found in Table 3.14.

Table 3.14: Symbols used in formulas.

Symbol	Description
t_{packet}	The relative time a broadcast packet occurs
$t_{arrival}$	Time between two broadcast packets of one device
$\bar{t}_{arrival}$	Mean interarrival time of broadcast packets of one device
$t_{arrival_{max}}$	The maximum interarrival time between two broadcast packets of one device
$t_{coverage}$	The maximum interarrival time of broadcast packets within a network, so that every device can be found

Estimating the time for complete device discovery coverage, the interarrival time $t_{arrival}$ was found to be an important key figure.

$$t_{arrival} = t_{packet_n} - t_{packet_{n-1}} \tag{3.7}$$

Generally, $t_{arrival}$ is defined as the elapsed time between the arrival of two consecutive packets containing the same information. It is calculated in the manner shown by Section 3.5.4.2. The mean interarrival time $\bar{t}_{arrival}$ comprises constituent measurements ($measure$) shown by Section 3.5.4.2.

$$\bar{t}_{arrival} = \frac{1}{n} \cdot \sum_{n=1}^{n=measure} t_{arrival_n} \tag{3.8}$$

Equation 3.9 calculates the maximum time to get a packet of one device.

$$t_{arrival_{max}} = \max_{\forall measure \in device} t_{arrival}(measure) \tag{3.9}$$

Finally, the expected time for a complete network device discovery $t_{coverage}$ is calculated (see Equation 3.10) by using the maximum interarrival time of all devices within the monitored network segment.

$$t_{coverage} = \max_{\forall device \in network} t_{arrival_{max}}(device) \tag{3.10}$$

3.5.5 Using Media Access Control Addresses for Device Discovery and Identification

This section introduces the methodology developed for the MAC address-based security assessments. Essentially, it builds up on capturing MAC broadcasts. The MAC addresses are extracted from these captures and fed to an address-identification process. The information about the device are mapped to known vulnerabilities.

3.5.5.1 Passive Scan Utilizing Address Resolution Protocol Broadcast Messages

In an industrial network, there are mostly static routes with fixed network configurations. Consequently, DHCP and routing protocol broadcasts are rare. In contrast, ARP requests are performed in cases where no MAC address is cached for a certain device. The ARP cache contains a four-column table containing the protocol type, the protocol address of the sender, the hardware address of the sender, and the entry time. The expiration time for the entries is not specified by the relevant RFC 826 [Plu82].

3.5.5.2 Media Access Control Address Correlation

To successfully correlate unknown MAC addresses to devices, a prebuilt data-pool of known devices is essential.

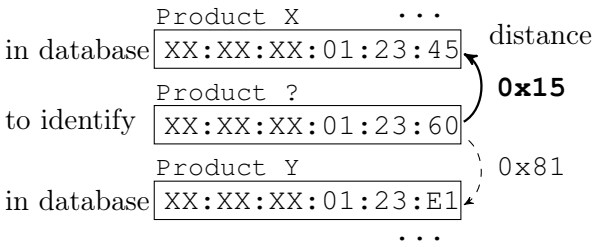


Figure 3.36: Method of the MAC based identification.

Figure 3.36 illustrates the developed device identification scheme. In this example, there are two initially known MAC addresses stored in the data-pool. Both addresses are close to a MAC address of an unknown device. The illustrated distance value is calculated from the difference between the addresses. A smaller distance increases the chance of a correct device identification. In the present case, the unidentified device is more likely to be the same product family as "Product X". This distance is used for the passive identification of ICS components.

3.5.5.3 Mapping of Vulnerabilities

With the determined device correlation, the identification of known vulnerabilities is feasible. Figure 3.37 shows the relation of the vulnerability mapping. With information concerning the vendor and product name, the corresponding Common Platform Enumeration (CPE) is searched using pattern recognition. The relationship between CPE and CVE allows the assignment of vulnerabilities to a device.

3.5.6 Framework and Evaluation

To evaluate the feasibility of the presented method, the outlined methodology is implemented in a framework.

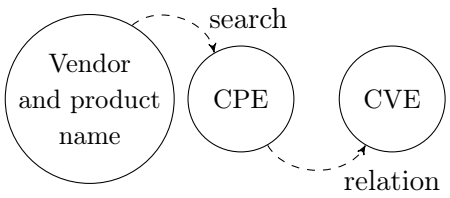


Figure 3.37: Method of vulnerability mapping within the framework.

3.5.6.1 Testbed Setup / Preliminary Investigations

To facilitate an examination of the MAC address distribution, the CoRT introduced in Section 3.1 is used. For this evaluation, a selection of 12 devices of each rack is used, which were available at the moment of evaluation. By using different configurations in the testbed, but keeping the numbering in the document consistent, the numbering is not continuous (see Table 3.2). The components used for the evaluation are listed in Table 3.15. The devices with the number identifier ending with R1 is in rack 1 and ending with R2 is in rack 2. The last column denotes the distance, which is the absolute value calculated from the subtraction of the MAC addresses of devices of the same type. Except for those from Moxa, the devices were bought at the same time. Therefore, the distances between the MAC addresses are generally small. Further inspection of the Moxa devices revealed different production dates and higher values for the distance.

Table 3.15: Devices employed within the testbed for this evaluation.

Vendor	Product	No.	MAC	IP	No.	MAC	IP	Distance
			Rack 1			Rack 2		hex
Wago	750-889	01-R1	00:30:DE:0C:AA:68	192.168.0.30	01-R2	00:30:DE:0C:AA:6C	192.168.0.130	0x000004
Wago	750-8100	02-R1	00:30:DE:41:B9:F0	192.168.0.31	02-R2	00:30:DE:41:B9:E6	192.168.0.131	0x00000A
Wago	750-880	03-R1	00:30:DE:0C:AE:84	192.168.0.32	03-R2	00:30:DE:0C:AE:68	192.168.0.132	0x00001C
Siemens	CPU 1211C	05-R1	28:63:36:C6:C7:D4	192.168.0.10	05-R2	28:63:36:C6:CC:67	192.168.0.110	0x000493
Siemens	LOGO! 24RCE	10-R1	E0:DC:A0:1C:35:85	192.168.0.23	10-R2	E0:DC:A0:1C:35:4F	192.168.0.123	0x000036
Phoenix	ILC 151	11-R1	00:A0:45:9D:40:74	192.168.0.20	11-R2	00:A0:45:9D:42:54	192.168.0.120	0x0001E0
ABB	PM554-T	14-R1	00:24:59:0A:4C:B7	192.168.0.21	14-R2	00:24:59:0A:58:B0	192.168.0.121	0x000BF9
Crouzet	em4 B26-2GS	15-R1	84:AC:FB:00:05:E0	192.168.0.22	15-R2	84:AC:FB:00:05:E6	192.168.0.122	0x000006
Schneider	TM221CE16T	16-R1	00:80:F4:0E:58:89	192.168.0.50	16-R2	00:80:F4:0E:59:BC	192.168.0.150	0x000133
Siemens	KP 300	17-R1	00:1C:06:35:C0:7C	192.168.0.11	17-R2	00:1C:06:35:C0:7B	192.168.0.111	0x000001
Schneider	HMISTU855	18-R1	00:01:23:2D:BA:A3	192.168.0.51	18-R2	00:01:23:2D:BD:7B	192.168.0.151	0x0002D8
Moxa	NP5110	20-R1	00:90:E8:2A:E5:34	192.168.0.70	20-R2	00:90:E8:56:78:5D	192.168.0.170	0x2B9329

3.5.6.2 Interarrival Time of ARP Packets

First, $t_{arrival}$ between consecutive ARP broadcasts for each of the connected devices is determined. To this end, MAC addresses are extracted from the ARP broadcasts captured from a 12-hour pcap.

The results of this experiment consist of 3559310 packets, where 220988 (6.21 %) are ARP packets. Figure 3.38 illustrates the calculated $t_{arrival}$ for the 24 devices. In 70 % of the cases, an ARP packet shows up every 16–37 seconds ($\bar{t}_{arrival}$), not exceeding the boundary of one minute ($t_{arrival_{max}}$) at the worst cases. With this data, $t_{coverage}$ of this network is about one minute. The total number of ARP packets varies from about 1200

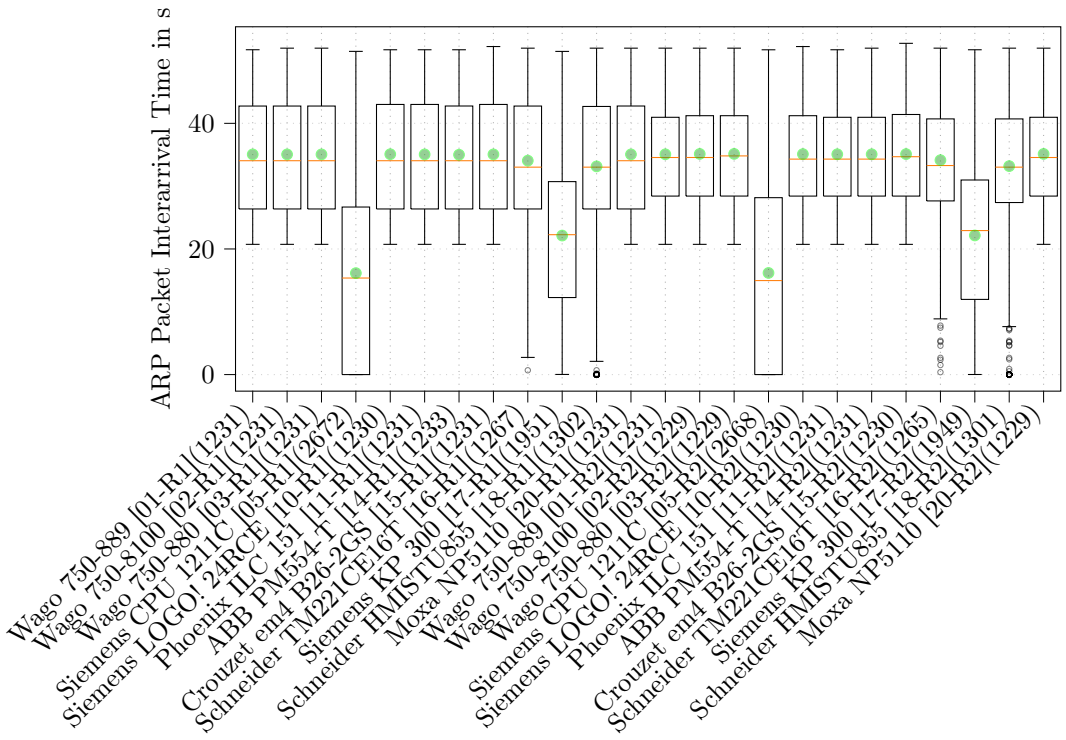


Figure 3.38: ARP packet occurrence by device over time [Device number](Total ARP packets).

to 2700 over a measurement time of 12 hours. In the testbed, a continuous SCADA monitoring process is implemented, which queries the status of all devices every second. This communication generates constant traffic, which could lead to a homogeneous plot. There are four HMIs – the Siemens KP 300 [17] and the Schneider HMISTU855 [18], which communicate with the controllers Siemens CPU 1211C [05] and Schneider TM22ICE16T [16]. Because of the constant communication, there are more ARP packets of these devices in the pcap.

The second capture, which is used is the publicly available "4SICS-GeekLounge-151022.pcap" [4SI15] from NETRESEC [NET] captured at the 4SICS conference (now renamed to CS3STHLM [OMN]). It is about 15h long and has 2274747 packets in total, of which 16197 (0.71 %) are ARP packets. The number of ARP packets varies from 31 (Phoenix Contact FL IL 24), 162 (Hirchmann EAGLE 20 Tofino), 220 (Siemens S7-1200) to 343 (Moxa EDS-508A) ARP packets over the complete duration of the 4SICS capture. Figure 3.39 illustrates the distribution of interarrival time of ARP requests. In this capture, there is not as much traffic as in the first pcap. The difference in the number of captured packets is mainly caused by the continuous monitoring process of the first testbed and the different network structure. As a result, the time between two ARP packets ranges from 2 to 20 minutes.

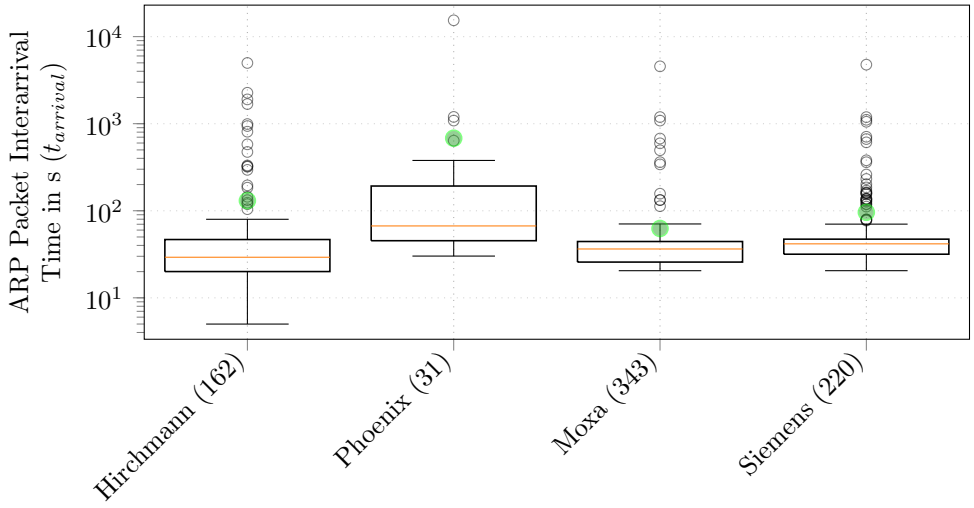


Figure 3.39: ARP packet occurrence by device over time.

The different results from the captures show that the occurrence of ARP packets depends on the network and the communication behavior. It follows, therefore, that after a passive network dump of approximately one hour all devices could be detected.

3.5.6.3 MAC Address Database

The initial device database (see Table 3.16) was built using data from *Censys* [Dur+15], *Shodan* [Mat09], Google Images (`images.google.com`) and marketplace search (`www.ebay.com`), as well as previous scans from our research group. This database is locally stored and is used for the identification. Figure 3.40 illustrates the distribution of known devices in the local database over the complete MA-L range of the corresponding vendor, with more than 500 entries. Some vendors have more than one MA-L range, resulting in more company identifiers.

The dataset plots of Moxa and Allied Telesis are bundled across the lower MAC address range of the complete MA-L space. This indicates that the examined manufactures have yet not exceeded their assigned address space. Moreover, the available database entries suggest a linear assignment process.

The results for Lantronix devices indicate a different assignment process. The MAC addresses are spread over the full MA-L space, while the majority of addresses are concentrated at the upper range. This indicates a kind of randomization, as there are no larger connected groups of addresses. However, the entries in the used dataset show a possible sequential block-wise assignment, which could also be a sign that the used dataset is not large enough to include many devices of the same range. Eventually, it is not possible to make a final statement on the MAC distribution policy applied by Lantronix.

Table 3.16: Number of devices and known vulnerabilities in the database.

Vendor	Products	Devices	Known Vuln.
MOXA Technologies	Networking equipment	5242	80
Lantronix	Networking equipment	2126	6
Allied Telesis	Networking equipment	950	9
Siemens	Automation equipment	75	242
WAGO Kontakttechnik	Automation equipment	49	5
Star Micronics	(Receipt) printer	20	1
Schneider Electric	Automation equipment	14	113
Phoenix Contact	Automation equipment	11	6
ABB	Automation equipment	6	18
Hirschmann Industries	Networking equipment	3	8
Crouzet	Automation equipment	3	0
	Total:	**8499**	**488**

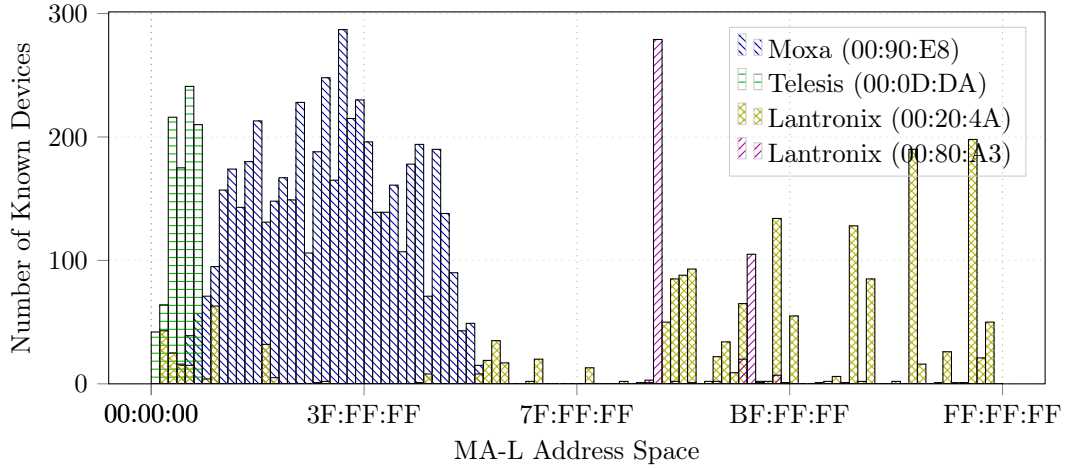


Figure 3.40: Histogram of known devices MAC addresses in their MA-L address space.

3.5.6.4 Proof of Concept Framework

The Proof of Concept (PoC) framework is written in Python 3 and implements automatic evaluation of MAC-based discovery and identification of ICSs. It takes a pcap file or input, or captures live traffic, analyzes it, and creates a security report on the detected network devices. The pseudo program structure of the framework is shown in Figure 3.41.

At startup, the local MAC databases (see Section 3.5.6.3) are imported. These are extensible Comma-Separated Values (CSV) files that store devices with their product name and MAC address. Current vulnerabilities are fetched automatically from cve details [MITb]. In the next step, the IEEE MAC vendor lists, which are publicly available on the internet, are downloaded and parsed. Subsequently, logged or live network traffic

is analyzed using the *pcapy* tool [COR], which provides access to the *pcap* packet capture library.

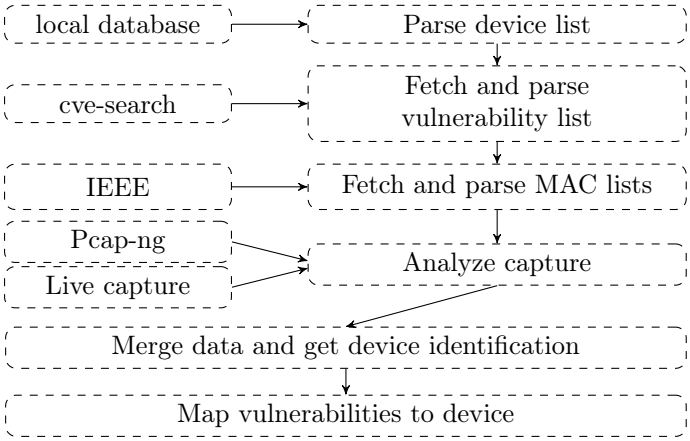


Figure 3.41: Dataflow within the framework.

The MAC addresses of the devices found in the capture are now compared with the local database of known devices. If a MAC address of a device from the network capture is at a predefined distance from a MAC address of a known device, it is possible that these devices are the same product. Based on this, the CVE database is used to map possible vulnerabilities to the devices.

3.5.6.5 Network Integration

The network integration for this method is easier than most of the available passive network monitoring tools. This is because no mirror port or network TAP is necessary. Basically, all broadcast messages containing MAC addresses are sufficient for device discovery. Figure 3.42 shows the integration possibility of the proposed framework in an existing network.

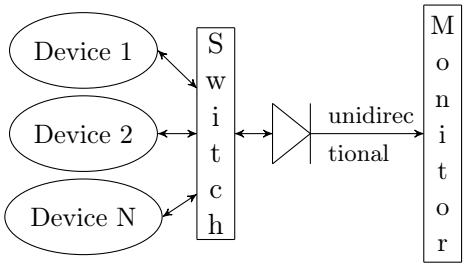


Figure 3.42: Passive network monitoring setup.

To ensure a feedback-free passive scan, a unidirectional network diode can be placed between the switch and the scan device. For each broadcast domain, only one monitoring connection is needed.

3.5.6.6 Validation of the Proposed Scheme

To validate the method and the framework, the devices in Table 3.15 of the CoRT are used. By probing real hardware, the methodology introduced in Section 3.5.5 and the PoC implementation are verified. The used database comprises 8499 known devices excluding the devices from the CoRT that are blacklisted for the validation. The network traffic used to validate the accuracy is the same 12 hours pcap as that used for interarrival time calculation (see Figure 3.38). Table 3.17 shows the results of the validation categorized into four possible results.

Correct identification (✓): As illustrated in Section 3.5.5.2, a small distance to a known device indicates a correct identification. A distance of 0x000000 indicates that the device was clearly identified. This happens if an already identified device should be identified again, for example, in a previous scan or in a continuous monitoring process.

Correct vendor, wrong device (✧): In a false positive identification, the identified device does not match the real one. With the distance increasing between the MAC of the captured device and the entries in the database, the likelihood of a correct identification decreases.

Only vendor identified (❍): Furthermore, it is possible that there is no match, e.g. if there is no device in the MA-L address space, leading to a distance higher than 0xFFFFFF. It is still possible to identify the vendor by looking up the MAC vendor list(s).

No identification (✗): If there is no known device in the address space and no entry in the vendor lists, no identification is possible.

Table 3.17: Validation of the introduced approach in the Communication Robustness Testbed.

Device	Distance	Ident.	Device	Distance	Ident.
01-R1	0x00A214	✓	01-R2	0x00A218	✓
02-R1	0x001789	✧	02-R2	0x00177F	✧
03-R1	0x00A630	✧	03-R2	0x00A614	✧
05-R1	0x19CE87	✧	05-R2	0x19D31A	✧
10-R1	0x117EDC	✓	10-R2	0x117EA6	✓
11-R1	0x0FE94D	✧	11-R2	0x0FE76D	✧
14-R1	0x00049C	✓	14-R2	0x00075D	✓
15-R1	0x000481	✓	15-R2	0x000487	✓
16-R1	0x000102	✓	16-R2	0x000235	✓
17-R1	0x089F19	✓	17-R2	0x089F18	✓
18-R1	0x033E07	✓	18-R2	0x0340DF	✓
20-R1	0x0001AC	✓	20-R2	0x00003D	✓

✓ Correct identification ❍ Only vendor identified
✧ Correct vendor, wrong device ✗ No identification

In all, the validation leads to a discovery rate (find the device) of 100 % with a correct identification (associate product) rate of 66.67 %, demonstrating the feasibility of the

introduced method and software. Also, these results could be improved with a larger device database.

3.5.6.7 Comparison with Other Tools

Three active and three passive network scanner tools are used and are compared with the results of the framework introduced here. A popular active scanner used in the *Censys* project is *ZGrab* (Git version 6c81ce4) [Dur+15], which supports the following SCADA protocols: BACnet, DNP3, Niagara Fox, Modbus, and SimaticS7. *PLCScan* (Git version 014480c) [Efa12] is one of the first active scanners for industrial networks. Furthermore, the active scan of *Nmap* (Version 7.60) [Lyo09] with Nmap Scripting Engine (NSE) scripts for BACnet, Ethernet/IP, Modbus, Nigara Fox, Omron, PcWorx, ProConOS, and SimaticS7 is used. For the passive scan, *Netdiscover* (Version 0.3), *SinFp* (Version 1.22), and *p0f* (Version 3.09b), which are introduced in Section 3.5.2, are used.

Table 3.18: Comparison of existing tools within CoRT.

	Device (No./Rack)																							
	01	01	02	02	03	03	05	05	10	10	11	11	14	14	15	15	16	16	17	17	18	18	20	20	Ident.
Scanner	R1	R2	R1	R2	R1	R2	R1	R2	R1	R2	R1	R2	R1	R2	R1	R2	R1	R2	R1	R2	R1	R2	R1	R2	Rate
zgrab	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✓	✓	✗	✗	25.00 %
PLCScan	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗	✓	✓	✗	✗	25.00 %
Nmap	❍	❍	✓	✓	✓	✓	✓	✓	❍	❍	✓	✓	❍	❍	❍	❍	✓	✓	❍	❍	❍	❍	❍	❍	41.67 %
SinFp	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	00.00 %
p0f	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	00.00 %
Netdiscover	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	❍	00.00 %
Introduced tool	✓	✓	✧	✧	✧	✧	✧	✧	✓	✓	✧	✧	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	66.67 %

✓ Correct identification ❍ Only vendor identified ✧ Correct vendor, wrong device ✗ No identification

To get comparable results, all tools are evaluated in the CoRT, with the devices introduced in Table 3.15. The results of the comparison are shown in Table 3.18. The identification rate of the active scanners depends on the implementation of the protocol used in SCADA environments. If the protocol is available for the scanner, then it mostly discovers and identifies the device. *ZGrab* and *PLCScan* have found fewer PLCs, because there are fewer protocols implemented than *Nmap* with additional NSE scripts. Similar to the introduced tool, *Netdiscover* used the IEEE OUI vendor list and could detect the vendors of the devices but not the specific product. *SinFp* and *p0f* could not identify any ICS device correctly in the used testbed, instead showing only office components such as the server used for the evaluation. Similar results of the existing fingerprinting tools were achieved by others in the past [Cas+13][Hah+11]. The introduced framework was able to discover all the devices in the testbed by their vendor, and about 66 % of the products were correctly identified.

3.5.6.8 Discussion of Limitations

The proposed MAC-based device identification has some limitations regarding the rate of discovery and identification. Some of these limitations, however, do not apply to industrial networks due to their structure and specific use of the connected devices.

MAC Address Spoofing IT devices often allow specifying a MAC address of a network interface. For example, a network administrator wants to clone a device. Nevertheless, this is generally not possible for ICS devices, because vendors do not provide this feature. Of course, attackers can bring in new devices into a network by sending ARP requests with spoofed MAC addresses. However, attackers cannot spoof ARP requests of existing devices without physical access or access to the switching hardware. Therefore, MAC address spoofing has only minor relevance for the proposed scheme.

Database Quality To identify devices by their MAC address, there must be a similarly known device in the local database. Hence, the number of entries in the known MAC address database is significant for the quality of device identification. This could be improved by community contributions.

MAC Randomization for Connections To make it difficult to identify devices, it is possible to choose MAC addresses randomly for every new connection. This could lead to a false positive identification of a device. However, for stationary devices, especially in industrial networks like PLCs, it is uncommon to randomize the MAC address.

Vendor Assignment Process Vendors sometimes assign the same MAC address to multiple devices, maybe by mistake or intentionally to save money. This is not a major problem if the devices are shipped to different parts of the world. In the case of MAC-based identification, this could lead to false identifications if different products have the same MAC.

Static ARP Table In a static ARP table, the MAC addresses of some or all network participants have fixed entries. It is used in static environments, to prevent MitM attacks by ARP poisoning, to use a network diode, or to reduce the broadcast traffic [Sto+11]. In this case, the ARP broadcasts used to identify the devices will not be sent, and thus, no identification is possible.

Firmware/Software Version The firmware and software versions are not detected by the MAC-based identification of a device. Consequently, the vulnerability allocation is done without considering the software version.

However, in many ICSs, the devices are still in the delivery software state and are not patched. This information could be used to roughly estimate the firmware version of the device, which is sufficient for an initial assessment.

3.5.7 Conclusion

In this section a new method for passive scanning of fragile networks is introduced and a functional PoC is provided. With a MAC-based discovery and identification of ICS components, a safe passive scan within ICSs is possible. The feasibility of the method with a validation resulting in a total discovery rate of 100 % and an identification quote

of more than 66 % is shown. In comparison with existing tools, the MAC address correlation approach performed well. The minimal integration effort and the extensibility of the proposed framework has advantages over classical scan methods and deep packet inspection. Moreover, the code of the presented framework is published and open for contribution to the MAC databases [HSA18]. The material includes the source code of the framework, network captures, and additional information about the current configuration of the CoRT.

With a detailed recognition of the transitions between products during the MAC address assignment, a more detailed identification could be achieved. Therefore, some details provided by the vendors or a larger database of each vendor are necessary due to the vendor-specific assignment process. Additionally, the users of the proposed approach can add known devices from an industrial plant with which similar devices in other plants could be identified.

Chapter 4

Modular Building Blocks to Enhance Industrial Control System Security

Contents of this chapter

In this Chapter, building blocks to enhance the IT security level of ICS are introduced. It is important that industrial systems are protected against the current threats. This means, among other things, bringing new possibilities and functions into future products, for example, to detect attacks and to take countermeasures. Furthermore, it is important to build up awareness for automation engineers who are not familiar with security and to train people, such as students, in this area. Since productive plants are often in use 24 hours a day at 365 days a year, neither training nor tests and analyses can be done there. Furthermore, no change can be made to the mostly proprietary ICS devices. For these reasons, open components and testbeds are needed, which are also accessible at a low cost to the general public.

The Chapter is structured as follows. At the outset, the open testbed and components are described in Section 4.1. This is followed by two building blocks for network monitoring on industrial edge node devices. The first building block introduces a distributed IDS on low performance MCUs in Section 4.2. The second building block presented in Section 4.3 demonstrates the possibility of a network scanner on industrial edge nodes. At the end, a low-cost ICS testbed for education and research is presented based on the previous experiences of the author in recent years (Section 4.4). This closes the gap to find a quick starting point into the complex topic of industrial it security.

4.1 Open Industrial Control System Components for Secure Operation

Contents of this section

As proprietary industrial components were mostly used in the previous chapter, it is difficult to make changes to them in order to evaluate new concepts to enhance the IT security of a single component. Owing to the closed-source devices, usually no personal software can be installed, except the user program. However, this does not allow any changes to the operating system or the control software which come from the manufacturer. This is only possible in newer PLC generations, which were released on the market in the last years using Linux as an operating system. Currently, there are only a few devices from a small number of manufacturers offering this functionality (e.g. Phoenix PLCnext, WAGO PFC, Siemens IoT and Raspberry Pi-based PLCs).

4.1.1 Introduction

To examine new network architects in the age of Industry 4.0, there is almost no way around open components. One idea of Industry 4.0 is that the PLC runs completely in software (soft-PLC) and is mostly hardware and software independent. For example, this allows that the PLC software runs in the local data center, in the cloud, or simply on a small computer such as a Raspberry Pi. With centralized PLCs, so-called remote IOs devices are used for inputs and outputs, which are controlled by the soft PLC via network. In order to cover this scenario, the open components that are used in this work are presented in this section. These differ from the previously presented PLC implementation in Section 3.4, because the architecture is now divided into soft PLCs and remote IOs.

4.1.2 Remote IO/Edge Node

An open-source remote IO (edge node) for the evaluation of IT security building blocks for ICSs is presented in this section and explained in detail.

4.1.2.1 Hardware

The edge node devices are implemented on ST NUCLEO-F767ZI [STMa] development boards. They have an ARM Cortex-M7 core, which operates at a frequency of 216 MHz with 512 kB of RAM and 2 MB flash. The board is equipped with an Ethernet transceiver and corresponding RJ45 jack. This device was used because more performance is available

for tests, even if it is not needed later. Figure 4.1 shows the used development board from STM and the custom designed PCB on top. The schematic of the custom PCB can be found in Appendix A.5. This is used to control the IOs and the display, which is controlled over Inter Integrated Circuit (I^2C). Additionally, LEDs are mounted to indicate suspicious behavior. This can be used, as in a data center where servers which require maintenance flash an LED. This allows a quick search when many devices are installed in a plant.

