Information Security for Industrial Applications

- Detection of Anomalous Values in Industrial Automation Technology Infrastructures -

vorgelegt von
M.Sc.
Christian Horn

von der Fakultät V - Verkehrs- und Maschinensysteme
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr.-Ing. Roland Jochem
Gutachter:	Prof. Dr.-Ing. Jörg Krüger
Gutachter:	Prof. Dr. Jean-Pierre Seifert
Gutachter:	Prof. Dr. Michael Meier

Tag der wissenschaftlichen Aussprache: 23. Mai 2019

Berlin, 2019

Kurzfassung

Die Gesellschaft, in der wir heute leben, stellt wesentliche Anforderungen an die Versorgungssicherheit der Menschen, speziell in großen urbanen Metropolen. Zeitgleich wird versucht, diesen Anforderungen mithilfe der technologischen Entwicklung, insbesondere in der Automatisierungstechnik, beizukommen. Die Implementierung von zahlreichen Systemen und Kommunikationsverbindungen zur intelligenten Steuerung von Maschinen und Anlagen in Domänen wie kritischen Infrastrukturen, Warenproduktion, Transport oder Zuhause schreitet dabei zunehmend voran. Diese Cyber-Infrastruktur als Sekundärinfrastruktur hat einen hohen Grad an Komplexität und Automatisierung erreicht und die Abhängigkeit der Primärinfrastrukturen von dieser wächst stetig. Die Absicherung dieser Infrastrukturen, insbesondere in sicherheitskritischen Bereichen, stellt eine der zentralen Herausforderungen für die Betreiber dar.

Angriffe auf diese Cyber-Infrastrukturen in sicherheitskritischen Bereichen wie Stuxnet oder Duqu demonstrieren die prinzipielle Verwundbarkeit selbst gekapselter Systeme. Die zunehmende Vernetzung mit dem Internet, etwa über kabellose Technologien auf der Feldebene, erhöht die Verwundbarkeit der Systeme weiter. Dies bezieht sich sowohl auf die einzelnen Schichten der IT-Architektur und die verbaute Steuerungstechnik, als auch auf die organisationalen Strukturen und Arbeitsprozesse sowie die Sicherheitstechnik, die die Anlagen vor unautorisiertem Zugang physisch schützen soll. Die zunehmende Bedrohung der Systeme durch die tendenziell steigende Ausweitung der Angriffsflächen erfordert neue Verfahren zur Detektion von Angriffen, um effizient und kontrolliert reagieren zu können.

Diese Arbeit analysiert den aktuellen Stand der Grundprinzipien, Bedrohungen, Angriffsmodelle und Gegenmaßnahmen für Automatisierungsinfrastrukturen. Daraus ergibt sich, dass insbesondere Gegenmaßnahmen zur Erkennung von Anomalien notwendig sind und verwandte Arbeiten nicht mit Blick auf Anforderungen aus dem praktischen Anwendungskontext entwickelt wurden. Daher wurde dem Bedarf der Betreiber der Entwicklung einer **Methodik** zur Ableitung eines Detektionskonzepts entsprochen. Diese Methodik wird auf der Grundlage von Anforderungen und Daten aus elf realen Anwendungsfällen, vier kritischen Infrastrukturen, sechs Produktionsunternehmen und einem akademischen Beispiel, entwickelt. Das resultierende verallgemeinerte **Konzept** kann als Vorlage in Verbindung mit der Methodik verwendet werden, um anwendungsspezifische Konzepte für neue Anwendungsfälle wesentlich schneller zu generieren. Eine prototypische Implementierung wird anhand von Anforderungen und Angriffsszenarien aus der Praxis bewertet. Das dafür zur Verfügung stehende realitätsgetreue **Testfeld** arbeitet dabei mit realen Datenerfassungen aus dem jeweiligen An-

wendungsszenario. Es wird gezeigt, dass eine **Entscheidungsfusion** verschiedener **Detektionsdienste und Datenquellen** eine bessere Erkennungsleistung hat als individuelle Lösungen. Darüber hinaus führen Analyse, Implementierung und Bewertung zu mehreren neuen Erkenntnissen, die wiederum zu neuen Forschungsfragen führen, welche von nachfolgenden Forschungsarbeiten beantwortet werden könnten.

Zuletzt wurde bei den Experimenten auch ein neuer Angriffsvektor gefunden: **der schnellste gewinnt**. Dieser Angriff erfordert ein gut berechnetes Timing in Verbindung mit dem Zugriff auf das Subnetz der Speicherprogrammierbaren Steuerung.

Abstract

The society that we are living in today makes essential demands on the security of supply for people, especially in large urban metropolitan areas. At the same time, attempts are being made to meet these requirements with the help of technological developments, particularly in automation technology. The implementation of information and communication systems for smart control of machines and plants in domains such as critical infrastructures, goods production, transport or home is making increasing progress. This cyber infrastructure as a secondary infrastructure has reached a high degree of complexity and automation and the dependence of primary infrastructures on it is constantly growing. Securing these infrastructures, especially in security-critical areas, is one of the central challenges for operators.

Attacks on these cyber infrastructures in security-critical areas such as Stuxnet or Duqu demonstrated the fundamental vulnerability of even encapsulated systems. The increasing connection to the Internet, for example via wireless technologies at the field level, further increases the vulnerability of these systems. This applies not only to the individual layers of the IT architecture and the control technology installed, but also to organizational structures and work processes as well as the security technology, which is intended to physically protect systems from unauthorized access. This increasing threat situation to systems due to increasing expansion of attack surfaces requires new methods for detection of attacks in order to be able to react efficiently and in a controlled manner.

This work analyses the current state of basic principles, threats, attack models and countermeasures for automation infrastructures. It is revealed that measures are particularly necessary to detect anomalies and that related work has not been developed with practical application context requirements in mind. Therefore, the need of operators to develop a **methodology** to derive a detection concept was addressed. This methodology is developed on basis of requirements and data from eleven real use cases, four critical infrastructures, six production companies and one academic example. The resulting generalized **concept** can be used as a template in conjunction with the methodology to generate application-specific concepts for new use cases much faster. A prototypical implementation is evaluated based on practical requirements and attack scenarios. The realistic **test environment** utilized in this works is parametrized with real data from the respective application scenario. It is shown that a **decision fusion** of different **detection services and data sources** has a better detection performance than individual solutions. Furthermore, analysis, implementation and evaluation lead to several new findings that lead to new research questions that could be answered by further work.

Last, but not least, a new attack vector was found during experiments: **the fastest wins**. This attack requires a well calculated timing in conjunction with access to the subnet of a Programmable Logic Controller.

Contents

Glossary

AOC Attacker Operating Characteristics. 36

APT Advanced Persistent Threat. 4, 19, 141

ARP Address Resolution Protocol. 38

ATI Automation Technology Infrastructure. xi, xiii, 7–13, 16, 17, 19, 25–27, 31, 33–35, 45, 47–49, 55, 61, 62, 65, 69–72, 74, 76, 79, 80, 85–89, 91, 97, 105, 117, 118, 121, 127, 132–135, 137, 146, 147

CI Critical Infrastructure. xi, 1–4, 11, 12, 62, 69, 72, 135, 139, 141

CIP Common Industrial Protocol. 42, 44

CPN Colored Petri Net. viii, xiii, 121, 132, 144, 145

CPU Central Processing Unit. 67, 80, 87, 91, 116, 118, 119, 121, 122

DAG Directed Acyclic Graph. 51, 54, 146–148

DEM Discrete Event Model. 121, 122

DFA Deterministic Finite Automata. 38, 86, 95, 118, 130

DNN Deep Neural Network. 43, 120, 121

DoS Denial of Service. xii, 40, 77, 102, 109, 111, 122, 128, 129

ERP Enterprise Resource Planning. 71

EtherNet/IP EtherNet Industrial Protocol. 43

GC Granger-Causality. 52

GSM Global System for Mobile communications. 79

HIDS Host based Intrusion Detection System. 13, 29, 30, 37, 43, 44, 87, 93, 96, 98, 131, 137

HIL Hardware-in-the-Loop. 88, 106

HMI Human-Machine-Interface. xii, 2, 4, 38, 94, 106, 109–111, 120, 122, 128

ICS Industrial Control System. xi, 2–5, 13, 16, 17, 25–27, 31–33, 35, 42, 76, 89, 96, 133

ICT Information and Communication Technology. xi, 2, 6, 7

IDS Intrusion Detection System. xi, 29–31, 34, 35, 37–39, 41–44, 66, 80, 90, 102, 129, 136

IL Instruction List. 100

IoT Internet of Things. 8, 9, 72, 77

IPS Intrusion Prevention System. 43

ISMS Information Security Management System. 22, 57

IT Information Technology. 2, 5, 7, 9, 11, 16, 22, 26, 31, 33, 35, 72, 134

LQG Linear Quadratic Gaussian. 39

MES Manufacturing Execution System. 71

MITM Man-In-The-Middle. 37

NDA Non-Disclosure Agreement. 70, 139

NIDS Network based Intrusion Detection System. 13, 29, 42, 44, 51, 85

OCSVM One-Class Support Vector Machine. 40, 42, 43, 121

PCBAD Process Causality based Anomaly Detection. viii, xii, 120, 121, 137, 146, 147, 149

PID Proportional Integral Derivative. 39

PLC Programmable Logic Controller. 6, 13, 23, 31, 36, 38–43, 47, 64, 66, 67, 71, 78, 80, 85–89, 96, 98, 100–102, 105, 106, 109, 111, 113–115, 118–122, 125, 129, 131–133, 135, 137, 143–145, 147

RAM Random Access Memory. 13, 87, 116

ROC Receiver Operating Characteristics. 36

SCADA Supervisory Control and Data Acquisition. 4, 25–27, 31, 33, 35–37, 40, 42, 80, 86, 106

SPOF Single Point of Failure. 23

SSH Secure Shell. 91

TechMap Technological Map. vii, xi, 9, 12, 58–61, 63–65, 69, 72, 85, 134, 139, 140

ToE Theory of Everything. 51

vSoftPLC virtual Software Programmable Logic Controller. 100, 114, 121, 122, 132

WAN Wide Area Network. 3, 4, 8, 11, 20, 86

WLAN Wireless Local Area Network. 20

List of Figures

List of Tables

1. Introduction

Over the last centuries population has increased and formed urbanizations. The United Nations report on urbanization [United Nations, 2014] states:

> *Globally, more people live in urban areas than in rural areas, with 54 per cent of the world's population residing in urban areas in 2014. In 1950, 30 per cent of the world's population was urban, and by 2050, 66 per cent of the world's population is projected to be urban.*

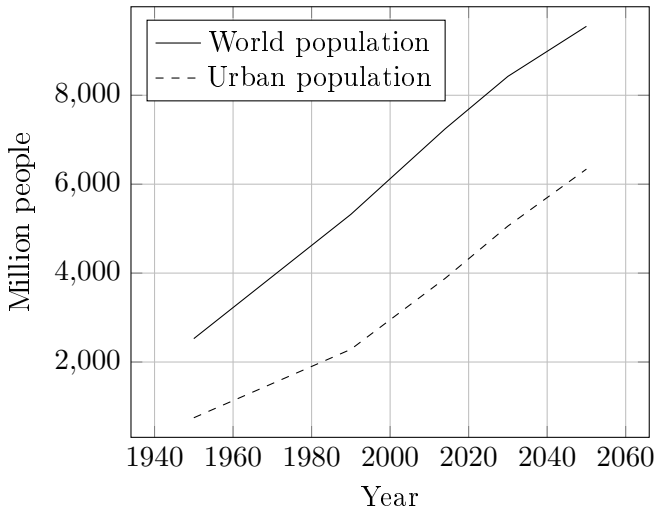

Figure 1.1.: Urbanization of world population

Figure 1.1 shows respective numbers for the world population. The report estimates a percentage of urbanized population for Europe by 2050 over 80 per cent. This increasing urbanization possibly leads to huge conurbations. Within these huge metropolitan areas completely new demands for supply structures due to a much higher population density and therefore increased consumption rate will arise. This also increases the complexity level and the requirements regarding a robust and secure supply within these infrastructures.

Future requirements can be satisfied by implementing automation, since it offers a huge potential for fast, robust, flexible and cost effective structures [Adamczyk et al., 2015]. For example several facilities can be monitored and controlled by a few people using automation and fast networking technology. These systems are nowadays already implemented in most supply structures, so called Critical Infrastructures (CIs), such as water and wastewater, energy (e.g. electricity, gas and oil), transportation, food and beverage. Furthermore these systems can be found in industrial production environments for manufacturing all kinds of mer-

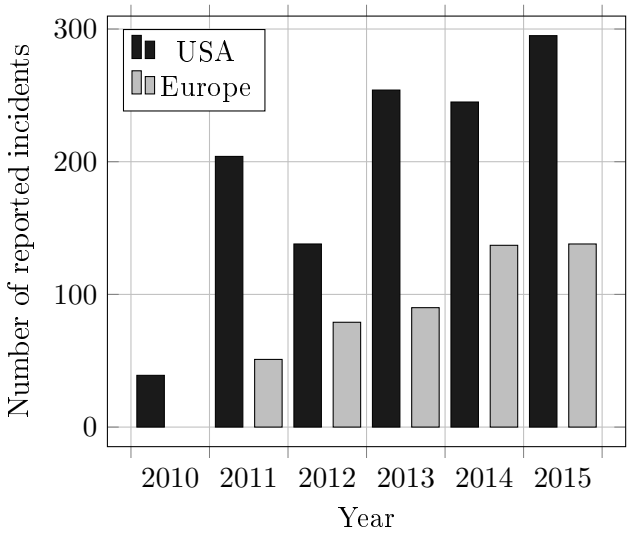

Figure 1.2.: Reported cyber security incidents in CIs

chandise (e.g. aircraft, household appliances or automobiles). Information and Communication Technology (ICT) forms an infrastructure of a second degree inside these application domains. Its importance is growing in the light of an increasing automation of the primary processes due to economic pressure and ease of use for the end-user. Protection of these infrastructures, particularly in safety-critical areas, represents a key challenge to society.

Up-to-date ICT-infrastructures already possess a high level of complexity and automation. The rise of industrial information and automation technology in the last decades offers great potential for optimization of efficiency and expenses. Further increasing economic pressure possibly leads to higher demands for implementing these systems in new areas, where they used to be unthinkable before. These cost-effective structures have already been implemented in most of the CIs. On the one hand this leads to more efficient structures and decreased expenses, but on the other hand this increases the risk of failures through attacks, malfunction or misuse. A single failure can hereby lead to a serious cascading effect [Becker et al., 2012]. Therefore the robustness and security of ICT Infrastructures are the foundation for an uninterrupted supply as well as for a sustainable and robust value creation chain. This applies particularly to critical systems in automation technology such as Industrial Control Systems (ICSs). Furthermore current research towards an Internet of Things (IoT), smart production technologies or intuitive HMIs pushes further the adoption of concepts and approaches from computer science to automation environments. The combination of modern machinery (e.g water pumps or even industrial robots), fast networking technologies based on widely-used standards (e.g. Ethernet), central processing capacities (e.g. cloud-infrastructures) with smart algorithms and sensor technologies form a new architecture. The implementation of new Information Technology (IT) concepts triggers massive changes and the ongoing exposure of ICSs to the Internet raises additional attack surfaces as shown in

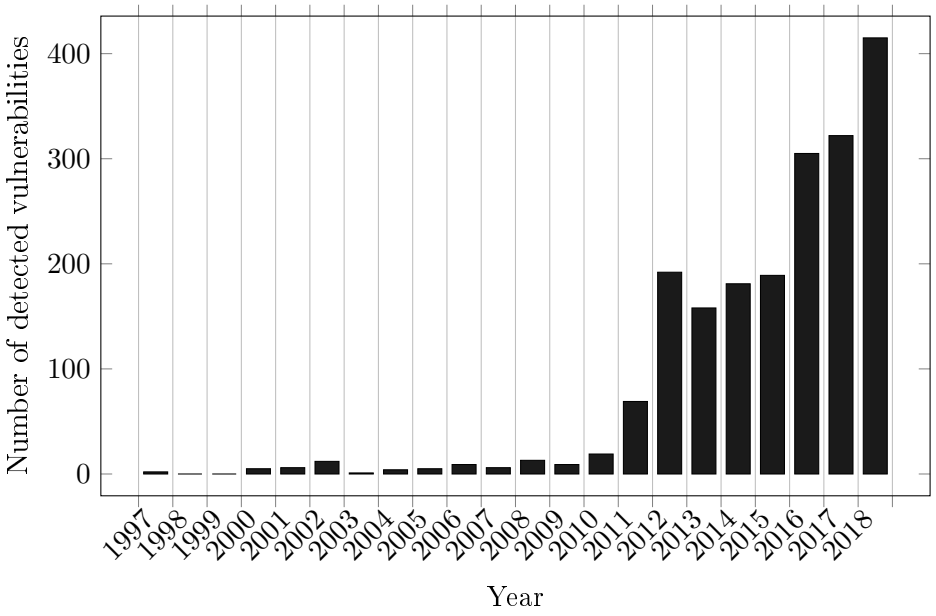

Figure 1.3.: Detected vulnerabilities in ICSs

[Radvanovsky, 2014] and [Andreeva et al., 2016a].