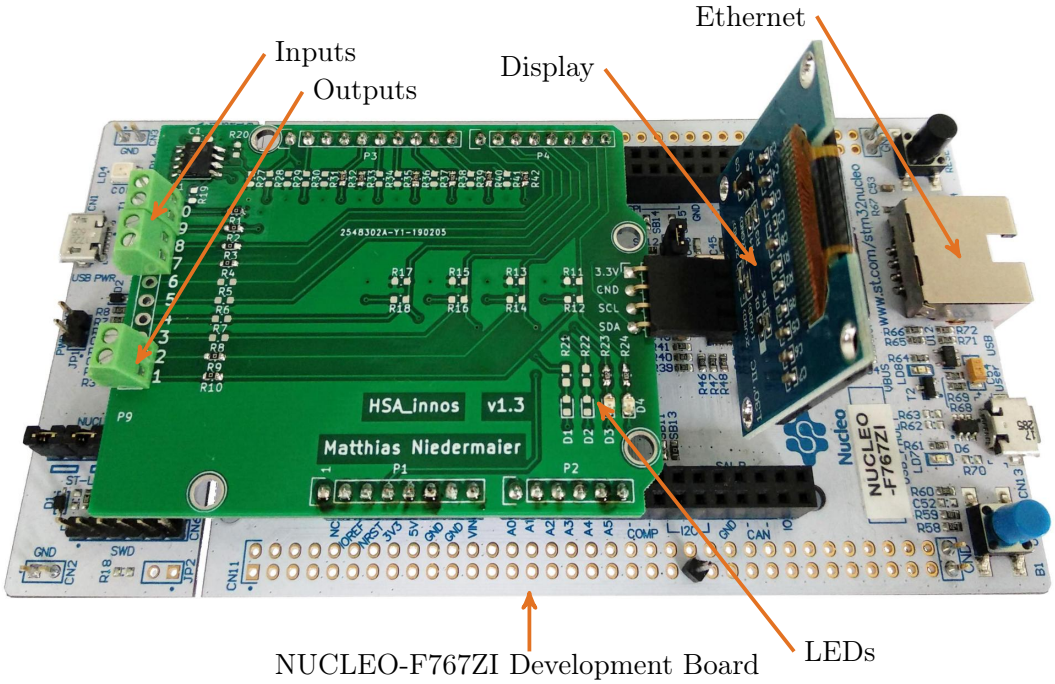


Figure 4.1: Picture of the baseboard and the custom PCB of the used edge node device.

On the right side of the development board, the Ethernet connector is placed, and on the left side, the USB connection for power supply and programming is mounted. By using the Arduino™ Uno V3 header for the custom PCB, the underlying prototyping board can be easily changed with other compatible boards. This can be interesting, for example, if more performance or an energy-saving solution is needed. Table 4.1 lists the features of the development board with the MCU. The ARM®Cortex®-M7 MCUs are the high-performance series of the energy-efficient Cortex®-M product range.

4.1.2.2 Software Stack

The software architecture used on the edge nodes is illustrated in Figure 4.2. As operating system *FreeRTOS* is used. The MCU software is configured with *CubeMX* [STMb] by STM, enabling an easy portability to a wide variety of STM MCUs. Additionally, using *FreeRTOS* also allows adding new tasks easily, for example, an additional feature or user code.

Table 4.1: Specification of the used edge node hardware.

Hardware	**IIoT Edge Node**
Board design	STMicroelectronics
MCU	STM32F767ZIT6
Core	ARM® Cortex®-M7
Clock	up to 216 MHz
RAM	512 kB
Flash	2 MB

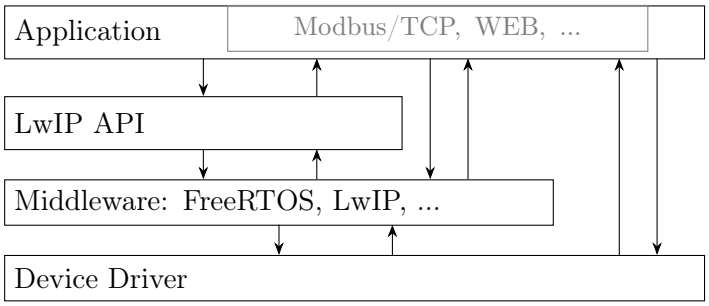


Figure 4.2: Overview of software integration onto the edge node devices.

The used network stack, *LwIP*, provides TCP/IP functionality, for example. On the application side, a self-developed Modbus/TCP slave is implemented as well as a rudimentary web server. Additionally, a thread-safe logging function is implemented, which prints the log message over Universal Asynchronous Receiver Transmitter (UART). As a whole, this forms the basis for further developments and provides easy extension of the edge node.

4.1.3 Open-source Testbed

The open-source testbed in this chapter, is based on IIoT and cloud-based ICS architectures [Mah19; Col+14]. Figure 4.3 shows such aICS architecture where the PLC serves as a central data hub. In the future, the PLC services may be hosted in a local data center, cloud, or fog.

The architecture used in this open-source testbed contains eight sensors (S1-S8), one actuator (A1), a PLC, an HMI, and a connection to other services like SCADA, logging and cloud applications. Modbus/TCP is used as the industrial communication protocol because it is widely used [Swa+99]. Additionally, Modbus/TCP offers manifold attack paths [Hui+08]. Thus, there are different possibilities to evaluate and benchmark multiple realistic attack scenarios.

Table 4.2 summarizes the components used in the open-source testbed. In contrast to existing testbeds, software can be changed easily. Furthermore, as standard components are be used for evaluation, this testbed can be realized at a low cost. In contrast with standard Modbus/TCP sensors, it is usually not possible to make changes such as

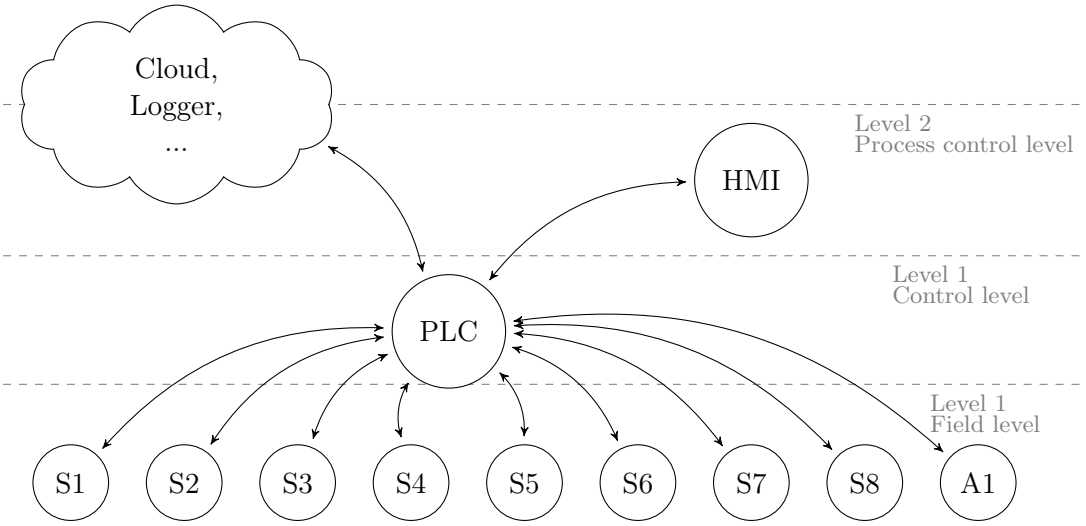


Figure 4.3: Network system view on a "standard" industrial network mapped to the used PoC testbed implementation. Eight "intelligent" edge node sensors, one "intelligent" actuator, a PLC, an HMI, and possibilities for cloud services.

inserting an IDS. Furthermore, measurements on the system are therefore not feasible, because no measurement routines can be used within the software on the device.

Table 4.2: Overview of devices used in the testbed.

Identifier	Device	Software	Hardware	IP
S1-S8	Sensor	FreeRRTOS, LwIP	STM32F7	192.168.1.101-108
A1	Actor	FreeRRTOS, LwIP	STM32F7	192.168.1.109
PLC	PLC	OpenPLCv3	Raspberry Pi	192.168.1.50
HMI	HMI	Custom	Raspberry Pi	192.168.1.40
Cloud	Cloud	ScadaLTS, Logging, ...	APU2C4	192.168.1.1

Figure 4.4 shows the open-source testbed for the measurement, which is set up in a 19-inch rack. At the bottom, there is the physical process with a motor controlling a disc. This disc is sensed by eight sensors, each of which is connected to one edge node. Additionally, the HMI is placed next to the physical process. In the middle, the edge nodes are placed, each with its own display. Next to this, Raspberry Pis are mounted, where one controls the HMI and the other runs the PLC software. At the top of the rack, the network switch and a server representing the cloud are placed. In reality there would be several (hundred) meters between the components.

In total, the testbed consists of 12 Modbus/TCP devices (eight sensors, one motor, one PLC, one HMI, one SCADA) controlling a physical process.

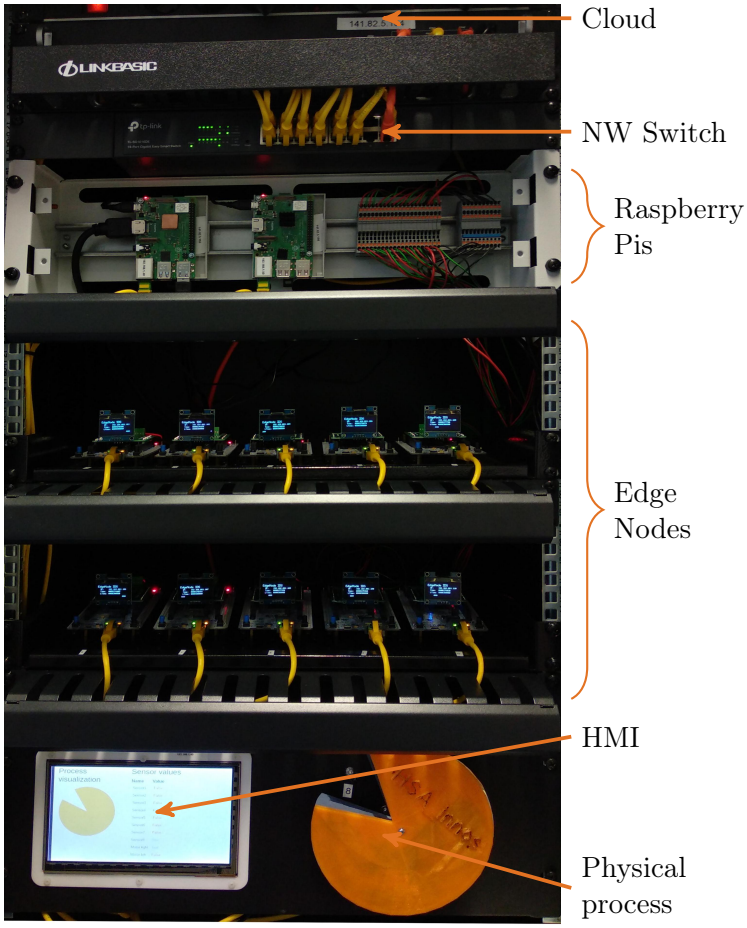


Figure 4.4: Pictures of the open-source ICS testbed, which is controlling a physical process.

4.1.3.1 OpenPLC

As the central control unit, a Raspberry [Ras12] Pi with *OpenPLC* [Alv+14] is used. This PLC is used here to cover the scenarios with remote IOs and soft-PLCs. The project provides a free open-source solution for PLCs. The PLC is configured to poll all sensors and actuators every 100 ms. Using current sensor states as an input, the program, which runs on the PLC, is executed and the new output values are calculated. New values are then written back by the next poll cycle.

4.1.3.2 Human Machine Interface

HMIs enables technicians to view the current status of a plant and the associated processes. Based on this, technicians can make decisions and interact with the control process. The HMI is a custom implementation based on Flask [Ron10] with *Pymodbus* [Col11]. With *Pymodbus*, the input and output registers are polled from the *OpenPLC* every 100 ms.

4.1.3.3 ScadaLTS

ScadaLTS [Rok+16] is an open-source SCADA system that accesses the *OpenPLC* to gather data. This is used as a historian and runs on a dedicated server (APU2C4). The SCADA system polls the *OpenPLC* every 100 ms via Modbus/TCP. This makes it possible not only to record the data and make it usable for later usage, but also to set alerts when values are exceeded or unwanted conditions occur.

4.1.4 Conclusion

In this section the open source components used in this chapter were presented. On the one hand, open edge nodes were introduced, which are used as remote IOs, and on the other hand, an open-source testbed was described. This is important because changes to the software of the proprietary components, such as those mostly used in Chapter 3, are difficult to make. Thus, the use of open source makes research possible directly on the components. Additionally, with the setup shown in Figure 4.3, Industry 4.0 scenarios with a centralized PLC and network capable remote IOs can be realized and investigated. Furthermore, an entire industrial system with a simple physical process can be analyzed with regard to security using the open testbed.

4.2 Intrusion Detection on Industrial Internet of Things Edge Devices

Contents of this section

Parts of this section have already been made publicly available in the paper "Efficient Intrusion Detection on Low-Performance Industrial IoT Edge Node Devices" on the arXiv preprint [Nie+19c].

4.2.1 Introduction

The increased connectivity in the times of IIoT allows remote attackers to directly target IIoT devices, for example, with DoS attacks (See Section 3.3) or packet injection to send control commands [Ber11]. Thus, centralized defense at the perimeter of the network do often not provide sufficient protection. Decentralized defenses, where each part of the network protects itself, are needed. Network IDSs monitor the network and report suspicious activity. They usually run on a single host, cannot capture all events in the network, and are associated with a great integration effort. To bridge this gap, a method for intrusion detection that combines distributed agents on IIoT edge devices with a centralized logging is introduced in this section. In contrast to existing IDSs, the distributed approach is suitable for industrial low-performance MCUs.

In classical centralized monitoring systems, low-performance edge devices forward all information to a high performance system such as a server [Ana17]. There, data is stored and processed, as shown in Figure 4.5. However, this approach has several drawbacks. High bandwidth is required to forward data from sensor nodes to the server. This places high burden on the network. Second, with more and more sensor nodes, tremendous amounts of data need to be processed on the server, bringing serious scalability issues.

For the above-mentioned reasons, it is beneficial to analyze network data already at the edge device. As Garcia et al. [Gar+15] note, the edge node devices are getting smarter again, because, on the one hand, they have enough performance. On the other hand, this offers advantages in terms of privacy and security. Figure 4.6 shows a distributed IDS approach where every edge node can observe the network, preprocess data and decide autonomously. While a centralized logging system might still be present, only few data is sent to it. This has several benefits over the centralized approach: Bandwidth requirements are lower, less data is transmitted, and computational resources at the server are saved. Thus, this approach scales better for large networks. Moreover, anomalous net-

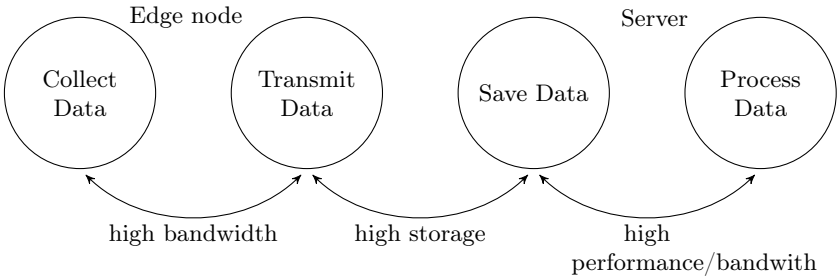


Figure 4.5: Centralized data collection approach with system requirements.

work traffic can be detected everywhere in the network. This increases network coverage and the robustness of the IDS, as there is no more single point of failure. Lastly, the IDS can respond to malicious traffic quicker, because the decision is made on the edge device itself and has no or less network delay on top. Furthermore, multiple IDSs could operate concurrently in a single network and exchange information.

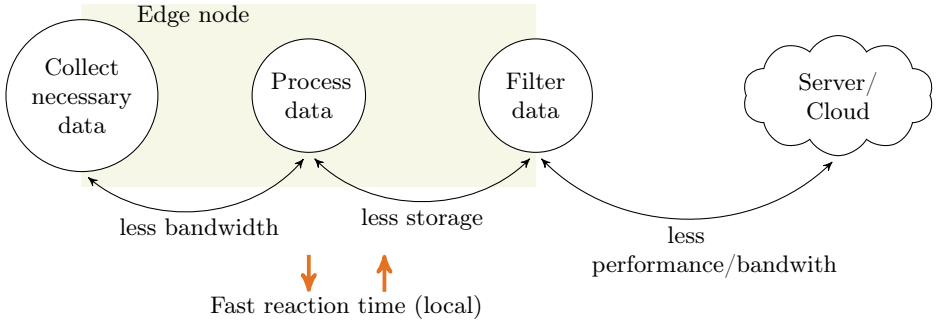


Figure 4.6: Distributed data collection approach. Preliminary data processing is conducted at the edge devices.

Currently, there is a lack of distributed network-based IDSs, which can both run on low-power edge devices and account for the special networking requirements of ICSs.

To overcome this gap, in this section, a distributed IDS tailored for industrial and sensor applications, using a statistical approach suitable for embedded low performance MCUs, is presented. Compared to signature and rule-based approaches, this concept has some advantages, such as dynamic learning without fixed rules, and the lack of periodic signatures updates of malicious software/traffic.

This work answers the following research questions about distributed network-based IDSs in industrial and sensor environments. Special concerns for distributed intrusion detection are:

- How long does an ICS network have to be observed to get a sufficient amount of networking data to learn its regular behavior?
- How to handle user interaction, e.g. through an HMI with its influence on system timings?

- How to retrieve the incident message from the edge node and inform an operator about it?
- What degree of deviation from the regular behavior of the network is tolerable before detecting it as an incident?

Key contributions of the approach include the following:

- An easily portable implementation, since only the widely used LwIP stack with *FreeRTOS* is needed.
- A detailed performance analysis of an actual IDS implementation and an evaluation of the capabilities of the approach in a realistic industrial control system environment.

The remainder of this section is structured as follows: Section 4.2.2 provides necessary background knowledge. Section 4.2.3 explains the concept for this approach. The PoC implementation is described in Section 4.2.4. Validation and benchmarking in an industrial testbed are done in Section 4.2.5. Finally, Section 4.2.6 concludes this section and gives an outlook.

4.2.2 Background

There is a large body of research in IDSs, ICS and SCADA networks. Works can roughly be divided into two groups – one detecting compromise of networked devices, e.g. caused by malware or control-flow anomalies [Ree+12; Jin+18], and the other concerned with detecting intrusive network traffic, into which group this work falls.

The first criterion is the type of data used for classifying traffic into benign and intrusive. For example, Liu and Liu show how voltage drops reveal the presence of an attacker in RS485 daisy chain networks [Liu+18]. The simplicity of their approach comes at the cost of being tailored towards a particular protocol and requiring modeling of the network beforehand. This illustrates one of the key problems for designing a generalized IDS for SCADA networks: A large number of networking protocols is encountered in the field. However, to be able to properly scan the network, a monitoring system must accurately dissect the traffic. Thus, it requires a precise model of every single protocol it captures, most of which are proprietary [Gol+13]. In contrast, the system proposed here uses a metadata-based approach which operates independently of the underlying transport protocol. This overcomes the need for modeling the protocol. Further, this approach brings flexibility and permits the introduced system to be retrofitted to already deployed networks.

Network traffic can be acquired and processed either centralized or decentralized. As stated in the introduction, the latter is preferable. Additionally, passive acquisition has the benefit of not interfering with the network traffic and thus avoiding the risk to interrupt production processes.

While the strict timing requirements in ICS traffic hinder the introduction of some security mechanisms, they can be exploited for distinguishing between normal and intrusive

network traffic. Barbosa et al. utilize periodic cycle time to distinguish between normal and intrusive traffic [Bar14]. However, they do not implement a distributed method. This must be considered insufficient with respect to an insider attacker, who can access the local network from anywhere within. Lin et al. also present a method which attempts timing-based intrusion detection [Lin+17]. Yet again, as opposed to this work, their system captures data centrally. Further, their system uses network traffic capture files as an input with no real world testbed. In contrast, the system introduced in this work is deployed in a real-life testbed and can adjust the baseline, which distinguishes between normal and intrusive traffic, during runtime.

Haller et al. [Hal+19] show the feasibility of an IDS based on a monitoring task and the statistical cumulative sum, running on a Phoenix Contact ILC 350-PN controller. However, this system is a basic approach for this specific PLC and mostly only handles these two detection possibilities.

There is a lot of research going on in the field of intrusion detection. Some of the published concepts are summarized in a survey on IDSs in wireless networks [But+14]. These methods have also been used within industrial networks with adjustments to its specific requirements. Another survey on IDSs and Intrusion Prevention Systems (IPSs) in SCADA networks has been published by Zhu et al. [Zhu+10]. However, in these surveys no distributed network analysis on low performance MCUs are handled.

Further work was done by Zimmer et al. [Zim+15], who introduce security building blocks for real-time CPSs. These are based on measurements of the real-time operation system with no focus to the network data analysis.

Payer shows a state-driven IDS implemented within the LwIP Stack in 2003 [Pay03]. It analyzes the states of a connection and compares them with stored database. However, this work covers only a few scenarios and does not perform well as a network IDS.

The previous work mostly focuses on a purely network-based approach with a high effort necessary for integration, or host based systems with high requirements to the computing power of the host. In contrast to previous work, the proposed approach in this work provides the following advantages:

- Distributed IDS on industrial edge node devices, e.g. sensors. This does not require changes to the network infrastructure while listening to network traffic.
- Analysis of the periodic occurrence of packets/requests.
- Approach and implementation are protocol-independent, because they utilize meta information. This enables later usage with cryptographic protection mechanisms.

4.2.3 Concept

This section sheds light on what the IDS should protect against, which data can be used for the IDS analysis and how the intrusion is announced to the operator. Further, the strengths and weaknesses of the introduced approach are discussed.

4.2.3.1 Attacker Model

In this section, both local and remote attackers are considered.

An attacker with **local access** could, for example, be an employee or a visitor. This attacker has both physical and network access. Being physically present, the attacker is able to remove edge nodes from the network by, for example, unplugging the Ethernet cable. Furthermore, he can launch simple attacks such as pressing the emergency stop button at a machine, resulting in a denial of service attack. From within the network perspective, the goal of the attacker is to be able to eavesdrop and manipulate messages from sensor nodes. Also, he can inject arbitrary messages or delay messages in order to launch a denial of service attack.

The **remote access** attacker has gained access to the internal network over the Internet or can directly attack exposed devices. Flooding attacks, which lead to a DoS, as well as spoofing and injecting messages can be conducted. MitM attacks are not possible for the remote attacker.

4.2.3.2 Overview on Workflow

The IDS operates by firstly observing the network traffic in a non-corrupted network, learning normal or benign traffic. Utilizing the deterministic timings in traffic flows, it derives periodicity-thresholds, which separate normal from intrusive traffic. Calculating the thresholds is accomplished in two ways. First, the metadata of each connection are analyzed and categorized. Second, these categorized connections are analyzed based on their periodicity. After the learning phase, the traffic of the live ICS is compared to these thresholds and classified as either normal or intrusive. To account for slight changes in the ICS traffic behavior, thresholds are adjusted during runtime.

Meta Data Selection First, it is discussed, which features of the network traffic are suitable to be used for characterizing and classifying the connections. Basically the introduced IDS approach can analyze, train, and then make decisions using all network data it receives. Figure 4.7 shows the meta information, which can be analyzed by the IDS, mapped to the layers of the network stack according to the Open Systems Interconnection (OSI) model.

Analyzed metadata is chosen such, that the IDSs operates completely protocol-independent. In this case, metadata is all data, which resides below the application layer. Thus, no further information is needed during the deployment, which means that the operator does not have to set rules. Specifically, the following information is used:

- Source and destination ports are used from the **TCP** and **UDP** header.
- From the **IP** header the source address and destination address are used.
- **ARP** requests and responses contain MAC and IP addresses, which are mapped to each other.

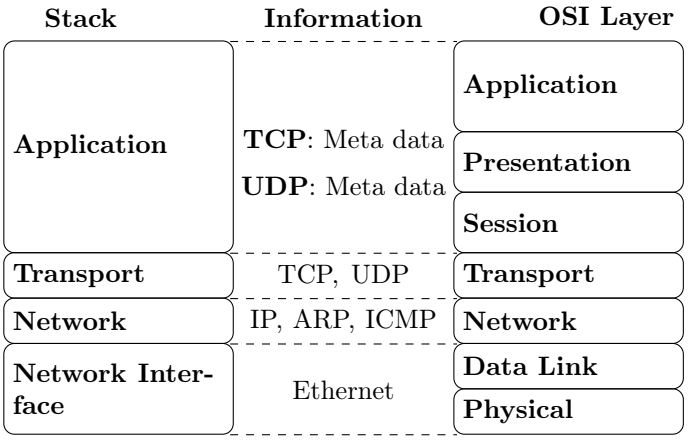


Figure 4.7: Used information from the stack for intrusion detection.

- The **Ethernet** header contains the destination MAC address and the source MAC address. Those must be consistent with each other in order to recognize e.g. ARP poisoning.
- Optionally, **meta data** from the application layer could be used.

The source and destination ports are typically unfeasible for intrusion detection because they vary: As soon as a connection is terminated, usually a new port is used by the client. This must be taken into account when categorizing the connections. It is still feasible to use the port on the server side, e.g. port 502 for Modbus/TCP to analyze the network packets. The other metadata, however, remains consistent and does not change at reconnections. Allocating metadata from connections is already a good starting point for intrusion detection. This is similar to rule-based detection, which is also used in firewalls.

Exploiting and Learning the Network Timing Behavior A specific characteristic of an ICS network is periodical polling of inputs and outputs. This creates a homogeneous timing picture of the connections in the network. In the example in Figure 4.8, first a TCP/IP connection is established (SYN). The timing pattern here is considered irregular. The same holds true for closing the connection (FIN). However, after having established the connection, in the center of Figure 4.8 periodic timing can be observed caused by periodic polling.

Periodic behavior is exploited in the proposed approach and observed in the initial training phase. During this training phase, the network needs to remain free from malicious traffic. Two methods to derive thresholds, which separate normal behavior from an intrusion are used. By filtering one specific connection, e.g. from one edge node to the central PLC, the time series is getting periodic. This mostly depends on the network infrastructure and implementation. Nevertheless, this only requires a longer learning time, if the traffic is more irregular.

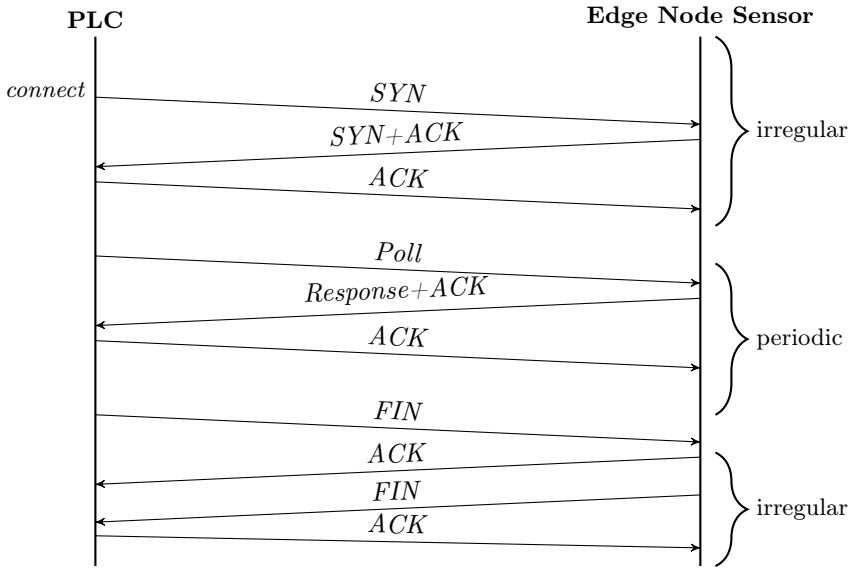


Figure 4.8: Timing in sensor networks with polling, separated in periodic and irregular timings.

Statistical Analysis of Normal Traffic Behavior During the learning phase, the minimum and maximum interarrival time is calculated. The values t_l are recorded during the learning phase, which is the interarrival time between packets of the same connection type. The IDS forms a cumulative moving average (see Equation 4.1) over the interarrival time of the packets. The mathematical symbols used in the formulas can be found in Table 4.3. The interarrival time during learning is t_l and in active mode t_a.

Table 4.3: Symbols used in formulas.

Symbol	Description
t_l	Interarrival time ($t_{arrival}$) between two packets with the same meta data during learning
t_a	Interarrival time ($t_{arrival}$) between two packets with the same meta data, when the IDS is active
n	Number of packets used for the calculation
Δ_a	Delta to configure the mean time deviation, which is tolerable when the IDS is active
Δ_{mm}	Delta to configure the minimal and maximal time deviation, which is tolerable when the IDS is active

To calculate a specific mean value, the amount of used interarrival times is divided by n_l during learning n in the active mode. This is used after the learning phase as a reference to detect changes in the frequency of packet transfer in the specific connections.

Since there may still be minimal deviations after the learning phase, a relative offset (Δ_a) is added on top.

$$\frac{t_{l1} + ... + t_{ln}}{n_l} * (1 - \Delta_a) < \frac{t_{a1} + ... + t_{an}}{n} < \frac{t_{l1} + ... + t_{ln}}{n_l} * (1 + \Delta_a) \tag{4.1}$$

After the learning phase, the moving average is further calculated and compared with the trusted reference. If this current moving average is outside of this trusted reference, an anomaly is assumed. Those calculations must be done for each network connection to the edge node device.

Minimum and Maximum Classification of Traffic After the IDS is switched from the learning phase to active, the current packet interarrival time during the activated IDS (t_a) is compared with the previous calculated maximum and minimum. If this is outside of those boundaries, this is seen as an intrusion (see Equation 4.2). In addition, an adjustable relative offset (Δ_{mm}) is specified for the minimum and maximum limits.

$$min(t_{l1}, t_{l2}, ..., t_{ln}) * (1 - \Delta_{mm}) < t_a < max(t_{l1}, t_{l2}, ..., t_{ln}) * (1 + \Delta_{mm}) \tag{4.2}$$

Network traffic is categorized and statistically evaluated using the parameters described in Section 4.2.3.2. There are parameters like IP and MAC address, which must not deviate after the learning phase. In contrast, there is metadata such as the time behavior which accepts a certain tolerance to avoid false reports. A recognized intrusion should be processed and displayed directly at the edge node, as well as transmitted to a central logging server.

4.2.3.3 Intrusion Announcement

To announce an intrusion to the network, broadcasts messages are used. As a result, the central intrusion logger only needs to be in the same broadcast domain and no configuration is necessary. Additionally, in order to detect DoS attacks which are supposed to block the messages, a keep alive message is sent at a certain time interval. If the central logger does not receive this message within a predetermined timeout interval, an intrusion is assumed to take place. Furthermore, the message must contain a signature and a changing variable to prevent replay attacks. Besides the centralized logging, local signaling can be used to warn operators within the plant.

4.2.3.4 Discussion of Strenghts and Limitations

One of the biggest strengths of this approach is the easy integration into existing networks, because no special network hardware such as as mirror ports are necessary. In addition, each edge node device can defend itself and does not have to trust other parties. Of course, this is only possible if the manufacturer implement this or an open-source component is used.

In order to get a trusted comparison base, it has to be ensured that there is no attacker in the network during the initial learning phase. To launch an attack undetectable to the IDS, the attacker has to generate the same traffic used in the training phase; otherwise, it will be recognized as an anomaly.