This increased attack surface and threat situation is also reflected in figure 1.2, which shows the numbers of reported cyber security incidents in CIs from 2011 to 2016 in USA[1] and Europe[2]. A more general business survey from 2015 in the UK [PwC, 2015] showed that *"90% of large organizations reported that they had suffered a security breach, up from 81% in 2014"*, while 74% of small businesses had suffered a security breach, compared to 60% one year earlier. In contrast to the numbers in figure 1.2 this indicates that the number of incidents not reported is quite higher and probably more valid numbers will arise with new legislations forcing operators of CIs to report incidents. Also an IBM Managed Security Services Research Report [McMillen and Li, 2017] states that attacks targeting industrial control systems (ICS) increased since 2013 in numbers.

Aggravating this situation, the ICSs itself were not developed with security in mind and different security challenges and potential attack vectors have been found already in the basic concepts [Johnson, 2010] [Igure et al., 2006]. Also the control systems of industrial robots gain more interest to security research due to the connection to Wide Area Networks (WANs) [Maggi et al., 2017]. These security flaws by design can be exploited by different individuals, groups or organizations with divergent interests ranging from economic benefits up to cyber-warfare [Nicholson et al., 2012]. In Fig. 1.3 [Andreeva et al., 2016b] the discovered vulnerabilities in ICSs are displayed for the past two decades. Since the discovery of the computer worm STUXNET [Falliere et al., 2011] in 2010 and

[1] [NCCIC/ICS-CERT, 2013],[NCCIC/ICS-CERT, 2014],[NCCIC/ICS-CERT, 2015], [NCCIC/ICS-CERT, 2016]

[2] [ENISA, 2013],[ENISA, 2014], [ENISA, 2015], [ENISA, 2016]

the subsequent Advanced Persistent Threats (APTs), such as FLAME [Virvilis and Gritzalis, 2013] and DUQU [MacKinnon et al., 2013] researchers focus more and more on security vulnerabilities in industrial control systems. This is also reflected in figure 1.3. In the past a common approach of ICS vendors to achieve the illusion of a secure environment was using a concept known as *"security by obscurity"* [Mercuri and Neumann, 2003]. It means that vendor keeps knowledge about functionality of their systems and products confidential. At that time people apparently believed that nobody is able to reverse engineer systems without the original blueprints. Nowadays by using standardized components within products and the availability of information through the Internet this approach becomes evidently inappropriate. Furthermore a vulnerability can be found and exploited easily because of an increasing connection of these systems to WANs, as shown by the aforementioned numbers and figures.

An analysis of discovered malware did not only show that these systems do have security holes, but it also has a high maturity level and therefore was likely developed by specialists [Symantec, 2011]. Other prominent examples for incidents in ICS are:

2013 Probably politically motivated, the on Bowman Dam, Rye, New York was attacked [U.S. Attorney's Office, 2016]. With unauthorized access to the SCADA systems the attacker gained access to information and control of water levels, temperature and the sluice gate. Fortunately the sluice gate could not be controlled from the Supervisory Control and Data Acquisition (SCADA) system at this time since it had been manually disconnected for maintenance.

2014 A German steel mill was attacked [BSI, 2014]. The intruders used spear-phishing methods to obtain access to the office network of the company. From that point the attackers exploited the connection of the IT-infrastructure to achieve access to the production network. The attackers caused the failure of several control modules, which led to a breakdown of the production line. The Remote Access Trojan (RAT) HAVEX [F-Secure, 2014] was also discovered in 2014 and was developed to spy on industrial plants and CIs [Symantec, 2014].

2015 Attackers force a power outage in Ukraine that impacted a large area that included the regional capital of Ivano-Frankivsk leaving 225.000 people without energy [Auchard and Finkle, 2016]. The APT-malware Blackenergy3[3] [Shamir, 2016] was used for Spear-Phishing. In a coordinated attack command and control systems in four different supervision facilities were targeted to switch off electric power transformation substations.

2016 A HMI system in the nuclear power plant Gundremmingen (Germany) was infected via USB by the computer worms Win32.Conficker[4] and Win32.-Ramnit[5] [BSI, 2016]. The computer system was running data visualization software of machinery for moving nuclear fuel rods. The infection

[3] BlackEnergy1 was discovered in 2007 and Blackenergy2 in 2010.
[4] Win32.Conficker was first discovered in 2008.
[5] Win32.Ramnit was first discovered in 2010.

did not threaten the facility's operations because of isolation from the Internet. Furthermore a sophisticated malware (SFG) was discovered that year which infected at least one European energy company. [Landry and Shamir, 2016]: *"The malware is most likely a dropper tool being used to gain access to carefully targeted network users, which is then used either to introduce the payload, which could either work to extract data or insert the malware to potentially shut down an energy grid."* At then end of 2016 another blackout in Ukraine (Kiev) happened and is believed to be the result of a cyber-attack using the malware CrashOverride/Industroyer [Lipovsky et al., 2017].

2017 The Ransomware Win32.WannaCrypt aka WannaCry affected systems worldwide including ICSs [BSI, 2017]. Known affected companies include Renault (several factories), Nissan (at least one production line), Honda (a whole plant was halted in Sayama, Japan), Gas Natural (spanish gas supplier), Iberdrola (spanish electric power company) [Kaspersky Lab, 2017] and Deutsche Bahn (failing displays at train stations). Also several Hospitals in the UK were affected with effects up to suspended surgery due to the lack of information. Following WannaCry the Ransomware ExPetr (Petya) spread and *50% of the companies attacked were from manufacturing and oil and gas industries* [Kaspersky Lab, 2017].

2018 At the beginning of the year the vulnerabilities Meltdown [Lipp et al., 2018] and Spectre [Kocher et al., 2018] for most recent processor generations of Intel, AMD, ARM, possibly NVIDIA have been published. This will affect all ICS using these processor generations.

Apart from these incidents, operating companies of complex industrial Information Technology (IT)-infrastructures in general face novel threats. Examples for these novel threats are Return-Oriented-Programming (ROP) [Buchanan et al., 2008], where the malware does not even need to be transferred to the targeted system. Instead the sequential arrangement of the software running on the targeted computer is changed in a way that leads to the desired behavior. Likewise there are side-channel attacks [Kocher, 1996], which extract data from a system undetected by exploiting previously unknown connections. Furthermore do modern SoCs have a hidden secondary platform with a separate processor, but complete access to the central memory of the system. These secondary platforms can be used to permanently store and run malware [Stewin and Seifert, 2010]. These threats can not be detected by conventional methods, which is also reflected by the recent discovery of the vulnerabilities Meltdown [Lipp et al., 2018] and Spectre [Kocher et al., 2018].

The aforementioned malicious software has in common that it is highly specialized and therefore its existence was detected late. Common methods of IT-infrastructure protection like black listing reach their limits with this highly specialized malware. At the same time the vulnerability of those systems is growing simultaneously with the extension of the network structures and the increasing usage of network technologies. This leads to much more possible weak spots and with the constantly growing number of remote stations, an

attack from vast distance to a target is possible. Furthermore the effort for attackers to perform a complex attack is decreasing constantly. There are several freely available software libraries like the Metasploit-Framework [Rapid7, 2014], which can be used to exploit a broad variety of weak spots for various systems without extensive system or programming knowledge. On one hand this is highly relevant for operating companies and developers to test their systems (penetration testing), but on the other hand it also provides unwanted assistance to malicious attackers. For example the process of locating a target and acquiring information with tools like the Shodan Computer Search Engine [Matherly, 2009] became much easier than it was a couple of years ago. With the help of freely available software libraries like Snap7 [Nardella, 2015] or ADS [Beckhoff, 2017], that can be used to communicate with Programmable Logic Controllers (PLCs), it is possible to alter the code or data on a PLC easily. The combination of these tools can lead to complex and severe attacks, as shown in [Klick et al., 2014]. Here the source code running on tht PLC is altered to achieve the own desired behavior. Another possible scenario is the alteration of important process-values by an attacker, which remains undetected too. These changes in deviating process-values cannot be detected manually by the operators, since they have to rely on the sensor values. Attackers can use the boundaries of automation layers (ref. to figure 1.4) to hide and they probably remain undetected.

1.1. Motivation

Automation technology is the heart of nowadays supply chain, since it offers potential for implementation of flexible and cost effective structures [Adamczyk et al., 2015]. Since the demand for customized and still affordable products is rising, these potentials can be well utilized. The ICT used in this supply chain is structured in different levels. [ISO/IEC62264, 2013] describes a functional and hierarchical model for Enterprise-Control System Integration, where standard levels are described as follows:

- Level 4: defines all business-related activities to run the enterprise (e.g. business planning and logistics). Level 4 usually works in the time frame of months, weeks, and days.
- Level 3: defines the activities within production work-flow. (e.g. manufacturing operations and control). Level 3 usually works in the time frame of days, shifts, hours, minutes and seconds.
- Level 2: defines the activities of monitoring and controlling the physical processes. Level 2 usually works in time frames of hours, minutes, seconds, and less than seconds.
- Level 1: defines activities for measuring and controlling physical processes. Level 1 usually works within the time frame of seconds and faster.
- Level 0: defines the physical processes itself.

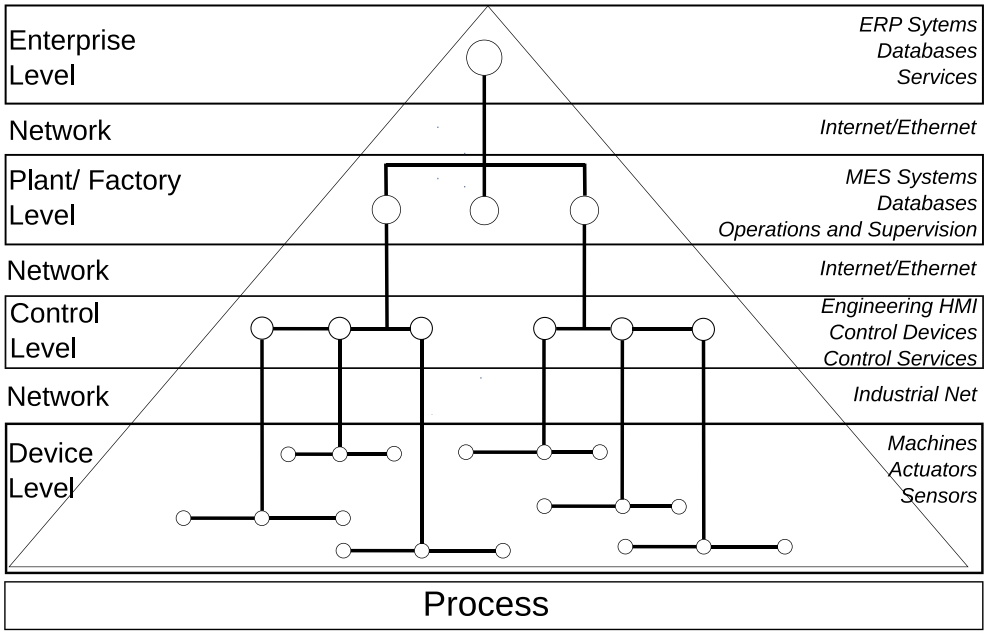

Figure 1.4.: Automation pyramid structure of industrial ICT

Figure 1.4 shows the so called automation pyramid structure. Each level contains different systems and networks, leaving the whole infrastructure for the enterprise complex and heterogeneous. Industrial networks from level 2 and below differ significantly from business and office networks (level 3 and above) especially concerning the primary function to control and monitor real-world actions and conditions. Other differences are *application domains, architecture, failure severity, Real-time requirements, determinism, data size, periodicity, temporal consistency and event order* [Galloway and Hancke, 2013].

Currently industrial networks (level 2 and below), referred to as **ATIs** throughout the thesis, are subject to fundamental changes, since the concept of cloud manufacturing was introduced by [Li et al., 2010]. Following that several concepts have been published for different applications and factory layers ([Tao et al., 2011], [Xu, 2012], [Wu et al., 2013], [Langmann and Meyer, 2014], [Breivold and Sandstrom, 2015], etc.). These changes are characterized by an increasing integration of information and communication concepts and technologies into ATIs. Well-established companies as well as startup companies in IT and automation technology are focusing their products and services on support and development of industrial applications and they are also currently developing specialized cloud-platforms [Esler, 2017] [KUKA, 2018]. This development focuses on processes of Enterprise Resource Planning (ERP) like order management, resource management, logistics management or reporting. But also more and more potentials of cloud-based data acquisition for comprehensive data analysis and visualization open up and corresponding products and services arise as well. The potential of cloud-based industrial control services is also recognized and subject to developments [Vick et al., 2015]. A platform to support virtual controllers has to deal with real-time capabilities for the control

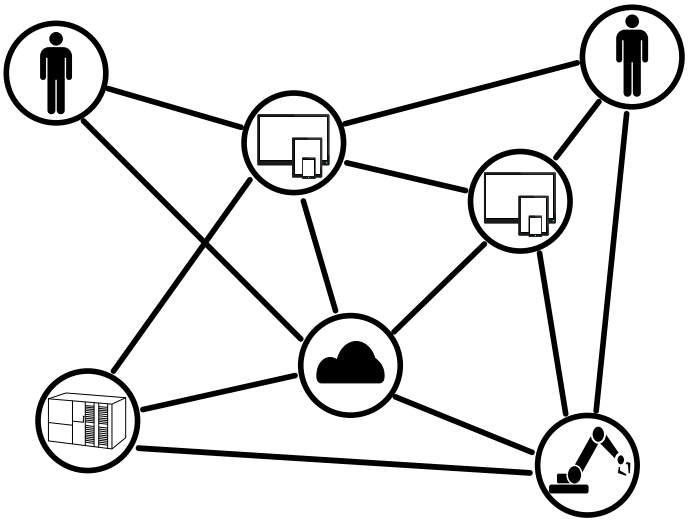

Figure 1.5.: Internet of Things approach for automation

of industrial robots and machine tools. The application of virtualization technologies ensures further usage of existing industrial software. In figure 1.5 this networked structure is outlined. In addition to networking, the integration of consumer technologies (smart-phones, tablets, etc.) is also indicated.

The limiting factor for an extensive implementation of cloud platforms in production environments was located in missing standard interfaces between machinery, robots, production systems and the novel cloud infrastructures and services. Especially machinery without Ethernet-based communication interfaces cannot be integrated. New interfaces are subject to developments and could lead to an extensive usage of Internet of Things (IoT)-devices in ATIs as cloud-connectors. [Horn and Krüger, 2016a] show the feasibility in different connection scenarios and the results of experiments with realistic setups. Even the usage of out-of-the-box network solutions including even low-cost consumer hardware shows that already cycle timing requirements for industrial processes around 10 Milliseconds can be fulfilled from a private cloud. The measured delays for community or public cloud scenarios over the Internet imply a possible cycle time under 100 milliseconds. With further adaption and tuning it will be possible to port industrial processes to cloud infrastructures soon with stable timing demands of 1 ms.

These technologies, which are novel to the field of automation, meet security concepts that have been developed for the classic form of ATI. These include isolation and "security by obscurity" (security through incomprehensibility). In these times devices and systems were hard-wired directly with their controllers and these were only interlinked at specific nodes in local and isolated environments with special and non-open protocols. The connection of individual systems and controller devices to WANs such as the Internet was not taken into account. New requirements for information security concepts and tools arise in these new structures for ATIs, in particular since manipulation, technical

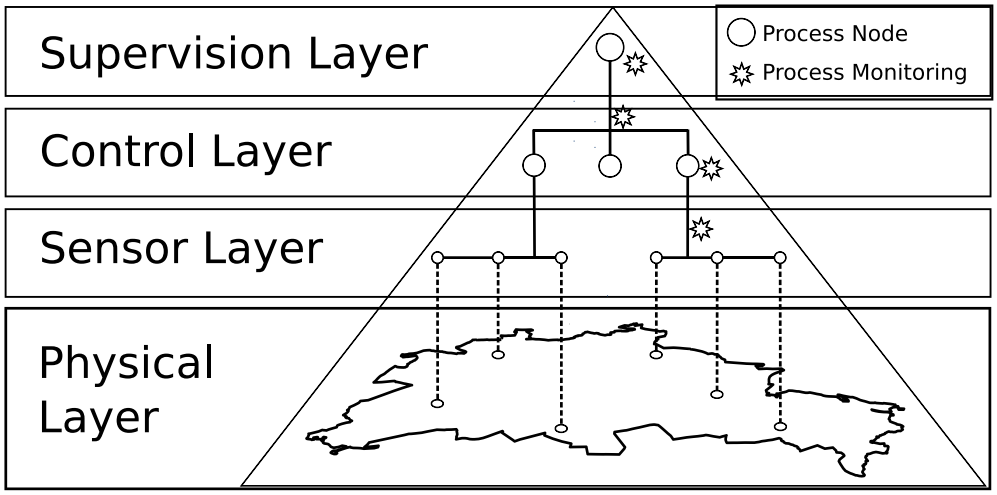

Figure 1.6.: Monitoring system as sketched in STEUERUNG

malfunctions or attacks have far greater effects than for a single office computer. Additionally, tools and security concepts for office networks cannot be transferred to a production environment without modifications, since these were developed for environments with completely different requirements and conditions. For example inadequate security regulations of an IT department could directly jeopardize the assets of an entire company, if just copied to the ATI. Furthermore conventional security tools (e.g. network and system monitoring programs) could disturb a particular process cycle time thus leading to safety or economical issues.

The progressing interconnection of systems, the connection to public networks and the use of IoT-devices and standard consumer technologies broadens the attack surface towards these processes. The temptation of increased cost-efficiency, functionality and ease-of-use fuels this development. New concepts and especially tools for securing Automation Technology Infrastructures (ATIs) are sorely needed.

1.2. Thesis outline

The basic idea for this research was developed in 2012 in context of the project proposal for the research project STEUERUNG within the sub-project *Process Security* [Horn and Krüger, 2013]. Already in the proposal a *Process-Fingerprint* was formulated which refers to a description of the process itself through mathematical statements based on process-relevant parameters and causalities. These parameters and causalities can be elaborated using Technological Map (TechMap)[6]. Within the live running process, the Process-Fingerprint can then be used to detect deviations from a normal behavior using several

[6]Descriptions of the whole infrastructure that contain all relevant technological components, their characteristics and relations to each other. (ref. to section 4.2.2)

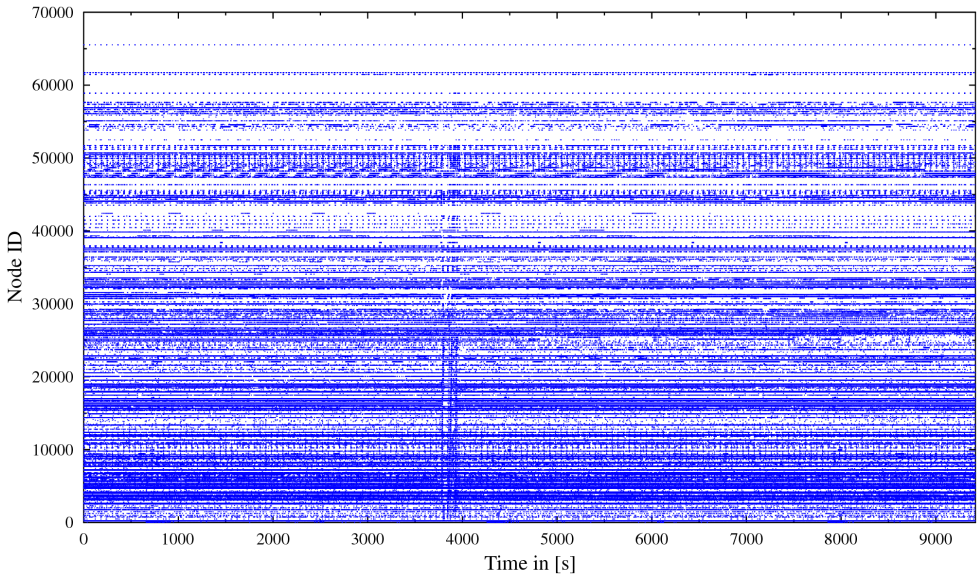

Figure 1.7.: Network activity of more than 65,000 communication nodes

on-line data captures at different critical points within the infrastructure. The intended monitoring system itself is sketched in figure 1.6.[7] Network traffic and process nodes should be monitored to capture data at different spots of the whole process. This data can then be cross-validated to reveal anomalies.

Within the research project STEUERUNG [Horn et al., 2014] three different application scenarios, gas, water and power distribution for a huge metropolitan area each containing a complex and distributed ATI were available. Within these, all sensor nodes (sensor layer) transmit their data to a local controller (control layer), which aggregates the information and sends it to a central control room (supervision layer), where human operators make decisions based on the current situation.[8] The control values get then transmitted back to the local controller. The most crucial aspect for operators is being able to trust the process values they get, but at this point the process values could have been manipulated at the local controllers, network infrastructure or sensors.

Within all application scenarios data was collected from different sources. Figure 1.7 shows an example containing $1,504,457$ network packets and respective communication frequencies from more than 65,000 sensor and controller nodes in the field arriving at the main operations tele-control station (supervision layer). Due to organizational challenges, as sketched in example 1.2.1 it was not possible to capture data at the local controllers (control layer) or sensors (sensor layer). This made clear that it is not possible to develop an all-purpose detection

[7] The structure differs in number and labels from the layers of the classic automation pyramid, as shown in figure 1.4, since the project only focused on these layers. The layers were chosen according to the levels 0,1,2 and 3, where 0 is the process, 1 is sensor/device, 2 is control and 3 is supervision/ enterprise.

[8] Automated decisions are also made, if human reaction is too slow for some cases, i.e. automatic load shedding in power distribution to protect the grid.

concept based on a *Process-Fingerprint* that fits all application scenarios. The intended detection of deviations from a normal behavior using several on-line data captures at different critical points within the infrastructure simply fails because of a missing possibility for on-line data captures at necessary points in the infrastructure. It is more important to develop a concept of a system that contains different modules and can be modified according to practical circumstances and available data.

Example 1.2.1. In a CI, the pipelines, sensors and possibly parts of control nodes belong to company A, while central control-nodes and supervision technology and personnel belong to company B. If company B changes its security policies and eventually uses new monitoring software it does not necessarily mean that company A applies also similar or compatible measures. The effectiveness of software used by company B is limited through the organizational boundary to company A.

Example 1.2.2. In a CI company A owns all infrastructure, from supervision layer to pipelines. The software running on servers was developed and licensed by company B, the software for the control nodes by company C. Company A does not have the authorization by license, the source code itself or personnel resources to perform changes to any of the software products. A new security software which needs interfaces to the products from company B and C requires the cooperation of all companies. Companies B and C eventually cooperate if more than one customer demands these security interfaces. The effectiveness of software used by company A is limited through the organizational boundaries to company B and C.

Examples 1.2.1 and 1.2.2 outline the organizational challenge in a simplified manner, real world scenarios are usually more complex and involve additionally organizational challenges between different departments within companies (i.e. automation versus IT department). The boundaries of organizational layers cannot be overcome with technological measures.

Following STEUERUNG two more research projects could be used to focus the development. Research project pICASSO [Vick et al., 2015], where novel cloud-based control concepts were developed, could especially be used to include future production architectures into the concept. Project RetroNet [Horn and Krüger, 2016c], where methods to connect cloud-based services to machinery with legacy communication interfaces were subject to research, contributed four more different ATI application scenarios from production. These showed additional requirements and methods that were developed before could be applied in real-world production scenarios.

Information from both projects was used to focus the development towards the control layer, since both research projects push the interconnection of (control-) systems of ATIs at this layer to cloud-architectures in WANs.

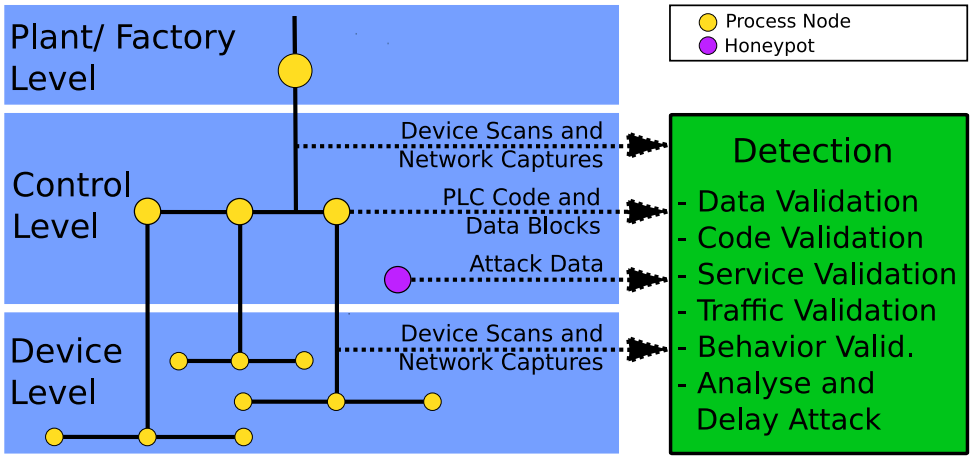

Figure 1.8.: Data sources and detection services

1.3. Research focus and contribution

This thesis aims to answer the research question "***How can a previously unknown attack on a distributed Industrial Information and Automation Technology Infrastructure be detected?***" and focuses on developing a concept to detect anomalous values in complex and distributed automation processes. According to [ISO/IEC62264, 2013] and figure 1.4 this includes Level 2: control and Level 1: devices.

Based on qualitative expert interviews [Mayring, 2010] as well as an analysis of the *TechMaps* the requirements of eleven different ATIs were identified. The ATIs include four distributed CIs: Gas, water (2x) and power distribution and six production use cases (huge, medium and small companies). Additionally another academic application scenario, where an industrial robot gets directly connected to the Internet, was examined. Future requirements as sketched in the project pICASSO as well as safeguarding of models and algorithms were included. Possible data sources were identified within the application scenarios and a methodology to develop a modular detection concept based on different services was introduced and a generalized concept derived. Some parts of the concept were developed to compensate the lack of access to the closed system platforms itself, since vendors like SIEMENS[9] explicitly prohibit that. The respective data sources and detection services are shown in figure 1.8 and include:

Process Values are the most crucial aspect in an ATI. Their correctness is essential for functionality and safety reasons, as several expert interviews indicated (ref. to appendix A.1). Furthermore no reliable method is currently available for operators to validate process values. They simply have to trust that everything is correct. A *Process Value Validation Service* is part of the concept that uses a novel approach based on process causality

[9] [SIEMENS, 2016]: *"Customer will not reverse engineer, decompile, translate, disassemble, or otherwise attempt to discover the source code of the Software.[..] Customer may not otherwise modify, alter, adapt, or merge the Software."*

models to verify the correctness of process values.

PLC Soure and Binary Code is available at two points as data source: the source form can be obtained directly from the programmer before integration and the binary form is available on the devices at all times. In the source form the code is human readable and simple diff-tools can detect and highlight modification [MacKenzie et al., 1997]. In binary form other approaches perform well to detect modification. A *Code Validation Service* is part of the concept to detect unauthorized modification in the code.

Host based Services are programs running on a process node itself like the interface services on a PLC using ISO on TCP at Port 102 [Rose and Cass, 1987]. Several tools exist to actively (nmap, [Lyon, 2009]) or passively (p0f, [Zalewski, 2016]) fingerprint a host concerning its services from a distant location. This is also referred to as OS fingerprinting. A *Service Validation Service* utilizes these technologies and is part of the concept. Depending on the possibilities of devices within the ATI, software agents can monitor features like processor load or Random Access Memory (RAM) consumption and report these to the service to enhance the detection (ref. to infrastructure monitoring tools like Nagios [Galstad, 2017]). For open platforms like Linux-based devices a Host based Intrusion Detection System (HIDS) like Open Source Tripwire [Tripwire-Community, 2017] can additionally be used.

Network Traffic is usually used by Network based Intrusion Detection Systems (NIDSs) like Snort [Roesch, 1999] and Bro [Paxson, 1998] as data source. In general two different approaches can be used: protocol analysis and deep packet inspection. A *Network Traffic Validation Service* is part of the concept and detects anomalies.

Behavior of Controller Nodes for closed internal platforms can be monitored using a concurrent simulation of the application software in a virtual environment or using a state space model of it. Furthermore the simulation can be run anywhere to achieve a decoupling of the security service from the monitored system. Several different technologies were evaluated for a *Behavior Validation Service*, that can process complex PLC Code (ref. to EN 61131-3:2014).

Honeypot systems are valuable data sources if they can attract attackers and capture their actions. For ICS simple systems [MushMush-Foundation, 2015] are available to mimic industrial communication protocols. More advanced attackers may not be fooled by these systems. The developments for the *Behavior Validation Service* can be used standalone as honeypot. Using this approach not only the communication protocols, but also a user scenario written in PLC Code can be used to attract advanced attackers and capture their actions. The attack can be analyzed and delayed.

Besides developing the concept itself and transferring different existing detection methods used in office networks for some services, a specific contribution

to increase the state of technology and science is made in the development and evaluation of methods to detect anomalous or malicious behavior in routines of automated processes by (a) concurrent simulation of process nodes [Horn and Krüger, 2014] and (b) using causal models on process data [Horn and Klein, 2017]. Furthermore the best-practices from international standards were adopted to form a methodology, which could be enhanced by a new tool called technological maps [Horn and Krüger, 2016b].

2. Current state of technology and science

"If I have seen further it is by standing on ye shoulders of Giants."
– Isaac Newton, Letter to Robert Hooke, 1675

This section provides some background to improve the readers' comprehension and it shows existing standards and methods for security specialists to secure their infrastructures (state of technology). Furthermore the current research towards new approaches and technologies (state of research) is outlined. The focus is on the threat situation and resulting research question previously asked in chapter 1.

2.1. Terms and basic principles of information security

Several different terms appear in conjunction with information security. To distinct between different terms the following definitions are given.

- Safety is a widely used term and is defined in [Oxford, 2004] as the *"[..] state of being safe and protected from danger or harm [..]"*. In automation technology and engineering the term can be used for safe interaction of two entities, e.g. man and machine. Several international standards have been established, e.g. [ISO12100, 2010].
- Privacy is defined in [Oxford, 2004] as *"[..] peace and quiet, lack of disturbance, freedom from interference [..]"* and its protection in terms of information refers to the protection of personal data against misuse. The right of each individual entity of self-determination of its own personal data is often implied.
- Security has several definitions in literature. [Oxford, 2004] refers to the word safety and further meanings like *"shielding"*, *"guarding"* or *"invulnerability"*. In [ISO/IEC27000, 2014] *"[..] information security involves the application and management of appropriate security measures that involves consideration of a wide range of threats, with the aim of ensuring sustained business success and continuity, and minimizing impacts of information security incidents. [..]"*.

In this thesis security of information and data refers to protection against unauthorized access, detection and defense of particular targeted attacks or more generally spoken the protection of the object from the environment.

The international standard [ISO/IEC27000, 2014] defines since its first edition in 2009 three basic principles of information security:

- Confidentiality
  "property that information is not made available or disclosed to unauthorized individuals, entities, or processes"
- Integrity
  "property of accuracy and completeness"
- Availability
  "property of being accessible and usable upon demand by an authorized entity"

[Cherdantseva and Hilton, 2013] extended these three major security goals, also known as CIA-triad, by analyzing recent security and system engineering literature. More major security goals were defined as follows and include

- Accountability
  "ability of a system to hold users responsible for their actions (e.g. misuse of information)"
- Auditability
  "ability of a system to conduct persistent, non-bypassable monitoring of all actions performed by humans or machines within the system"
- Authenticity/Trustworthiness
  "ability of a system to verify identity and establish trust in a third party and in information it provides"
- Non-repudiation
  "ability of a system to prove (with legal validity) occurrence/non-occurrence of an event or participation/non-participation of a party in an event"
- Privacy
  "A system should obey privacy legislation and it should enable individuals to control, where feasible, their personal information (user-involvement)"

Currently these eight goals form the basic principles of information security. A system that fulfills these principles is considered secure.

Especially for ATIs and ICSs reliability is most important. Associated principles are availability and integrity [Drias et al., 2015]. More strictly availability is necessary in real-time [Cardenas et al., 2009] [Cárdenas et al., 2011].