The determination of Δ_a and Δ_{mm} depend on how high the acceptable false detection could be. If the Δ is chosen too small, there will be many false positive (normal activity mistakenly identified as an intrusion). Otherwise, if Δ is chosen too high, false negatives, where an attack is not detected will occur. False positives and false negatives are often a problem of IDS, which also exists in products on the market [Ho+12]. This is often the reason why IDS are mostly preferred instead of IPS in industrial networks, to not disrupt production. This means that alarms can be checked first to determine whether they are an attack or a fault alarm.

4.2.4 Implementation

In this section, the feasibility of the introduced distributed IDS approach is demonstrated by providing an implementation on low-performance MCUs. This sets this work apart from many other proposed IDSs, which only use simulation for validation. The used edge node hardware is explained in detail in Section 4.1.

4.2.4.1 Stack integration

The IDS must be integrated into the software stack using some functions (e.g. system timer) of the operating system. The basic structure of the software stack on the edge nodes is explained in Section 4.1.2.2. As shown in Figure 4.9, the IDS is integrated into the widely used LwIP [Dun01] stack. All Receive (RX) and Transmit (TX) data is processed by the IDS before being forwarded to the regular Application Programming Interface (API) of the *LwIP* stack. This design is highly beneficial, as the IDS can be used without changes to existing projects, which use the *LwIP* API.

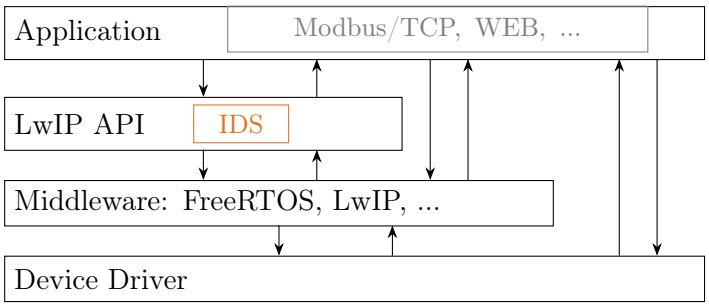


Figure 4.9: Overview of the system with integration of the IDS into LwIP.

Another advantage of using the existing LwIP API is that the IDS can also be used as a IPS. In this case, packets, which are detected as an intrusion, are not passed to the application layer and discarded instead.

4.2.4.2 Centralized Logging of Intrusions

In the introduced decentralized IDS, every sensor node individually captures and preprocesses network data. However, to permit technicians centralized administration of edge nodes, a central logging is necessary. Every sensor is reporting the current "security" state in a periodic way to this central logger. If the sensor does not send the status message within a certain time frame, because, for example, an attacker is flooding the network, the monitoring and logging server must detect and report this as an incident. The key features of the centralized intrusion and alive notification are:

- The intrusion message is send to the network via UDP broadcast.
- The message is signed with a performance efficient Keyed-Hash Message Authentication Code (HMAC)-based using a Pre-shared Key (PSK).
- The message is sent out every 10 seconds, with status information and keep alive message.
- The system time is part of the message for replay protection.

These messages are gathered by the logging server, and, if there is no keep alive message within 20 seconds, the host is regarded as contaminated.

4.2.4.3 Intrusion Notification on the Edge Node

In addition to the centralized intrusion logging, each edge node device can visualize its intrusion status using a display and/or an LED. This helps technicians in the control room, who have been warned of an intrusion, to quickly localize the affected edge nodes in the field. Without information at the devices or dedicated tools such as *EyeSec*, this has been proven to be a tedious task [Str+19]. Figure 4.10 shows the current status of the edge node device. This shows the IP of the node, the current time (Time) and learning time (LTime) in milliseconds. Furthermore, when an anomaly is detected, this is shown on the Organic Light Emitting Diode (OLED) (!!!INTRUSION!!!).

Regular behavior Intrusion detected

Figure 4.10: Current state of the IDS on the edge node display.

After network startup, the current system time and the configured learning time is displayed. If an intrusion occurs, this will be displayed on the OLED and the red LED on the baseboard lights up. All these features are additional tools, which aid operators in managing network security.

4.2.5 Evaluation and Measurement Results

To show the performance of the proposed approach, the edge node IDS is evaluated in the open-source industrial testbed introduced in Section 4.1. Additionally, measurements were carried out and the detection scenarios were identified.

4.2.5.1 Evaluation in an Open Source Testbed

Please note that the here introduced approach does not depend on a particular network architecture. For the introduced approach in this work, only periodic network traffic patterns are required. However, the network architecture only plays a minor role for the approach introduced in this work, as long as the interaction is "predictable" in a periodic manner.

IDS Webserver on the Edge Nodes The webserver enables easy remote maintenance of the edge devices from the control room. Figure 4.11 shows the web page running on each edge node.

At the top the debug information is shown. To the bottom left, the learned connections are listed. The background color of the web page displays the current status of the IDS: Learning (blue), active without intrusion (green) and intrusion detected (red). This simple yet effective visualization permits the operator to grasp the network status at a single glance.

Centralized Logger The central logger first checks the HMAC signature and whether the message time, which is used as replay attack protection, is consecutive. Listing 4.1 shows the status outputs of the central logger. In line 1, the edge node device is down, which could be caused by a failure or a DoS attack, for example. Line 2 shows the desired status, where the device is up and no intrusion is detected. An intrusion on edge node 3 is illustrated in line 3.

```
1 ID: 1 is down Intrusion: ???
2 ID: 2 is up   Intrusion: no
3 ID: 3 is up   Intrusion: yes
4 ...
```

Listing 4.1: Output of the central logger

As explained in Section 4.2.3.3, the UDP broadcast can be received from everywhere in the broadcast domain. Furthermore, the message outputs can be easily integrated in different logging mechanisms and tools, offering great flexibility to the user.

4.2.5.2 Interarrival Time in the ICS Testbed

In the following, it is described, how to statistically assess the deterministic timings in the testbed, which are then utilized by the IDS to detect intrusions.

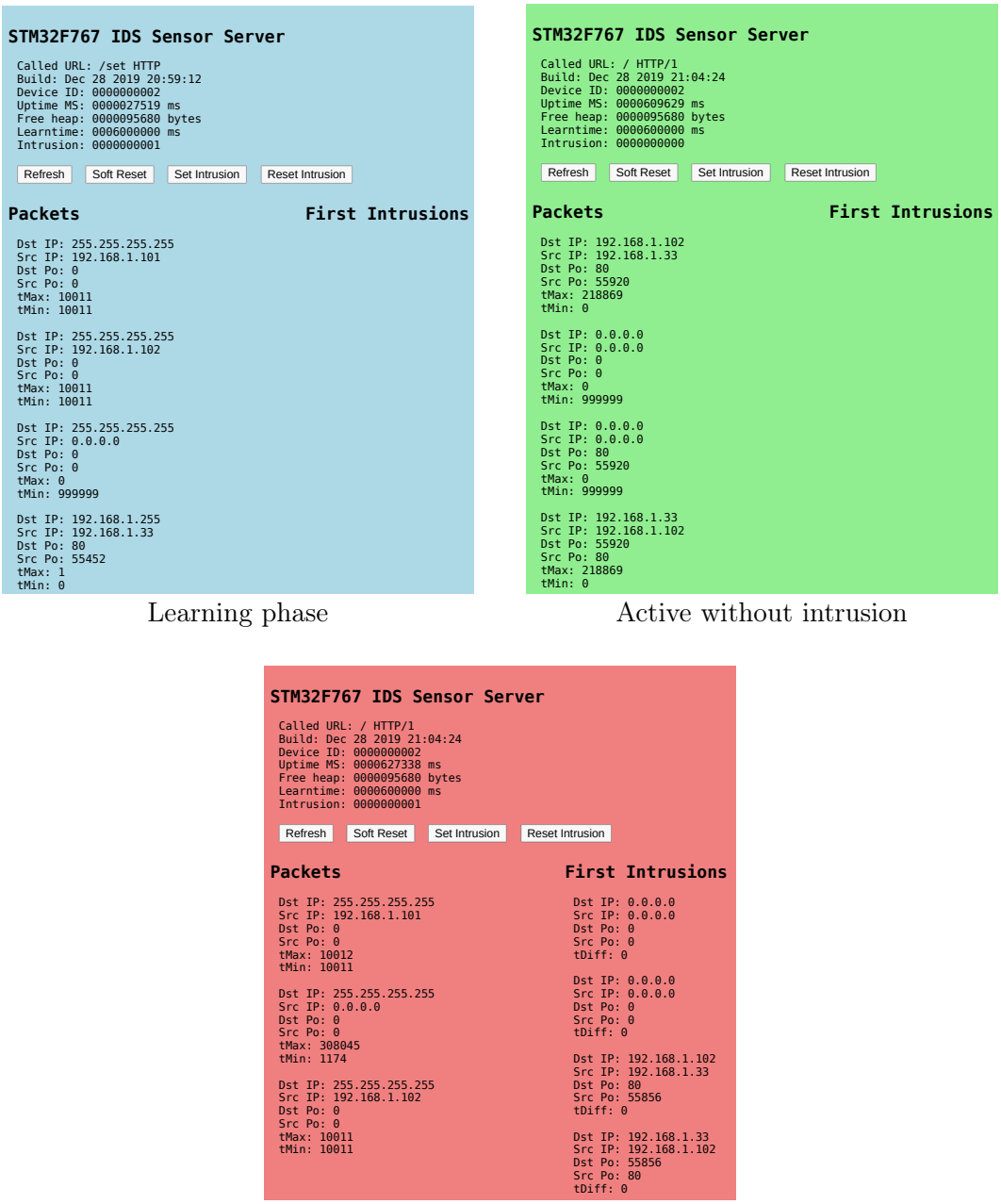


Figure 4.11: Webpage running on each edge node, displaying the current IDS status and debug output.

Assessing the Packet Interarrival Time The interarrival time is the time between packets of one connection. Figure 4.12 shows the interarrival time of Modbus/TCP packets to the sensor system in the testbed. The results of the measurements are shown in a boxplot with calculated arithmetic mean (▼) and median (–). The quantiles are 25 % and

75 %, with whiskers up to factor 1.5 of the box. The plot only contains communication on port 502, namely Modbus/TCP. The arithmetic average is about 100 ms where the Modbus/TCP "read discrete input" is executed. This reflects a normal polling behavior of an industrial PLC. The *OpenPLC* implementation polls each node. The timeout is set to 1000 ms by default. Most of the controllers have implemented such a timeout, which, if exceeded, indicates problems in the network communication.

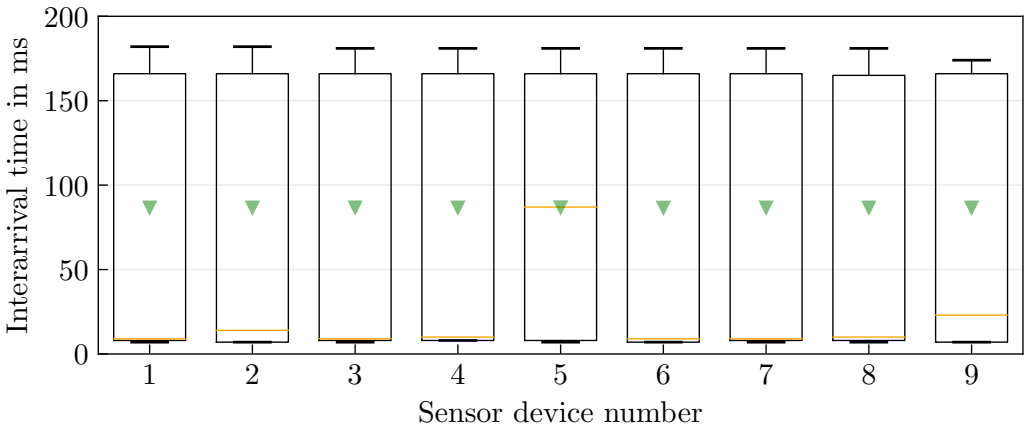


Figure 4.12: Interarrival time of modbus packets.

Assessing ARP Request Interarrival Time Figure 4.13 illustrates the interarrival time of ARP packets in the testbed. In this figure, all ARP requests and responses to the specific sensor are plotted. This results in a mean of 270 seconds every 4.5 minutes when the ARP cache is cleared. This makes it possible to perform a host-up detection of devices in the network. For example, if there is no ARP request for a long time, the device is probably offline.

Figure 4.14 shows the Kernel Density Estimation (KDE) plot with two high density areas. In this case, this type of plot represents the frequency of connections over time. The *OpenPLC* v3 implementation uses a default polling sleep time of 100 ms, which is added on top of the interaction of each node. The first peak is generated, because of the default response time of a query. This first peak represents the request from the PLC and the fastest possible answer by the sensor (request → response). The second peak is the delay not to flood the edge nodes, with the default of 100 ms. This is, in other words, the refresh rate of the PLC values.

All Modbus/TCP interarrival times of the nine edge nodes are actually shown in the plot, but they overlap so much, that almost no difference is visible. This could be different, if, for example, two PLCs poll the sensors. However, this does not make any difference in the classification per connection and does not change the core statement of this plot.

Through these illustrations it can be seen, that there is only a little time deviation in this testbed. Thus, the Δ can be chosen small. In our evaluations the Δ was set to 0.1

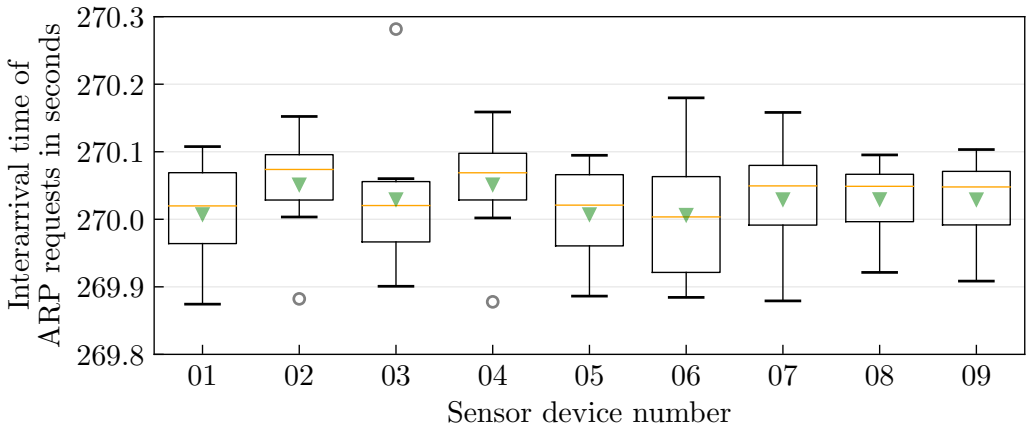


Figure 4.13: Interarrival time of ARP request packets.

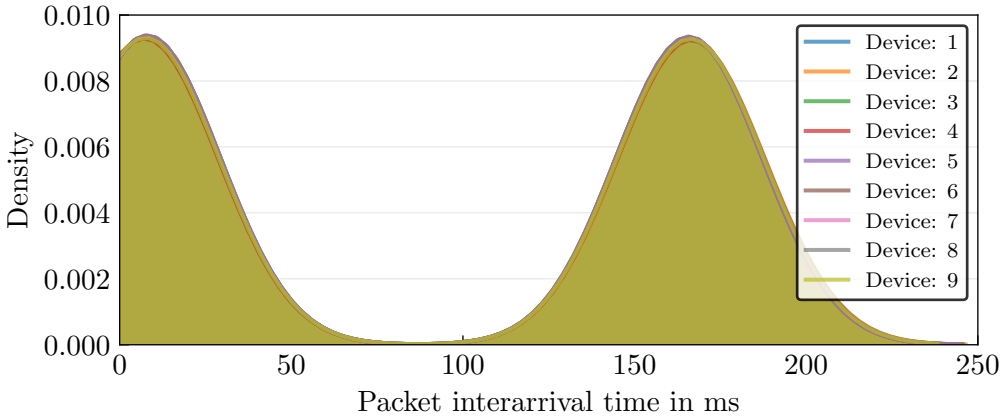


Figure 4.14: KDE of the interarrival time of Modbus/TCP packets.

which corresponds to 10 %. In this way, the IDS performed well and no false positives or false negatives occurred during our benchmark (see Section 4.2.5.3).

4.2.5.3 Intrusion Detection Benchmark of the PoC Implementation

The parameters used to detect attacks essentially depend on what has been learned and how the attack is executed.

Attack Detection The introduced attacker models in Section 4.2.3.1 are used to measure the attack detection of the IDS on the embedded edge nodes. The following scenarios are covered by these two attacker models, which are summarized in Table 4.4. These are evaluated within the open testbed.

The local attacker **removes** ❶ an edge node from the network. The centralized logger will detect this incident after some seconds, because the keep-alive message of the edge node is missing.

The network is **actively sniffed** ❷ by the local attacker with e.g. ARP poisoning. Owing to the ARP poisoning, ARP requests are sent out. Those packets are new to the IDS, resulting in an incident report. Since this attack already requires some knowledge, the attacker knowledge is categorized as medium.

Input, output, or keep-alive commands are **spoofed** ❸ by the local attacker. If an attacker connects his own device to the network and sends out ARP requests, the edge node will receive those requests from an unknown source and will report this.

The local or remote attacker **inject** ❹ packets into the network to send commands to the edge nodes. New connections will be detected by the IDS.

An attacker is flooding an edge node to perform a **DoS attack** ❺. The edge node IDS detects, that packets are either new or occur too often. Additionally, this will be detected by the logging server, because no alive message are received anymore.

The attacker is **passively sniffing** ❻ the network with a unidirectional network diode. If this is done completely passively, the IDS cannot recognize this. For this kind of attack, the attacker knowledge is low.

An attack is already executed during the **learning** ❼ phase of the IDS. Of course, if an attacker manages to get involved during the learning phase, the IDS naturally also learns it and regards this as regular. However, the attacker must continue this traffic after the learning phase; otherwise, the IDS will detect the attack.

The attacker **captures** ❽ an edge node. The IDS is only capable to detect the attack during the attack phase. After a successful attack and if the edge node is captured, the attacker can manipulate the data on the captured edge node. If the attacker opens new connections, the other still trusted edge nodes will detect this.

Table 4.4: Summary of the evaluated attack scenarios and detection capabilities.

Model	Short description	Attacker	Detection
❶	Node removed	weak	✓
❷	Active sniffing	medium	✓
❸	Spoofing attack	medium	✓
❹	Injection attack	weak	✓
❺	DoS attack	weak	✓
❻	Passive sniffing	weak	✗
❼	Learning attack	strong	❍
❽	Capture edge node	strong	❍

✓detected ❍depends ✗not detected

Time for Learning Regular Behavior The time needed by the IDS to learn regular connections depends first of all on the use case. In the case of the open-source testbed introduced in Section 4.1, there is normally a distinction between direct connections, such

as the read and write of the register through the PLC and packets like ARP requests that the edge node receives, because it is in the broadcast domain. In the testbed, the time for learning is approximately twice the time between the longest interarrival time of an ARP request, which is approximately 10 minutes. This is the absolute minimum and must always be chosen to also collect rarer events. In practice, it will make no difference whether the initial learning time is a few minutes, hours, or even one day, as long as the devices are not battery-operated.

4.2.5.4 MCU Performance Data

Figure 4.15 shows the ping of the edge node device with enabled and disabled IDS. This is measured with *fping* [Sch] every 100 ms over 100 samples. The average ping without the IDS is about 0.31 ms and 1.13 ms with the enabled IDS.

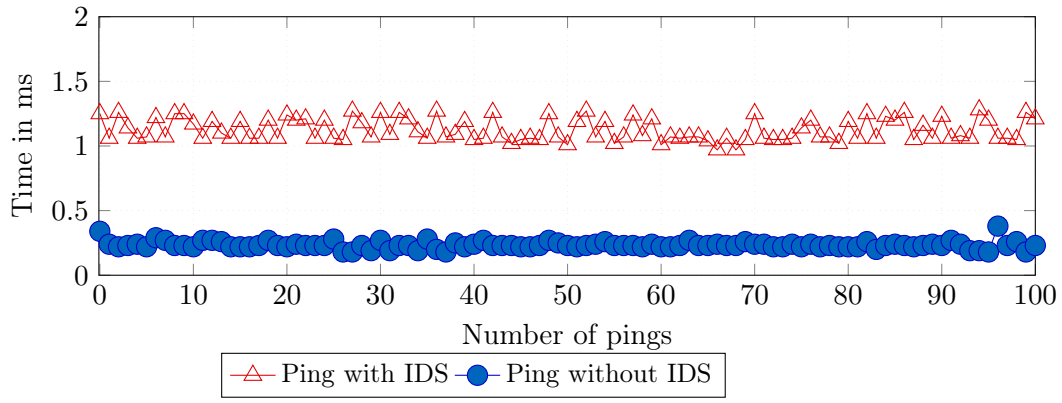


Figure 4.15: Measurement of the ping behavior with and without the IDS.

The data throughput measured with *iperf* drops from 28.2 Mb/s without the IDS to 4.23 Mb/s with the IDS. However, in the used testbed, this performance reduction does not have any impact on the controlled process, because there are only about 100 kb/s with a PLC cyclic refresh time of 50 ms. Such low traffic rates are common in industrial plants, but also with higher rates the IDS is capable of handling the traffic.

Table 4.5 shows the differences in the software build with IDS and without IDS. It is only the function analyzing the received and transmitted packets, because the other functions depend on the configuration. This means, for example, the webserver or the logging function can consume more or less RAM and ROM depending on the configuration. Hence, the comparison is limited to the analyze function of the IDS.

This analysis shows, that the actual IDS functionality requires quite few resources and thus does not waste scarce flash memory.

Table 4.5: Binary comparison of example application with and without the IDS building block in bytes.

Information	text	data	bss	dec	hex
With IDS	145176	12592	285848	443616	6c4e0
Without IDS	141872	12592	285832	440296	6b7e8
Difference	3304	0	16	3320	cf8

4.2.6 Conclusion

This work shows that network-based IDSs on low-performance MCUs are a workable and potential way of detecting intrusions in industrial networks. Especially the periodical nature of polled communication between PLCs and remote sensors and actors allow effective detection mechanisms. The measurements help to detect numerous network-based cyber-attacks, while the detection itself affects the network traffic only slightly. The introduced IDS approach can either be used as a single point of defense or be easily combined with firewalls, for example. Combined with the protocol neutrality of the metadata approach proposed in this work, the distributed IDS can be effortlessly integrated within already existing projects as an autonomous level of security. Being able to base the implementation of the IDS as a function on the widely used *LwIP* stack provides a decent foundation to upgrade the system to an IPS. This is possible because as an intrusion a detected packet can be dropped before it is being processed by the regular application running on the MCU. Good detection results paired with the modularity and easy-to-integrate nature of the proposed approach make it a reasonable choice to tackle the upcoming security problems of Industry 4.0.

4.3 Network Scanning on Industrial Internet of Things Edge Devices

Contents of this section

Parts of this section have already been published in the paper "Network Scanning and Mapping for IIoT Edge Node Device Security" at the 24th International Conference on Applied Electronics (AE) 2019 [Nie+19a].

4.3.1 Introduction

In a common industrial system at the field level, which is generally IP-based nowadays, the network structure rarely or never changes. This means that changes to the network are either caused by maintenance, malfunction, or an attack. Independent of what caused these changes in the network environment, the incident must be detected, because this is a deviation from regular behavior and an operator has to decide how to react to it.

In this section, an active network mapping tool is introduced as a building block for embedded low-cost edge node devices. This approach differs from the passive scan option in Section 3.5 and the IDS introduce in section Section 4.2. Here the previously introduced problems of active scanning are revisited, and an approach with slow and distributed scanning is presented.

The here introduced security building block enables probing devices (hosts) and services in the network directly from an edge node device connected to this network, without the requirement of additional components. After the edge node is placed into the network and the system is put into operation for the first time, the edge node scans the network and learns the structure of the network architecture. This default network "fingerprint" is stored locally on the edge and is compared with further scans. If the network changes during further scans, this indicates an anomaly, which will be reported to the operator, for example.

The concept of edge node based network mapping presented here has the following properties:

- Easy integration with current network state recognition of other participants.
- New devices in the network can be found.
- New services with open ports will be detected.

- Hosts and services which change the status are recognized.
- The information of the network scan is only on the "intelligent" edge node so that attackers cannot exploit this feature. Thus, the edge node only has to trust itself and no third parties.
- The network scan is done in a pseudo random manner for load balancing, and the attacker cannot retrace and exploit the scan process.

The section is structured as follows: The related work is discussed in Section 4.3.2. Section 4.3.3 explains the concept of network scanning and mapping. Section 4.3.4 introduces the PoC implementation. To show the feasibility, an evaluation is done in Section 4.3.5. At the end, a conclusion is given in Section 4.3.6.

4.3.2 Related Work

There are different active and passive network scanners available on the market and discussed in research:

One of the best-known network scanners is *Nmap* [Lyo09]. It offers a wide variety of scan options, as well as various scripts for further analysis. However, this requires a comparatively high-performance computer compared to an embedded MCU used in an IIoT edge node. Additionally, this is usually done by scanning the network from a central point, meaning that either an additional scan node must be placed in each separate subnet or only certain subnets can be scanned.

Wedgbury et al.[Wed+15] gave an overview of passive network scanners for ICSs. Furthermore, different SCADA network monitoring tools and scanners are compared by Coffey et al. [Cof+18]. All these specialized ICS scanners also call for high system performance and produce a huge amount of data to process.

An internet-wide search engine (*Censys*) for SCADA devices was introduced by Durumeric et al. [Dur+15]. This is also capable of scanning internal networks with the help of *zgrab2* [Bas+16]. However, this is a full featured active scanner that does not run on small edge node devices either.

ModScan, a Modbus/TCP enumeration scanner, was introduced by Bristow et al. [Bri08]. A specialized vulnerability network scanner for Siemens devices was introduced by Antrobus et al. [Ant+16], which is based on a modified version of *PLCScan* [Efa12]. However, these basic scanners are written in the Python scripting language and are again not suitable for MCUs.

Wang et al. [Wan+06] and Radhappa et al. [Rad+18] summarized the current problems and open research questions existing in wireless sensor networks. However, they did not handle intrusion detection with regard to port scanner and network mapping on low performance edge node devices.

Webster et al. [Web+06] describing their experience using active and passive mapping. The passive scanning approach often took several days to discover a new service and the active scanner was not able to find services, without the corresponding protocol implementation. Table 4.6 shows an overview of properties of active and passive scanners.

Table 4.6: Generalized comparison of passive network monitoring (e.g. IDS) and active scanning.

Property	Passive network monitoring	Active scanning
Necessary CPU performance	high	low
Results clearance	medium	high
Time to get results	high	low
Additional network traffic	none	yes

Another possibility is to combine active and passive systems with each other in order to use the advantages of both systems [Jar+16]. In this way, for example, the concept presented here, the IDS in Section 4.2, and the passive scanning in Section 3.5 could be combined.

4.3.3 Concept

Figure 4.16 shows an example of network mapping seen from an edge node device. In this example network, there are four nodes (N1-N4), and N1 scans the network in a pseudo random periodic manner. The dashed arrows show the first scan results with 4 devices (N1-N4) found. This first scan is used as a reference and it must be ensured that this network is not already contaminated. Thereafter, an additional edge node device gets into the network (A). Thus, the second scan (dotted) detects the expected devices (N1-N4) as well as the new device (A). This could indicate an intruder or other processes causing a change in the network, e.g. maintenance work.

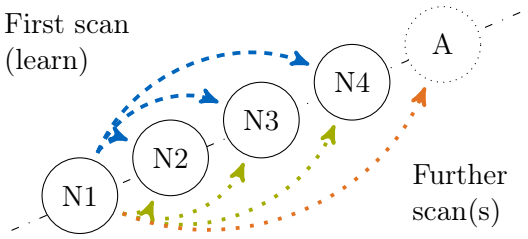


Figure 4.16: View of the network mapping from an IIoT edge node device of two different scans.

For network scanning, followed by mapping, where the connections of the hosts are analyzed, the following parameters can be used. These are grouped into two classes and introduced in the following section:

- **Host alive discovery**: With an Internet Control Message Protocol (ICMP) ping sweep, the active hosts in the network can be detected.
- **SYN/connect scan**: Open ports and services are detected with SYN and connect scanning.

- Optional: The **ping timing** can be used to detect redirection, e.g. MitM attacks.
- Optional: At the application level, e.g. Modbus/TCP, a more detailed **fingerprint** is possible.

In this work, the approach introduced combines the methods presented here and integrates them as a security building block for IIoT edge node devices. To the best of the author's knowledge, no network scanner for low-performance MCU devices is available at present.

One of the biggest advantages of distributed scans is that different paths and subnets can be easily scanned. This is, for example, the case if networks are strongly segmented, and the already existing IIoT edge devices in this subnet take over the scanning and no additional scanning hardware is necessary due to a software update.

4.3.3.1 Host Alive Discovery

The first step that is executed in a network scan is the detection of whether a device is active or not. Called "host alive discovery," this method is done via a ping sweep at the IP ICMP level. If this is done in a local network, it results in an ARP request, and if the host answers (ARP response), it is up and is tried to be pinged. If a host does not respond to ICMP ping messages, it does not necessarily mean that it is non-existent; it may also be possible that it has just disabled ICMP echo. In this case, it is possible to do a port scan anyway, which is time-consuming.

4.3.3.2 SYN/Connect Scan

Figure 4.17 shows the flowchart of SYN and connect scanning. If the port is closed but the host is up, the target responds with a RST directly after the SYN packet. If the host is up, but does not send any packet at all, then a packet filter is active. After the target host has answered, a RST packet is send if SYN scanning is used. In contrast, if a connect scan is executed the SYN+ACK message from the target host is acknowledged, and it is possible to get the data/banner from the target. This data/banner is sent by many services once the TCP handshake is complete.

The advantage of SYN scanning is that the data does not reach the application level, and therefore, there are no log entries in the application. However, since this is not used in preparation for an attack and the goal is not to stay under the radar. The preferred method is a connect scan, because more information from the target host can be collected. In addition, Soulie [Sou14] recommends performing connect scans within ICS networks to reduce influences on the process. This is the preferred scanning method, especially in fragile ICS networks.

4.3.3.3 Pseudo Random Scanning

On the one hand, the selection of the target host to be scanned must be chosen randomly, as an attacker might otherwise hide himself. On the other hand, if more scanners are in

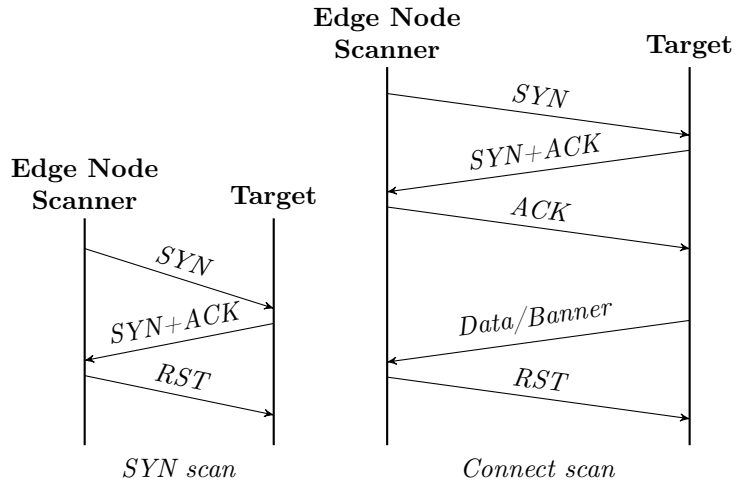


Figure 4.17: Flowchart of a SYN and a connect scan with an open port on the target.

the network, they should not flood one target. Furthermore, the start time of the scan is randomly chosen between one and five minutes after the edge node is switched on. This delay after the startup is necessary so that other devices can finish booting, and if there are multiple scanners in a network, the network load will be further distributed. For attackers, the pseudo random scanning on distributed edge nodes makes it difficult to guess the scan pattern.