2.1.1. Threats

General threats to IT-Systems were discussed extensively in the past decades. For example [Whitman, 2003] names threats like: *deliberate software attacks, technical software failures or errors, acts of human error or failure, deliberate acts of espionage or trespass, deliberate acts of sabotage or vandalism, technical hardware failures or errors, deliberate acts of theft, forces of nature, compromises to intellectual property, QoS deviations from service providers, technolog-*

ical obsolescence, deliberate acts of information extortion. [Stouffer et al., 2011] later name two major categories: adversarial sources (hostile governments, terrorist groups, industrial spies, disgruntled employees, malicious intruders) and natural sources (system complexities, human errors and accidents, equipment failures and natural disasters). For the European electronic communications sector the European Union Agency for Network and Information Security identifies in [ENISA, 2013], [ENISA, 2014], [ENISA, 2015], [ENISA, 2016] four root cause categories for all reported incidents: *system failures, human errors, natural phenomena and malicious actions.*

For ATIs and especially ICSs the European Union Agency for Network and Information Security states [ENISA, 2011]:

> *"According to the respondents, the biggest technical challenges regarding ICS security are: legacy issues, ICS and ICT convergence issues (including common viruses, Stuxnet-like malware and increasing interest in hacking), practical difficulties in patching/vulnerability management, and unintentional human errors due to a lack of interest or understanding of ICS security issues."*

The Federal Office for Information Security in Germany ranks its *"Top 10 threats"* for ICS in 2016 [BSI, 2016a] as follows:

1. Social engineering and phishing
2. Infiltration of malware via removable media and external hardware
3. Malware infection via Internet and Intranet
4. Intrusion via remote access
5. Human error and sabotage
6. Control components connected to the Internet
7. Technical malfunctions and force majeure
8. Compromising of extranet and Cloud components
9. (D)DoS attacks
10. Compromising of smart-phones in the production environment

[Kaspersky Lab, 2016] confirms the aforementioned change in ATIs and concludes the threat landscape

> *"The industrial network is increasingly similar to the corporate network [..][and] the cyber threat landscape for industrial systems is increasingly similar to the threat landscape for corporate networks. [..] [We] can expect [..] the emergence of new threats specifically designed for industrial enterprises [..][and the] evolution of existing, traditional IT threats [..] for attacks against industrial enterprises and physical world objects."*

The different categories of threats in literature are based on the implementation or impact. Here, threats are categorized according to their source as follows:

Force majeure contains all threats that have extraordinary circumstances be-

yond control of the affected parties, i.e disasters (thunderstorm, hurricane, lightning etc.), strikes or riots.

Organizational flaw contains threats that arise because an organization lacks a security concept or the one used is of defective design. This includes all technological and organizational rules, behavioral guidelines/ code of conduct, responsibilities, roles and measures to achieve security goals.

Human failure refers to all threats that result from human error. Even if effective technological an organizational security measures are in place, the human being operating these tends to make mistakes. These mistakes can result in a threat for an organization.

Technical failure takes place within hard- and software components. The latter can have bugs and hardware can have damages due to regular wear and tear.

Intentional actions contain any malicious intent of a person to harm an organization. This includes sabotage, destruction or attacks on any layer including the cyberspace.

2.1.2. Cyber-attacks

Cyber-attack is defined in [Oxford, 2004] as *"an attempt by hackers to damage or destroy a computer network or system"*. A cyber-attack can be classified into four types by means of their purpose: information theft, manipulation, disturbance and destruction.

Information theft : The attacker tries to extract information from an organization. This can be achieved by extracting the information through technological (systems) or organizational (people) infrastructures. A simple and well known example is a noble spy trying to get the world domination plans of an evil adversary to prevent this attempt and save the world. For industrial espionage this could lead to huge economic loss for a company due to a competitor, which uses the stolen information wisely.

Manipulation : The attacker changes specific information contained within the targeted organization. A simple and well known example is a lazy student who is changing his scores in the schools database to get a better school report. In an industrial scenario an attacker could change the quality parameters of manufactured products slightly, so the companies customers will get dissatisfied in the medium to long term.

Disturbance : The attacker tries to inhibit regular functionality of a targeted organization. A simple and well known example is a distributed denial of service (DDOS) [Lau et al., 2000] attack on a website of a company, which prevents anybody to access the website. A (D)DOS-attack within an industrial automation environment could lead to huge financial loss in a production line, because availability is the most important security goal.

Destruction : Actions of an attacker intending to destroy specific assets of an

organization which usually leads to substantial financial loss and disrupted functionality. An example is the destruction of a manipulator, e.g. a robot in a production line or a water-pump in a water supply infrastructure. This manipulator needs to be replaced, integrated, programmed and calibrated to re-establish functionality.

An attack is usually a sequence of actions and consists of different phases. In Ethical Hacking [EC-Council, 2017] [Dregier, 2017] five general phases are defined: reconnaissance, scanning, gaining access, maintaining access and covering tracks. For ATIs and other industrial environments APT attacks [Symantec, 2011] [Chen et al., 2014] have more recently gained attention. [Symantec, 2011] defines: incursion, discovery, capture and exfiltration. In [BSI, 2014] the given phases are: reconnaissance, infection, network investigation, privilege escalation, accomplishment and covering tracks. Similar to the aforementioned [Chen et al., 2014] defines phases like: reconnaissance and weaponization, delivery, initial intrusion, command and control, lateral movement and data exfiltration.

A more general definition for the sequence of actions used throughout this thesis is shown in figure 2.1 (ref. to [Horn and Krüger, 2015]). First, the attacker needs to identify its target, which can be the operators of critical infrastructure, a company or an authority. This is followed by the provision of information about the target, so that systems are deployed where and on which communications interfaces they are connected. The identification of a particular server with its IP address, operating system is running and services and any resulting vulnerability could be the result of such phase of gathering information. In the next step methods and tools need to be used or developed to take advantage of the identified weakness and infiltrate into the system. In many cases the classic stack overflow attack or similar storage area conflicts are used to enable running own malicious code. This might enable an attacker to get full access to the system which has then to be made persistent and hidden from any monitoring system. Through this back-door the attack itself is then executed, for example confidential information extracted or changed and the system disturbed in its operation. After the attack, the attacker wants to eliminate his tracks, so that either the attack itself goes unnoticed, or the attacker can not be identified.

1. Target identification is the step of getting the knowledge who the target will be. An organization can become a target for several reasons [BSI, 2014], e.g. disgruntling its customers or the public (Hacktivism), being an attractive financial target (Cyber-Criminals), offering the potential for political influence or pressure (Intelligence Services) or mistreating or not valuing its members or affiliates (Insiders).
2. Reconnaissance means to gain as much information as possible about a target. This usually includes Internet research, social engineering, database querying and active network scanning. Usable information ranges from services and addresses of critical systems to user credentials and organizational or technological structures. The more information an attacker can get, the more likely an exploitable vulnerability can be found.
3. Exploitation uses the information gathered in the reconnaissance phase to

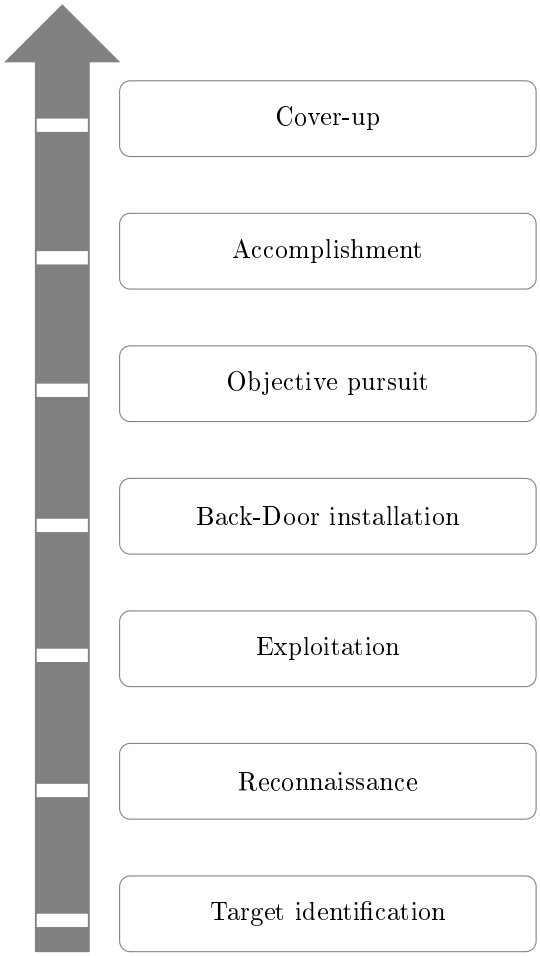

Figure 2.1.: Phases of an attack

identify and exploit vulnerabilities. The tools to do so are nowadays publicly available. Only specialized tools or exploits have to be developed, if necessary. The attack vector can vary from dropping an USB-Stick within a building and waiting for an employee to find and use it out of curiosity, exploiting a software bug from a distance over a WAN or installing rogue Wireless Local Area Network (WLAN) access points within the organization to deliver the malware.

4. Back-Door installation is the step to maintain access for the whole time of the operation. To do so, the attacker needs to install a Back-Door. This enables full and usually stealthy access to the target using e.g. a new communication channel, by adopting existing credentials or creating new ones. The access has to be available if needed and should not be alterable by the organizations personnel.
5. Objective pursuit means that all preparations are done and the attacker can execute all necessary actions to reach the desired goal. In this phase the real intentions can become clear and a good detection followed by

respective countermeasures can avoid these.

6. Accomplishment refers to the fulfillment of the attackers goals, which usually goes along with a specific negative effect for the victim, depending on the type of attack. This is typically the point where the victim recognizes that something happened. It is not unusual that the attack itself is classified as a technical malfunction or similar by the target.
7. Cover-up is the last step. Since the attacker itself usually does not want to be recognized or the attack vector needs to remain undetected, the traces of the attack are removed during that phase. This usually includes log-file manipulation, simple deletion of files or even destruction of storage or information technology.

Depending on how advanced the attacker is, defense-capable the target or ambitious the goal of the attack is, this sequence can miss some of the presented items. Furthermore the sequence can be interrupted by the target itself at any point. For example the identification can be avoided for an organization by not becoming an attractive target to certain groups of attackers. A company that changes its general terms and conditions in favor of some clients, e.g. the ones paying more money, eventually displeases the other clients. This could be one reason for identifying the company as target for an attack by these disgruntled customers, which could be avoided by the company in the first place by acting more carefully. After this point every step can be organizationally or technologically detected to some point, since the attacker has to take extraordinary actions.

2.2. Countermeasures

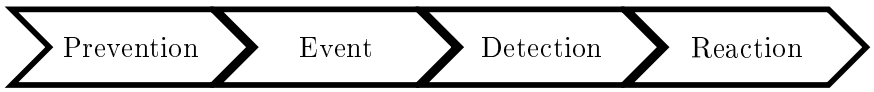

Figure 2.2.: Timely structure of countermeasures

Current State-of-Technology security methods and tools to secure information, automation and communication infrastructures are divided into two different categories: *prevention* and *reaction* [Horn and Krüger, 2015]. The timely structure is outlined in figure 2.2. Examples for preventive and reactive measures can be found in table 2.1. The following sections only describe technological measures, organizational and social ones like raising awareness, user training or action guidelines are not focus of this thesis.

Detection is part of a reaction, since it occurs after the event. It is covered separately, since it is essential for any reactive countermeasure and focus of this research.

Category	Prevention	Reaction
Organisation	Policies, Training, Awareness, Liability	Depends on employee
Software	Penetration testing, Patches, Access control	Monitoring, Analysis
Hardware	Interface & Access Limitations	Monitoring, Analysis, Shut-Down
Communication	Filter, Encryption, Deflection	Monitoring, Analysis, Slow Down, Cut-Off
Data	Access Control, Encryption, Pseudonymisation	Monitoring, Analysis, Validation

Table 2.1.: Preventive and reactive countermeasures

2.2.1. Prevention

Prevention is the attempt to prepare and protect a system against attacks. However preventive measures can only protect against known threats and can be overcome in time. The goal usually is to make an attack more complicated and discourage and put off some attackers. But the advantage is usually on the attacker's side because they have to detect and exploit at best only one vulnerability at a certain critical point. Furthermore, they can easily switch between the technological, human and organizational levels in a complex infrastructure and also take different roles. The defenders are forced to take various measures at all levels within of a complex IT infrastructure and stay in their role. A fundamental countermeasure is the establishment of security concepts, i.e. by implementing authentication, authorization or access control for different roles. This is usually done by defining technological and organizational rules, codes of conduct, responsibilities, roles and actions. Another example is the usage of technological filters, such as packet or content filters, where only certain parts of communication are permitted. Also widely spread is the usage of cryptology (encryption) to secure the transport or storage of messages and data against eavesdropping, tampering and forgery. A preventively conducted vulnerability analysis, as done in penetration testing, in conjunction with the elimination of detected vulnerabilities can prepare the infrastructure against attacks as well. Lastly mentioned in the area of prevention is the usage of deception, for example using honeypots or -nets. By simulating a productive and worthwhile target the attackers are tricked to perform the attack there, slowing it down and being able to analyze it.

Security concepts and policies

Different concepts exist to secure IT-infrastructures against threats. Usually the implementation of these go along with implementing an Information Security Management System (ISMS) for the organization. These mechanism are standardized in several international and national standards for security of in-

formation, automation and communication infrastructures. A brief overview can be found in table 2.2.

Topic	Standard	Description
Definitions	ISO/IEC 2382	Information technology – Vocabulary
	ISO/IEC 27000	Terms and Definitions
Information security management systems (ISMS)	ISO/IEC 27001	Requirements
	ISO/IEC 27002	Code of practice for information security management
	ISO/IEC 27004	ISMS Measurement
	ISO/IEC 27006	Requirements for bodies providing audit and certification of ISMS
Domain specific	ISO/IEC 27011	Information security management guidelines for telecommunications
	VDI/VDE 2182	Information security for industrial automation
	DIN/IEC 62443	Industrial Communication Networks and System Security
Security risk management	ISO/IEC 27005	Information security risk management
	ISO/IEC 27014	Governance of information security
Evaluation	ISO/IEC 15408	Evaluation criteria for IT security (Common Criteria)
	ISO/IEC 18045	Methodology for IT security evaluation
	ISO/IEC 19791	Security assessment for operational systems
Specific functions	ISO/IEC 18033	Encryption algorithms
	ISO/IEC 10118	Hash functions
	ISO/IEC 18031	Random bit generation

Table 2.2.: Exemplary standards for IT security related to this work

Different security concepts and policies are presented in [Saltzer and Schroeder, 1975], [Schneider, 2000], [Liu et al., 2001], [Höne and Eloff, 2002], [Berghel, 2007], [Cardenas et al., 2009] or [ENISA, 2017] and include

Redundancy is a concept in safety engineering explained well in [Oxford, 2004]: *"[Engineering] The inclusion of extra components which are not strictly necessary to functioning, in case of failure in other components."* For security purposes a redundant system can prevent a Single Point of Failure (SPOF).

Diversity is a concept where different system components are used for a specific task. An example would be to use PLCs from different vendors. This way it can be avoided that a single attack vector can compromise similar systems in the infrastructure [Cardenas et al., 2009].

Access control is a concept to maintain different roles and access rights for users, software services or devices, including the principles of *least privilege* and *separation of privilege* [Saltzer and Schroeder, 1975]. The former means that every entity should only get the minimum amount of access rights to perform tasks. The latter describes that critical assets should be protected by multi-entity authentication[1].

KISS (Keep it Simple and Straightforward)[2] means to keep a system manageable by human operators. The complexity should be reduced to a necessary minimum [Rich, 1995]. This way it can be ensured to be able to detect vulnerabilities and eliminate them.

Security-by-obscurity is often used by vendors for product designs and means "*secret protection for their products in order to extend ownership beyond the terms afforded under copyright and patent law*" [Mercuri and Neumann, 2003]. It is necessary for keeping cryptographic keys secret, but "*a good practice in this regard is to avoid the use of close-source and proprietary protocols, as their security cannot be verified, and many incidents have already proven that security through obscurity does not necessarily equate proper security coverage*" [ENISA, 2017].

Open design is a well known principle as used in *Free Software* [Stallman, 2002]. It means to share the design openly, e.g. for software this means to disclose the source code. Advantages are decoupling of protection mechanisms, examination by many reviewers and the necessity for security-by-design. "*Finally, it is simply not realistic to attempt to maintain secrecy for any system which receives wide distribution*" [Saltzer and Schroeder, 1975].

Security-by-design is achieved if a system or infrastructure fulfills all basic principles of information security (ref. to section 2.1). It is "*an approach to information security which [..] is at once holistic, creative, anticipatory, interdisciplinary, robust, accountable and embedded into systems. It stands in direct contrast to security through obscurity, which approaches security from the standpoints of secrecy, complexity or overall unintelligibility.*" [Cavoukian and Dixon, 2013]

Time based Security was introduced by [Schwartau, 1998] and its basic model states $P_t > D_t + R_t$, where P_t is the time a protection system protects from an adversary, D_t is the time necessary to detect an attack and R_t is the reaction time. If the statement is fulfilled by all security measures of an infrastructure it is considered secure.