4.3.3.4 Intrusion Detection Handling

If a new device is detected in the network, a known device is no longer reachable, or ports/services have changed, this should be regarded as an incident. In this case, the edge node could go into a safe state or report the incident to a centralized logger. This depends on the respective field of application. The advantage here is that the direct processing on the edge node allows a fast and independent reaction. This is because there are no dependencies and long runtimes, e.g. through network communication to a central server.

4.3.4 Proof of Concept Implementation

To prove that the approach is feasible on an embedded MCU, it was implemented for the usage in the test environment, which was introduced in Section 4.1.

4.3.4.1 Hardware

The hardware for the edge node scanning is the same as described in Section 4.1.2.1. The custom PCB provides input and output capabilities and an I^2C display connector to show current scan details. The current scan progress and intrusion message can be

displayed on the display of the edge node device itself. Figure 4.18 shows the 1.3 inch OLED display with an SH1106 I^{2}C driver.

Regular scan Intrusion detected

Figure 4.18: Scan progress and results on the edge node display.

4.3.4.2 Software

As shown in Section 4.1.2.2, the software on the MCU uses *FreeRTOS* [Bar+08] with the *LwIP* [Dun01] stack. The edge node device provides a Modbus/TCP slave, to control the IOs and a web server to provide current information.

The scan process is shown in Figure 4.19. The data of the first network scan is regarded as a secure state and will be used as a reference for later scans. After this, the scans are executed periodically and the results are compared with the results of the initial scan, which is treated as a trusted dataset. In case any mismatch is detected during the following scans, intrusion handling is initiated.

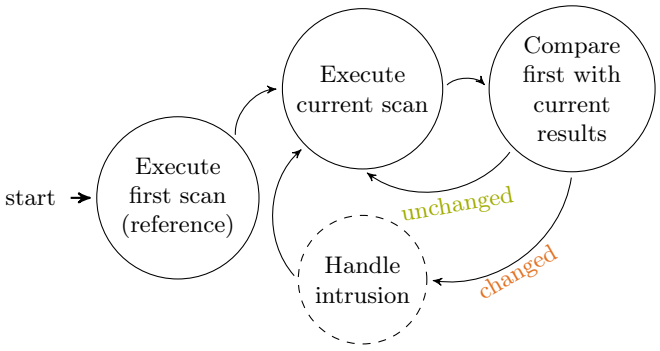
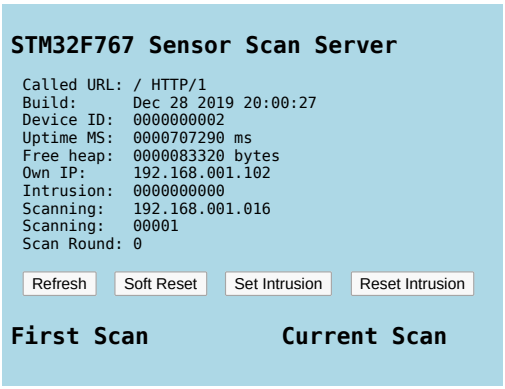


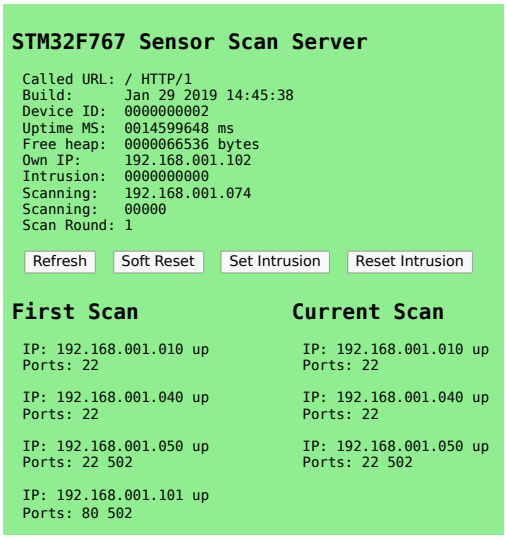
Figure 4.19: High-level view of the scan process. After the first/trusted scan, a continuous monitoring is run.

The scan module is designed as a task in *FreeRTOS* and could be used as a building block in other devices as well.

Figure 4.20 shows the current debug output of a scanning edge node device. On the lower left side, the data of the trusted scan can be seen, which serves as a reference dataset. On the lower right side, the output of the current scan progress is illustrated. The IP with the alive status is printed as well as the open TCP ports. During the learning phase (first scan), the background color is blue. If there is no network change detected the background color is green, and otherwise red.



Learning phase (first scan)



Current scan status without intrusion

Figure 4.20: Webpage running on the edge node, displaying the current scan status and debug output.

This representation is not for productive usage, because attackers could use this information for aimed attacks. Preferably, only the intrusion message with the changes is sent cryptographically protected to a centralized logger or only allows access to authenticated users. However, this setup depends largely on the integration of the scanners, e.g. if there are local operators with access to HMIs or a centralized control who can react to the incident.

4.3.5 Evaluation

To show the feasibility of the approach presented here, an evaluation is done. This is divided into four parts. First, the feasibility in the previously introduced open-source industrial testbed is measured, then the network performance is evaluated. After this, the MCU requirements are measured, and at the end the attack detection is evaluated.

4.3.5.1 Industrial Testbed

The PoC implementation is evaluated in the introduced open-source industrial testbed (see Section 4.1). There, the introduced network scanner runs on each IIoT edge node, which is accessed and controlled by an *OpenPLC* [Alv+14] instance over Modbus/TCP running on a Raspberry Pi. Eight edge nodes are each connected to one sensor and one edge node is connected to a motor rotating a disc. Furthermore, there is an HMI displaying the current state of all edge node devices.

4.3.5.2 Network Performance Measurement

Figure 4.21 shows the number of packets per second during scanning of one edge node in the open-source testbed. As shown in Section 3.3, high scan rates can affect the control behavior of PLCs. Therefore, the number of packets must be low, depending on the components in the network. As an example, the parameter in our testbed is set to 100 ms delay between pings. This affects the number of packets per second of ARP and ICMP packets. This wait time between packets is set to a high value, because ARP requests are broadcast to the complete broadcast domain and, as a result of this, it affects all devices within this subnet. Further, the delay between each single port scan is also set to 100 ms to reduce the network load. Both delays can be changed easily to fulfill the custom requirements of a certain industrial network.

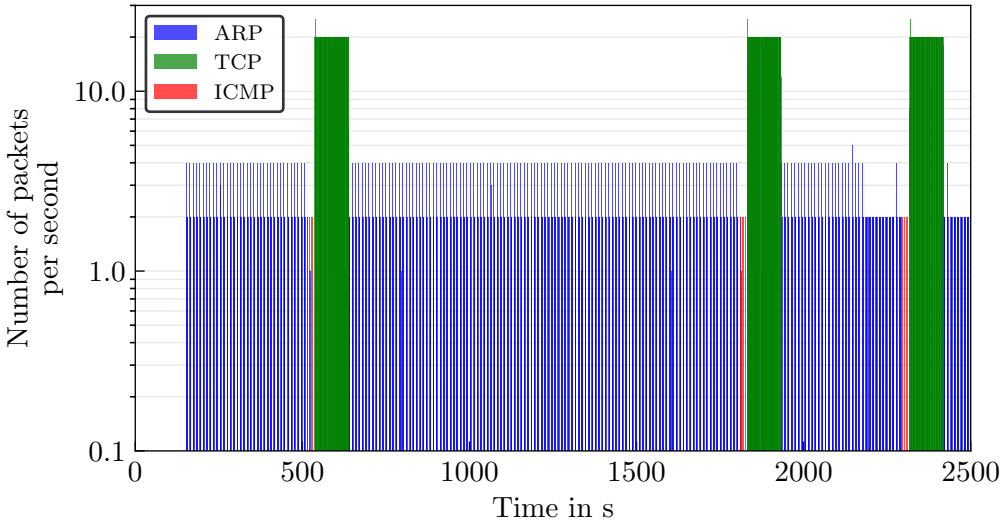


Figure 4.21: Plot over time, with packets per second of an edge node scanning the network. ARP and ICMP ping requests are used to check if the hosts are up. TCP connect scans are done if the host is up.

Initially, the scan is delayed for a few seconds. After a power-up, not all edge nodes start scanning the same IP at a time (see Section 4.3.3.3). After this, ARP requests for each IP address are sent out, resulting in a maximum of 4 packets/s. If an ARP response is received, an ICMP ping is executed (max 4 packets/s). This means that the host is reachable and the first 1024 TCP ports are scanned, which is done with a maximum of about 25 packets/s, as shown in Figure 4.21. This depends on the state of the port, for example, if it is open or closed. In comparison, the standard Modbus/TCP traffic in our testbed is about 400 packets/second between each node and the PLC. To distribute the load, if more nodes are scanning, the host selection is pseudo randomized (see Section 4.3.3.3).

An overview of packet sizes is given in Table 4.7. For example, 25 *SYN* packets/s with a size of 60 bytes each generate a throughput of 1500 bytes/s (12 kb/s). The 25 packets/s is a mixed calculation for open and closed ports. This is because, with closed ports, only

one ACK+RST returns from the target. In contrast, an open port results in a three-way handshake as illustrated in Figure 4.17.

Table 4.7: Data size of different packets from our scanner or as a response to it.

Packet	Bytes
ARP request	60
ARP reply	60
ICMP ping request	74
ICMP ping reply	74
TCP SYN	60
TCP SYN/RST	60
TCP SYN/ACK	60
TCP FIN	60
Example SSH banner	95

4.3.5.3 MCU Requirements

Table 4.8 shows the build output of the different sections in bytes. The *FreeRTOS* task uses no more than 2,048 words of stack and can run with a low priority. Additionally, the time between packets can be set to a high value, which results in a sleep (blocked state) of the scan task, whereby other operations can be performed.

Table 4.8: Binary comparison of example application with and without scanner in bytes.

Information	text	data	bss	dec	hex
With scanner	140040	12588	293704	446332	6cf7c
Without scanner	129368	12588	293552	435508	6a534
Difference	10672	0	152	10824	2a48

Most MCUs-enabling networking should have enough performance to handle the additional scan task due to the relatively low RAM and ROM requirements. Nevertheless, by optimizing the code, the requirements of the scan building block can be further reduced.

4.3.5.4 Attacker and Detection Consideration

The detection in the testbed depends on the scenario and the configuration of the attacker device. Furthermore, a trusted scan with a clean network at the beginning must be ensured. For this reason, five possible attack scenarios are modeled and evaluated within the open testbed:

❶ One edge node is removed from the network. This can happen, for example, when an attacker removes the device or by a malfunction. The type of attack requires little knowledge of the specific target. Therefore, the attacker is considered weak.

❷ Services offered in the network have disappeared or new services have been added. This can happen when an adversary attacks services which crash or introduces back-doors that open new ports. This type of attack requires moderate attacker knowledge, because changes to the network are made.

❸ An attacker attaches a standard configured computer to the network. There is no special configuration made by the attacker to be undetectable. Adding a computer to perform a port scan, for example, requires little knowledge and can be done by a weak attacker.

❹ A MitM attack is executed. In this case, the attacker has complete control over the traffic between two or more network participants. This enables viewing and manipulating the data. For this scenario, the attacker knowledge is medium because of the necessary high privileges.

❺ An attacker is performing a "stealth" attack [Sin+15]. For example, the attacker is passively listening to the network traffic and makes a "stealth" port scan. This passive attack is a special attack on a network, where a system is secretly monitored and scanned passive, for example, for open ports and vulnerabilities. The purpose is solely to collect information about the network and hosts. No data is being injected into the destination network by the attacker. This scenario requires a strong attacker due to the necessary knowledge.

For scenario ❶, our network scanner detects the changes, because the host is not reachable by pings anymore and can handle the intrusion. If services with open ports change (scenario ❷), they are detected by the port scan. A standard configured computer (scenario ❸) even without open ports can be found by ICMP pings. If a MitM attack (scenario ❹) is executed, in some cases the latency of pings is getting higher. In this case, the MitM attack could be detected by analyzing the ping timing; otherwise, it is not possible with this approach. Stealth attacks or passive listening (scenario ❺) cannot be detected by active scanning methods, such as the edge node scanner presented here. Table 4.9 summarizes the detection of different scenarios.

Table 4.9: Summary of the evaluated attack scenarios and detection capabilities.

Model	Short description	Attacker	Detection	Mechanism
❶	Node removed	weak	✓	ICMP ping
❷	Service changed	medium	✓	SYN scan
❸	Standard attack	weak	✓	ICMP ping
❹	MitM attack	medium	❍	timing
❺	Stealth attack	strong	✗	–

✓detected ❍depends ✗not detected

4.3.5.5 "Stealth" Attacker Configuration

Case ❺ is possible if the attacker suppresses any network interaction, has no open ports, and disables ICMP echo (Listing 4.2).

```
1 echo "1" >  /proc/sys/net/ipv4/icmp_echo_ignore_all
2 ip link set dev enp0s31f6 arp off
```

Listing 4.2: Command to disable ARP and ICMP echo in Linux.

In this case, it is not possible for the network scanner to detect the device. The attacker must be aware that there is a continuous network scanning so as not to get detected. Additionally, knowledge is required about how the network is configured and if local networks are not allowed to respond to ARP requests.

4.3.6 Conclusion

In this section, a network scanning and mapping building block for embedded low-cost IIoT edge node devices was introduced. Furthermore, the feasibility of the presented approach was evaluated in the previously introduced open-source industrial testbed (Section 4.1). The introduced network mapping concept is lightweight and the results are clear and detailed in contrast to most passive network monitoring approaches.

The amount of additional traffic in the network with the open testbed sample configuration with a mean of 4 packets/s and peeks up to 25 packets/s from a single edge node is low and could be adjusted, if necessary. Furthermore, the integration within a *FreeRTOS* scanning task and the configuration for the network can easily be done in other projects. Using the here presented building block, the security level of low-cost edge node devices, e.g. for securing the IIoT, can be increased.

4.4 Low-cost Industrial Control System Testbed for Education and Research

Contents of this section

Parts of this section have already been published in the paper "LICSTER – A Low-cost ICS Security Testbed for Education and Research" at the 6th International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR) 2019 [Sau+19].

4.4.1 Introduction

In order to make ICSs more secure, research is needed, which is, however, difficult to conduct on productive systems, since these often have to operate 24 hours on 7 days a week. Testbeds are mostly very expensive or based on simulation with no real-world physical process. In this section, Low-cost ICS Security Testbed for Education and Research (LICSTER), an open-source low-cost ICS testbed is introduced, which enables researchers and students to get hands-on experience with industrial security for about 500 euros. All necessary materials to quickly start ICS hacking are provided, with the focus on low cost and open source for education and research.

In principle, there are three types of testbeds – virtualized, real-world, and hybrid. Just as there are different types, there are different tasks for which a testbed can be used. For security scenarios and attacks on an ICS in particular, a real-world testbed incorporating a physical process is preferred to fully understand the effects and attack vectors in a production environment. Unfortunately, purchasing real industrial hardware for a testbed is very expensive, and particularly for education and research often not affordable. Additionally, the proprietary devices prevent pervasive changes, which makes research partly difficult.

In this section, LICSTER, an open-source, low-cost ICS testbed is presented with the following contributions:

- Testbed components for **about 500 Euro**, which is affordable by most researchers and students.
- A real-world physical process controlled by an ICS, which enables to demonstrate and analyze the impacts of cyber-attacks in the real-world.
- The **feasibility** of the testbed is shown and ideas for **research** are discussed.

- The components are **open-source** and **open-hardware**, as far as possible. This allows a wide range of further research.

- **Attacker models and attacks** to understand threat scenarios in industrial environments are provided.

The section is structured as follows: In Section 4.4.2, the concept of LICSTER is presented. Section 4.4.3 describes the proposed implementation of the components. The section continues with an evaluation in Section 4.4.4, with a discussion about further training and research questions. Eventually, Section 4.4.5 concludes this work.

4.4.2 Industrial Control System Testbed

Setting up a testbed is not the final goal but simply a tool to achieve a bigger objective. This makes it crucial to have a clear understanding of the objectives and their constraints before beginning to design a testbed. Especially for an ICS security testbed for education and research, having clear attacker models and a prepared list of desired attack scenarios is valuable.

4.4.2.1 Testbed Requirements

As Green et al. [Gre+17] concluded, a testbed, sophisticated and versatile as it may be, has little use unless it is broadly accessible. Therefore, open source is an essential requirement. Additionally, many scenarios should be covered with LICSTER, resulting in the following requirements:

- In order to make the testbed affordable for teaching and research, it must be designed for **low-cost**.

- A **physical process** must be represented to study consequences of cyber-attacks on ICSs.

- In order to obtain repeatable results, the entire process must be **reproducible**.

- The testbed must be **portable**, e.g. for teaching and demonstration to gain awareness. Furthermore, a small footprint is easier to handle when modifying components.

- By using **open-source** software and hardware, research on components is feasible and widely accessible.

- The testbed implementation should cover **level 0 to level 2** of a common ICS, focusing on the physical process and SCADA environment.

4.4.2.2 Related Work

Owing to the interest in ICS security, a lot of testbeds were created around the world in the past years. Holm et al. [Hol+15] described and compared 30 testbeds in a survey. However, the testbeds listed there are either expensive, closed-source or virtualized. LICSTER clearly distances itself from those testbeds with its clear focus on open-source and low-cost components.

Testbeds for ICS are expensive, especially when they are built with standard hardware, for example the testbed used in Section 3.1 is about 35 000 euros. This testbed fulfill the task of specific robustness tests of ICS components. However, the size and the cost involved make it unattractive for most researchers and students.

Queiroz et al. [Que+09] describe a modular testbed based on Modbus/TCP. However, they only show simple DoS attacks and make changes difficult because of proprietary hardware.

Green et al. [Gre+17] describe ten lessons learned by setting up an ICS testbed. They built up a huge testbed, with the conclusion that local access and a mobile demo unit are essential.

McLaughlin et al. [McL+16] summarize the ICS landscape and also describe the requirement of testbeds. They highlight the need for real-world physical consequences and their monitoring. LICSTER matches these requirements and additionally enables monitoring the physical process.

Maynard et al. [May+18] and Formby et al. [For+18] introduced an open framework for SCADA virtualization and simulation. However, a pure simulation or virtualization does not fulfill the requirements of the testbed introduced in this work. Nevertheless, this can be taken into consideration as an expansion.

Foley et al. [Fol+18] use a *Fischertechnik* simulation model for cyber security science hackathons. The basic idea is similar, but in this testbed proprietary components are used and thus it is significantly more expensive.

4.4.2.3 Attacker Models

There is no single defense mechanism to mitigate all threats to a digital system. Depending on the nature and origin of an attacker, some defenses might be less useful than others. Therefore, before defining the possible attack scenarios against LICSTER, the potential attackers must be defined.

The **remote attacker** has network access to the ICS through a router. That means that the attacker can reach the system only via its IP address and thus preventing attacks below the OSI network layer three [Day+95]. This replicates the scenario of exposed control systems [Dur+15], for example, when devices are connected to the internet for maintenance reasons.

In contrast to the capabilities of the remote attacker, the **local attacker** has direct access to the ICS. Being present at the plant site allows, on the one hand, the possibility for physical attacks on the individual ICS components, e.g. sensor tampering. On the other hand, there is the direct access to the network switch where ICS components are

connected to. This enables an attacker to perform ARP spoofing and all the attacks that rely on it.

4.4.2.4 Attack Scenarios

A central distinction between traditional office networks and production networks is that an ICS network has to manage and control the physical processes. That makes it all the more important for a relevant ICS testbed to incorporate a physical process, as it allows attacks on the system from an entirely different perspective. With LICSTER, various attack scenarios on the ICS level 0 to level 2 are possible. This means, as shown in Figure 2.1, that attacks from the process level to the SCADA level can be performed within LICSTER.

The act of network **sniffing** can be separated into two methods. The first is a passive approach. An attacker can utilize a mirror port or network tap to capture the traffic or simply receive and read broadcast messages. The second method of network sniffing is an active technique, where traffic is redirected over the host of the attacker by manipulation on the MAC layer through ARP poisoning, for example.

Less complicated is the **DoS** attack, where the target is flooded by network packages it needs to react to. As the number of requests is high enough, the accumulated network and/or CPU load reacts to each and every package, eventually causing the regular execution of the target to slow down or stop completely.

With a **MitM** attack, an intruder manages to place himself between two communication partners through the manipulation of routing information on the IP layer or MAC layer. There the attacker is able to capture and/or manipulate the exchanged packages.

Additionally, a **manipulation** over the network of ICS components is possible, as shown in Section 3.2, where the network interface of a PLC is fuzzed to manipulate the PLC.

Apart from network-based attack vectors, a culprit with **physical access** has a wide range of attacks at his disposal such as manipulating devices, sensors, plug- and unplugging systems, and straightforward destroying components of the ICS. In the following, a list of possible attacks mapped to the ICS levels is provided:

Attacks on level 0 (process level):

- Manipulating the physical process, for example, by removing the commodity
- Physically damaging machines so that the process is not performed properly, e.g. with raw violence

Attacks on level 1 (field level):

- DoS attack on sensors or actuators
- Interfering with availability by disconnecting the network or power plug
- Manipulating a sensor physically so it transmits spoofed values

- MitM to manipulate the values between the PLC and remote IO

Attacks on level 1 (control level):

- DoS attack on the PLC or HMI
- Sniffing network traffic, e.g. to get sensitive production information
- MitM to manipulate the values of the PLC or HMI
- Physical access to the HMI, e.g. to stop the process

Attacks on level 2 (process control level):

- DoS attack on the SCADA and historian systems
- Sniffing network traffic, e.g. to get sensitive process details
- MitM to manipulate the values of the SCADA or historian server
- Manipulating the state of the SCADA system, e.g. to reduce the daily order count

Attacks at various ICS levels require different access rights and tools, which are described in detail in Section 4.4.4 of the evaluation of LICSTER.

4.4.3 Testbed Implementation

The presented testbed handles the physical, field, control and supervisory levels of the industrial automation pyramid as described in Figure 4.22. The implementation of the testbed is designed in a way to allow for Industry 4.0 scenarios, like using a smartphone as an HMI, as well as the more traditional ICS cases represented by the industrial automation pyramid. Thus, the entire communication between the PLC, the HMI and even the remote IOs is based on TCP/IP protocols, namely Modbus/TCP. The sensors themselves communicate with the remote IOs on a fieldbus protocol, for which Modbus/TCP was chosen which is broadly used in the industry.

An overview of the devices in the testbed is given in Table 4.10 and described in detail in the following sections (Section 4.4.3.2 to Section 4.4.3.4). The total amount of 577 euros is not necessarily the cheapest choice, because it depends on the prices of the distributor.

Figure 4.23 shows the testbed mounted in a 3d printed case, which makes it easily portable. The physical process is placed on top. In the front panel, the HMI and the OLED screens of the remote IOs are mounted. The network switch, the two remote IOs, and the two Raspberry Pis are fixed inside.

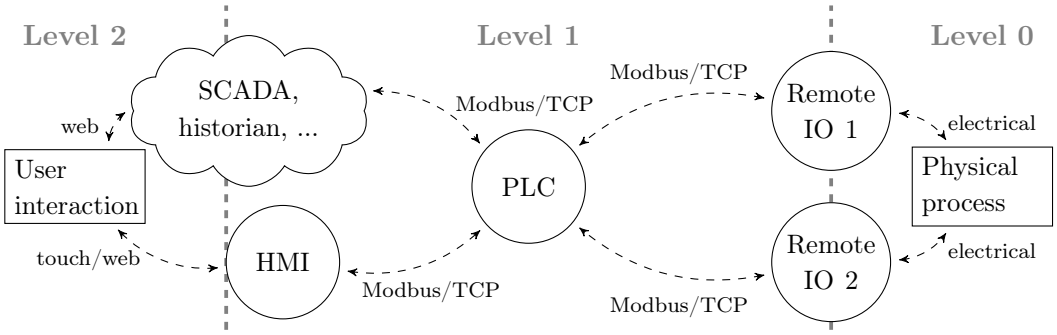


Figure 4.22: System view of LICSTER.

Table 4.10: Overview of devices used in the testbed. Prices are current prices on Amazon.

Component	Software	Hardware	IP	Price ca.
Remote IO	FreeRTOS, LwIP	Custom, STM32F7	192.168.0.51/52	79 Euro
PLC	OpenPLCv3	Raspberry Pi 3	192.168.0.30	56 Euro
HMI	Custom, PyModbus	Raspberry Pi with Display	192.168.0.20	139.- Euro
SCADA	ScadaLTS, Logging	Raspberry Pi	192.168.0.10	56 Euro
Switch	-	TP-Link	192.168.0.1	32 Euro
Process	-	Fischertechnik	–	195 Euro
Others	-	e.g. cables	–	20.- Euro
Total				577 Euro

4.4.3.1 PLC

As PLC the open-source solution *OpenPLC* from Alves et al. [Alv+14] is used. It is a soft PLC, meaning it can be run on various operating systems and on hardware with and also without IOs. Within LICSTER, *OpenPLC* runs on a Raspberry Pi, with a PLC procedure programmed in ST which is uploaded to the PLC over its own web portal.

The program contains a simple, repeatable process which can be triggered and monitored by the HMI as shown in Figure 4.24. The process consists of five stages in which it moves and processes a plastic cylinder as a workpiece example.

1 Before starting, the initial conditions must be fulfilled with everything at rest and the cylinder at its place. 2 The conveyor belt moves the plastic cylinder to the punching machine and stops right underneath it. 3 The punching machine moves downward until its lower limit switch is triggered. 4 then continues to move up again until its upper limit switch is triggered. 5 Finally, the conveyor belt moves the plastic cylinder away from the punching machine, back to its origin and then stops. 6 If a detectable error (e) occurs or the emergency stop is pressed, the machine goes in an error state and stops every movement. In this case, the process handling in the event of an error or an emergency stop are identical, in order to achieve a safe stop state. This can be reset on the HMI.

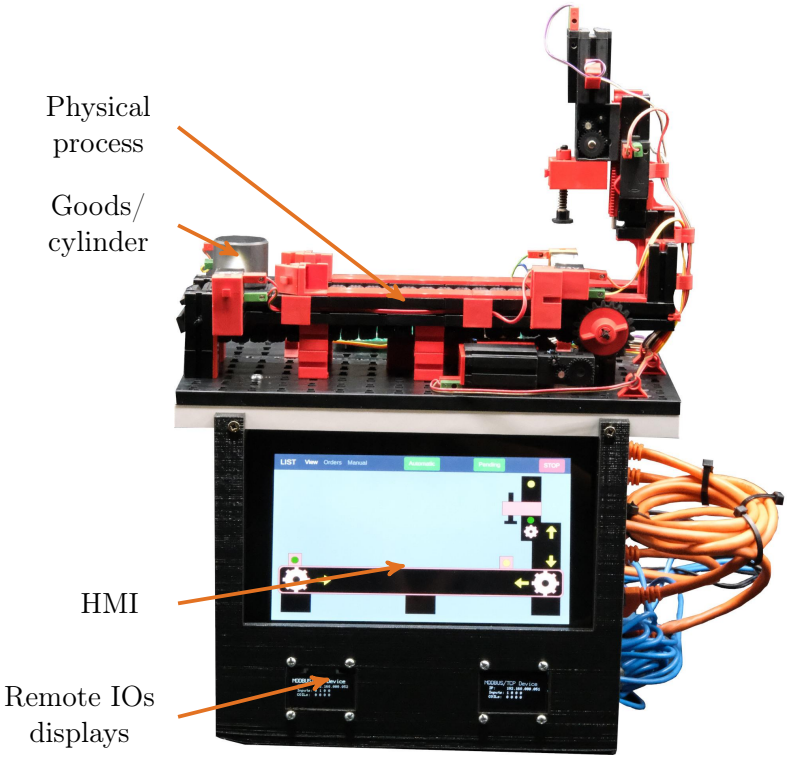


Figure 4.23: Front view on the complete LICSTER testbed. The process on top, represents a punching machine with a conveyor belt.

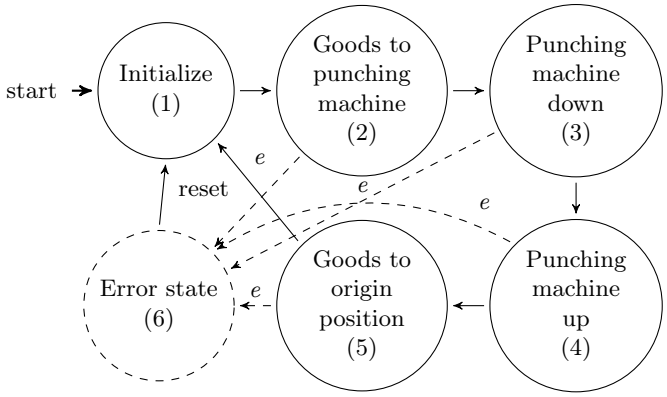
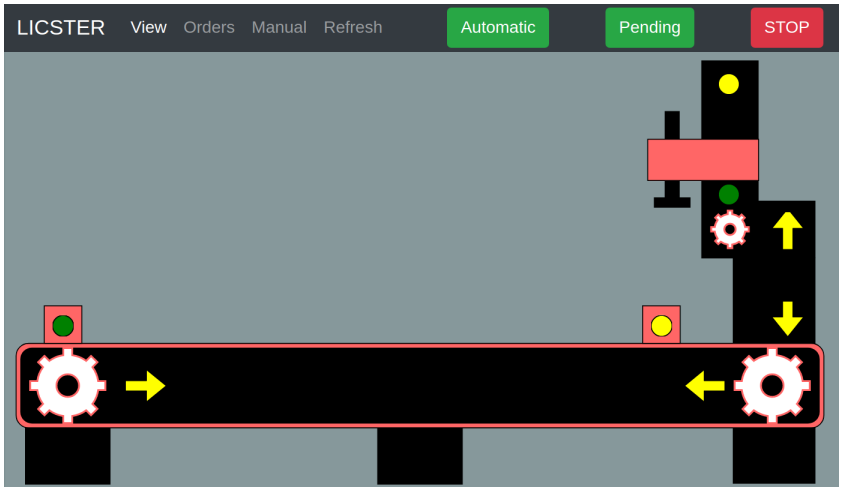


Figure 4.24: Program sequence of the process implemented on the PLC.