Perimeter defenses are established at access points of the infrastructure, separating internal from external domain. "*A perimeter defense is always the best first line of protection. Network firewalls, access control mechanisms, strong user authentication devices, virtual private networks, and antivirus and other content screening software can all be deployed as part of the*

[1] An example is launching nuclear missiles by two people, usually the president and a high-ranked general, inserting two individual keys at the same time, as seen in many movies.
[2] Originally by C. Johnson: "*Keep it simple, stupid!*"

network security perimeter" [Liu et al., 2001].

Defense in depth means to divide the whole infrastructure in different zones or compartments and establish *perimeter defenses* for each zone individually [Kuipers and Fabro, 2006]. The zones can be divided by physical, geographic, logical, layered and virtual means [Bass and Robichaux, 2001].

Risk management is a "*systematic approach to identify organizational needs regarding information security requirements and to create an effective information security management system (ISMS). This approach should be suitable for the organization's environment, and in particular should be aligned with overall enterprise risk management. Security efforts should address risks in an effective and timely manner where and when they are needed. Information security risk management should be an integral part of all information security management activities and should be applied both to the implementation and the ongoing operation of an ISMS. Information security risk management should be a continual process [..] to reduce the risk to an acceptable level.*" [ISO27005, 2011]

Incident/Crisis management should be triggered in case of an event (malfunction, attack or misuse). It means to return the infrastructure to a safe and manageable state by triggering the right actions. ''*Good crisis management involves using particular tactics to handle the specific situational contingencies which are present or arise during the course of a mass emergency. Clearly, it is usually impossible ahead of time to spell out in detail the particular tactics which have to be used because almost by definition they will be relatively specific to the actual situation encountered. Good crisis management to a considerable extent is the application of tactics which are relevant to the situational contingencies of a given disaster.*" [Quarantelli, 1988].

Specifically for ATIs and SCADA systems [Daneels and Salter, 1999] substantial studies were published to support operating companies, for example by the European Unions Agency for Network and Information Security (ENISA)[ENISA, 2011] or the United States National Institute of Standards and Technology (NIST) [Stouffer et al., 2011]. These serve as guidelines for integrating preventive measures and raising awareness among employees and executives. Furthermore research on optimal ICS network design can be found exemplary in [Zhang et al., 2011] and [Genge et al., 2015].

Filter

Filtering techniques have a long tradition in human history. Already at times where written notes and letters were passed on by messengers filtering, modification or censorship occurred. A fairy tale told to children [Grimm and Grimm, 1812] ("KHM 29: Der Teufel mit den drei goldenen Haaren") illustrates that well. In the story a letter gets intercepted, modified and the resulting action is, luckily to the messenger, different to the intended one. Also in modern times filtering was widely used within mailing and telecommunication, as stated by

[Foschepoth, 2009].

These well established concepts were adapted for computer hosts and network traffic with so called *firewalls* [Bellovin and Cheswick, 1994]. The network-packets can be filtered and modified, analogical to the aforementioned letters. A network packet consists of the package frame, depending on the network protocol used, and the payload or package content data. A basic differentiation is made between *packet-filtering* and *content-filtering*.

Packet-filtering uses the package frame data, which usually contains network address, network port or similar properties. This package frame can be seen as the envelope of the aforementioned letter, which contains information about source, destination and sometimes additional a description of the content data, i.e. what application format is used. Filtering on that basis was used first, since it is easier to implement.

Content-filtering enables the filtering based on the application content. This can be a complex task, since the payload of one network packet is usually only a part of a data stream, which is distributed over many network packets and could eventually even be encrypted. The stream has to be decrypted, reassembled and evaluated by the filter-application (i.e. firewall). To do so the filter-application has to maintain a state, which means the use of additional computing resources and an increased attack surface.

The topic filtering and firewalls for ATIs, ICSs and SCADA systems has been explored very deeply in literature [Stouffer et al., 2015] and vendors like Trend-Micro, Symantec, Kaspersky Labs and many more offer corresponding commercial products. Also free alternatives are commonly used in security research and are able to handle industrial protocols. [Nivethan and Papa, 2016a]

Cryptology

Cryptology is a well researched subject and dates back to ancient times. Well known examples from history can be found from the Caesar cypher used by the Roman Empire until the Enigma-system used by German forces in World War II.

Two major research areas are covered in the field of cryptology: cryptography and cryptanalysis. "*The cryptographer seeks to find methods to ensure the secrecy and/or authenticity of messages. The cryptanalyst seeks to undo the former's work by breaking a cipher or by forging coded signals that will be accepted as authentic*" [Massey, 1988].

While encryption is well established in common IT infrastructures, in ATIs, ICSs and SCADA systems cryptology is usually only used for Know-How protection of source code[3]. Industrial network communication is usually unencrypted [Fauri et al., 2017], since vendors don't offer this feature. Nowadays a separate device has to be integrated additionally into the infrastructure to encrypt the industrial network traffic [BSI, 2016b]. For future architectures encryption is intended in

[3] For example Siemens offers Code Blocks encryption.

industrial communication [Alves et al., 2017], but "*unique characteristics of SCADA networks make it difficult to adapt existing cryptographic techniques into these systems*" [Igure et al., 2006]. Furthermore [Fauri et al., 2017] stated three key findings against the usage of encryption in industrial networks: "*First, in the majority of cases, the introduction of encryption does not yield extra security. Second, encryption can actually have negative consequences for security by hindering other security mechanisms such as NIDS. Third, encryption can raise the costs of troubleshooting and recovery considerably.*"

Penetration testing

Penetration testing is a methodical approach to test software [Arkin et al., 2005], an application [Thompson, 2005], a system, network or infrastructure for vulnerabilities [Bishop, 2007]. It "*is the art of finding an open door. It is not a science as science depends on falsifiable hypotheses. The most penetration testing can hope for is to be the science of insecurity - not the science of security - inasmuch as penetration testing can at most prove insecurity by falsifying the hypothesis that any system, network, or application is secure*" [Geer and Harthorne, 2002]. It is also often referred to as *Ethical Hacking* [Dregier, 2017] [EC-Council, 2017] and certified training for professionals and many free software tools and frameworks are available. Examples are the metasploit framework [Rapid7, 2014], Nmap scanner [Lyon, 2009] or Shodan search engine [Matherly, 2009].

In ATIs, ICSs and SCADA systems "*identifying the vulnerabilities requires a different approach than in a normal IT network. In most cases, systems on an IT network can be rebooted, restored, or replaced with little interruption of service to their customers. Their world is mostly virtual-based, connecting only peripherally to the physical world. SCADA systems control physical processes and therefore have real world consequences associated with their actions. Some actions are time critical, while others have a more relaxed timeframe. One shouldn't connect a test machine to the network and perform scans of a SCADA system without understanding the possible consequences of this testing.*" [Duggan et al., 2005]. A real world example is also presented: "*While a ping sweep was being performed on an active SCADA network that controlled 9-foot robotic arms, it was noticed that one arm became active and swung around 180 degrees. The controller for the arm was in standby mode before the ping sweep was initiated. Luckily, the person in the room was outside the reach of the arm.*"

Hardening

Hardening relates originally to creating more robust electronic systems, that are able to withstand tough environmental conditions or exposure to nuclear detonations [Poll, 1970]. Software as used in operating systems [Granberg et al., 2016] [The Debian Project, 2018] and other applications can be hardened against typical attack mechanisms like buffer overflows, apart from applying constant vulnerability patching. Hardening has to be done by the developers of a specific software solution, since it is done by programming techniques. Specific addi-

tions are made to support secure execution of code [Strackx and Piessens, 2012], also for micro-kernel based systems like L4/Fiasco.OC [Borchert and Spinczyk, 2016]. Furthermore the gap between rapid prototyping of software using scripting languages and the utilization of these in later products is an important matter [Wrigstad et al., 2009]. The application of hardened mechanisms to binary software is discussed in [Fraser et al., 1999].

2.2.2. Reaction

All measures that are taken after an attack are called *reaction* (ref. to figure 2.2). Reactive measures including a successful detection are crucial to enable counter-measures, since a reaction to an incident requires its detection. After a successful detection, suitable mitigation, defense and forensic measures can be set in motion. In order to do this successfully, the detection should be followed by localization and classification. The former means to identify the precise location of affected systems and the latter means to determine the type of event, i.e. malfunction, misuse or attack [Horn and Krüger, 2015]. Due to this process, the reaction to the event has a certain delay, depending on the detection method used. In detection the subcategories monitoring, for example by virus scanners or intrusion detection systems, and analysis by honeypots/nets exist.

Detection

Detection can be done if there is a system that monitors a specific aspect of an organization and analyzes it. [Ahmad et al., 2014] distinguishes hereby between surveillance and detection: "*Surveillance is the systematic monitoring of the security environment aimed at developing situational awareness to adapt to fast- changing circumstances and threats. Situational awareness enables security decision makers to better cope with information security incidents and develop more effective defenses. [..] Detection is an operational-level strategy aimed at identifying specific security behavior. The objective of detection is to allow the organization to react in a targeted manner. This strategy contrasts with surveillance in that the latter aims to understand the overall situation. Detection therefore, focuses on a specific event whereas surveillance observes the status as a whole.*"

Intrusion Detection dates back to the development of more powerful computer systems in the 1960's. [Kemmerer and Vigna, 2002] describe the historic development: *"Originally, system administrators performed intrusion detection by sitting in front of a console and monitoring user activities. They might detect intrusions by noticing, for example, that a vacationing user is logged in locally or that a seldom-used printer is unusually active. Although effective enough at the time, this early form of intrusion detection was ad hoc and not scalable. The next step in intrusion detection involved audit logs, which system administrators reviewed for evidence of unusual or malicious behavior. In the late '70s and early '80s, administrators typically printed audit logs on fan-folded paper, which were often stacked four- to five-feet high by the end of an average week.*

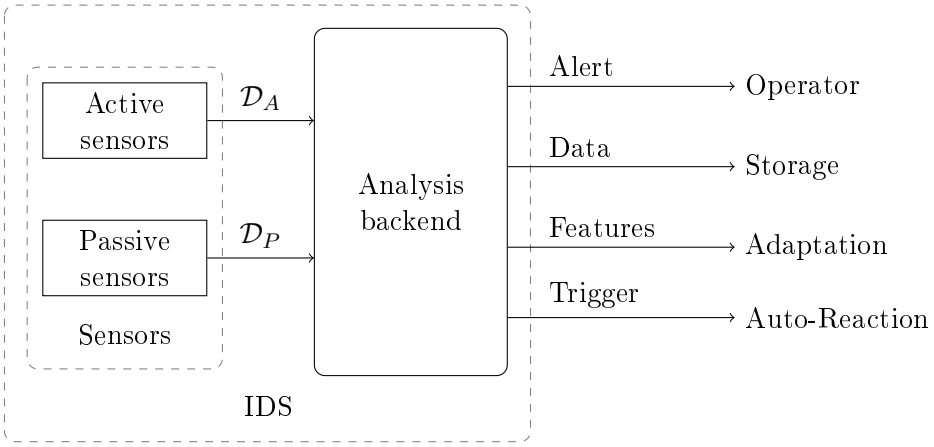

Figure 2.3.: Architecture of an Intrusion Detection System

Searching through such a stack was obviously very time consuming. With this overabundance of information and only manual analysis, administrators mainly used audit logs as a forensic tool to determine the cause of a particular security incident after the fact. There was little hope of catching an attack in progress."

[Anderson, 1980] introduced the first computer program to automate these tasks, an Intrusion Detection System (IDS). Subsequently different models for intrusion detection were developed. [Denning, 1987] introduced an early *"model of a real-time intrusion-detection expert system"*. Following that a great number of publications covering that topic can be found in literature. Since these first Intrusion Detection Systems (IDSs) were developed to protect a computer system or *host*, the potential to extend intrusion detection to networks was recognized. [Mukherjee et al., 1994] describe *"Intrusion detection [as] a new, retrofit approach for providing a sense of security in existing computers and data networks, while allowing them to operate in their current open [remark of the author: unprotected] mode."* Existing IDS were divided into Host based Intrusion Detection System (HIDS) and Network based Intrusion Detection System (NIDS).

A very important issue in monitoring is the data collection itself. "*For accurate intrusion detection, we must have reliable and complete data about the target system's activities. Reliable data collection is a complex issue in itself. [..] The amount of system activity information a system collects is a trade-off between overhead and effectiveness. A system that records every action in detail could have substantially degraded performance and require enormous disk storage*" [Kemmerer and Vigna, 2002]. Design and placement of a sensor "*is not only able to process packets at a higher rate, but can also apply more sophisticated detection techniques to reduce the number of false alerts [..but..] on high-speed networks, even highly effective sensors may produce alerts at a rate greater than the analysis backend can absorb. Consequently, it is important that the performance of the analysis components of network intrusion detection systems be accurately quantified as well*" [Schaelicke et al., 2003]. Furthermore *multisensor data fusion* from different sources is necessary "*to effectively create cyberspace*

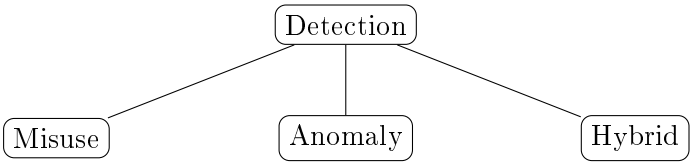

Figure 2.4.: Common categorization of detection techniques

situational awareness" [Bass, 2000].

Existing commercial detection tools are virus scanners, network/ device/ OS / service discovery, log analyzers, vulnerability scanners or Spam-filters, just to name a few. These categorizations are usually made based on the purpose of the product. All these can be summarized under the category IDS, for example a virus scanner is also referred to as a HIDS and different scanners are often utilized as active or passive sensors. The architecture shown in figure 2.3 outlines different modules of an IDS. Data sources, like network traffic, system log files or binary entities are monitored by sensors, which can be active or passive. The resulting data sets are analyzed by an algorithm which alerts an operator, archives the data, adapts its core engine or triggers automatic reactions.

In literature and in the context of this work the differentiation between different IDSs is made based on the underlying method utilized, i.e. how the tool analyzes collected data [Kemmerer and Vigna, 2002]. Commonly a differentiation is made between misuse/ signature-based detection, anomaly detection and a third category [Axelsson et al., 2000], which has properties of both aforementioned. [Patcha and Park, 2007] name signature/ misuse-based, anomaly-based and hybrid systems. For anomaly based systems the following techniques are mentioned: Statistical, Machine learning based (System call based sequence analysis, Bayesian networks, Principal components analysis, Markov models), Data mining based (Classification-based, Clustering and outlier detection, Association rule discovery). [Zhu and Sastry, 2010] also differentiate between signature-based and anomaly-based approaches. "*In between these two approaches, there lie the probabilistic- and specification-based methods for intrusion detection.*" Furthermore behavioral detection approaches are mentioned as the third main category. [Idika and Mathur, 2007] describe malware detection as "*a component of a complete IDS*" and categorize also signature-based, anomaly-based and additionally specification-based approaches. Each category splits further into dynamic, static and hybrid approaches. Figure 2.4 outlines the categorization used in this work described as follows:

Misuse detection or signature-based detection systems use a model of the intrusion process as knowledge basis. A signature is formed that can be recognized by the detection system [Axelsson et al., 2000]. An existing and well known attack can be well detected, but attacks that have no signature yet cannot be recognised by the system.

Anomaly detection refers to a contrary approach to signature based detection. The knowledge basis for the detector is formed by generating a model of normal state for the observed subject. Deviations from that exceeding spe-

cific boundaries generate an abnormal condition, thusfar getting flagged as intrusion [Axelsson et al., 2000].

Hybrid approaches combine misuse and anomaly detection approaches. "*These detectors have—at least in theory—a much better chance of correctly detecting truly interesting events in the supervised system, since they both know the patterns of intrusive behaviour and can relate them to the normal behaviour of the system*" [Axelsson et al., 2000].

Since the discovery of the malware STUXNET [Falliere et al., 2011] there is a growing number of publications focussing on IDSs algorithms especially designed for SCADA and ICS in ATIs. Before that event, IDSs were not utilized in these environments or transferred from general purpose IT structures without modification [Peterson, 2004]. An overview of current research towards new approaches can be found in section 3.2.

Deception

Deception means to divert the attack to a non-critical system or infrastructure to slow it down and/ or analyze it. The first scientifically documented deception mechanism can be found in [Cheswick, 1992], where services, password files and service activity in log files were faked within the production system at AT&T. Using that the defender was able to record hacks involving SMTP, FTP, finger, and multiple attempts to escalate privileges. The setup of a *chroot*[4] "*jail*" to monitor and record an intruders activity was later recognized as alternative approach to standard IDSs [Mukherjee et al., 1994]. The terms honeypot and honeynet were later adopted by [Spitzner, 2003].

Especially for SCADA and ICS in ATIs the following honeypots are available[5]

- Conpot [MushMush-Foundation, 2015], see listing 1
- SCADA HoneyNet Project [Pothamsetty and Franz, 2005]
- GridPot [sk4ld, 2015]
- GasPot [Hilt, 2016]
- SCADA honeynet [Digital Bond, 2010]

Furthermore some publications can be found over that topic. [Vollmer and Manic, 2014] presents a "*design and implementation for self-configuring honeypots that passively examine control system network traffic and actively adapt to the observed environment.*" Existing tools were evaluated and a "*novel four-step algorithm was developed for autonomous creation and update of a Honeyd configuration. This algorithm was tested on an existing small campus grid and sensor network by execution of a collaborative usage scenario.*" [McClean et al., 2013] utilized an honeypot based emulating a robot for their experiments on "*Cyber-Physical Security Assessment of the Robot Operating System (ROS)*". [Buza et al., 2014] present a PLC honeypot for smart energy systems called

[4]Standard unix command.
[5]To the best of the authors knowledge.

```
                    _
 ___ ___ ___ ___ ___ | |_
|  _|  .|   |  .|  .|  _|
|___|___|_|_|_  |___|_|
            |___|
Version 0.5.1
MushMush Foundation
------------------------------------------------------
Available templates:
------------------------------------------------------
--template default
Unit:         Siemens - S7-200
Desc:         Rough simulation of a basic Siemens S7-200 CPU with 2 slaves
Protocols:    HTTP, MODBUS, s7comm, SNMP
Created by:   the conpot team
--template guardian_ast
Unit:         Guardian - Guardian AST tank-monitoring system
Desc:         Guardian AST tank-monitoring system
Protocols:    guardian_ast
Created by:   the conpot team
--template ipmi
Unit:         IPMI - 371
Desc:         Creates a simple IPMI device
Protocols:    IPMI
Created by:   Lukas Rist
--template kamstrup_382
Unit:         Kamstrup - 382
Desc:         Register clone of an existing Kamstrup 382 smart meter
Protocols:    Kamstrup
Created by:   Johnny Vestergaard
--template proxy
Unit:         None - Proxy
Desc:         Sample template that demonstrates the proxy feature.
Protocols:    Proxy
Created by:   the conpot team
```

Listing 1: Output of conpot in a shell

CryPLH. [Wilhoit, 2013b] and [Wilhoit, 2013a] present honeypot systems that emulate a water distribution, production environment and factory controls to gain knowledge about attackers. [Vasilomanolakis et al., 2016] present an evaluation between their own honeypot called "*HosTaGe ICS HONEYPOT*" and conpot. [Lau et al., 2016] present an ICS honeypot called "*XPOT*". [Ding et al., 2018a] introduces a honeypot based on SNAP7 and IMUNES.

3. Thesis objective

"Matters of great concern should be treated lightly. Matters of small concern should be treated seriously." – Tsunetomo Yamamoto, *Hagakure* (1710-1716)

In parallel to business office networks developed by computer scientists, ATIs and SCADA systems were developed by engineers for reasons of cost efficient automation, functionality, extensive development support for end-user, investment security, reliability and robustness for mission critical industrial processes and the related technical support or maintenance by the vendor [Daneels and Salter, 1999]. This concurrent development also affected security technologies and research. Current methods shown in chapter 2 were initially developed for standard IT infrastructures as found in business office networks and then transferred to industrial applications. The threat situation outlined in chapter 1 requires a novel approach to secure Automation Technology Infrastructures (ATIs).

This chapter analyzes this situation and provides research questions, related work and a research hypothesis for this thesis.

3.1. Problem statement and research questions

The malicious software for ICSs mentioned in section 1.1 has in common, that it is highly specified and therefore its existence was detected respectively proven late [Symantec, 2011]. Traditional methods of IT-Infrastructure protection that rely on known attack vectors or signatures reach their limits with these highly specialized malware [Chen et al., 2014]. At the same time vulnerability of Automation Technology Infrastructures (ATIs) is growing (ref. to Fig. 1.3) simultaneously with the expansion of network structures and the increasing intensity of communication technologies used in all areas of life. This leads to much more possible weak spots and attack vectors [Johnson, 2010] [Cardenas et al., 2009]. Furthermore the pressure of cost-effective operation with a constantly growing number of remote stations enables attacks from vast distance to the target [Wilhoit, 2013b]. Aggravating this situation the effort for an attacker to carry out an attack is lower than several years ago, since several freely available software libraries like the Metasploit-Framework [Rapid7, 2014] exist. These tools can be used to exploit a broad variety of weak spots for various systems without extensive system or programming knowledge. On one hand this is highly relevant for operating companies and developers to test their systems (penetration testing), but on the other hand it also provides unwanted assistance to malicious attackers. For example the process of locating a target and acquiring information

with tools like the Shodan Computer Search Engine [Matherly, 2009] became much easier than it was a couple of years ago. Freely available software libraries like Snap7 [Nardella, 2015] or ADS [Beckhoff, 2017] can additionally be used to develop specialized malware.

With these tools at hand, an attacker is able to alter process-values undetected. These changes in deviating process-values cannot be detected manually by the operators, since they have to rely on sensor-information. An attacker can use the boundaries of automation layers (ref. to figure 1.4) to hide and probably remains undetected. As shown in section 2 there are already many tools and methods for securing ATI components against malfunction, misuse or attacks that rely on known attack vectors or signatures. These measures can only protect against known threats Furthermore all preventive measures can be overcome in time and "*Security is a combination of protection, detection, and response*" [Schneier, 2014]. The basic assumption of this research is:

Postulate 1. *An attacker will get access.*

Therefore especially a successful detection is crucial to enable counter-measures. This leads to the general research question of this work:

How can a previously unknown attack on an Automation Technology Infrastructure be detected?

Based on this further detailed research questions arise:

1. What would a sound and practice-oriented concept for detecting unknown attacks look like?
2. What methodical steps are necessary to develop such a concept?
3. Which limitations and constraints exist and how do they alter a possible concept?
4. Which parameters influence the quality of detection and which are modifiable?
5. How can a possible concept be integrated within the heterogeneous infrastructures?
6. How to scale the possible concept depending on the need for each application?

In section 2.2.2 it has be shown that a detection of unknown attacks is possible using anomaly detection. A comprehensive overview about general anomaly detection methods and algorithms can be found in [Chandola et al., 2009] and [Chandola et al., 2012]. [Tsai et al., 2009] analyzes underlying machine learning techniques and classifiers as used in IDSs. Methods and algorithms designed especially for ATIs are presented in the next section.

3.2. Related work

Caused by emerging threats (ref. to section 1.1), new and improved security measures in automation systems are mandatory, specifically in critical sectors. The boundaries of an automation hierarchy between the supervision-, control, and device level (refer to Fig. 1.4) are a challenge for Intrusion Detection Systems (IDSs) in the detection of malfunctions, misuses or attacks. Especially field bus networks using proprietary protocols are still a challenge for existing systems.

Since the discovery of the malware STUXNET [Falliere et al., 2011] there is a growing number of publications focusing on IDSs algorithms especially designed for SCADA and ICS in ATIs. Before that event, IDSs were not utilized in these environments or transferred from general purpose IT structures without modification [Peterson, 2004]. An overview of current research towards new approaches can be found in [Zhu and Sastry, 2010], [Mitchell and Chen, 2014], [Urbina et al., 2016], [Ding et al., 2018b] or [Hu et al., 2018].

[Yang et al., 2006] discusses the application of anomaly-based intrusion detection in SCADA systems. The chosen method uses auto associative kernel regression (AAKR) model to estimate *"predetermined features that represent network behavior"* and the statistical probability ratio test (SPRT) is used to classify a *"normal or an anomalous distribution"*. The approach was tested in a simulated SCADA system *"consisting of several SUN servers and workstations on a local network"*. Detection rates are not given.

[Cheung et al., 2007] presents a model-based IDS using expected communication patterns of protocol-level models for characterizing Modbus TCP requests and responses. Furthermore a component that discovers supported function codes on the Modbus devices to detect availability of services was proposed. The Snort IDS [Roesch, 1999] is used to implement the protocol-level rules, signature-based rules and service discovery. Detection rates are not given by the author. It is stated: *"Results from this experiment provide evidence that the model-based intrusion detection approach is effective for monitoring networks, and that it is complementary to the signature-based approach."*

[Verba and Milvich, 2008] demands a hybrid approach between signature and anomaly based IDS, that is able to understand the proprietary protocols in SCADA environments. *"A system implementing signature matching, flow analysis, and data inconsistency detection tuned specifically for SCADA and process control environment should be developed."*

[Rrushi and Campbell, 2008] presents an anomaly detection approach using stochastic activity network formalism to "*model and analyze the operation of a boiling water reactor [..] along with its digital I&C systems that engage the Modbus protocol.*" No evaluation is given.

[Linda et al., 2009] introduces an anomaly-based IDS using a combination of two neural network learning algorithms for normal behavior modeling: Error-Back Propagation and Levenberg-Marquardt. Window-based features for the supervised training process are *"the number of IP addresses in the window, the*

maximum and minimum number of packets per single IP, the average interval between packets, the time length of the whole window, the data speed, the number of protocols in the window, the maximum and minimum number of packets per protocol, the number of flag codes, the maximum and minimum number of packets per flag code, the number of packets with window size attribute set to 0, the number of packets with data length attribute set to 0, the average value of the window size attribute, and the average value of the data length attribute". The approach uses network data from a testbed that contains one single PLC connected to a Computer Workstation via a hub. *"The PLC unit was responsible for controlling valves in a fluid flow structure system."* Further Information of the testbed is not given. The attacks are *"simulated intrusion attempts".* With some fine tuning the authors were able to achieve a detection rate of 100%.

[Kosut et al., 2010] considers covert attacks on smart grid meters by injecting malicious data and respective countermeasures for a control center. The proposed detector uses hypothesis testing based on the generalized likelihood ratio test. An evaluation is done using numerical simulations and results are presented using Receiver Operating Characteristics (ROC) and Attacker Operating Characteristics (AOC) curves.

[Sandberg et al., 2010] introduces two security indices for state estimators in power networks, attack vector sparsity α_k and attack vector magnitude β_k. The attack scenario focuses on *"small-effort attacks [..] to manipulate one power flow measurement and to change related measurements in a consistent manner so that no alarms are triggered".* A comparable evaluation is not given, the authors state *"that simple measurement redundancy quantities seem to give security in terms of attack vector magnitude, but not in terms of attack vector sparsity".*

[Carcano et al., 2011] presents a concept for Intrusion Detection in SCADA systems based on *"a system representation language to describe in a formal way the system under analysis, a system state language to describe in a formal way the critical states associated to the system under analysis, a state evolution monitor to follow the evolution of the system, a critical state detector to check whether the state of the system is evolving toward a defined critical state and a critical state distance metric to compute how close any state is with respect to the critical states.".* The model itself is a set of rules in the form *condition* $\rightarrow$ *action*. The monitor polls PLC registers to parametrize the model. The distance metrics used are Minkowski L_1 (also known as *Manhattan*) distance and a *discrete distance*, where the number of system components whose values differ among two states are counted.

[Cárdenas et al., 2011] presents risk assessment, vulnerabilities, classification scheme and architecture for an anomaly detection system utilizing "*knowledge of the physical system under control*" The detection is done by modeling the system with a linear function and testing between two hypotheses: H_0 (normal behavior) and H_1 (attack). Simulation experiments are done using urge, bias and geometric attacks. False alarm rates are shown in figures and stated to be dependent on a threshold τ, that is based on the assumed (and fixed) false alarm constraint. Numbers taken from the figures range from 0 to 20. A trade

off between detection time and false alarm rate is further mentioned.

[Giani et al., 2011] shows characterization and countermeasures for unobservable low-sparsity cyber attacks on meters in smart grids. Detection is done using a DC power flow model and the placement of known-secure phase measurement units to render these attacks observable. Examples are given instead of an evaluation.

[Reeves et al., 2012] proposes design and implementation of a HIDS for embedded control systems."*Autoscopy, an experimental host-based intrusion detection mechanism that operates from within the kernel and leverages its built-in tracing framework to identify control-flow anomalies, which are most often caused by root-kits that hijack kernel hooks.*" An evaluation with experiments using real root-kits is presented and validates the functionality of the approach.

[Hadžiosmanović et al., 2012] presents an approach that analyzes log files within SCADA systems on layer 3 and above (ref. to Fig. 1.4). The proposed system is called MELISSA (Mining Event Logs for Intrusion in SCADA Systems) and focuses on user activity. Evaluation results are given for functionality, usability and system performance.

[Papa et al., 2012] introduces a Transfer Function based IDS. *"A TFIDS uses one or more time or frequency domain based transfer functions or system models that are used to estimate signals within the system."*. The validation scenario is a Matlab simulation of a waste water treatment system and the focus is on detecting Man-In-The-Middle (MITM) attacks. The model itself is roughly outlined and physical parts, like "Level estimator model" or "control valve model" are not given. It is furthermore not explained how transfer functions are constructed and where the used examples come from. The evaluation is non-comparable.

[Lin et al., 2012a] presents a specification based IDS for the DNP3 protocol using Bro [Paxson and Bro-Community, 2017] in detail. [Lin et al., 2012b] and [Lin et al., 2013] further show other details of the same approach. Experiments are made using a "*laboratory-scale emulated SCADA system consisting of proprietary hardware and software commonly found in todays power grid environment*". Attack scenarios are based on modifications of commands and measurements within the DNP3 protocol. Signature based and anomaly based policies are evaluated. For the anomaly detection weighted least square state predictions are made for the whole system. Experiments conclude with "*zero false positives or negatives are found*" with the exception of the anomaly based setup, where no statement regarding detection rates is given at all.

[Mashima and Cárdenas, 2012] focus on detecting electricity theft from Advanced Metering Infrastructure (AMI) systems. The hypothesis is that an attacker gains access to a device and manipulates measurements. Possible detectors based on simple averaging, Auto-Regressive Moving Average (ARMA), Exponentially-weighted Moving Average (EWMA), Non-parametric CUmulative SUM (CUSUM) and Local Outlier Factor (LOF) are evaluated using a meter. False positive rate and (financial) loss per attack are given.

[Amin et al., 2013b] develops a model-based scheme based on analytically ap-

proximated models of canal hydrodynamics for detection and isolation of a wide class of faults and attacks in automated canal systems. The presented attack scenario describes "simultaneous and uncoupled cyber-physical faults", i.e. an attacker withdraws water from the canal system unauthorized and manipulates several sensors to remain undetected. Concerning *"security of water SCADA systems"* the approach has according to the authors *"fundamental limitations [..] in isolating attacks to distributed physical infrastructures."*

In [Hahn and Govindarasu, 2013] colored stochastic Petri net models are used to identify configured communication patterns in the IEC 61850 protocol data flow for Smart grids. The models are built from the IEC 61850 standard XML configuration language, the Substation Configuration Language. Malicious behavior is detected if the traffic diverges from configured communication pattern.

[Genge et al., 2013] shows an approach for detecting anomalies based on data fusion using Dempster-Shafer's "Theory of Evidence" . A performance evaluation is given.

[Goldenberg and Wool, 2013] presents an approach to model the network traffic patterns between HMI and PLC using Deterministic Finite Automata (DFA). Later [Kleinmann and Wool, 2016] extends the construction method using spectral analysis and [Kleinmann and Wool, 2017] extends this work subsequently to Statechart DFA. Each cyclic pattern is modeled with one DFA while incoming traffic gets de-multiplexed and sent to its respective DFA. In the unsupervised learning phase a Discrete-Time Markov Chain is constructed from a network stream, from which the individual DFAs are constructed using Euler cycles. Evaluation is done using network captures of "*S7-0x72 protocol from a control network of a solar power plant*" and synthetic datasets. For the real dataset false alarm rates are given up to 14.54%. In [Markman et al., 2017] Burst-DFA models are introduced, because "*cyclic-DFA models have limitations*".

[Han et al., 2014] discusses techniques and challenges for Intrusion Detection in Cyber-Physical Systems. Furthermore an IDS design is outlined with the following requirements: "1) it performs in a distributed manner, where many components cooperate with each other; 2) it performs in an on-line manner, where the detection is done in real time and the latest dynamics of normal profile is captured in time; 3) it should be effective against both known and unknown cyber attacks, as well as the random failures; 4) it should be fault tolerant; 5) it should not expose any privacy during the analysis and decision procedures."