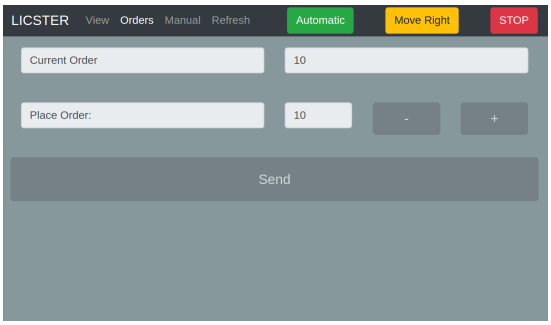
4.4.3.2 Human Machine Interface (HMI)

The HMI is provided by a webserver which runs on a dedicated Raspberry Pi attached to a touchscreen. The web application is split into three areas – view, order, and control – where the user has the possibility to monitor the process, place orders, and thus trigger the described process. The user can also manually control the conveyor belt and the

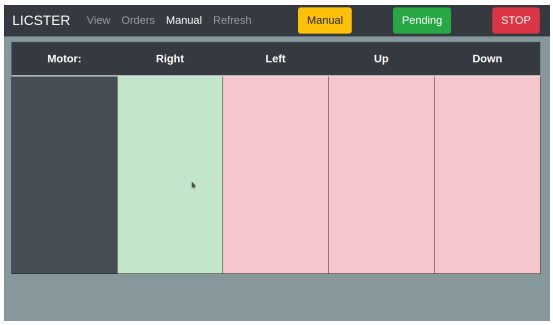
punching machine via the touch panel. This behavior resembles a real HMI. Figure 4.25 shows a screenshot of the HMI, where the process is observed. Through this functionality and usage, authentic attack scenarios can be set up. Basing the HMI on a webserver has several perks: First, it reflects the change of technology introduced by industry 4.0. Companies like Siemens are already pursuing this approach with their *WinCC/Web Navigator* [Sie]. Second, knowledge about web technologies is widespread and conveniently accessible, making this HMI easy to understand, extend, and exploit. Third, with small modifications, such as introducing Wi-Fi to the Raspberry Pi, the HMI can be effortlessly ported to tablets or smartphones, which introduces new attack vectors and, again, represents the shift to contemporary technologies.



Process view mode



Order management mode



Manual control mode

Figure 4.25: Pictures of the different HMI views.

Besides monitoring the process, orders can be placed and the controller can be set in a manual mode, where the process is controlled by the operator. This also brings the human component into play in the testbed.

4.4.3.3 Remote IO

The remote IOs are custom, open-source solutions based on a development board from STMicroelectronics. These remote IOs are based on the edge node devices introduced in Section 4.1, with 24 V IOs. The base board is a STM32F767ZI [STMa] with an Arduino Uno V3 header, which is connected to a self-designed custom add-on board. Using the Arduino Uno V3 header for the custom PCB allows the underlying prototyping board to be changed easily with other compatible boards. This is relevant, for example, if higher performance, an energy-saving solution, or cheaper hardware is needed.

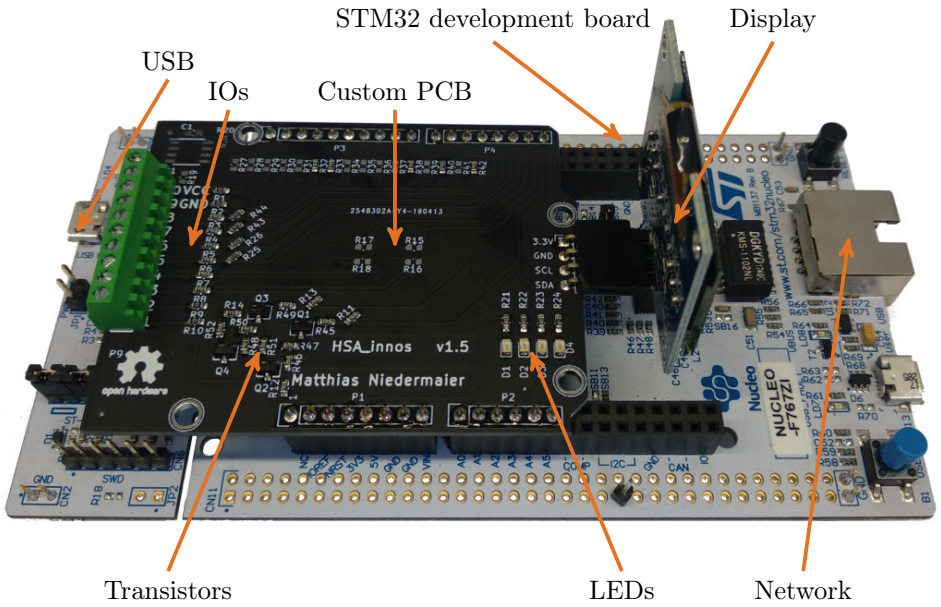


Figure 4.26: PCB of the remote IOs.

The USB connector for programming and power is placed on the left side of the development board. The RJ45 Ethernet connector for networking is mounted on the right side. The custom PCB is necessary to convert the 24 V of the physical process to the 3.3 V of the STM32 development board. Additionally, each remote IO is connected to a display as illustrated in Figure 4.27. There the operator can monitor the current state of the Modbus/TCP input registers and coils.

Applying displays to field level components is a trend that can also be seen by real-world components. In larger plants and systems, it simplifies identifying erroneous devices for the operator. Moreover, these devices often allow a simple on-site basic configuration during commissioning.

4.4.3.4 Physical Process (Fischertechnik)

One of the main requirements is to use a real physical process so that impacts are directly visible. However, in order to keep the necessary skills low, an affordable as

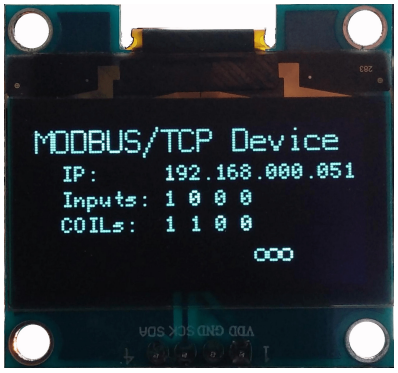


Figure 4.27: Picture showing the display mounted on each remote IO.

well as manageable solution is used in the LICSTER testbed. Another requirement to the process is that it should be automatically repeatable without the need for human interaction. That way the researcher or student has enough time to execute his attacks and to observe the effects. The selected device is a *Fischertechnik* punching machine 96785_sim [fis], as shown in Figure 4.28 and is costing 195 euros.

The *Fischertechnik* system consists of a conveyor belt, two light-barriers, two limit-switches, and two motors. The light-barriers are placed at each end of the conveyor belt and the limit-switches control the upper and lower limits of the punching machine. One electric motor drives the conveyor belt clockwise and counterclockwise, while the other electric motor lifts and lowers the punching machine from and to the conveyor belt.

4.4.3.5 SCADA/Historian

The software used as SCADA and historian system for the testbed is *Scada-LTS* [Rok+16]. It is an open-source solution that supports Modbus/TCP and is entirely web-based. It also represents the shift to contemporary technologies in Industry 4.0. The software runs on a Raspberry Pi. Its webpage can be accessed by any system within the network. *Scada-LTS* also offers the possibility to store values over a longer period and to offer them as process history.

4.4.3.6 Necessary Skills

Although the testbed is designed to keep the entry hurdle for new students as low as possible, basic knowledge about the following three aspects is necessary. ❶ Basic electrical knowledge is required to safely connect a few wires to the system. However, this is very limited, since only four sensors and two motors need to be connected. ❷ When the STM32-based remote IOs should be used, soldering skills and a soldering iron are necessary. This can be avoided by using, for example, Raspberry Pis with 24 V IOs, as recommended by the *OpenPLC* project, but this results in a smaller testbed and also a higher cost. ❸ Basic Linux skills are of benefit to set up the Raspberry Pis and use

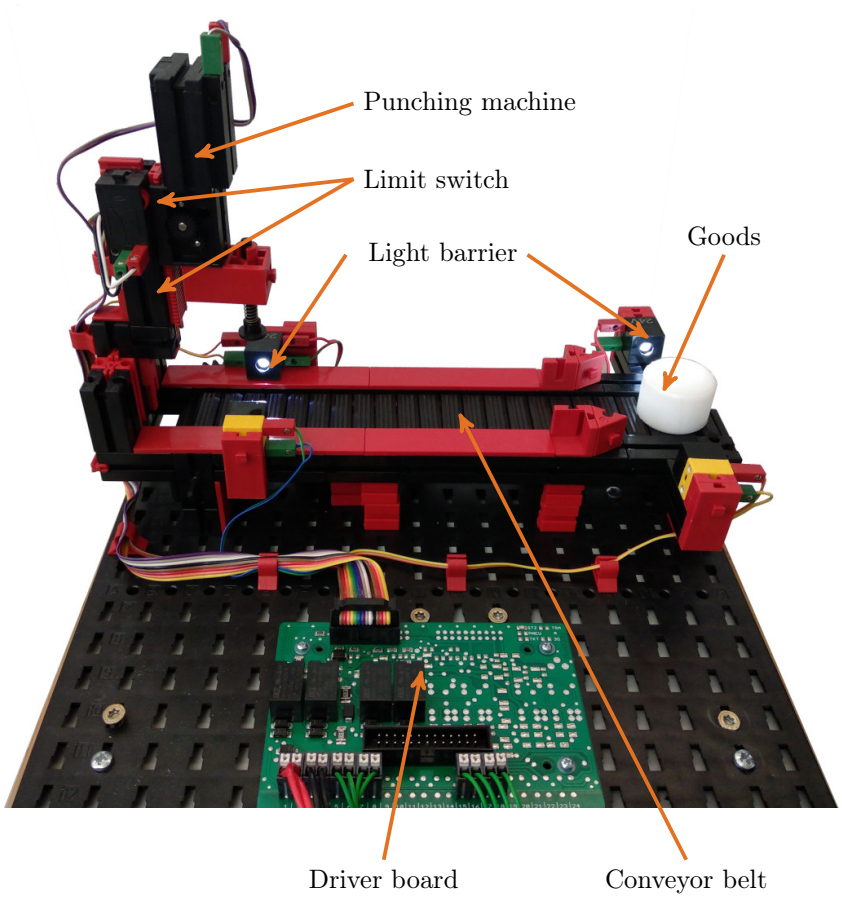


Figure 4.28: Picture showing the Fischertechnik setup.

the attacking tools. However, necessary materials and guides to set up the testbed are provided, which reduce the initial hurdles to a minimum.

4.4.3.7 Connections and Wiring

The network cabling is simple, because only all components need to be connected to the network switch. There are no special requirements for the networking, but it is recommended to start at network port 1, for example to set up a mirror port on port 8 later. The OLED screens are connected directly to the remote IO PCBs, or if extensions are used one to one. The power supplies of the Raspberry Pis, the network switch and the 24 V power supply must be connected as regular and the remote IOs are powered over USB from one of the Raspberry Pis. The most difficult part is the wiring of the *Fischertechnik* process, but every connection is labeled with a number or named, which make the wiring straight forward. Table 4.11 shows the wiring matrix.

To test whether the cabling is correct, it is possible control and check everything individually with the manual control mode of the HMI.

Table 4.11: Electrical wiring within the testbed.

Fischertechnik	Remote IO 1	Remote IO 2	Power
1	VCC	VCC	24 V
2	nc	nc	24 V
3	GND	GND	GND
4	nc	nc	GND
5	nc	8	nc
6	nc	7	nc
7	8	nc	nc
8	7	nc	nc
15	nc	1	nc
16	nc	2	nc
17	1	nc	nc
18	2	nc	nc

nc = not connected

4.4.4 Evaluation and Benchmarking of the Testbed

The evaluation of the LICSTER testbed consists of three tiers. First, it is assured that the overall concept is viable and the communication between the components works as expected. Second, the previously identified attacks are systematically applied to the testbed and evaluated for their feasibility and effects to the system. Finally, it is evaluated which open research questions could be assessed with the help of the proposed testbed.

4.4.4.1 Evaluation of the Implementation

For some research questions and teaching exercise, network traffic is a critical element. For example, intrusion detection can be configured and evaluated based on the network traffic captures of executed attacks. Figure 4.29 shows the density distribution of outgoing packets per second per component during a one-minute capture while the testbed runs its process.

It clearly shows that the number of packets per second differs, depending on the device. The PLC (192.168.0.30) contains most of the logic and therefore is the most communicative component with about 110 packets/s. This is because the PLC polls the remote IOs every 100 ms while simultaneously being polled by the SCADA and HMI. In comparison, the communication of the HMI, which updates the values only once every 500 ms, amounts to significantly less traffic. Each remote IO takes about 55 packets/s to communicate, which is mostly due to the constant polling of the PLC. The similar behavior of the remote IOs leads to a correlative density representation. Lastly, the SCADA system, which has no hard timing requirements, only amounts to about 20 packets/s.

4.4.4.2 Attack Validation within the Testbed

Table 4.12 shows an overview of selected attacks performed on the testbed.

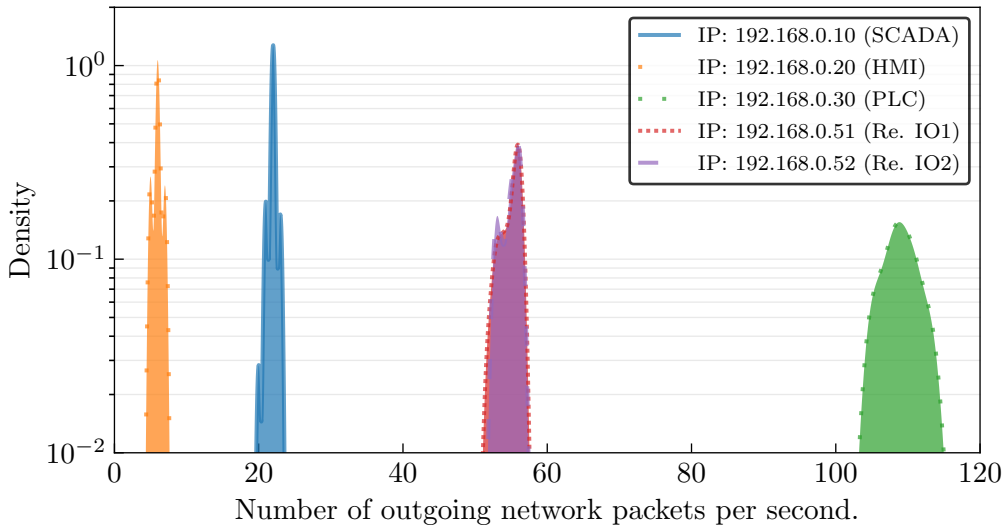


Figure 4.29: Density plot showing the number of packets per second of each device within LICSTER.

In order to correlate the attacks with the **ICS levels** ❶, the first column enumerates the levels zero to two of the industrial automation pyramid. The second column **Attack description** ❷ of the table contains a list of attacks derived from Section 4.4.2.4. These are the scenarios for a potential attacker. The three basic protection goals of IT security are Confidentiality, Integrity and Availability. Therefore, the third column, **CIA** ❸, maps each attack to the protection goals it compromises. The **STRIDE** ❹ threat model by Kohnfelder et al. [Koh+99] in the fourth column uses the indicators **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service and **E**levation of privilege as a more detailed threat mapping. Next, column 5 maps the **attacker model** ❺ introduced in Section 4.4.2.3 to each attack, to clarify which attacks actually need physical access and which do not. Column 6 contains **Tools** ❻ which are used to evaluate and perform attacks on the testbed are. Here viable readymade and open-source solutions are preferred. Additionally, customized scripts are provided to execute attacks where no ready to use tools is available. Here is a brief overview of the tools and the corresponding attacks:

- Network scanning is done with *nmap* [Lyo09]. The specific Modbus/TCP NSE script [Bri10] is used to identify Modbus/TCP devices.

- For network sniffing the Linux tool *tcpdump* [Jac+89] and *wireshark* [Com+08] is used. Wireshark, a piece of software for traffic capture and protocol dissection for various protocols including Modbus/TCP, facilitates easy package analysis.

- For flooding and DoS attacks the tool *hping3* [San99] is used. It supports a multitude of protocols and configurations to perform different forms of DoS attacks.

Table 4.12: Evaluation of a selection of possible devices in the testbed.

ICS Level ❶	Attack description ❷	CIA ❸	STRIDE ❹	Remote attacker ❺	Local attacker ❺	Tools (selection) ❻	Skill level ❼	Impact ❽	Detection difficulty ❾
0 process level	Manipulate	A	TD	✗	✓	—	low	high	easy
	Physically Damage	A	TD	✗	✓	—	low	high	easy
1 field level	DoS Sensor	A	D	✓	✓	hping3	low	high	easy
	Disconnect IO power/network	A	TD	✗	✓	—	low	high	easy
	Manipulate IO physical	A	TD	✗	✓	—	low	high	easy
	MitM spoof values IO-PLC	CIA	STRIDE	✗	✓	script	high	high	medium
1 control level	DoS PLC	A	D	✓	✓	hping3	low	high	easy
	DoS HMI	A	D	✓	✓	hping	low	medium	easy
	Sniffing network	C	I	✗	✓	Tcpdump	low	low	difficult
	MitM spoof values HMI-PLC	CIA	STRIDE	✗	✓	script	high	high	medium
	Physical access HMI	CIA	STRIDE	✗	✓	—	low	low	medium
2 process control level	DoS SCADA	A	D	✓	✓	hping3	high	low	easy
	Sniffing network	C	I	✗	✓	Tcpdump	low	low	difficult
	MitM spoof values SCADA-PLC	CIA	STRIDE	✗	✓	script	high	high	medium
	Attack SCADA	CIA	STRIDE	✓	✓	script	medium	high	medium

- For MitM attacks a custom Python script for easy editing is used, based on the libraries pymodbus and scapy [Bio+11].

- For active manipulation of values in the Modbus/TCP communication, a custom Python script or interactive Python shell with pymodbus is employed.

The seventh column, **Skill level** ❼, refers to the knowledge required by the attacker to achieve his malicious goals. Each attack scenario is rated low, medium, or high. The rating primarily represents the amount of system-specific insight an attacker needs to follow through with his attack. Other aspects such as the basic technical knowledge and the expertise in available tools do not weigh in the rating quite as much, since most tools and relevant documentation are freely and sufficiently available. To measure the **Impact** ❽ in column 8, again a rating of low, medium, or high is used to reflect the consequences to the system and the physical process of each attack. The severity of the rating depends on factors such as whether or not the plant can be damaged as a direct or indirect result of the attack, e.g. when the punching machine does not stop at the limit switch and crashes into the ground. Finally, the **Detection difficulty** ❾ is assessed in the ninth column on three levels low, medium, and high, as mentioned previously. It represents a rough estimation of the likelihood of successful detection of an attack by defensive mechanisms, e.g. an IDS. The skill level, impact, and detection difficulty are rated on three levels. Attacks in real scenarios depend on many circumstances and can vary heavily when compared with each other. That is why, the table can only give a tendency, which should be taken with a grain of salt.

This evaluation has been shown that even low-cost testbeds offer many possibilities of attacks. It is important to see direct effects for new researchers and students, such as the physical process. This shows ICS specifics, as digital devices interact with the real world and attacks on network devices can have an impact on a process.

4.4.4.3 Discussion of Extensions and Research Questions

This subsection elaborates on how LICSTER's functionality can be enhanced by extensions and what research questions can be investigated on the foundation of the here presented testbed. Also, the research ideas introduced by Cardenas et al. [Cár+08a] are picked up in this testbed. This demonstrates how adaptive LICSTER really is and how much room for research and discussion it provides, despite its simplicity. In fact, it is this simplicity that makes this testbed so easy to use and promising for students and beginners.

Extensions LICSTER can be easily extended, for example, by virtual ICS components [Ant+15] or standard office clients in virtual machines. The extensibility of the introduced LICSTER testbed allows for the simulation of potentially huge environments, always with the physical process integrated. Apart from enhancing LICSTER by further components and clients, its communication capabilities can be extended by additional protocols such as, for example, OPC UA. Hence, LICSTER can be used as a platform to evaluate the security aspects of upcoming ICS protocols [Ren+10]. Evaluations like these could lead to the establishment of requirements for secure protocols in ICSs. One topic that was discussed by Givehchi et al. [Giv+14], involves operating parts of an ICS, such as PLCs, in the cloud or fog. It is interesting to examine further security research in addition to the evaluation of availability and control timings. All that is needed is to move the *OpenPLC*, which runs on any Linux-based computer, into the cloud or fog.

Offensive Scenarios Morris et al. [Mor+13] presented 17 attacks against ICSs that use the same Modbus/TCP protocol as LICSTER. Hence, these attacks can be executed on the testbed introduced here and evaluated with the real-world impacts on the physical process. One of the most serious dangers to critical infrastructures is an Advanced Persistent Threat (APT), as explained, among others, by Gouglidis et al. [Gou+18]. LICSTER provides an elementary but sufficient platform to further investigate these types of attacks, since it provides all of the relevant components of an ICS, including the physical process.

Protection Measures A protection mechanism often used for ICSs is network monitoring [Zhu+10]. Since LICSTER spans over multiple ICS layers, it incorporates various types of network communication, as can be seen in Section 4.4.4.1. For example, the communication between the PLC and the remote IOs shows clear timing criticality, while the traffic between the SCADA system and the PLC does not. With these distinctive communication characteristics, LICSTER can be used to evaluate IDS implementations and to test their detection mechanisms.

Moreover, the remote IO can be programmed with custom firmware, which is mostly not possible when proprietary hardware is used. With this, intelligent IIoT edge nodes, e.g. for intrusion detection, can be placed into the testbed.
This small selection of topics shows that it is often not the number of components and

size that counts, but mapping an entire industrial process within a testbed is more important. This allows a simple demonstration of impacts on physical processes caused by cyber attacks. Particularly for learning purposes, it is important to have simple tools to comprehend complex topics.

4.4.5 Conclusion

In this section, LICSTER, an open-source, low-cost ICS testbed for education and research, was presented and made public available [HSA19]. It is shown that the concept can be set up for about 500 euros. This way, the entry barrier is lowered so that more people can get hands-on experience with ICS security. To enhance the learning experience of how the physical world interacts with the digital environment, suitable attacker models and possible attacks are introduced. These measures are intended to provide students and researchers an easy access to the topic of ICS security.

Furthermore, even with a low-complexity testbed as LICSTER, current and relevant research questions can be assessed, due to the open-source nature of the project and its components. With LICSTER, offensive as well as defensive techniques can be tested and evaluated on different ICS levels. The physical process is a key segment of LICSTER, which allows for a haptic understanding of the effects of cyber-attacks on ICSs.

Chapter 5

Conclusion and Future Work

Contents of this chapter

In this chapter, a conclusion of this work is given. This is followed by an outlook on open and further research challenges in the area of ICS security. Since 2008, when Cárdenas et al. [Cár+08a] presented the current status and open research questions in the field of ICS security, a lot of research has been done. Nevertheless, more than a decade later, much remains unanswered. A part of these research questions were picked up in this work and current problems of ICS with regard to the communication robustness of ICS components like PLCs have been analyzed. Furthermore, secure ICS architectures as well as other building blocks were introduced. All the here shown solutions have been implemented in a PoC and evaluated. The presented results enhance the state of research in the area of industrial IT security.

5.1 Conclusion

In this work, I analyzed problems and vulnerabilities of commercial PLCs and then introduced building blocks to enhance the ICS security in general and PLCs in particular.

First, I introduced the proprietary testbed, that was used for most analyzes in this work. This testbed is equipped with PLCs from different vendors and a custom designed logic analyzer to observe the electrical outputs of the PLCs. With this, I analyzed the automated fuzzing of proprietary TCP/IP protocols for programming PLCs. With this framework, these protocols can be analyzed without high reverse engineering effort, which was demonstrated by detecting vulnerabilities in Phoenix Contact PLCs. Furthermore, possible impacts of network communication load to PLCs were demonstrated. This is a new way of looking at things, as most research does not cover the interference between the network and electrical side of industrial controllers. Here also vulnerabilities of ICS products were discovered, and a warning for careless network scanning in ICS was given. This shows that network traffic in critical control systems can lead to unintended states and that there is still potential for further research. Subsequently, a dual MCU

architecture was introduced, which solves the above-mentioned problems regarding an influenceable cycle time and allows a robust control of physical processes. For existing ICSs, a secure passive network scanning method was presented to enable asset identification without influences. This captures ARP broadcasts and uses MAC addresses to identify the corresponding device. Using this it is possible to get a first impression of the network participants without active network scanning.

In the second part, open components to enhance security in ICSs and for education and research are introduced. At the beginning, self-developed open components and an open-source testbed were presented. These were used for the research and for PoC evaluation of the introduced security building blocks. First, a method for a distributed IDS on low performance MCUs was presented. This allows each ICS device to detect cyber-attacks on its own by monitoring network timing behavior and packet meta data. Additionally, I presented a network scanner for low-performance MCU evaluated on the previous introduced open testbed. As a result, attackers in industrial networks can be found without the requirement of special scanning hardware in each subnet. At the end, a Low-cost ICS Security Testbed for Education and Research (LICSTER) is presented to enable an easy start into ICS security. This is intended, for example, to enable research in the field of industrial IT security at low costs, as the hurdle in knowledge and cost is usually high.

On the whole, in this work I identified problems in the area of ICS security, which has led to numerous vulnerability reports as well as a new awareness of attacks and active scanning in industrial networks. Furthermore, I demonstrated appropriate solutions to increase the security level of ICS devices and networks, contributing significantly for secure Industry 4.0 architectures.

5.2 Future Work

Future industrial plants are being exposed to more and more threats due to increasing demands in connectivity caused by Industry 4.0 scenarios. To withstand this, new architectures, tools, and concepts will be required. For this reason, the International Electrotechnical Commission (IEC), introduced the IEC 62443 [IEC08] series that deals with security in industrial plants. For these new ICSs, security concepts based on the IEC 62443 have to be designed and set up from the beginning and devices have to be developed according to these principles. In order to be able to implement these measures, specific implementations and evaluations for these are needed. In this way, the flooding tests and evaluations that were made using the proprietary testbed (Section 3.3) can be used for standardized tests. This enables operators to compare which PLC meet the requirements for a specific task and also can be used for manufacturers within a continuous integration test. This is partly a task of further research to provide and evaluate independent generic solutions. In the end, it would be optimal if each device is secure by itself (Secure by Design) and does not need any external protection mechanisms like firewalls.

However, old plants should not be forgotten, since industrial systems have a lifetime of several decades. A completely different way of looking at things needs to be chosen, in which every component is considered insecure. It will take years, if not decades, for these legacy devices to go out of operation. A first step would be a complete asset identification with associated vulnerability mapping. This would allow operators of ICSs to get an overview of possible hazards. A cornerstone for asset management was laid in Section 3.5, where a completely passive scanning mechanism was demonstrated. This scanner can serve as the basis for an asset management tool in fragile networks. Another step is the detailed monitoring of networks, for example, with an IDS. This is difficult, since the normal state has to be defined in order to detect anomalies. Furthermore, this often goes along with deep packet inspection and thus with great effort in implementation and configuration. For this purpose, the IDS presented in Section 4.2 must be expanded so that the real world process is also mapped virtually in the IDS. These examples show that the research in the field of ICS security is far from complete, and that there is a necessity to continue research in this area.

Bibliography

[4SI15] 4SICS. *Capture files from 4SICS Geek Lounge*. 2015. URL: https://www.netresec.com/?page=PCAP4SICS (visited on August 6, 2019).

[Abh14] Kumar Abhishek. *Beaglelogic – Beaglebone Logic Analyzer*. 2014. URL: https://github.com/abhishek-kakkar/BeagleLogic/wiki (visited on August 6, 2019).

[Alv+14] Thiago Rodrigues Alves, Mario Buratto, Flavio Mauricio de Souza, and Thelma Virginia Rodrigues. "OpenPLC: An Open Source Alternative to Automation". In: *Global Humanitarian Technology Conference (GHTC), 2014 IEEE*. IEEE. 2014, pp. 585–589. DOI: 10.1109/GHTC.2014.6970342.

[Alv+18] Thiago Rodrigues Alves, Rishabh Das, and Thomas Morris. "Embedding Encryption and Machine Learning Intrusion Prevention Systems on Programmable Logic Controllers". In: *IEEE Embedded Systems Letters* 10.3 (2018), pp. 99–102. ISSN: 1943-0663. DOI: 10.1109/LES.2018.2823906.

[Ami+07] Pedram Amini, Aaron Portnoy, and Ryan Sears. *Sulley – A Pure-python Fully Automated and Unattended Fuzzing Framework*. 2007. URL: https://github.com/OpenRCE/sulley (visited on August 6, 2019).

[Ami+09] Saurabh Amin, Alvaro A Cárdenas, and S Shankar Sastry. "Safe and Secure Networked Control Systems under Denial-of-Service Attacks". In: *International Workshop on Hybrid Systems: Computation and Control*. Springer. 2009, pp. 31–45. DOI: 10.1007/978-3-642-00602-9_3.

[Ana17] Analog Devices Inc. *Intelligence at the Edge Part 1: The Edge Node*. 2017. URL: https://www.analog.com/en/technical-articles/intelligence-at-the-edge-part-1-the-edge-node.html (visited on December 22, 2019).

[Ant+15] Daniele Antonioli and Nils Ole Tippenhauer. "MiniCPS: A Toolkit for Security Research on CPS Networks". In: *Proceedings of the First ACM workshop on cyber-physical systems-security and/or privacy*. ACM. 2015, pp. 91–100. DOI: 10.1145/2808705.2808715.

[Ant+16] Rob Antrobus, Sylvain Frey, Benjamin Green, and Awais Rashid. "Simaticscan: Towards a Specialised Vulnerability Scanner for Industrial Control Systems". In: *Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research 2016*. ICS-CSR '16. Belfast, United Kingdom: BCS Learning & Development Ltd., 2016, pp. 1–8. ISBN: 9781780173573. DOI: 10.14236/ewic/ICS2016.2. URL: https://doi.org/10.14236/ewic/ICS2016.2.

[Auf10] Patrice Auffret. "SinFP, Unification of Active and Passive Operating System Fingerprinting". In: *Journal in Computer Virology* 6.3 (2010), pp. 197–205. ISSN: 1772-9904. DOI: 10.1007/s11416-008-0107-z.

[Bar+08] Richard Barry et al. *FreeRTOS*. 2008. URL: https://www.freertos.org (visited on December 22, 2019).

[Bar14] Rafael Ramos Regis Barbosa. "Anomaly Detection in SCADA Systems: A Network Based Approach". English. PhD thesis. University of Twente, April 2014. ISBN: 978-90-365-3645-5. DOI: 10.3990/1.9789036536455.

[Bas+16] Justin Bastress et al. *Go Application Layer Scanner*. 2016. URL: https://github.com/zmap/zgrab2 (visited on December 22, 2019).

[Ber11] Dillon Beresford. *Exploiting Siemens Simatic S7 PLCs*. 2011. URL: https://media.blackhat.com/bh-us-11/Beresford/BH_US11_Beresford_S7_PLCs_WP.pdf (visited on August 7, 2019).

[Bil+13] Olivier Bilodeau, David LaPorte, and Eric Kollman. *FingerBank*. 2013. URL: https://fingerbank.org/ (visited on August 6, 2019).

[Bio+11] Philippe Biondi et al. *Scapy Packet Manipulation*. 2011. URL: https://scapy.net/ (visited on August 7, 2019).

[Bod+14] Roland Bodenheim, Jonathan Butts, Stephen Dunlap, and Barry Mullins. "Evaluation of the Ability of the Shodan Search Engine to Identify Internet-facing Industrial Control Devices". In: *International Journal of Critical Infrastructure Protection* 7.2 (2014), pp. 114–123. DOI: 10.1016/j.ijcip.2014.03.001.

[Bov17] John Boville. *Productivity Secrets: Don't Underestimate the Power of PLC Scan Time*. 2017. URL: https://blog.se.com/machine-and-process-management/2017/02/23/productivity-secrets-dont-underestimate-power-plc-scan-time/ (visited on August 6, 2019).

[Bow+05] Calvert L. Bowen, Timothy K. Buennemeyer, and Ryan W. Thomas. "A Plan for SCADA Security to Deter DOS Attacks". In: *Proceedings of the Department of Homeland Security: R&D Partnering Conference*. Citeseer. 2005. DOI: 10.1109/TELSKS.2013.6704448.

[Bri08] Mark Bristow. *ModScan: A SCADA Modbus Network Scanner*. 2008. URL: https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-bristow.pdf (visited on August 7, 2019).

[Bri10] Mark Bristow. *File modbus-discover*. 2010. URL: https://nmap.org/nsedoc/scripts/modbus-discover.html (visited on December 22, 2019).

[Bro+13] Christopher Bronk and Eneken Tikk-Ringas. *The Cyber Attack on Saudi Aramco*. 2013. DOI: 10.1080/00396338.2013.784468.

[But+14] Ismail Butun, Salvatore D Morgera, and Ravi Sankar. "A Survey of Intrusion Detection Systems in Wireless Sensor Networks". In: *IEEE Communications Surveys & Tutorials* 16.1 (2014), pp. 266–282. DOI: 10.1109/SURV.2013.050113.00191.

[Cár+08a] Alvaro A Cárdenas, Saurabh Amin, and Shankar Sastry. *Research Challenges for the Security of Control Systems*. 2008. URL: http://dl.acm.org/citation.cfm?id=1496671.1496677 (visited on October 16, 2019).

[Cár+08b] Alvaro A Cárdenas, Saurabh Amin, and Shankar Sastry. "Secure Control: Towards Survivable Cyber-Physical Systems". In: *Distributed Computing Systems Workshops, 2008. ICDCS'08. 28th International Conference on*. IEEE. 2008, pp. 495–500. DOI: 10.1109/ICDCS.Workshops.2008.40.

[Cár+09] Alvaro A Cárdenas, Saurabh Amin, Bruno Sinopoli, Annarita Giani, Adrian Perrig, and Shankar Sastry. "Challenges for Securing Cyber Physical Systems". In: *Workshop on Future Directions in Cyber-physical Systems Security*. Vol. 5. 1. 2009.

[Cas+13] Marco Caselli, Dina Hadžiosmanović, Emmanuele Zambon, and Frank Kargl. "On the Feasibility of Device Fingerprinting in Industrial Control Systems". In: *Critical Information Infrastructures Security: 8th International Workshop, CRITIS 2013, Amsterdam, The Netherlands, September 16-18, 2013, Revised Selected Papers*. Ed. by Eric Luiijf and Pieter Hartel. Springer International Publishing, September 2013, pp. 155–166. ISBN: 978-3-319-03964-0. DOI: 10.1007/978-3-319-03964-0_14.

[Cen] Censys.io. *Frequently Asked Questions (FAQ)*. URL: https://support.censys.io/en/articles/1294848-frequently-asked-questions-faq (visited on December 12, 2019).

[CER18a] CERT@VDE. *PHOENIX CONTACT ILC 1x1 ETH Denial of Service*. 2018. URL: https://cert.vde.com/de-de/advisories/vde-2018-012 (visited on December 22, 2019).

[CER18b] CERT@VDE. *WAGO 750-8xx Controller Denial of Service*. 2018. URL: https://cert.vde.com/de-de/advisories/vde-2018-013 (visited on December 22, 2019).

[Cof+18] Kyle Coffey, Richard Smith, Leandros Maglaras, and Helge Janicke. "Vulnerability Analysis of Network Scanning on SCADA Systems". In: *Security and Communication Networks* (March 2018). DOI: 10.1155/2018/3794603. URL: https://doi.org/10.1155/2018/3794603.

[Col+14] Armando W Colombo, Thomas Bangemann, Statmatis Karnouskos, Jerker Delsing, Petr Stluka, Robert Harrison, Francois Jammes, Jose L Lastra, et al. "Industrial Cloud-based Cyber-physical Systems". In: *The IMC-AESOP Approach* 22 (2014). DOI: 10.1007/978-3-319-05624-1.

[Col11] Galen Collins. *A full modbus protocol written in python*. 2011. URL: https://github.com/riptideio/pymodbus (visited on December 22, 2019).

[Com+08] Gerald Combs et al. *Wireshark-network Protocol Analyzer*. 2008. URL: https://www.wireshark.org/ (visited on August 7, 2019).

[COR] CORE Security. *Pcapy*. URL: https://github.com/CoreSecurity/pcapy (visited on August 6, 2019).

[Cyb11] Global Energy Cyberattacks. *Night Dragon*. 2011. URL: https://www.mcafee.com/wp-content/uploads/2011/02/McAfee_NightDragon_wp_draft_to_customersv1-1.pdf (visited on December 15, 2019).

[Day+95] John D. Day and Hubert Zimmermann. "The OSI Reference Model". In: *Conformance Testing Methodologies and Architectures for OSI Protocols*. Washington, DC, USA: IEEE Computer Society Press, 1995, pp. 38–44. ISBN: 0818653523.

[Dit15] Ditecting. *Detecting everything among the industrial control cyberspace, probing malicious vulnerability then mending the "heaven"*. 2015. URL: http://www.ditecting.com (visited on January 3, 2020).

[Dro97] Ralph Droms. "RFC 2131-Dynamic Host Configuration Protocol, March 1997". In: *Obsoletes RFC1541. Status: DRAFT STANDARD* 3.1 (1997). URL: https://tools.ietf.org/html/rfc2131 (visited on August 7, 2019).

[Dud17] Marcin Dudek. *TRISIS/TRITON/HatMan Malware Repository*. 2017. URL: https://github.com/MDudek-ICS/TRISIS-TRITON-HATMAN (visited on December 15, 2019).

[Dun01] Adam Dunkels. *Design and Implementation of the lwIP TCP/IP Stack*. 2001. URL: https://www.academia.edu/32820259/Design_and_Implementation_of_the_lwIP_TCP_IP_Stack (visited on August 7, 2019).

[Dur+13] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. "ZMap: Fast Internet-wide Scanning and Its Security Applications". In: *USENIX Security Symposium*. Vol. 8. 2013, pp. 47–53. URL: https://zmap.io/paper.pdf (visited on August 7, 2019).

[Dur+15] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J Alex Halderman. "A Search Engine Backed by Internet-wide Scanning". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2015, pp. 542–553. DOI: 10.1145/2810103.2813703.

[Edd09] Michael Eddington. *Peach Fuzzing Framework*. 2009. URL: http://www.peachfuzzer.com (visited on August 6, 2019).

[Efa12] Dmitry Efanov. *PLCScan the Internet*. 2012. URL: http://scadastrangelove.blogspot.de/2012/11/plcscan.html (visited on August 6, 2019).

[Ele98] Electro Cam Corporation. *Scan Times*. 1998. URL: https://static1.squarespace.com/static/58e274951b631bf0ffe7f6d8/t/592e5dc46b8f5bd5ce006522/1496210884591/SCANTIME.PDF (visited on August 6, 2019).

[Eur11] European Network and Information Security Agency (ENISA). *Protecting Industrial Control Systems*. 2011. URL: https://www.enisa.europa.eu/publications/annex-v/at_download/fullReport (visited on August 18, 2019).

[Fed13] Federal Office for Information Security (German: Bundesamt für Sicherheit in der Informationstechnik, abbreviated as BSI). *ICS Security Kompendium*. 2013. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/ICS/ICS-Security_kompendium_pdf.pdf?__blob=publicationFile (visited on August 19, 2019).

[fis] fischertechnik GmbH. *Punching Machine with Conveyor Belt 24V - Simulation*. URL: https://www.fischertechnik.de/en/products/simulating/training-models/96785-sim-punching-machine-with-conveyor-belt-24v-simulation (visited on December 22, 2019).

[Fol+18] Simon N Foley et al. "Science Hackathons for Cyberphysical System Security Research: Putting CPS testbed platforms to good use". In: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. ACM. 2018, pp. 102–107. DOI: 10.1145/3264888.3264897.

[For+18] David Formby, Milad Rad, and Raheem Beyah. "Lowering the Barriers to Industrial Control System Security with GRFICS". In: *2018 USENIX Workshop on Advances in Security Education (ASE 18)*. 2018. URL: https://www.usenix.org/node/219741 (visited on August 7, 2019).

[Fru+05] Terry L Fruehling and Troy L Helm. *Secured Microcontroller Architecture*. US Patent 6,981,176. December 2005.

[Gar+15] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. "Edge-centric Computing: Vision and Challenges". In: *ACM SIGCOMM Computer Communication Review* 45.5 (2015), pp. 37–42. DOI: 10.1109/JIOT.2016.2579198.

[GE 17] GE Digital. *Achilles Test Platform*. 2017. URL: https://www.ge.com/digital/sites/default/files/download_assets/achilles-test-platform-from-ge-digital-datasheet.pdf (visited on December 22, 2019).

[Giv+14] Omid Givehchi, Jahanzaib Imtiaz, Henning Trsek, and Juergen Jasperneite. "Control-as-a-Service from the Cloud: A Case Study for using Virtualized PLCs". In: *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*. IEEE. 2014, pp. 1–4. DOI: 10.1109/WFCS.2014.6837587.

[Gol+13] Niv Goldenberg and Avishai Wool. "Accurate Modeling of Modbus/TCP for Intrusion Detection in SCADA Systems". In: *International Journal of Critical Infrastructure Protection* 6.2 (2013), pp. 63–75. DOI: 10.1016/j.ijcip.2013.05.001.

[Gou+18] Antonios Gouglidis, Sandra König, Benjamin Green, Karl Rossegger, and David Hutchison. "Protecting Water Utility Networks from Advanced Persistent Threats: A Case Study". In: *Game Theory for Security and Risk Management: From Theory to Practice*. Ed. by Stefan Rass and Stefan Schauer. Cham: Springer International Publishing, 2018, pp. 313–333. ISBN: 978-3-319-75268-6. DOI: 10.1007/978-3-319-75268-6_13. URL: https://doi.org/10.1007/978-3-319-75268-6_13.

[Gre+17] Benjamin Green, Anhtuan Lee, Rob Antrobus, Utz Roedig, David Hutchison, and Awais Rashid. "Pains, Gains and PLCs: Ten Lessons from Building an Industrial Control Systems Testbed for Security Research". In: *10th USENIX Workshop on Cyber Security Experimentation and Test CSET 17)*. 2017. URL: https://www.usenix.org/node/205860 (visited on August 7, 2019).

[Gre19] Andy Greenberg. *The Untold Story of NotPetya, the most Devastating Cyberattack in History*. 2019. URL: https://www.wired.com/story/notpetya-cyberattackukraine-russia-code-crashed-the-world (visited on December 15, 2019).

[Had+07] Sami Haddadin, Alin Albu-Schäffer, and Gerd Hirzinger. "Safety Evaluation of Physical Human-Robot Interaction via Crash-Testing". In: *Robotics: Science and Systems*. Vol. 3. 2007, pp. 217–224. DOI: 10.15607/RSS.2007.III.028.

[Hah+11] Adam Hahn and Manimaran Govindarasu. "An Evaluation of Cybersecurity Assessment Tools on a SCADA Environment". In: *Power and Energy Society General Meeting, 2011 IEEE*. IEEE. 2011, pp. 1–6. DOI: 10.1109/PES.2011.6039845.

[Hal+19] Piroska Haller, Béla Genge, and Adrian-Vasile Duka. "Engineering Edge Security in Industrial Control Systems". In: *Critical Infrastructure Security and Resilience: Theories, Methods, Tools and Technologies*. Ed. by Dimitris Gritzalis, Marianthi Theocharidou, and George Stergiopoulos. Cham: Springer International Publishing, 2019, pp. 185–200. ISBN: 978-3-030-00024-0. DOI: 10.1007/978-3-030-00024-0_10. URL: https://doi.org/10.1007/978-3-030-00024-0_10.

[Hel16] Aki Helin. *Radamsa Fuzzing Test Case Generator*. 2016. URL: https://github.com/aoh/radamsa (visited on August 6, 2019).

[Hem+18] Kevin E. Hemsley and Ronald E. Fisher. *History of Industrial Control System Cyber Incidents*. December 2018. DOI: 10.2172/1505628.

[Hje08] Erik Hjelmvik. "Passive Network Security Analysis with NetworkMiner". In: *(IN) Secure* 18 (2008), pp. 1–100. URL: https://www.forensicfocus.com/passive-network-security-analysis-networkminer (visited on August 7, 2019).

[Ho+12] Cheng-Yuan Ho, Yuan-Cheng Lai, I-Wei Chen, Fu-Yu Wang, and Wei-Hsuan Tai. "Statistical Analysis of False Positives and False Negatives from Real Traffic with Intrusion Detection/Prevention Systems". In: *IEEE Communications Magazine* 50.3 (2012), pp. 146–154. DOI: 10.1109/MCOM.2012.6163595.

[Hol+15] Hannes Holm, Martin Karresand, Arne Vidström, and Erik Westring. "A Survey of Industrial Control System Testbeds". In: *Secure IT Systems*. Springer, 2015, pp. 11–26. ISBN: 978-3-319-26502-5. DOI: 10.1007/978-3-319-26502-5_2.

[HSA18] HSA_innos. *macDetec - Device Identification by MAC Address*. 2018. URL: https://github.com/hsainnos/macDetec (visited on December 22, 2019).

[HSA19] HSA_innos. *LICSTER - A Low-cost ICS Security Testbed for Education and Research*. 2019. URL: https://github.com/hsainnos/LICSTER (visited on December 22, 2019).

[Hui+08] Peter Huitsing, Rodrigo Chandia, Mauricio Papa, and Sujeet Shenoi. "Attack Taxonomies for the Modbus Protocols". In: *International Journal of Critical Infrastructure Protection* 1 (2008), pp. 37–44. DOI: 10.1016/j.ijcip.2008.08.003.

[ICS16] ICS-CERT. *ICS Advisory (ICSA-16-313-01) - Phoenix Contact ILC PLC Authentication Vulnerabilities*. 2016. URL: https://www.us-cert.gov/ics/advisories/ICSA-313-01 (visited on December 22, 2019).

[ICS17] ICS-CERT. *ICS Advisory (ICSA-17-264-04) - iniNet Solutions GmbH SCADA Webserver*. 2017. URL: https://www.us-cert.gov/ics/advisories/ICSA-17-264-04 (visited on December 22, 2019).

[ICS19] ICS-CERT. *ICS Advisory (ICSA-19-106-03) - PLC Cycle Time Influences (Update A)*. 2019. URL: https://www.us-cert.gov/ics/advisories/ICSA-19-106-03 (visited on December 22, 2019).

[IEC08] DIN IEC. *IEC 62443 Industrial Communication Networks – Network and System Security*. 2008. URL: https://www.beuth.de (visited on January 7, 2020).

[IEC93] DIN IEC. *IEC 61131-3 – Programmable Controllers - Part 3: Programming Languages*. 1993. URL: https://www.beuth.de (visited on January 7, 2020).

[Ina+11] Rafia Inam, Jukka Mäki-Turja, Mikael Sjödin, and Moris Behnam. "Hard Real-time Support for Hierarchical Scheduling in FreeRTOS". In: *23rd Euromicro Conference on Real-Time Systems*. 2011, pp. 51–60. URL: http://www.es.mdh.se/publications/2166- (visited on August 7, 2019).

[Ind12] Industrial Control Systems Cyber Emergency Response Team. *NCCIC/ICS-CERT Year in Review FY 2012*. 2012. URL: https://www.us-cert.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2012_Final.pdf (visited on August 14, 2019).

[Ind15] Industrial Control Systems Cyber Emergency Response Team. *NCCIC/ICS-CERT Year in Review FY 2015*. 2015. URL: https://www.us-cert.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2015_Final_S508C.pdf (visited on August 14, 2019).

[Ind16] Industrial Control Systems Cyber Emergency Response Team. *NCCIC/ICS-CERT Year in Review FY 2016*. 2016. URL: https://www.us-cert.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2016_Final_S508C.pdf (visited on August 14, 2019).

[Ins] Institute of Electrical and Electronics Engineers. *IEEE Registration Authority*. URL: https://regauth.standards.ieee.org/standards-ra-web/pub/view.html (visited on August 6, 2019).

[Ins14] Institute of Electrical and Electronics Engineers. "IEEE Standard for Local and Metropolitan Area Networks". In: *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)* (2014), pp. 1–74. DOI: 10.1109/IEEESTD.2014.6847097.

[Int02] International Electrotechnical Commission and others. "IEC 62278:2002. Railway Applications - Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)". In: *IEC, Genf* (2002). URL: https://www.vde-verlag.de/iec-normen/210148/iec-62278-2002.html (visited on August 18, 2019).

[Int03] International Electrotechnical Commission and others. "IEC 62264-1 Enterprise-control system integration–Part 1: Models and terminology". In: *IEC, Genf* (2003). URL: https://www.iso.org/standard/57308.html (visited on August 7, 2019).

[ISA19] ISA Security Compliance Institute. *IEC 62443 - CSA Certification*. 2019. URL: https://www.isasecure.org/en-US/Certification/IEC-62443-CSA-Certification (visited on December 22, 2019).

[Jac+] Van Jacobson, Craig Leres, and Steven McCanne. *LibPcap*. URL: http://www.tcpdump.org (visited on August 6, 2019).

[Jac+89] Van Jacobson, Craig Leres, and Steven McCanne. "The tcpdump Manual Page". In: *Lawrence Berkeley Laboratory, Berkeley, CA* 143 (1989).

[Jar+16] William Jardine, Sylvain Frey, Benjamin Green, and Awais Rashid. "SENAMI: Selective Non-Invasive Active Monitoring for ICS Intrusion Detection". In: *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*. CPS-SPC '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 23–34. ISBN: 9781450345682. DOI: 10.1145/2994487.2994496. URL: https://doi.org/10.1145/2994487.2994496.

[Jin+18] C. Jin, S. Valizadeh, and M. van Dijk. "Snapshotter: Lightweight Intrusion Detection and Prevention System for Industrial Control Systems". In: *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. May 2018, pp. 824–829. DOI: 10.1109/ICPHYS.2018.8390813.

[Kal+16] Rajesh Kalluri, Lagineni Mahendra, RK Senthil Kumar, and GL Ganga Prasad. "Simulation and Impact Analysis of Denial-of-Service Attacks on Power SCADA". In: *Power Systems Conference (NPSC), 2016 National*. IEEE. 2016, pp. 1–5. DOI: 10.1109/NPSC.2016.7858908.

[Kar11] Stamatis Karnouskos. "Stuxnet Worm Impact on Industrial Cyber-Physical System Security". In: *IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2011, pp. 4490–4494. DOI: 10.1109/IECON.2011.6120048.

[Kel+09] Ingmar Kellner and Ludger Fiege. "Viewpoints in Complex Event Processing: Industrial Experience Report". In: *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*. DEBS '09. Nashville, Tennessee: ACM, 2009, 9:1–9:8. ISBN: 978-1-60558-665-6. DOI: 10.1145/1619258.1619271.

[Kim+16] SungJin Kim, WooYeon Jo, and Taeshik Shon. "A Novel Vulnerability Analysis Approach to Generate Fuzzing Test Case in Industrial Control Systems". In: *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*. May 2016, pp. 566–570. DOI: 10.1109/ITNEC.2016.7560424.

[Kli+15] Johannes Klick, Stephan Lau, Daniel Marzin, Jan-Ole Malchow, and Volker Roth. "Internet-facing PLCs as a Network Backdoor". In: *Communications and Network Security (CNS), 2015 IEEE Conference on*. IEEE. 2015, pp. 524–532. DOI: 10.1109/CNS.2015.7346865.

[Kno16] Knownsec Inc. *ZoomEye – Cyberspace Search Engine*. 2016. URL: https://www.zoomeye.org (visited on January 3, 2020).

[Koh+99] Loren Kohnfelder and Praerit Garg. *The Threats to our Products*. 1999. URL: https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx (visited on August 7, 2019).

[Lan11] Ralph Langner. "Stuxnet: Dissecting a Cyberwarfare Weapon". In: *IEEE Security & Privacy* 9.3 (2011), pp. 49–51. DOI: 10.1109/MSP.2011.67.

[Lee+14] Robert M Lee, Michael J Assante, and Tim Conway. *German Steel Mill Cyber Attack*. 2014. URL: https://ics.sans.org/media/ICS-CPPE-case-Study-2-German-Steelworks_Facility.pdf (visited on August 14, 2019).

[Lee+16] Robert M Lee, Michael J Assante, and Tim Conway. *Analysis of the Cyber Attack on the Ukrainian Power Grid*. 2016. URL: https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf (visited on August 14, 2019).

[Lei08] T. Leimbach. "The SAP Story: Evolution of SAP within the German Software Industry". In: *IEEE Annals of the History of Computing* 30.4 (October 2008), pp. 60–76. ISSN: 1934-1547. DOI: 10.1109/MAHC.2008.75.

[Lin+17] Chih-Yuan Lin, Simin Nadjm-Tehrani, and Mikael Asplund. "Timing-based Anomaly Detection in SCADA Networks". In: *Critical Information Infrastructures Security*. 2017, pp. 48–59. DOI: 10.1007/978-3-319-99843-5_5.

[Liu+18] Pengfei Liu and Ting Liu. "Physical Intrusion Detection for Industrial Control System". In: *2018 IEEE Conference on Communications and Network Security, CNS 2018, Beijing, China, May 30 - June 1, 2018*. 2018, pp. 1–2. DOI: 10.1109/CNS.2018.8433194.

[Lon+05] Men Long, Chwan-Hwa Wu, and John Y Hung. "Denial of Service Attacks on Network-based Control Systems: Impact and Mitigation". In: *IEEE Transactions on Industrial Informatics* 1.2 (2005), pp. 85–96. DOI: 10.1109/TII.2005.844422.

[Lyo09] Gordon Lyon. *Nmap–Free Security Scanner For Network Exploration & Security Audits*. 2009. URL: https://nmap.org/ (visited on August 7, 2019).

[Mad00] Angelika Mader. "A Classification of PLC Models and Applications". In: *Discrete Event Systems: Analysis and Control*. Ed. by R. Boel and G. Stremersch. Boston, MA: Springer US, 2000, pp. 239–246. ISBN: 978-1-4615-4493-7. DOI: 10.1007/978-1-4615-4493-7_24. URL: https://doi.org/10.1007/978-1-4615-4493-7_24.

[Mah19] Magdi S. Mahmoud. "Architecture for Cloud-Based Industrial Automation". In: *Third International Congress on Information and Communication Technology*. Ed. by Xin-She Yang, Simon Sherratt, Nilanjan Dey, and Amit Joshi. Singapore: Springer Singapore, 2019, pp. 51–62. DOI: 10.1007/978-981-13-1165-9_6.

[Mar+13] Jasna D Markovic-Petrovic and Mirjana D Stojanovic. "Analysis of SCADA System Vulnerabilities to DDoS Attacks". In: *Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), 2013 11th International Conference on*. Vol. 2. IEEE. 2013, pp. 591–594. DOI: 10.1109/TELSKS.2013.6704448.

[Mat09] John C Matherly. *SHODAN the Computer Search Engine*. 2009. URL: http://www.shodanhq.com/help (visited on August 6, 2019).

[May+18] Peter Maynard, Kieran McLaughlin, and Sakir Sezer. "An Open Framework for Deploying Experimental SCADA Testbed Networks". In: *Proceedings of Proceedings of ICS & SCADA* (2018), p. 92. DOI: 10.14236/ewic/ICS2018.11.

[McL+16] Stephen McLaughlin, Charalambos Konstantinou, Xueyang Wang, Lucas Davi, Ahmad-Reza Sadeghi, Michail Maniatakos, and Ramesh Karri. "The Cybersecurity Landscape in Industrial Control Systems". In: *Proceedings of the IEEE* 104.5 (2016), pp. 1039–1057. URL: http://dl.acm.org/citation.cfm?id=3241074.3241078 (visited on August 7, 2019).

[Meu17] Rob van der Meulen. *Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016*. 2017. URL: https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016 (visited on August 13, 2019).

[Mic19] Microsoft. *Troubleshoot blue screen errors*. 2019. URL: https://support.microsoft.com/en-us/help/14238/windows-10-troubleshoot-blue-screen-errors (visited on October 14, 2019).

[Mil+19] Steve Miller, Nathan Brubaker, Daniel Kapellmann Zafra, and Dan Caban. *TRITON Actor TTP Profile, Custom Attack Tools, Detections, and ATTACK Mapping*. April 10, 2019. URL: https://www.fireeye.com/blog/threat-research/2019/04/triton-actor-ttp-profile-custom-attack-tools-detections.html (visited on August 14, 2019).

[Mil+90] Barton P. Miller, Louis Fredriksen, and Bryan So. "An Empirical Study of the Reliability of UNIX Utilities". In: *Commun. ACM* 33.12 (December 1990), pp. 32–44. ISSN: 0001-0782. DOI: 10.1145/96267.96279.

[MITa] MITRE Corporation. *About CVE*. URL: https://cve.mitre.org/about/ (visited on December 11, 2019).

[MITb] MITRE Corporation. *Browse Vulnerabilities By Date*. URL: https://www.cvedetails.com/browse-by-date.php (visited on August 6, 2019).

[Mor+13] Thomas H Morris and Wei Gao. "Industrial Control System Cyber Attacks". In: *Proceedings of the 1st International Symposium on ICS & SCADA Cyber Security Research*. 2013, pp. 22–29. URL: https://dl.acm.org/citation.cfm?id=2735341 (visited on August 7, 2019).

[Nao+17] Laura Nao, Pierluigi Passaro, Egidio Gioia, and Matteo Petracca. "Asymmetric Multiprocessing Techniques in Smart Devices: Application in a Drone Navigation System". In: *Software, Telecommunications and Computer Networks (SoftCOM), 2017 25th International Conference on*. IEEE. 2017, pp. 1–5. DOI: 10.23919/SOFTCOM.2017.8115511.

[Nat16a] National Vulnerability Database. *CVE-2016-8366 Detail*. 2016. URL: https://nvd.nist.gov/vuln/detail/CVE-2016-8366 (visited on December 22, 2019).

[Nat16b] National Vulnerability Database. *CVE-2016-8371 Detail*. 2016. URL: https://nvd.nist.gov/vuln/detail/CVE-2016-8371 (visited on December 22, 2019).

[Nat16c] National Vulnerability Database. *CVE-2016-8380 Detail*. 2016. URL: https://nvd.nist.gov/vuln/detail/CVE-2016-8380 (visited on December 22, 2019).

[Nat17] National Vulnerability Database. *CVE-2017-13995 Detail*. 2017. URL: https://nvd.nist.gov/vuln/detail/CVE-2017-13995 (visited on December 22, 2019).

[Nat19] National Vulnerability Database. *CVE-2019-10953 Detail*. 2019. URL: https://nvd.nist.gov/vuln/detail/CVE-2019-10953 (visited on December 22, 2019).

[Nel16] Nell Nelson. *The Impact of Dragonfly Malware on Industrial Control Systems*. 2016. URL: https://www.sans.org/reading-room/whitepapers/ICS/impact-dragonfly-malware-industrial-control-systems-36672 (visited on December 16, 2019).

[NET] NETRESEC. *Network Forensics and Network Security Monitoring*. URL: https://www.netresec.com (visited on January 7, 2020).

[Ngu+15] Tan-Sy Nguyen and Thai-Hoang Huynh. "Design and Implementation of Modbus Slave based on ARM Platform and FreeRTOS Environment". In: *Advanced Technologies for Communications (ATC), 2015 International Conference on*. IEEE. 2015, pp. 462–467. DOI: 10.1109/ATC.2015.7388372.

[Nie+17] Matthias Niedermaier, Florian Fischer, and Alexander von Bodisco. "PropFuzz – An IT-security Fuzzing Framework for Proprietary ICS Protocols". In: *2017 International Conference on Applied Electronics (AE), Pilsen*. 2017, pp. 1–4. DOI: 10.23919/AE.2017.8053600.

[Nie+18a] Matthias Niedermaier, Alexander von Bodisco, and Dominik Merli. "CoRT: A Communication Robustness Testbed for Industrial Control System Components". In: *4th International Conference on Event-Based Control, Communication, and Signal Processing EBCCSP 2018*. 2018. URL: http://arxiv.org/abs/1904.04286 (visited on August 6, 2019).

[Nie+18b] Matthias Niedermaier, Thomas Hanka, Sven Plaga, Alexander von Bodisco, and Dominik Merli. "Efficient Passive ICS Device Discovery and Identification by MAC Address Correlation". In: *Proceedings of the 5th International Symposium for ICS & SCADA Cyber Security Research*. 2018. DOI: 10.14236/ewic/ICS2018.3.

[Nie+18c] Matthias Niedermaier, Jan-Ole Malchow, Florian Fischer, Daniel Marzin, Dominik Merli, Volker Roth, and Alexander von Bodisco. "You Snooze, You Lose: Measuring PLC Cycle Times under Attacks". In: *12th USENIX Workshop on Offensive Technologies (WOOT 18)*. 2018. URL: http://dl.acm.org/citation.cfm?id=3307423.3307435 (visited on August 5, 2019).

[Nie+19a] Matthias Niedermaier, Florian Fischer, Dominik Merli, and Georg Sigl. "Network Scanning and Mapping for IIoT Edge Node Device Security". In: *2019 International Conference on Applied Electronics (AE)*. 2019.

[Nie+19b] Matthias Niedermaier, Dominik Merli, and Georg Sigl. "A Secure Dual-MCU Architecture for Robust Communication of IIoT Devices". In: *2019 IEEE 8th Mediterranean Conference on Embedded Computing (MECO)*. 2019, pp. 1–5. DOI: 10.1109/MECO.2019.8760188.

[Nie+19c] Matthias Niedermaier, Martin Striegel, Felix Sauer, Dominik Merli, and Georg Sigl. *Efficient Intrusion Detection on Low-Performance Industrial IoT Edge Node Devices*. 2019. eprint: 1908.03964. URL: https://arxiv.org/abs/1908.03964 (visited on August 15, 2019).

[Nov+08] T. Novak and A. Treytl. "Functional Safety and System Security in Automation Systems – A Life Cycle Model". In: *2008 IEEE International Conference on Emerging Technologies and Factory Automation*. September 2008, pp. 311–318. DOI: 10.1109/ETFA.2008.4638412.

[OFl] Colin O'Flynn. *Pico-python*. URL: https://github.com/colinoflynn/pico-python (visited on August 6, 2019).

[OMN] OMNISIENS. *CS3STHLM - The Premier Cyber Security Conference for ICS/SCADA and Critical Infrastructure*. URL: https://cs3sthlm.se (visited on January 7, 2020).

[Pay03] Udo Payer. "State-driven Stack-based Network Intrusion Detection System". In: *Proceedings of the 7th International Conference on Telecommunications, 2003. ConTEL 2003.* Vol. 2. June 2003, 613–618 vol.2. DOI: 10.1109/CONTEL.2003.176969.

[Per08] Chad Perrin. *The CIA Triad*. 2008. URL: http://www.techrepublic.com/blog/security/the-cia-triad (visited on August 7, 2019).

[Per17] Joshua Pereyda. *Boofuzz Fuzzing Framework*. 2017. URL: https://github.com/jtpereyda/boofuzz (visited on August 6, 2019).

[Pic] Pico Technology. *Pico Oscilloscope*. URL: https://www.picotech.com/products/oscilloscope (visited on December 22, 2019).

[Pla+16] Sven Plaga, Stefan Tatschner, and Thomas Newe. "Logboat - A Simulation Framework Enabling CAN Security Assessments". In: *2016 International Conference on Applied Electronics (AE)*. September 2016, pp. 215–218. DOI: 10.1109/AE.2016.7577276.

[Pla+18] Sven Plaga, Norbert Wiedermann, Matthias Niedermaier, Alexander Giehl, and Thomas Newe. "Future Proofing IoT Embedded Platforms for Cryptographic Primitives Support". In: *2018 12th International Conference on Sensing Technology (ICST) - Wireless Sensor Networks*. 2018. DOI: 10.1109/ICSensT.2018.8603610.