[Yang et al., 2014] introduces preventive and detection measures for power networks. The preventive framework outlines security enclaves, a SCADA-IDS management system and perimeter defense with interior detection. The detection method consists of access-control white-lists, protocol-based white-lists and behavior-based rules. The latter perform a deep packet inspection of IEC 60870-5 network traffic and apply specific rules based on function code, packet length, value ranges, frequency of commands, correlating alarms to sensor information and state variables to measuring variables. The attack scenario is based on Address Resolution Protocol (ARP) spoofing using the Metasploit Framework

[Rapid7, 2014]. To verify the reliability a statistical distribution analysis of the execution time for the IDS using Gumbel distribution obtained by experiments is given. Detection rates are given with 100% as *"validated through a series of realistic scenarios performed in a SCADA-specific testbed"*.

[Alcaraz et al., 2014] surveys common anomaly detection approaches and discuss the applicability to smart grid domains.

In [Hadžiosmanović et al., 2014] variable-specific prediction models for process-values are used to predict future activity patterns. The process values are extracted from network traffic and categorized into control, reporting, measurement and program-state variables to differentiate activity patterns. The prediction is done using two complimentary approaches: autoregressive modeling and control limits. An alert is triggered, *"if the value reaches outside of the control limits or produces a deviation in the prediction of the autoregressive model"*. The approach is evaluated non-comparably, as the authors are not *"interested [..] in specific true/false positives rates"*.

The Spear-Framework [Genge et al., 2014] is able to model networks of Critical Infrastructures and generates anomaly detection rules for Snort [Roesch, 1999] based on that.

[Sridhar and Govindarasu, 2014] develop a model-based anomaly detection and attack mitigation algorithm for Automatic Generation Control (AGC) on power system frequency and electricity market operation and evaluate it through simulation studies. The focus is on scaling and ramp attacks and false positive ($0 < FP < 0.65$) and negative rates ($0.14 < FN < 0.28$) are presented in figures.

[Mo et al., 2014] shows an approach to detect replay attacks from within the control program of a PLC by adding a zero-mean Gaussian authentication signal to an Linear Quadratic Gaussian (LQG) control algorithm. Theoretically evaluated results vary depending on the design parameters, rendering an *"attack without being detected [..] feasible"*. Furthermore the approach provides *"detection at the expense of control performance"*. The approach gets generalized by [Mo et al., 2015], where the "*problem of designing the optimal watermark signal in the class of stationary Gaussian processes to maximize a relaxed version of the expected Kullback-Leibler divergence between the distributions of the compromised and healthy residue vector, while satisfying a constraint on the control performance*" gets investigated This theoretical works applicability in practice is limited to the usage of a LQG control strategy [Bryson, 1996]. The companies and organizations providing the application scenarios for this thesis utilize more conservative control paradigms like Proportional Integral Derivative (PID) [Minorsky, 1922] or bang-bang [Sonneborn and Vleck, 1965]. Furthermore [Rubio-Herna´n et al., 2016] reexamine this work and present another adversary model to show the weaknesses of [Mo et al., 2014]'s approach. The authors conclude: *"[..] we have shown that the detection strategy is not sufficiently robust from a security standpoint."*

[DO et al., 2014] presents an approach to detect cyber/physical attacks on

SCADA systems. A discrete-time state space model without process noise is utilized to describe the system and an attack vector comprised of state attack and sensor attack vectors. Detection is done by finding transient changes of known profiles using the Variable Threshold Window Limited CUmulative SUM (VTWL CUSUM) test. A comparison between the FMA test and the WL CUSUM test by multiple repeated Monte Carlo simulations is presented.

[Wang et al., 2014] proposes a relation-graph-based detection scheme called SRID: State Relation based Intrusion Detection. It is designed to detect false data injection attacks to SCADA systems and utilizes alternation vectors (alternation relations between two continuous states) and an autoregressive graph representation of these called state relation graph. An evaluation with a simulated boiler system as part of a coal power plant is given. The attack used for evaluation is "*randomly choose a variable and inject with arbitrary data that falls in its valid range*". True positives and false positives are given.

In [Schuster et al., 2015] One-Class Support Vector Machines (OCSVMs) are applied to important attributes of industrial network traffic (Profinet) to detect anomalies. [Maglaras and Jiang, 2014] uses a similar approach, where important features of network traffic in SCADA systems get extracted and classified by a OCSVM. But to reduce the number of false alarms a post-processing step, based on k-means clustering is applied.

[Janicke et al., 2015] introduces a PLC monitoring system using Interval Temporal Logic (ITL). *"Interval Temporal Logic (ITL) is a flexible notation for both propositional and first-order reasoning about periods of time [..] can handle both sequential and parallel composition and offers powerful and extensible specification and proof techniques for reasoning about properties involving safety, liveness and projected time. Timing constraints are expressible and furthermore most imperative programming constructs can be viewed as formulas in a slightly modified version of ITL."* From the PLC Values for Markers, Digital Inputs and Outputs, Counters and Timers are captured and sent to the logic for analysis. Two attacks are conducted for evaluation: DoS and malicious code uploading. Both attacks were detected. Further detection metrics were not given by the authors.

[Senyondo et al., 2015] shows an architectural design paradigm called "PLCLOUD" for power grid applications. *"Red line safety criteria"* have to be defined for a real plant and subsequently controller programs uploaded from the SCADA system to the PLC get intercepted and analyzed for code that violates *"red line"* safety conditions. The threat model focuses only on uploading malicious code to a PLC:*"We note that PLCLOUD is not secure against a privileged insider with physical access to the plant floor. PLCLOUD cannot defend against false data injection attacks, in which a PLC is given forged sensor data. [..] Additionally, PLCLOUD cannot defend against PLC firmware exploitation, in which case the verified control logic can be completely bypassed by the compromised PLC. Finally, attacks against the cloud-based platform are currently out of our scope..."*. The *"security monitoring of the program execution [..] through real-time emulation of the PLC device in the cloud [..is done i a way..] such that the*

controller program executes on the actual PLC device and cloud-based PLC emulator simultaneously while the cloud-based execution is heavy instrumented with security intrusion detection sensors." The monitoring agents log PLC device inputs and send them to the cloud-based emulator for replay. For the cloud-based emulator the authors state: *"We implemented a controller code in the low-level assembly statement list (STL) programming language."* Only a short functional evaluation is given.

[Malchow et al., 2015] presents a system called "PLC Guard" to intercept code from network traffic as transferred to a PLC and enables comparison of the code with previous versions by an engineer or operator.

[Erez and Wool, 2015] introduces an anomaly detection system that detects irregular changes in Modbus/TCP SCADA control register values. The registers are firstly classified into sensor, counter and constant type. An iterative algorithm using finite state machines improves this classification. A model of normal behavior is subsequently constructed differently for each register class. During evaluation (enforcement phase) "*an alert is triggered if the window contains a value that is different from the value saved during the learning phase*". Evaluation results are stated with a true positive classification rate of 93% and a false alarm rate of 0.86% during enforcement phase.

[Stone et al., 2015] presents an approach to collect low power electromagnetic emissions from PLCs using a near-field RF probe for each ladder logic command (e.g. ADD, DIV, SUB, etc.) and use a correlation based algorithm to differentiate between normal and anomalous behavior. This refers directly to the code execution on the PLC. An experimental evaluation shows that the detection rate is highly dependent on the signal-to-noise-ratio, where detection in Hilbert transformed sequences shows better results than in time domain sequences. The best average true anomaly detection rate was given with 90%.

[Drias et al., 2015] presents analysis of "*different types and architectures of an ICS, security requirements, different threats attacks, and existing solutions to secure Industrial control systems.*"

[Caselli et al., 2015] presents an approach to develop a sequence-aware IDS. Sequences consist hereby of a chronological list of events and can include information from network communications, host log entries and process variable values. The model built upon this information is based on discrete-time Markov chains. The evaluation is done using real application datasets from a water treatment and purification facility that uses Modbus communication and the evaluation model only includes network sequences. "*No malicious traffic is included in the dataset, therefore this test verifies detection resilience against false positives*" No detection metrics are given, numerous false positives are present.

[Kiss et al., 2015] introduces an approach to "*detect cyber attacks targeting measurements sent to control hardware, i.e., typically to Programmable Logical Controllers (PLC). The approach builds on the Gaussian mixture model to cluster sensor measurement values and a cluster assessment technique known as silhouette.*" An experimental evaluation comparing the approach to he k-means

clustering algorithm and attack simulation results are given.

[Shoukry et al., 2015] presents "*PyCRA, a physical challenge-response authentication scheme designed to protect active sensing systems against physical attacks occurring in the analog domain*". The system works using a challenge response mechanism (χ^2). Random physical probes are sent to the surroundings and responses to these are analyzed. The system is evaluated using case studies from magnetic sensors and commercial Radio Frequency Identification (RFID) tags. Detection rates are given in figures, where the FPR is plotted over TPR.

[Barbosa et al., 2016] presents an approach similar to the works of [Goldenberg and Wool, 2013], where patterns in industrial Modbus/TCP and MMS (ISO 9506) traffic get modeled on packet level to detect deviations from a normal condition. The evaluation was done by validating the communications model using attack-free datasets from real application network captures. Therefore no detection metrics are given, the alarm rate for normal traffic (false alarms) varies depending on the number of candidate cycle for the learning phase and the network protocol used. Notably the figure presenting the alarm rate has a logarithmic scale.

[Ghaeini and Tippenhauer, 2016] introduces the *"HAMIDS: Hierarchical Monitoring Intrusion Detection System for Industrial Control Systems"*, which uses instances of the Bro [Paxson, 1998] NIDS in different network segments to monitor industrial network traffic. The payload of EtherNet/IP and Common Industrial Protocol (CIP) is analyzed for malicious commands of known attacks to specific Allen-Bradley ControlLogix PLCs. The approach was evaluated in the SWaT testbed [Mathur et al., 2016]. Detection rates are not given.

[Nivethan and Papa, 2016b] proposes a SCADA IDS that uses process semantic descriptions to create monitoring profiles for the Bro NIDS to monitor network traffic. A new compiler based system description language is introduced to model process variables and their limits, PLCs, connections between process variables and PLC memory locations and SCADA protocol and network details.

[Cruz et al., 2016] develops a distributed SCADA ICS IDS within the research project CockpitCI. It consists of detection agents, correlators and an OCSVM classifier. The detection agents are data sources for the approach and include *Network IDS, Host IDS, Honeypots, Shadow security unit, Exec checker, Output traffic controls, Configuration checker and Behavior checker.* The correlator filters *"process and relate security events,[..] also delivering noise filtering and event reduction capabilities, aggregating alerts produced by several detection agents or multiple events from a single source."* The authors do not explain why this architecture and detection data sources were chosen, they state: *"This solution was also designed with existing infrastructures in mind, by supporting mature technologies and taking advantage of already deployed management and operation support systems to provide topology and asset information."* The testbed for evaluation contains a *"HMI station (for process monitoring), a managed switch (with port monitoring capabilities for network traffic capture), and two PLC units, for process control."* The application scenario is an energy distribution grid. Classification rates for the OCSVM analysis component are

evaluated separately and vary depending on the evaluation dataset. Detection accuracy ranges from 94.6% to 98.81% and the false alarm rate from 1.18% to 3.25%. Other measures are not given.

[Garcia et al., 2016] presents security models for *"open controller systems where programmable logic controllers are coupled with embedded hyper-visors running operating systems with much more computational power. [..] The embedded hyper-visor shares memory with the PLC and can run models with advanced calculations for protocol analysis and safety verification."*. Two solutions are proposed: a memory monitoring and access control agent called *"Cyber-physical Verification Solution"* and a HIDS. The Cyber-physical Verification Solution *"relies on the assumption that the proprietary protocol is not reverse-engineered. If the PLC's programming protocol(s) are reverse engineered, a hacker who is able to establish a programming connection to the PLC can just program any blocks to overwrite or skip over the security implementation."* This assumption does not hold, since even at the publication time of this paper the Snap7 Suite [Nardella, 2015] already existed for several years, which contains that feature. For the HIDS the works of [Kleinmann and Wool, 2014] are utilized. Both solutions are evaluated independently and only for functionality. Detection rates are not given.

[Stefanidis and Voyiatzis, 2016] introduces an anomaly detection system based on the Hidden Markov Model (HMM) for interconnected SCADA systems using Modbus based on TCP/IP. A model of the traffic is constructed for header and data sections. For each section multiple HMMs are generated. Detection is done by routing each network packet through these multiple classifiers and all classification results are fused. Evaluation is done using three freely available datasets containing "*measurements from a laboratory-scale gas pipeline, a laboratory-scale water tower, and a laboratory-scale electric transmission system.*" The classification efficiency (accuracy) is given with 93.4%, false positives with "*less than* 1%". Detailed results show that Complex Malicious Response Injection and Malicious State Command Injection attacks cannot be detected by the system at all. Overall attack detection rate is given with 77.6%.

[Dunlap et al., 2016] presents an anomaly detection approach using timing based side-channels. the execution time of a PLC firmware and application program is measured and compared to reference values recorded earlier. An evaluation with a minimal program versus a fully featured program is given. The true positive rate ranges between 0.88 and 1 and the false positive rate from 0.033 to 0.077.

[Inoue et al., 2017] evaluates two anomaly detection approaches using unsupervised machine learning for water treatment systems. The two compared approaches are OCSVM and Deep Neural Network (DNN). Training and evaluation are based on the Swat dataset [Goh et al., 2017]. Detection rates are given with precision $(0.925/SVM, 0.983/DNN)$, recall $(0.699/SVM, 0.678/DNN)$ and F-measure $(0.796/SVM, 0.803/DNN)$.

[Wong et al., 2017] enhances the IDS/ Intrusion Prevention System (IPS) Suricata [OISF, 2017] with modules to inspect EtherNet Industrial Protocol (Eth-

erNet/IP) and CIP protocols. The rule-sets used to parametrize the IDS were taken from [emergingthreats, 2018].

[Sándor et al., 2017] defines a detection methodology based on sensor fusion using Dempster-Shafer's Theory of Evidence and validates the approach in the context of a real natural gas transportation installation. The approach is related to [Genge et al., 2013] and [Kiss et al., 2015]. The fused measurements are:"*parameters of the TCP communication stack measured with network sniffers; Modbus/TCP throughput as recorded by the primary controller; and the system tick recorded by the tasks running on the primary controller.*" Detection metrics are not given for the stated evaluation.

[Lahza et al., 2018] evaluates "*a number of features described in the literature that may be used to detect distributed denial-of-service attacks on the GOOSE and MMS protocols*". Features are evaluated using decision tree, neural network and support vector machine classifiers. The results include 55 features from previous research and 17 identified "*new advanced features*". Accuracy and false positive rates are given.

Furthermore [Barbosa, 2014] contributed in terms of a PhD thesis, where the works were continued as shown above in [Barbosa et al., 2016]. Another PhD thesis in that field was done by [Redwood, 2015] which contributes in terms of surveying relevant research of "*Cyber Physical System Vulnerability*" and a novel methodology for "*Malware and forensics analysis of embedded Cyber Physical Systems*" was elaborated.

In table 3.1 the relevant research is outlined. The approaches chosen here are based on availability, possible implementability and number of citations from the aforementioned research.