[Plu82] David C Plummer. "RFC 826: An Ethernet Address Resolution Protocol". In: *InterNet Network Working Group* (1982). URL: https://tools.ietf.org/html/rfc826 (visited on August 2, 2019).

[Que+09] Carlos Queiroz, Abdun Mahmood, Jiankun Hu, Zahir Tari, and Xinghuo Yu. "Building a SCADA Security Testbed". In: *2009 Third International Conference on Network and System Security*. IEEE. 2009, pp. 357–364. DOI: 10.1109/NSS.2009.82.

[Rad+18] Harish Radhappa, Lei Pan, James Xi Zheng, and Sheng Wen. "Practical Overview of Security Issues in Wireless Sensor Network Applications". In: *International journal of computers and applications* 40.4 (2018), pp. 202–213. DOI: 10.1080/1206212X.2017.1398214.

[Ras12] Raspberry Pi Foundation. *Raspberry Pi*. 2012. URL: https://www.raspberrypi.org (visited on December 22, 2019).

[Rat+98] John W. Ratcliff and David Metzener. *Ratcliff-Obershelp Pattern Recognition*. 1998.

[Raw+17] Sanjay Rawat, Vivek Jain, Ashish Kumar, Lucian Cojocar, Cristiano Giuffrida, and Herbert Bos. *VUzzer: Application-aware Evolutionary Fuzzing*. 2017. DOI: 10.14722/ndss.2017.23404.

[Ree+12] Jason Reeves, Ashwin Ramaswamy, Michael E. Locasto, Sergey Bratus, and Sean W. Smith. "Intrusion Detection for Resource-constrained Embedded Control Systems in the Power Grid". In: *International Journal of Critical Infrastructure Protection* 5.2 (2012), pp. 74–83. DOI: 10.1016/j.ijcip.2012.02.002.

[Ren+10] Huang Renjie, Liu Feng, and Pan Dongbo. "Research on OPC UA Security". In: *2010 5th IEEE Conference on Industrial Electronics and Applications*. July 2010, pp. 1439–1444. DOI: 10.1109/ICIEA.2010.5514836.

[Ris15] RiskViz Consortium. *RiskViz – Risk Map of the Industrial IT-Security in Germany*. 2015. URL: https://www.hs-augsburg.de/RiskViz.html (visited on August 6, 2019).

[Rok+16] Michał Rokitiański et al. *Scada-LTS is an Open Source, web-based, multiplatform solution for building your own SCADA*. 2016. URL: https://github.com/SCADA-LTS/Scada-LTS (visited on December 22, 2019).

[Ron10] Armin Ronacher. *The Python micro framework for building web applications*. 2010. URL: https://github.com/pallets/flask (visited on December 22, 2019).

[Sad+15] Ahmad-Reza Sadeghi, Christian Wachsmann, and Christian Wachsmann. "Security and Privacy Challenges in Industrial Internet of Things". In: *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. June 2015, pp. 1–6. DOI: 10.1145/2744769.2747942.

[San99] Salvatore Sanfilippo. *Hping3 (8)-Linux Man Page*. 1999. URL: https://linux.die.net/man/8/hping3 (visited on August 6, 2019).

[Sau+19] Felix Sauer, Matthias Niedermaier, Susanne Kießling, and Dominik Merli. "LICSTER – A Low-cost ICS Security Testbed for Education and Research". In: *Proceedings of the 6th International Symposium for ICS & SCADA Cyber Security Research*. 2019. DOI: 0.14236/ewic/icscsr19.1.

[Say+13] Naoum Sayegh, Ali Chehab, Imad H Elhajj, and Ayman Kayssi. "Internal Security Attacks on SCADA Systems". In: *Communications and Information Technology (ICCIT), 2013 Third International Conference on*. IEEE. 2013, pp. 22–27. DOI: 10.1109/ICCITechnology.2013.6579516.

[Sch] David Schweikert. *fping*. URL: https://fping.org/ (visited on August 6, 2019).

[Sch+10] Ragnar Schierholz and Kevin McGrath. "Security Certification–A Critical Review". In: *ISA Automation Week 2010: Technology and Solutions Event* (2010).

[Sea16] Chad Seaman. *Threat Advisory: Mirai Botnet*. Tech. rep. Akamai, 2016. URL: https://www.akamai.com/de/de/resources/our-thinking/threat-advisories/akamai-mirai-botnet-threat-advisory.jsp (visited on August 7, 2019).

[Sen14] Tomás Senart. *HTTP load testing tool and library. It's over 9000!* 2014. URL: https://github.com/tsenart/vegeta (visited on December 22, 2019).

[Sha+11] Rebecca Shapiro, Sergey Bratus, Edmond Rogers, and Sean Smith. "Identifying Vulnerabilities in SCADA Systems via Fuzz-Testing". In: *Critical Infrastructure Protection V: 5th IFIP WG 11.10 International Conference on Critical Infrastructure Protection, ICCIP 2011, Hanover, NH, USA, March 23-25, 2011, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 57–72. ISBN: 978-3-642-24864-1. DOI: 10.1007/978-3-642-24864-1_5.

[Sie] Siemens. *SIMATIC WinCC V7*. URL: https://new.siemens.com/global/de/produkte/automatisierung/industrie-software/automatisierungs-software/scada/simatic-wincc-v7.html (visited on December 22, 2019).

[Sin+15] Rajni Ranjan Singh and Deepak Singh Tomar. "Network Forensics: Detection and Analysis of Stealth Port scanning Attack". In: *International Journal of Computer Networks and Communications Security* 4 (2015), p. 8. ISSN: 2410-0595. URL: http://www.ijcncs.org/published/volume3/issue2/p2_3-2.pdf (visited on August 7, 2019).

[Sla+08] Jill Slay and Michael Miller. "Lessons Learned from the Maroochy Water Breach". In: *Critical Infrastructure Protection*. Boston, MA: Springer US, 2008, pp. 73–82. ISBN: 978-0-387-75462-8. DOI: 10.1007/978-0-387-75462-8_6.

[Sou14] Arnaud Soullié. *Industrial Control Systems: Pentesting PLCs 101*. 2014. URL: https://www.blackhat.com/docs/eu-14/materials/eu-14-Soullie-Industrial-Control-Systems-Pentesting-PLCs-101.pdf (visited on August 7, 2019).

[Spi] Spirent. *Spirent - Promise. Assured.* URL: https://www.spirent.com (visited on December 22, 2019).

[STMa] STMicroelectronics. *NUCLEO-F767ZI*. URL: https://www.st.com/en/evaluation-tools/nucleo-f767zi.html (visited on December 22, 2019).

[STMb] STMicroelectronics. *STM32CubeMX*. URL: https://www.st.com/en/development-tools/stm32cubemx.html (visited on December 22, 2019).

[Sto+11] Keith Stouffer, Joe Falco, and Karen Scarfone. "Guide to Industrial Control Systems (ICS) Security". In: *NIST special publication* 800.82 (2011). DOI: 10.6028/NIST.SP.800-82r2.

[Str+19] Martin Striegel, Carsten Rolfes, Fabian Helfert, Max Hornung, Johann Heyszl, and Georg Sigl. "EyeSec: A Retrofittable Augmented Reality Tool for Troubleshooting Wireless Sensor Networks in the Field". In: *Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks*. 2019. URL: http://dl.acm.org/citation.cfm?id=3324320.3324343 (visited on August 6, 2019).

[Swa+99] Andy Swales et al. *Open Modbus/TCP Specification*. 1999. URL: http://irtfweb.ifa.hawaii.edu/~smokey/software/about/sixnet/modbus/Modbus_TCP_Standard.doc (visited on August 7, 2019).

[Tei+12] André Teixeira, Daniel Pérez, Henrik Sandberg, and Karl Henrik Johansson. "Attack Models and Scenarios for Networked Control Systems". In: *Proceedings of the 1st international conference on High Confidence Networked Systems*. ACM. 2012, pp. 55–64. DOI: 10.1145/2185505.2185515.

[Tex14] Texas Instruments. *Datasheet: SN74LVCH16245A 16-bit Bus Transceiver With 3-state Outputs*. 2014. URL: http://www.ti.com/lit/ds/symlink/sn74lvch16245a.pdf (visited on August 6, 2019).

[Til11] Filippo Tilaro. *Assessment and Testing of Industrial Devices Robustness against Cyber Security Attacks*. 2011. URL: http://cds.cern.ch/record/1398647/files/WEPMU029.pdf (visited on August 7, 2019).

[Tun+18] Andrea Tundis, Wojciech Mazurczyk, and Max Mühlhäuser. "A Review of Network Vulnerabilities Scanning Tools: Types, Capabilities and Functioning". In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ARES 2018. Hamburg, Germany: Association for Computing Machinery, 2018. ISBN: 9781450364485. DOI: 10.1145/3230833.3233287. URL: https://doi.org/10.1145/3230833.3233287.

[Wan+06] Yong Wang, Garhan Attebury, and Byrav Ramamurthy. *A Survey of Security Issues in Wireless Sensor Networks*. February 2006. DOI: 10.1109/COMST.2006.315852.

[Web+06] S. Webster, R. Lippmann, and M. Zissman. "Experience Using Active and Passive Mapping for Network Situational Awareness". In: *Fifth IEEE International Symposium on Network Computing and Applications (NCA'06)*. July 2006, pp. 19–26. DOI: 10.1109/NCA.2006.23.

[Wed+15] Adam Wedgbury and Kevin Jones. "Automated Asset Discovery in Industrial Control Systems: Exploring the Problem". In: *Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research*. ICS-CSR '15. Ingolstadt, Germany: BCS Learning & Development Ltd., 2015, pp. 73–83. ISBN: 978-1-78017-317-7. DOI: 10.14236/ewic/ICS2015.8.

[Xie+14] Feng Xie, Yong Peng, Wei Zhao, Yang Gao, and Xuefeng Han. "Evaluating Industrial Control Devices Security: Standards, Technologies and Challenges". In: *IFIP International Conference on Computer Information Systems and Industrial Management*. Springer. 2014, pp. 624–635. DOI: 10.1007/978-3-662-45237-0_57.

[Yoo+16] Hyunguk Yoo and Taeshik Shon. "Grammar-based Adaptive Fuzzing: Evaluation on SCADA Modbus Protocol". In: *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. November 2016, pp. 557–563. DOI: 10.1109/SmartGridComm.2016.7778820.

[Zal18] Michał Zalewski. *p0f v3*. 2018. URL: http://lcamtuf.coredump.cx/p0f3/ (visited on August 6, 2019).

[Zhu+10] Bonnie Zhu and Shankar Sastry. "SCADA-specific Intrusion Detection/Prevention Systems: A Survey and Taxonomy". In: *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*. Vol. 11. 2010. URL: https://pdfs.semanticscholar.org/1027/2f29fff747d7efccab3b58d64ffd1112c811.pdf (visited on December 19, 2019).

[Zhu+11] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. "A Taxonomy of Cyber Attacks on SCADA Systems". In: *Internet of things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing*. IEEE. 2011, pp. 380–388. DOI: 10.1109/iThings/CPSCom.2011.34.

[Zil+10] Amir Zilberstein and Lior Frenkel. *Protection of Control Networks Using a One-way Link*. US Patent 7,649,452. January 2010.

[Zim+15] Christopher Zimmer, Balasubramany Bhat, Frank Mueller, and Sibin Mohan. "Intrusion Detection for CPS Real-time Controllers". In: *Cyber Physical Systems Approach to Smart Electric Power Grid*. Ed. by James D. Khaitan Siddhartha Kumarand McCalley and Chen Ching Liu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 329–358. ISBN: 978-3-662-45928-7. DOI: 10.1007/978-3-662-45928-7_12. URL: https://doi.org/10.1007/978-3-662-45928-7_12.

Acronyms

ACK ..... acknowledgment

API ..... Application Programming Interface

APT ..... Advanced Persistent Threat

ARP ..... Address Resolution Protocol

ARP 1 ..... Address Resolution Protocol - Who has

ARP 2 ..... Address Resolution Protocol - Probe

ARP 3 ..... Address Resolution Protocol - Request TPA=SPA, THA=0

ARP 4 ..... Address Resolution Protocol - Reply TPA=SPA, THA=SHA

BSI ..... Federal Office for Information Security

CERT@VDE .. Cyber Emergency Response Team at the Association of Electrical Engineering, Electronics and Information Technology

CIA ..... Confidentiality, Integrity and Availability

CoRT ..... Communication Robustness Testbed

CPE ..... Common Platform Enumeration

CPS ..... Cyber-Physical System

CPU ..... Central Processing Unit

CRC ..... Cyclic Redundancy Check

CSV ..... Comma-Separated Values

CVE ..... Common Vulnerabilities and Exposures

DDoS ..... Distributed Denial of Service

DHCP ..... Dynamic Host Configuration Protocol

DMA ..... Direct Memory Access

DoS ..... Denial of Service

DuT ..... Device under Test

ERP ..... Enterprise Resource Planning

FTP ..... File Transfer Protocol

Gb/s ..... Giga bits per second

GRE ..... Generic Routing Encapsulation

HAL ..... Hardware Abstraction Layer

HTTP ..... Hypertext Transfer Protocol

HTTPS ..... Hypertext Transfer Protocol Secure

HMAC ..... Keyed-Hash Message Authentication Code

HMI ..... Human Machine Interface

I²C ..... Inter Integrated Circuit

IEEE ..... Institute of Electrical and Electronics Engineers

ICMP ..... Internet Control Message Protocol

ICS ..... Industrial Control System

ICS-CERT .... Industrial Control System Computer Emergency Response Team

IDE ..... Integrated Development Environment

IDS ..... Intrusion Detection System

IEC ..... International Electrotechnical Commission

IIoT ..... Industrial Internet of Things

ILC ..... Inline-Controller

IoT ..... Internet of Things

IO ..... Input/Output

IP ..... Internet Protocol

IPS ..... Intrusion Prevention System

IRQ ..... Interrupt Request

IT ..... Information Technology

JSON ..... JavaScript Object Notation

kb/s ..... kilo bits per second

KDE ..... Kernel Density Estimation

LED ..... Light-emitting Diode

LICSTER ..... Low-cost ICS Security Testbed for Education and Research

LwIP ..... Lightweight IP

M2M ..... Machine to Machine

MAC ..... Media Access Control

MCU ..... Microcontroller Unit

MES ..... Manufacturing Execution System

MitM ..... Man-in-the-Middle

MOSFET .... Metal-oxide-semiconductor Field-effect Transistor

Ms/s ..... Mega samples per second

Mb/s ..... Mega bits per second

NIC ..... Network Interface Card

NSE ..... Nmap Scripting Engine

OLED ..... Organic Light Emitting Diode

OS ..... Operating System

OSI ..... Open Systems Interconnection

OT ..... Operational Technology

PC ..... Personal Computer

pcap ..... packet capture

pcap-ng ..... packet capture - next generation

PCB ..... Printed Circuit Board

PDF ..... Probability Density Function

PLC ..... Programmable Logic Controller

PoC ..... Proof of Concept

PRU ..... Programmable Realtime Unit

PSK ..... Pre-shared Key

RA ..... Registration Authority

RAM ..... Random-Access Memory

RAMS ..... Reliability, Availability, Maintainability and Safety

RAMSST .... Reliability, Availability, Maintainability, Safety, Security and Testability

RAT ..... Remote Access Trojan

ROM ..... Read-only Memory

RST ..... reset

RTOS ..... Real-Time Operating System

RTU ..... Remote Terminal Unit

RX ..... Receive

SCADA ..... Supervisory Control and Data Acquisition

SD ..... Secure Digital

SHA ..... Sender Hardware Address

SNMP ..... Simple Network Management Protocol

SoC ..... System-on-a-Chip

SPA ..... Sender Protocol Address

SPI ..... Serial Peripheral Interface

SRA ..... Safety, Reliability and Availability

SSH ..... Secure Shell

ST ..... Structured Text

STM ..... STMicroelectronics

SWD ..... Serial Wire Debug

SYN ..... synchronize

TAP ..... Terminal Access Point

TCP ..... Transmission Control Protocol

THA ..... Target Hardware Address

TPA ..... Target Protocol Address

TTL ..... Time to Live

TX ..... Transmit

UART ..... Universal Asynchronous Receiver Transmitter

UDP ..... User Datagram Protocol

URL ..... Uniform Resource Locator

USB ..... Universal Serial Bus

US-CERT .... United States Computer Emergency Readiness Team

VLAN ..... Virtual Local Area Network

VM ..... Virtual Machine

Appendix A

Appendix

Contents of the appendix

Additional materials are included in the appendix for better understanding of the measurement results and demonstrator setups.

A.1 Search Engine Parameters

The search engine parameters and Uniform Resource Locator (URL) with which the queries were made are listed in the following. As far as specified, these are the search engine's standard calls.

Table A.1 shows the queries of the *Shodan* search engine, which are used to get an overview of Internet-facing PLCs (see Table 1.1).

Table A.1: Search engine parameters for Shodan.

Protocol	Port	Query	URL
Modbus	502 TCP	port:502	`https://www.shodan.io/search?query=port%3A502`
Siemens S7	102 TCP	port:102	`https://www.shodan.io/search?query=port%3A102`
DNP3	20000 TCP	port:20000 source address	`https://www.shodan.io/search?query=port%3A20000+source+address`
BACnet	47808 TCP	port:47808	`https://www.shodan.io/search?query=port%3A47808`
Niagara Fox	1911 TCP	port:1911,4911 product:Niagara	`https://www.shodan.io/search?query=port%3A1911%2C4911+product%3ANiagara`
Ethernet/IP	244818 TCP	port:44818	`https://www.shodan.io/search?query=port%3A44818`
Phoenix PCWorx	1962 TCP	port:1962 PLC	`https://www.shodan.io/search?query=port%3A1962+PLC`
Codesys	2455 TCP	port:2455 operating system	`https://www.shodan.io/search?query=port%3A2455+operating+system`

Table A.2 shows the queries of the *Censys* search engine, which are used to get an overview of Internet-facing PLCs (see Table 1.1).

Table A.2: Search engine parameters for Censys.

Protocol	Port	Query	URL
Modbus	502 TCP	protocols: "502/modbus"	`https://censys.io/ipv4?q=protocols%3A+%22502%2Fmodbus%22`
Siemens S7	102 TCP	protocols: "102/s7"	`https://censys.io/ipv4?q=protocols%3A+%22102%2Fs7%22`
DNP3	20000 TCP	protocols: "20000/dnp3"	`https://censys.io/ipv4?q=protocols%3A+%2220000%2Fdnp3%22`
BACnet	47808 TCP	protocols: "47808/bacnet"	`https://censys.io/ipv4?q=protocols%3A+%2247808%2Fbacnet%22`
Niagara Fox	1911 TCP	protocols: "1911/fox"	`https://censys.io/ipv4?q=protocols%3A+%221911%2Ffox%22`
Ethernet/IP	244818 TCP	Not available	Not available
Phoenix PCWorx	1962 TCP	Not available	Not available
Codesys	2455 TCP	Not available	Not available

Table A.3 shows the queries of the *ZoomEye* search engine, which are used to get an overview of Internet-facing PLCs (see Table 1.1).

Table A.3: Search engine parameters for ZoomEye.

Protocol	Port	Query	URL
Modbus	502 TCP	+port:"502"	`https://www.zoomeye.org/searchResult?q=%2Bport%3A%22502%22&t=all&is_dork=0`
Siemens S7	102 TCP	+port:"102"	`https://www.zoomeye.org/searchResult?q=%2Bport%3A%22102%22&t=all&is_dork=0`
DNP3	20000 TCP	+port:20000 +service:"dnp"	`https://www.zoomeye.org/searchResult?q=port:20000%20%2Bservice:%22dnp%22&t=all&is_dork=0`
BACnet	47808 TCP	+port:47808	`https://www.zoomeye.org/searchResult?q=port:47808`
Niagara Fox	1911 TCP	+port:1911	`https://www.zoomeye.org/searchResult?q=port:1911`
Ethernet/IP	244818 TCP	+port:44818	`https://www.zoomeye.org/searchResult?q=port:44818`
Phoenix PCWorx	1962 TCP	+port:1962	`https://www.zoomeye.org/searchResult?q=port:1962`
Codesys	2455 TCP	+service:"CoDeSyS"	`https://www.zoomeye.org/searchResult?q=%2Bservice%3A%22CoDeSyS%22&t=all&is_dork=0`

Table A.4 shows the queries of the *Ditecting* search engine, which are used to get an overview of Internet-facing PLCs (see Table 1.1).

Table A.4: Search engine parameters for Ditecting.

Protocol	Port	Query	URL
Modbus	502 TCP	service:Modbus	`http://www.ditecting.com/index.php/home/Result/index.html?query=service%3AModbus`
Siemens S7	102 TCP	service:Siemens s7	`http://www.ditecting.com/index.php/home/Result/index.html?query=service%3ASiemens%20s7`
DNP3	20000 TCP	service:DNP3	`http://www.ditecting.com/index.php/home/Result/index.html?query=service%3ADNP3`
BACnet	47808 TCP	service:BACnet	`http://www.ditecting.com/index.php/home/Result/index.html?query=service%3ABACnet`
Niagara Fox	1911 TCP	port:1911	`http://www.ditecting.com/index.php/home/Result/index.html?query=port%3A1911`
Ethernet/IP	244818 TCP	service:EtherNet/IP	`http://www.ditecting.com/index.php/home/Result/index.html?query=service%3AEtherNet/IP`
Phoenix PCWorx	1962 TCP	service:PCWorx	`http://www.ditecting.com/index.php/home/Result/index.html?query=service%3APCWorx`
Codesys	2455 TCP	port:2455	`http://www.ditecting.com/index.php/home/Result/index.html?query=port%3A2455`

A.2 Programmable Logic Controller Cycle Time Measurements

The appendix lists additional measurement results for every PLC in the testbed. The attack with the most influence for each controller is marked with a grey background. The red marking here shows the highest value..

A.2.1 Mean Idle Cycle Time

Table A.5 shows the mean cycle times without network load ($\bar{t}_{idle}$) of the different PLCs used in Section 3.3.

Table A.5: Overview of the mean idle cycle time of each PLC.

PLC	Mean idle cycle time
Wago 750-889 (1)	2000 μs
Wago 750-8100 (2)	10071 μs
Wago 750-880 (3)	1999 μs
Wago 750-831 (4)	2000 μs
Siemens S7-1211 (5)	223 μs
Siemens S7-1212 (6)	30500 μs
Siemens ET200-SP (7)	4500 μs
Siemens S7-314 (8)	206 μs
Siemens S7-1516F (9)	108 μs
Siemens Logo! 8 (10)	171 μs
Phoenix ILC 151 ETH (11)	1000 μs
Phoenix ILC 150 ETH (12)	1085 μs
Phoenix ILC 171 ETH (13)	1000 μs
ABB PM554-TP-ETH (14)	1000 μs
Crouzet EM4 (15)	1999 μs
Schneider M221 (16)	2000 μs

A.2.2 Overview

The measurement results of the network load tests of all measured PLCs are shown in the following tables.

Table A.6: Cycle time in μs during attacks against Wago devices

Device	Attack	Mean Pre	Mean Att	Mean Post	Median Pre	Median Att	Median Post	Min Pre	Min Att	Min Post	Max Pre	Max Att	Max Post
750-889 (1)	zgrab modbus	2000	2213	2000	1816	2356	2164	54	237	1467	2579	52024	2538
	zgrab http	2000	2000	2000	2167	1826	2227	1491	274	1447	2502	2539	2559
	syn	2000	2000	2000	1841	2010	2225	505	1579	1316	2562	2413	2697
	snmp	2000	2000	2000	2142	1800	2140	1393	973	1413	2620	2633	2594
	http	2000	2000	2000	2054	2168	2165	1430	1442	1483	2574	2568	2531
	hping udp flood	2000	140044	2000	2226	1633	2202	1406	414	1471	2603	8397008	2519
	hping S P U flood	2000	18282	2000	1806	2346	2166	279	1533	1447	2533	8067682	2628
	hping c1 1 C17 flood	2000	2000	2000	1784	1760	1772	425	775	463	2713	2620	2641
	arp 4	2000	199062	2000	2228	1789	1828	1445	604	243	2557	5526857	2559
	arp 3	2000	182637	2000	1758	1636	2235	684	597	1332	2530	5085928	2687
	arp 2	2000	306792	2000	1789	1639	1800	1408	584	386	2598	6540940	2511
	arp 1	2000	375307	2000	1831	1640	2167	1420	566	1487	2601	11540100	3365
750-8100 (2)	zgrab modbus	10101	19148	10093	10334	10398	10334	7160	8303	9553	30389	779529	30408
	zgrab http	10071	10073	10084	10327	10332	10334	9575	2486	7683	30407	30414	30428
	syn	10091	13545	10097	10330	10355	10334	7261	791	717	39742	90279	30434
	snmp	10063	10064	10057	10324	10330	10333	5876	9569	9569	30440	30432	30385
	http	10064	10057	10088	10334	10326	10329	9559	9575	8140	30403	30428	30407
	hping udp flood	10054	14724	10074	10334	10353	10334	9579	1431	9549	30432	89676	39737
	hping S P U flood	10066	14163	10066	10329	10348	10334	4387	8005	1819	30415	109617	30399
	hping c1 1 C17 flood	10091	10053	10078	10337	10333	10331	7087	8057	9569	30429	30394	30452
	arp 4	10113	11131	10085	10333	10348	10332	9451	9562	9571	30452	39632	40511
	arp 3	10054	11493	10054	10323	10350	10332	9580	4960	9521	30390	40462	30401
	arp 2	10081	11472	10060	10336	10341	10331	9549	9543	9562	49663	40400	30401
	arp 1	10081	11332	10074	10338	10340	10335	9557	7883	9563	30395	49598	30389
750-880 (3)	zgrab modbus	1999	2214	1999	1702	2337	1780	581	248	581	3388	61882	3556
	zgrab http	1999	1998	2000	1717	1701	2148	580	583	580	3384	3445	3472
	syn	1999	2019	1997	1712	2308	1733	581	1624	309	3426	2639	3392
	snmp	1997	1996	1996	1727	1723	1721	581	308	23	3475	3385	3453
	http	1999	1999	2000	1709	1716	2284	581	377	582	3430	3385	3451
	hping udp flood	1997	178903	1997	1731	1630	1727	580	332	580	3444	6283455	3491
	hping S P U flood	1998	87062	1996	1732	1634	1722	153	601	339	3384	42502647	3469
	hping c1 1 C17 flood	1998	1997	1996	1727	1728	1729	578	580	581	3410	3381	3484
	arp 4	1999	160511	1995	1774	1630	1724	577	575	580	3440	5400406	3392
	arp 3	1997	162632	1998	1717	1712	1740	581	557	580	3445	3761630	3446
	arp 2	1999	465689	1998	1734	1629	1734	574	612	580	3386	10973587	3487
	arp 1	1998	575663	1997	1732	2372	1725	581	617	580	3390	9987443	3392
750-831 (4)	zgrab modbus	2000	2233	2000	1890	2334	1903	320	337	1230	3448	96089	3226
	zgrab http	2000	2000	2000	2003	2192	2162	1439	1371	1385	3308	3367	3440
	syn	2000	1999	2000	1809	2006	2010	1326	1568	1447	2649	2448	2567
	snmp	2000	2000	2000	2232	1784	2233	547	578	1367	3342	3382	2646
	http	2000	2000	2000	1918	2002	1862	1407	1418	1336	3316	3208	3157
	hping udp flood	2000	75698	2000	1819	2336	2038	920	338	1457	2600	3869519	3043
	hping S P U flood	2000	18353	2000	1798	2326	2204	236	1521	1396	3424	7968669	3246
	hping c1 1 C17 flood	2000	2000	2000	2044	2180	2248	1085	1411	562	2678	3151	3399
	arp 4	2000	151997	2000	2200	1676	2020	1374	315	1074	2641	3321851	2650
	arp 3	2000	244520	2000	2002	2344	1910	1390	568	1484	2633	4065957	2531
	arp 2	2000	653732	2000	1777	2368	2133	1440	605	73	2606	7627166	2684
	arp 1	2000	467949	2000	1762	2366	2146	61	586	982	2646	6923920	2672

Table A.7: Cycle time in μs during attacks against Siemens devices

Device	Attack	Mean Pre	Mean Att	Mean Post	Median Pre	Median Att	Median Post	Min Pre	Min Att	Min Post	Max Pre	Max Att	Max Post
S7-1211 (5)	zgrab s7	223	233	223	192	141	192	110	108	110	1336	1572	1360
	zgrab https	223	223	223	192	192	192	109	109	54	1377	1320	1365
	zgrab http	223	223	223	192	192	192	92	37	108	1304	1324	1344
	szl	223	223	223	192	192	192	109	108	108	1353	1395	1354
	syn	223	325	223	192	161	191	91	108	109	1413	1638	1389
	snmp	223	223	223	192	192	192	110	110	109	1237	1297	1283
	http	223	223	223	192	192	192	89	111	26	1371	1252	1346
	hping udp flood	223	310	223	191	167	191	16	108	109	1372	1945	1363
	hping S P U flood	223	343	223	191	166	191	109	109	107	1312	1736	1337
	hping c1 1 C17 flood	223	223	223	191	191	191	104	109	106	1414	1348	1414
	arp 4	223	306	223	192	168	191	109	108	12	1340	1756	1207
	arp 3	223	301	223	192	156	191	85	62	108	1299	1645	1361
	arp 2	223	317	223	192	160	191	110	108	20	1003	1612	1371
	arp 1	223	317	223	192	160	191	109	108	57	1339	1587	1354
S7-1212 (6)	zgrab s7	30497	38125	30500	30968	39129	29073	20327	4211	28939	32061	44032	32063
	zgrab https	30500	30501	30500	29075	30501	29068	28940	28810	28937	32062	32069	32068
	zgrab http	30501	30500	30489	30502	29073	30501	28938	28938	5919	32064	32063	32068
	szl	30500	30494	30500	29072	30498	29077	28939	15975	28941	32066	32065	32063
	syn	30486	30760	30501	29075	31974	30500	276	28938	28450	32062	41043	32061
	snmp	30494	30489	30494	30502	30502	30499	15425	4873	15925	32064	32064	32068
	http	30489	30500	30490	30501	29076	30504	5396	28939	6947	32063	32061	32067
	hping udp flood	30487	30669	30496	29078	31967	30502	2286	28780	18996	33002	38999	32063
	hping S P U flood	30490	30676	30500	30498	31974	29074	6812	28937	28938	32065	39011	32064
	hping c1 1 C17 flood	30501	30500	30496	30498	29074	30501	28939	28936	19497	32064	32066	32063
	arp 4	30490	30671	30500	30501	31968	29074	7606	28940	28939	32057	38026	32062
	arp 3	30501	30671	30500	30500	31970	29075	28243	28499	28937	32065	38434	32063
	arp 2	30501	30699	30500	30504	31973	29070	28936	28418	28943	32063	39578	32062
	arp 1	30486	30720	30500	29075	31971	29074	895	28933	28938	32067	40011	32064
ET200-SP (7)	zgrab s7	4575	4530	4422	3947	3947	3944	1943	949	1054	33946	28054	36054
	zgrab https	4610	4571	4512	3947	3947	3947	1943	1943	1943	36054	35949	31943
	zgrab http	4468	4523	4608	3947	3947	3947	1054	945	1943	41949	30052	36054
	szl	4517	4634	4470	3947	3947	3947	1054	1943	1943	29948	31947	40056
	syn	4647	4395	4583	3947	3944	3946	1055	944	1053	36058	34059	44054
	snmp	4440	4383	4310	3947	3944	3943	1198	1943	1054	32055	38056	32055
	http	4477	4437	4572	3944	3944	3947	1055	1943	548	27945	34055	56056
	hping udp flood	4425	4551	4645	3944	3947	3947	948	945	1943	31945	35948	32055
	hping S P U flood	4506	4574	4505	3947	3947	3947	1943	945	945	26054	33945	28054
	hping c1 1 C17 flood	4475	4275	4620	3944	3943	3947	1943	945	794	37947	33944	33948
	arp 4	4451	3869	4587	3947	2058	3947	405	945	944	32056	26055	30057
	arp 3	4576	3921	4459	3947	2058	3947	1943	944	1943	34054	29947	36054
	arp 2	4541	4492	4505	3946	3947	3947	945	1054	1943	30056	33948	29948
	arp 1	4450	4400	4408	3947	3947	3947	1055	105	1943	30056	30052	37946
S7-314 (8)	zgrab s7	206	271	206	223	253	223	117	114	116	619	880	619
	zgrab https	206	206	206	223	223	223	117	117	117	674	660	619
	zgrab http	206	206	206	223	223	223	117	117	117	674	619	619
	szl	206	258	206	223	252	223	75	114	117	633	880	674
	syn	206	266	206	223	253	223	116	114	116	619	853	661
	snmp	206	206	206	223	223	223	117	117	117	619	646	619
	http	206	206	206	223	223	223	116	117	117	660	661	620
	hping udp flood	206	265	206	223	252	223	117	3	117	619	922	619
	hping S P U flood	206	267	206	223	253	223	102	112	117	660	853	633
	hping c1 1 C17 flood	206	206	206	223	223	223	117	117	117	619	620	619
	arp 4	206	264	206	223	252	223	16	115	117	674	840	619
	arp 3	206	265	206	223	252	223	117	114	117	619	840	661
	arp 2	206	267	206	223	253	223	117	114	117	674	881	619
	arp 1	206	267	206	223	253	223	75	115	117	619	840	633
S7-1516F (9)	zgrab s7	108	130	108	131	135	131	18	18	15	576	509	584
	zgrab https	108	108	108	131	131	131	18	18	18	616	588	588
	zgrab http	108	108	108	131	131	131	18	16	18	580	592	601
	szl	108	108	108	131	131	131	18	17	18	589	592	580
	syn	108	129	108	131	135	131	19	18	18	581	584	608
	snmp	108	108	108	131	131	131	18	19	18	621	580	561
	http	108	108	108	131	131	131	18	22	18	565	592	576
	hping udp flood	108	128	108	131	135	131	18	18	15	588	593	588
	hping S P U flood	108	128	108	131	135	131	16	18	18	600	573	619
	hping c1 1 C17 flood	108	108	108	131	131	131	22	18	18	565	564	617
	arp 4	108	129	108	131	135	131	18	18	19	577	585	581
	arp 3	108	129	108	131	135	131	18	18	22	592	597	635
	arp 2	108	129	108	131	135	131	17	19	18	569	581	576
	arp 1	108	129	108	131	135	131	22	22	18	604	608	621
Siemens Logo! 8 (10)	zgrab modbus	171	354	171	260	282	260	32	23	30	1420	2456	1443
	zgrab http	171	171	171	259	259	260	32	32	32	1459	1445	1436
	syn	171	420	171	260	285	260	32	20	32	1474	2170	1430
	snmp	171	171	171	260	260	260	18	32	32	1494	1446	1477
	http	171	171	171	260	260	260	22	32	32	1437	1454	1435
	hping udp flood	171	373	171	260	282	259	32	21	18	1440	2141	1481
	hping S P U flood	171	365	171	259	281	260	32	22	19	1429	2181	1502
	hping c1 1 C17 flood	171	171	171	260	260	260	18	32	18	1445	1429	1453
	arp 4	171	289	171	259	289	260	29	32	32	1479	1716	1460
	arp 3	171	290	171	259	289	259	32	19	32	1433	1714	1453
	arp 2	171	756	171	259	277	259	32	20	32	1454	96776	1435
	arp 1	171	791	171	260	276	260	19	17	32	1434	96160	1453

Table A.8: Cycle time in μs during attacks against Phoenix Contact devices

Device	Attack	Mean Pre	Mean Att	Mean Post	Median Pre	Median Att	Median Post	Min Pre	Min Att	Min Post	Max Pre	Max Att	Max Post
ILC 151 ETH (11)	zgrab modbus	1000	1001	1000	994	1044	945	636	10	498	1363	4227	1338
	zgrab http	1000	1000	1000	992	1097	1070	545	624	644	1329	1417	1351
	syn	1000	1003	1000	998	1029	1080	642	14	629	1330	4253	1385
	snmp	1000	1000	1000	916	1127	910	125	568	160	1404	1465	1337
	http	1000	1000	1000	899	1018	971	646	627	626	1368	1340	1336
	hping udp flood	1000	1001	1000	882	1012	980	470	31	329	1360	4228	1397
	hping S P U flood	1000	1002	1000	1142	1031	934	632	55	550	1423	3186	1385
	hping c1 1 C17 flood	1000	1000	1000	1068	926	1142	614	360	595	1387	1365	1409
	arp 4	1000	1002	1000	964	1072	854	84	13	519	1374	4231	1426
	arp 3	1000	1003	1000	1008	1072	946	702	27	630	1285	4226	1348
	arp 2	1000	1003	1000	1011	1063	925	133	18	84	1383	4260	1398
	arp 1	1000	1003	1000	873	1056	1056	253	7	568	1400	4238	1450
ILC 150 ETH (12)	zgrab modbus	1086	1086	1086	1111	1110	1111	953	269	952	4244	6313	4089
	zgrab https	1085	1085	1085	1111	1111	1111	434	6	341	4244	4240	4241
	zgrab http	1085	1085	1085	1111	1111	1111	929	953	954	4232	4244	4244
	szl	1084	1084	1084	1110	1111	1111	534	954	124	4242	4242	4240
	syn	1085	1088	1084	1111	1111	1111	953	952	287	4239	4260	4244
	snmp	1083	1084	1085	1111	1111	1111	917	954	952	4244	4239	4244
	http	1085	1085	1085	1111	1111	1111	954	520	93	4244	4244	4244
	hping udp flood	1086	1087	1088	1111	1111	1111	437	952	256	4246	4259	4244
	hping S P U flood	1086	1088	1085	1111	1111	1111	787	952	297	4244	4338	4246
	hping c1 1 C17 flood	1086	1086	1086	1111	1111	1111	809	953	232	4240	4240	4246
	arp 4	1086	1091	1086	1111	1111	1111	267	469	953	4241	4247	4245
	arp 3	1085	1090	1086	1111	1111	1111	952	455	953	4246	6160	4242
	arp 2	1084	1091	1085	1111	1111	1111	953	927	475	4242	6295	4243
	arp 1	1084	1090	1084	1111	1111	1111	841	953	540	4238	6156	4246
ILC 171 ETH (13)	zgrab modbus	1000	1000	1000	982	994	993	730	56	717	1275	4202	1279
	zgrab https	1000	1000	1000	949	988	992	719	718	727	1274	1275	1271
	zgrab http	1000	1000	1000	807	802	988	718	100	718	1274	1274	1278
	szl	1000	1000	1000	799	1190	799	625	701	157	1271	1297	1272
	syn	1000	1003	1000	994	1123	1001	701	54	774	1297	4219	1228
	snmp	1000	1000	1000	849	867	850	299	182	466	1232	1228	1227
	http	1000	1000	1000	799	994	799	261	727	272	1330	1271	1271
	hping udp flood	1000	1002	1000	1002	1070	964	755	40	722	1222	4221	1279
	hping S P U flood	1000	1001	1000	852	1013	856	254	22	88	1232	4239	1222
	hping c1 1 C17 flood	1000	1000	1000	856	1024	889	119	698	755	1222	1222	1222
	arp 4	1000	1003	1000	851	1085	852	540	22	147	1224	4217	1245
	arp 3	1000	1003	1000	806	1078	1004	261	9	769	1518	4235	1226
	arp 2	1000	1003	1000	855	1082	1191	206	25	715	1228	4244	1276
	arp 1	1000	1003	1000	850	1075	855	468	8	592	1228	4263	1229

Table A.9: Cycle time in μs during attacks against ABB devices

Device	Attack	Mean Pre	Mean Att	Mean Post	Median Pre	Median Att	Median Post	Min Pre	Min Att	Min Post	Max Pre	Max Att	Max Post
PM554-TP-ETH (14)	zgrab modbus	1000	1143	1000	1074	1079	1076	908	97	903	1916	3100	1097
	zgrab http	1000	1000	1000	926	1072	926	510	903	626	1099	1097	1097
	syn	1000	1107	1000	1000	1079	1074	903	895	232	1095	5089	1911
	snmp	1000	1000	1000	1074	1074	1076	523	902	904	1916	1095	1097
	http	1000	1000	1000	1073	1076	925	905	904	732	1097	1097	1095
	hping udp flood	1000	1070	1000	1074	1079	1074	905	662	903	1096	3919	1910
	hping S P U flood	1000	1073	1000	925	1078	926	747	231	542	2087	3099	1098
	hping c1 1 C17 flood	1000	1000	1000	1073	1074	924	906	903	839	1094	1915	1096
	arp 4	1000	1000	1000	1074	1002	1073	902	901	902	1097	1101	1095
	arp 3	1000	1000	1000	926	1000	926	114	901	238	1096	1099	1097
	arp 2	1000	1010	1000	927	1074	926	72	897	240	2079	3909	1096
	arp 1	1000	1014	1000	1000	1075	1072	903	322	903	2086	3091	1096

Table A.10: Cycle time in μs during attacks against Crouzet devices

Device	Attack	Mean Pre	Mean Att	Mean Post	Median Pre	Median Att	Median Post	Min Pre	Min Att	Min Post	Max Pre	Max Att	Max Post
em4 (15)	zgrab modbus	1999	1999	1999	1999	1999	1999	435	1841	1992	2006	2007	2006
	zgrab http	1999	1999	1999	1999	1999	1999	920	1992	1992	2006	2006	2006
	syn	1999	1999	1999	1999	1999	1999	565	1992	1991	2006	2006	2006
	snmp	1999	1999	1999	1999	1999	1999	615	793	1992	2006	2006	2006
	http	1999	1999	1999	1999	1999	1999	1634	861	1992	2006	2006	2006
	hping udp flood	1999	1999	1999	1999	1999	1999	1992	1992	170	2006	2006	2007
	hping S P U flood	1999	1999	1999	1999	1999	1999	768	984	1593	2010	2006	2006
	hping c1 1 C17 flood	1999	1999	1999	1999	1999	1999	1992	1992	1381	2006	2005	2006
	arp 4	1999	1999	1999	1999	1999	1999	1571	557	1992	2006	2005	2006
	arp 3	1999	1999	1999	1999	1999	1999	1992	1988	1992	2005	2010	2006
	arp 2	1999	1999	1999	1999	1999	1999	1992	1992	1991	2006	2006	2006
	arp 1	1999	1999	1999	1999	1999	1999	1069	1992	1992	2006	2006	2006

Table A.11: Cycle time in μs during attacks against Schneider devices

Device	Attack	Mean Pre	Mean Att	Mean Post	Median Pre	Median Att	Median Post	Min Pre	Min Att	Min Post	Max Pre	Max Att	Max Post
Modicon M2 (16)	zgrab modbus	2000	2152	2000	2000	2001	2003	134	1944	188	2034	54041	2045
	syn	2000	2852	2000	1998	2002	2000	644	1943	742	2045	52012	3008
	snmp	2000	2000	2000	2000	2000	2000	1464	1703	1953	2051	2056	2048
	hping udp flood	2000	10774	2000	2000	8003	2000	1208	1782	1948	2035	67019	2054
	hping S P U flood	2000	7773	2000	2000	4002	2000	1266	48	686	2035	76015	2050
	hping c1 1 C17 flood	2000	2000	2000	2000	2000	2000	1964	1317	1951	2051	2051	2051
	arp 4	2000	9004	2000	2000	8000	2000	1943	1621	1967	2057	102011	2033
	arp 3	2000	9050	2000	2000	8002	2000	1953	1946	1134	2047	97013	2048
	arp 2	2000	2628	2000	2001	2002	2000	1968	1935	1798	2033	36006	2033
	arp 1	2000	2494	2000	2000	2002	2001	1953	1275	1967	2048	35005	2034

A.2.3 Wago at Different Rates

The Wago PLCs offer the possibility to limit the bandwidth of the network traffic. In order to evaluate whether this is sufficient as a protective measure, various bandwidths were tested. However, the evaluation showed that only a very strong bandwidth limitation of 64 kb/s is effective.

Table A.12: Cycle time in μs during attacks against Wago devices at 64 kb/s

Device	Attack	Mean Pre	Mean Att	Mean Post	Median Pre	Median Att	Median Post	Min Pre	Min Att	Min Post	Max Pre	Max Att	Max Post
750-889 (1)	arp 4	2000	2001	2000	2132	2199	1755	1207	598	890	2783	3702	2665
	arp 3	2000	2001	2000	2149	2198	2178	1284	376	1292	2716	3691	2709
	arp 2	2000	2004	2000	2099	2152	1816	1342	598	1225	2684	4403	2782
	arp 1	2000	2004	2000	1786	2154	1754	481	244	1154	2775	4516	2629
750-8100 (2)	arp 4	9999	9999	9997	10320	9697	9715	9573	9565	214	10478	10456	10462
	arp 3	9999	9999	9998	10323	10008	9736	9568	9555	5315	10470	10462	10457
	arp 2	9999	9999	9998	10288	9699	9746	9570	9502	2781	10495	10448	10491
	arp 1	9997	10005	9999	9737	10016	10312	2408	6947	9566	10488	30394	10474
750-880 (3)	arp 4	2000	2009	1998	2276	2302	1715	580	352	580	3382	4387	3387
	arp 3	2000	2007	1998	1840	2298	1710	581	357	579	3417	4386	3419
	arp 2	1999	2010	1998	1757	2302	1704	580	518	581	3402	5377	3430
	arp 1	2000	2011	1998	1795	2300	1708	578	288	580	3384	4699	3391
750-831 (4)	arp 4	2000	2003	2000	2088	2163	2008	1377	322	1358	3042	4398	3368
	arp 3	2000	2003	2000	2226	2162	1981	1375	312	1411	3373	4385	3347
	arp 2	2000	2004	2000	2220	2112	2240	519	306	1353	3514	4433	3343
	arp 1	2000	2004	2000	2156	2112	2238	1383	58	638	2618	4413	3343

Table A.13: Cycle time in μs during attacks against Wago devices at 1 Mb/s

Device	Attack	Mean Pre	Mean Att	Mean Post	Median Pre	Median Att	Median Post	Min Pre	Min Att	Min Post	Max Pre	Max Att	Max Post
750-889 (1)	arp 4	2000	2002	2000	2034	2224	2140	1403	588	1481	2983	3686	2516
	arp 3	2000	2001	2000	1859	2227	2001	1321	543	1481	2678	3661	2531
	arp 2	2000	2004	2000	2002	2179	1810	1302	174	896	3293	4617	2518
	arp 1	2000	2004	2000	2026	2199	1826	1384	588	1115	2608	4585	2669
750-8100 (2)	arp 4	9998	9999	9998	9823	10259	9808	7934	9510	6628	10539	10514	10538
	arp 3	9999	9999	9997	10270	9752	9798	9515	9445	2508	10557	10516	10565
	arp 2	10002	10116	9999	10266	10333	10258	9523	9522	9520	30375	30404	10583
	arp 1	10001	10135	9998	9806	10331	9756	370	9567	6016	30432	30418	10560
750-880 (3)	arp 4	1998	2037	1997	1701	2363	1695	544	145	576	3419	4515	3423
	arp 3	1998	2036	1997	1714	2363	1695	582	581	425	3442	4491	3437
	arp 2	1998	2038	1998	1704	2311	1695	580	579	290	3386	5568	3414
	arp 1	1998	2041	1999	1711	2313	1747	576	584	576	3388	5434	3411
750-831 (4)	arp 4	2000	2003	2000	2009	2205	1993	812	525	1130	3412	4410	3364
	arp 3	2000	2003	2000	2192	2215	1773	1400	430	748	2631	4447	3403
	arp 2	2000	2004	2000	1828	2170	2226	23	417	1425	3341	4719	2958
	arp 1	2000	2004	2000	2166	2140	2215	1162	454	589	3370	4641	3439

Table A.14: Cycle time in μs during attacks against Wago devices at 8 Mb/s

Device	Attack	Mean Pre	Mean Att	Mean Post	Median Pre	Median Att	Median Post	Min Pre	Min Att	Min Post	Max Pre	Max Att	Max Post
750-889 (1)	arp 4	2000	2111	2000	1867	2362	1853	75	531	236	2612	8139	2533
	arp 3	2000	2111	2000	1916	2362	2066	1295	535	1489	2707	8923	2523
	arp 2	2000	2272	2000	2197	1690	2276	1360	343	1476	2608	107437	2538
	arp 1	2000	2278	2000	2195	1676	1756	1297	287	271	2716	92413	2590
750-8100 (2)	arp 4	9999	11163	9997	10305	10344	9764	9565	4078	85	10508	30475	10486
	arp 3	10002	11296	9999	10028	10344	10304	9546	9539	9562	30380	30493	10489
	arp 2	9999	11488	9999	10319	10351	9747	9550	2775	8522	10473	30476	10450
	arp 1	9999	11424	9998	10017	10347	9726	9534	5075	6532	10515	49692	10495
750-880 (3)	arp 4	1999	2116	2000	1709	2313	2299	576	410	580	3420	9521	3447
	arp 3	1997	2115	1998	1702	2343	1707	582	451	575	3380	9676	3414
	arp 2	1998	2267	1997	1702	1695	1730	580	226	579	3418	90386	3480
	arp 1	1998	2261	1996	1700	1693	1696	579	253	580	3385	142657	3449
750-831 (4)	arp 4	2000	2137	2000	2264	2334	2199	1436	329	1432	3346	9737	3403
	arp 3	2000	2139	2000	2092	2334	1994	1347	328	1348	2648	9769	3382
	arp 2	2000	2276	2000	2222	2107	2009	627	262	1441	3432	114800	3345
	arp 1	2000	2269	2000	2144	1719	1815	305	352	530	3398	100319	3335

Table A.15: Cycle time in μs during attacks against Wago devices at 16 Mb/s

Device	Attack	Mean Pre	Mean Att	Mean Post	Median Pre	Median Att	Median Post	Min Pre	Min Att	Min Post	Max Pre	Max Att	Max Post
750-889 (1)	arp 4	2000	2826	2000	2238	2288	1782	1493	555	1152	2519	493534	2602
	arp 3	2000	2791	2000	2161	2330	2162	1451	346	1480	2548	358603	2533
	arp 2	2000	2821	2000	1819	2302	1802	869	560	756	2601	479991	2659
	arp 1	2000	2822	2000	2169	2322	2244	1349	606	1413	2652	497548	2601
750-8100 (2)	arp 4	9998	9999	9998	9823	10259	9808	7934	9510	6628	10539	10514	10538
	arp 3	9999	9999	9997	10270	9752	9798	9515	9445	2508	10557	10516	10565
	arp 2	10002	10116	9999	10266	10333	10258	9523	9522	9520	30375	30404	10583
	arp 1	10001	10135	9998	9806	10331	9756	370	9567	6016	30432	30418	10560
750-880 (3)	arp 4	1999	2775	1997	1735	1693	1698	61	453	580	3448	360654	3382
	arp 3	1997	2785	1996	1728	1695	1726	577	566	578	3386	453659	3488
	arp 2	1996	2806	1998	1728	1692	1725	579	561	580	3429	384598	3386
	arp 1	1998	2833	1998	1735	1694	1732	580	567	582	3424	437690	3383
750-831 (4)	arp 4	2000	2798	2000	2218	1740	2203	1343	541	1387	2681	389171	3093
	arp 3	2000	2811	2000	2216	1733	2170	1399	564	133	3369	424686	3385
	arp 2	2000	2818	2000	2195	1731	2200	442	569	275	2647	451619	3389
	arp 1	2000	2850	2000	2006	2264	2060	1359	394	723	3337	511724	3285

A.3 Schematic of the BeagleBone Measurement Printed Circuit Board

For the IO signal measurements of the PLCs, a 24 V logic analyzer with continuous monitoring was developed. The Programmable Realtime Units (PRUs) of a BeagleBone device were used for the measurement task, which are only capable to handle 3.3 V as inputs. Due to this, a level shifter PCB from 24 V to 3.3 V was developed (see Figure A.1).

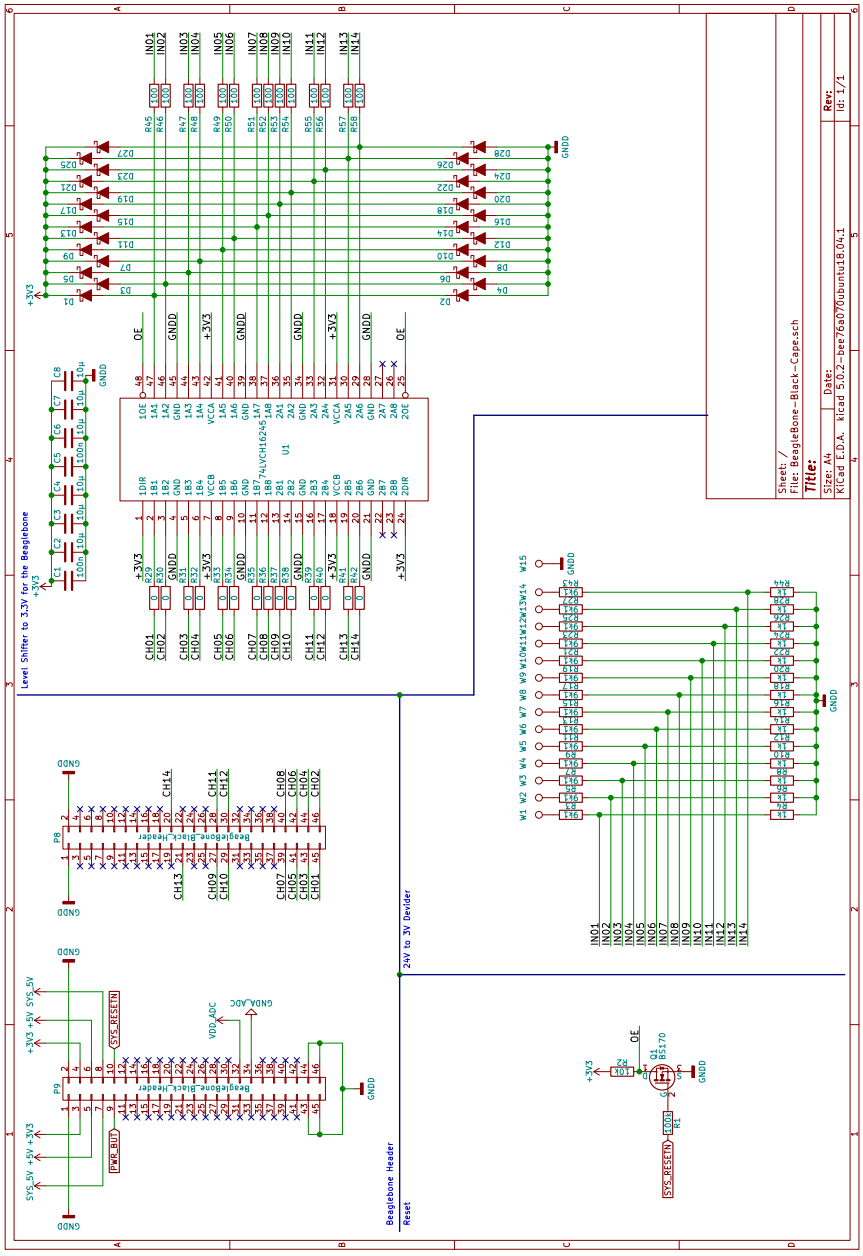


Figure A.1: Schematics of the developed measurement adapter for BeagleLogic.

A.4 Schematic of the Dual Microcontroller Unit Printed Circuit Board

To implement a robust PLC architecture, a dual MCU setup was developed. The baseboard is a NUCLEO-F767ZI development board from STM, which handles the network communication. A custom PCB with a STM32F030 controls IO operations. The schematics of the own developed PCB can be seen Figure A.2.

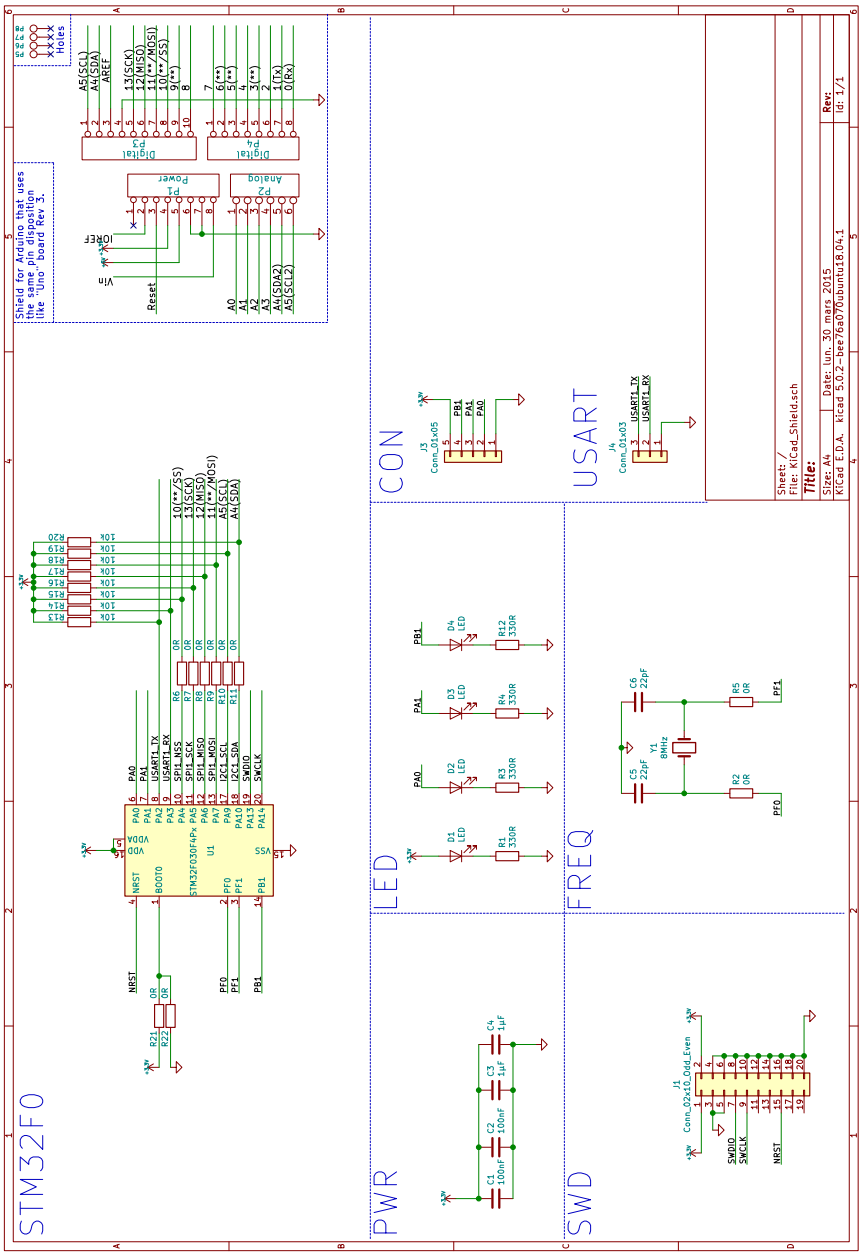


Figure A.2: Schematics of the developed dual MCU extension board.

A.5 Schematic of the Remote Input/Output Printed Circuit Board

As open components were required for this work, a remote-controlled IO device was developed on the basis of a NUCLEO-F767ZI development board. The schematic of the Remote IO extension board is shown Figure A.3.

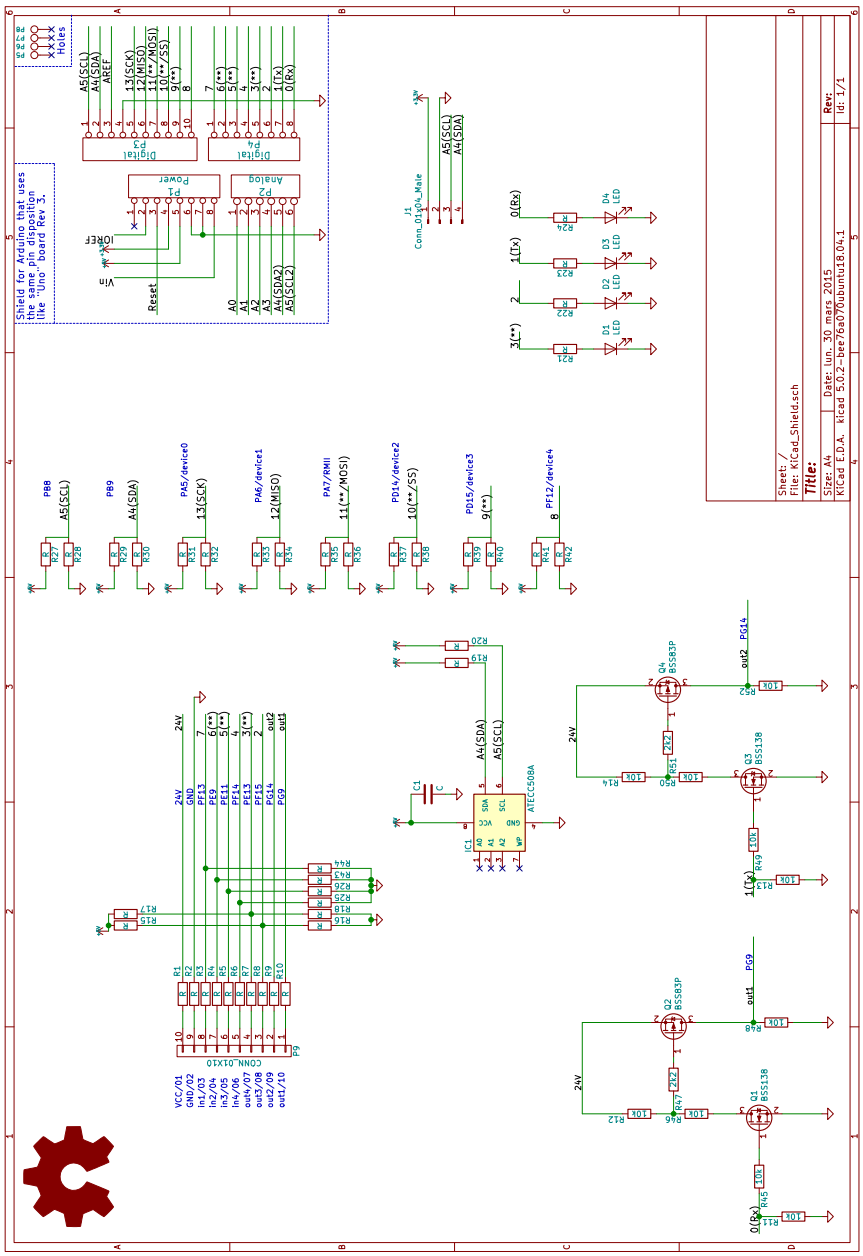


Figure A.3: Schematic of the Remote IO extension board.

Index