3.3. Analysis and research goal

The arguments for *general shortcomings* as stated in [Urbina et al., 2016] can be supported: "*(1) the vast majority of prior work uses stateless tests; (2) most control and power grid venues use [..specific models..] to model the physical system, while computer security venues tend to use a variety of models; several of them are non-standard and difficult to replicate by other researchers; (3) there is no consistent metric used to evaluate proposed attack-detection algorithms; (4) most papers focus on describing attacks to specific devices (i.e., devices that are not trusted) but they do not provide a fine-grain trust model that can be used to described what can be detected and what cannot be detected when the adversary is in control of different devices; and (5) no previous work has validated their work with all three options: simulations, testbeds, and real-world data.*"

The presented detection systems focus mainly on one specific data source, system, part or layer of a complex infrastructure. For example IDSs like HIDSs [Reeves et al., 2012] or NIDSs [Ghaeini and Tippenhauer, 2016] rely only on the data of a specific part of the whole infrastructure. Other works validate data of sensors [Sridhar and Govindarasu, 2014], actuators [Bai and Gupta,

Reference	Type	Domain	Evaluation
Cheung, 2007 [Cheung et al., 2007]	Network pattern	N/A (Modbus/TCP)	Testbed experiments
Linda, 2009 [Linda et al., 2009]	Communication pattern	N/A (Ethernet)	Static datasets
Carcano, 2011 [Carcano et al., 2011]	System state rule set	Power plant (Modbus/TCP)	Testbed experiments
Cardenas, 2011 [Cárdenas et al., 2011]	System and attack models	TE-PCS model (MATLAB)	Simulation experiments
Hahn, 2013 [Hahn and Govindarasu, 2013]	Communication pattern	Smart grid (IEC 61850)	N/A
Goldenberg, 2013 [Goldenberg and Wool, 2013]	Network pattern	Power grid (Modbus, S7)	Network captures
Hadžiosmanović, 2014 [Hadžiosmanović et al., 2014]	Process value model	Water boiler (Modbus)	Testbed experiments
Genge, 2014 [Genge et al., 2014]	Connection pattern	Smart grid (Ethernet)	Testbed experiments
Erez, 2015 [Erez and Wool, 2015]	Type of value model	Power grid (Modbus/TCP)	Static datasets
Caselli, 2015 [Caselli et al., 2015]	Sequence pattern	N/A (TCP/IP based)	Static datasets
Cruz, 2016 [Cruz et al., 2016]	Network pattern	Energy distribution	Testbed experiments
Inoue, 2017 [Inoue et al., 2017]	SWaT log pattern	SWaT: Water treatment	Static datasets

Table 3.1.: Overview of possibly applicable detection algorithms for ATIs

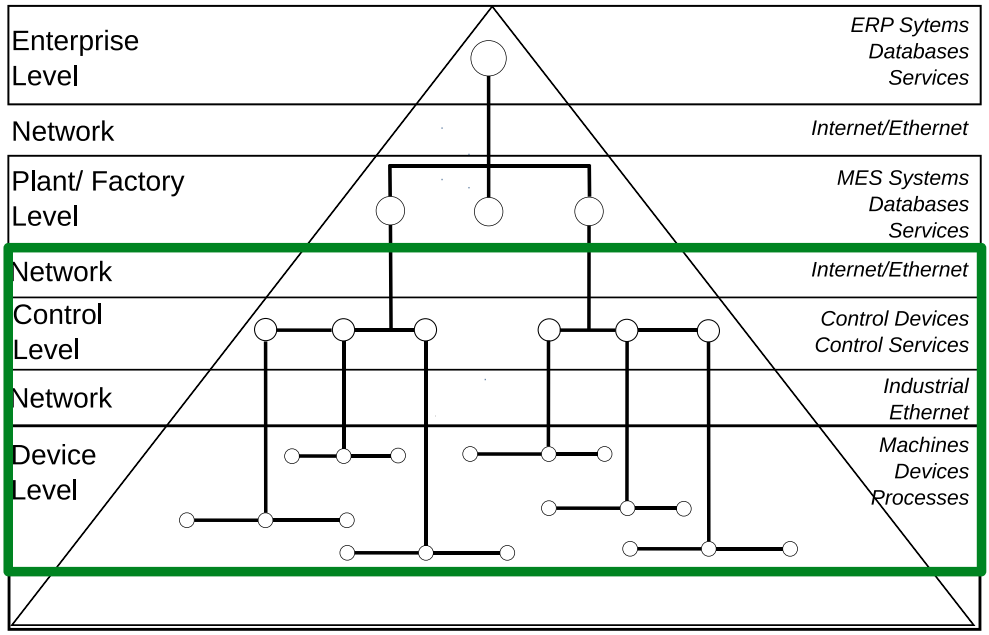

Figure 3.1.: Focus (green) for detection algorithm in automation pyramid structure

2014] or controllers [Vuković and Dán, 2013]. The correlation of data sources is proposed in some of the works presented, like [Caselli et al., 2015] and [Cruz et al., 2016], but interdependencies or causalities [Hlaváčková-Schindler et al., 2007] inside the infrastructure are usually not taken into account. Furthermore researchers tend to omit the methodology their work is based on and in some cases the hypothetico-deductive method [Gower, 1997] is implied. None of the approaches demonstrated methodological steps regarding requirements engineering along real application scenarios to achieve a better applicability in practice.

Postulate 2. *To achieve practical relevant results a detection concept has to be developed with respect to practical requirements.*

The goal of this work is development of a methodology and its subsequent prototypical application towards an evaluation use-case to derive a comprehensive

and application specific concept to detect anomalous values within the cyber-layer of complex and distributed automation processes based on real scenarios and requirements. As shown in figure 3.1 and according to [ISO/IEC62264, 2013] this includes Level 2: control and Level 1: devices.

The concept should fulfill the following requirements (ref. to section 2.2.2):

- usage of multiple data sources
- robust detection
- timely detection
- minimal interference to the process
- modular and scalable structure

To achieve relevance in practice of the desired concept a development has to be done close to real application scenarios with the help of experts within these. Also the concept should have the possibility to include different data sources and algorithms, since there is no perfect detection algorithm per se. The combination of different strengths could lead to a more robust detection. Furthermore the system has to be decoupled from the ATI infrastructure to avoid feedback effects if the detection system itself gets attacked.

3.4. Research hypothesis

The achievement of the desired objectives is coupled to the identification of data sources, features and causations of each process in every use case that can be monitored and analyzed. The challenge is complexity of processes in ATIs where deviations of normal process behavior have to be dynamically adopted without generating false alarms.

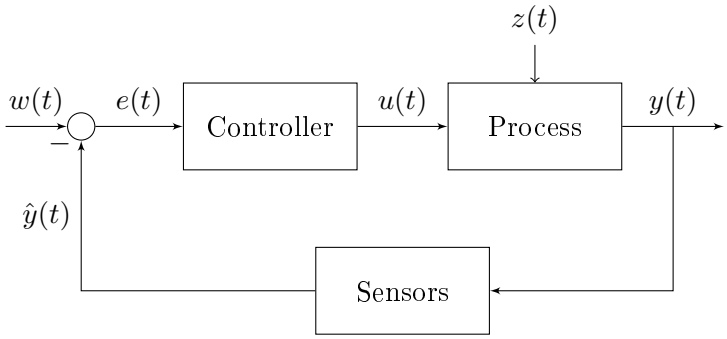

Figure 3.2.: Feedback control loop

Figure 3.2 shows a feedback control loop ([Lunze, 2008a]), where a specific physical process gets controlled by a controller which can be an automation node (i.e. a PLC with its specific software process) or a human operator. The reference input $w(t)$ defines the set-point for control-variables in which the process maintains a desired state. The error signal $e(t)$ is the input to the controller

and defined as

$$e(t) = w(t) - y(t) \tag{3.1}$$

The control signal $u(t)$ tunes the control variables of the process to keep it in a desired stable state. The output $y(t)$ reflects in the process values itself, which get measured by sensors. Since any measurement cannot be exact [Heisenberg, 1927], the measured values are represented by $\hat{y}(t)$. Any disturbance to the process (and that is why the controller is necessary in the first place) is represented by $z(t)$. The controller tries to minimize the error term over time:

$$\lim_{t \to \infty} e(t) = 0 \tag{3.2}$$

The stable control loop can only completely suppress a distortion or follow the reference input signal without a permanent control deviation (i.e. residual offset error) if it has an "internal model" of the distortion or reference input (i.e. internal-model principle, [Lunze, 2008a]). This would also apply to any kind of attack, which makes an automatic compensation of it a very complex task, since a model of it has to be available (ref. to misuse detection).

Since the goal of this work is to detect the attack, a second and more feasible approach can be used instead: using a concurrent instance of a process model itself for verification (ref. to anomaly detection). It can be built using

- experts knowledge
- analysis of software running throughout the ATI
- system identification methods

This first approach can be the most accurate, but requires cooperation of the experts and great effort. The second approach requires either systems to automatically transfer the knowledge contained inside the software into suitable models or a reproduction of the runtime environment of the original software. The third approach (system identification) has two major drawbacks already in its inherent approach: (i) it requires an assumption of the underlying model and (ii) it needs to perform controlled experiments using defined signal inputs to the system. This approach is mostly used for controller design, for system analysis it often produces nonlinear and complex models that cannot be simulated in real-time [Schauer, 2015]. Furthermore the models can often not be verified, since recalibration with further measurements leads to varying results.

Once the model is built it can be used for estimating the process output. It can then be used to detect deviations of the real process. The system is sketched in figure 3.3. Here an estimation is made in parallel to the controller and process branch. The estimation $\tilde{y}(t)$ will be used to validate possibly manipulated sensor values. Note that the summarization in figure 3.3 uses $y(t)$ only for drawing reasons. The calculation for difference $\delta(t)$ is made as follows:

$$\delta(t) = |\tilde{y}(t) - \hat{y}(t)| \tag{3.3}$$

Using equation 3.3 the difference between estimated- and measured values can be calculated. δ will take on the value zero in an ideal, non manipulated scenario.

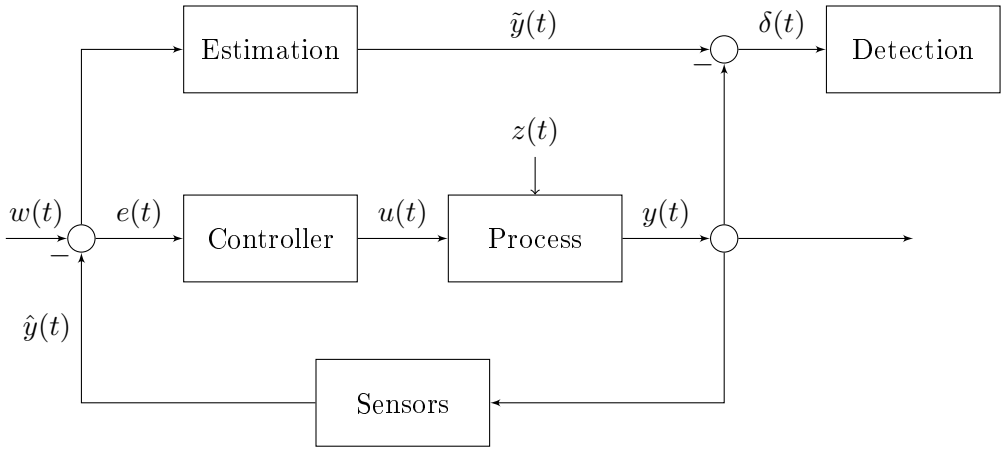

Figure 3.3.: Feedback control loop with attack detection

In practice, however, measurement inaccuracy will lead to deviant results close to zero. The basic assumption is based on the usage of a priori knowledge. It should be feasible that a defender of an ATI-infrastructure possesses more knowledge about it than the attacker.

Postulate 3. *A defender of an infrastructure possesses more knowledge about it than an attacker.*

The defenders can utilize this knowledge for building the aforementioned models. This way a more accurate model exists on the side of a defender which can detect the attack, even if the attackers use sophisticated and stealthy attacks.

Postulate 4. *Using a δ from well estimated $\tilde{y}(t)$ and respective live measured values $\hat{y}(t)$ in an ATI, it will notably deviate between manipulated values and non manipulated values.*

Postulate 4 is hereby the foundation of a successful detection using an estimation based approach for anomaly detection. The assumption that an anomaly is differentiable from the background noise so that a classifier or regression based approach can learn the normal pattern and differentiate it from outliers.

Furthermore a concept for a detection system should be designed using a fusion of different elements [Bass, 2000] to enhance detection. "*Besides, the inherent weakness of NIDSs, such as high false positives (FP) and high false negatives (FN), raises urgent requests on effective solutions. Data Fusion (DF), as a promising technology of big data, has been applied into the domain of network intrusion detection to overcome the [..] challenges in recent years*" [Li et al., 2018].

Hypothesis 1. *A detection concept with heterogeneous elements using different data sources and algorithms can lead to a more robust detection compared to an individual solution.*

3.4.1. Causality

As stated earlier, most of the work in literature (ref. to section 3.2) relies on the usage of statistical models, hypothesis testing, neural networks, support vector machines or state space models. A recently rediscovered approach using causal models [Granger, 2004] promises further potential for deterministic (man-made) systems and may be part of a well performing detection concept.

It is important to mention the difference between statistical and causal analysis. With the former it is possible to describe an observed behavior of a process and e.g. estimate variables based on the generated model or recognize patterns. Dynamic deviances of the process itself that were not observed cannot be predicted or recognized. The intention of causal analysis is the description of the generating process itself. A model derived from causal analysis is also able to estimate dynamic deviances. [Pearl, 2009] describes the difference as follows:

> *The aim of standard statistical analysis, typified by regression, estimation, and hypothesis testing techniques, is to assess parameters of a distribution from samples drawn of that distribution. With the help of such parameters, one can infer associations among variables, estimate beliefs or probabilities of past and future events, as well as update those probabilities in light of new evidence or new measurements. These tasks are managed well by standard statistical analysis so long as experimental conditions remain the same. Causal analysis goes one step further; its aim is to infer not only beliefs or probabilities under static conditions, but also the dynamics of beliefs under changing conditions, for example, changes induced by treatments or external interventions.*

Causality or causation is a widely used term and relates to the cause-and-effect principle. Several definitions have been published in different scientific domains like philosophy, medicine, mathematics, physics, computer science, engineering (system theory), economics or religious studies.

Generally spoken, causality means that two processes or events (also referred to as objects) are ordered and interconnected in a way, that the first one causes the second through a natural law or principle. In the case of an event this could mean that the second event had never happened, if the first did not occur [Hume, 1748]. Causality is also closely related to the term determinism, which is rooted in the 17th and 18th century and means the predetermination of an event by its previous conditions. Probably based on Gottfried Wilhelm Leibniz' principle of sufficient reason [Leibniz, 1686], Pierre Simon Laplace [Laplace, 1814] published

the idea of scientific determinism:

> *We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.*
>
> – Pierre Simon Laplace, A Philosophical Essay on Probabilities

The described *intellect* is widely known as Laplace's demon (in comparison to Maxwell's demon) and scientists like Werner Heisenberg and Albert Einstein tried to discover this stated *single formula*, also known as the Theory of Everything (ToE). Einstein tried to find the unified field theory, which combines his general relativity with electromagnetism, but failed. Heisenberg discovered instead the mathematical proof for his famous uncertainty principle which specifies a limit to measure multiple properties of a particle simultaneously, e.g. position and spin. This opposes the idea of determinism as stated by Laplace. Other discoveries of that time also opposed determinism, i.e. Wolfgang Pauli and his exclusion principle, which was named after him. These discoveries changed the basic concept of science towards a predictable nature. Developed in the 1920s, quantum theory contributed in this paradigm shift. In the late 20th century a major trend in science moved towards exploring chaotic systems. The foundations were laid much earlier by Henri Poincaré and Edward N. Lorenz, among others with the theory of nonlinear systems, which is a branch of mathematics of time.

While modern physicists [Hawking, 2016] tend to argue against the application of causality to describe natural processes within our universe, other scientific domains more recently and successfully demonstrated its application especially to man-made systems like economics [Granger, 2004]. Some basic definitions and models for causality in the field of statistics, probability theory, machine leaning and pattern recognition were given by, [Pearl, 2000, Pearl, 2009], [Spirtes et al., 2000, Spirtes, 2010] and a comprehensive introduction into the topic can be found in [Guyon et al., 2008].

[King et al., 2004] links network- and host based IDS together using causal dependencies. [Qin and Lee, 2004] applies Granger-Causality analysis to detect the "causal" relationship between types of alerts to detect attacks without prior knowledge of attack scenario patterns. [Alserhani et al., 2010] uses a Directed Acyclic Graph (DAG) as correlated attack graph for attack recognition in alerts of NIDSs. [Staniford-Chen et al., 1996] presents *GrIDS* that collects data about activity on computers and network traffic between them. It aggregates this information into activity graphs which reveal the causal structure of network activity. [Qiu et al., 2012] utilizes Granger graphical models for dependency anomalies in production fault detection.

3.4.2. Process Causality

Causality is defined in control theory [Lunze, 2008a] as characteristic of a system that is technically feasible. A system with the transfer function

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\sum_{m=0}^{Q} b_m \cdot s^m}{\sum_{n=0}^{R} a_n \cdot s^n} \tag{3.4}$$

is considered causal, if the order of the numerator Q is lower or equal to the order of the denominator R

$$Q \leq R \tag{3.5}$$

In that sense the output depends causally to the input, which means Y depends only on past and current values of U, but not future values.

$$\begin{aligned} u_1(t) = u_2(t) \; \forall \; t \leq \tau \\ G(u_1(t)) = G(u_2(t)) \; \forall \; t \leq \tau \end{aligned} \tag{3.6}$$

An example is shown equation 3.6. The dynamic system G is considered causal, if equates for any input function u_1 and u_2 within timespan $t...\tau$. This basically means that for the equal input values a causal system would respond with equal output values.

An extended definition was made by [Granger, 1969] which additionally defines the term Granger-Causality (GC) and includes further variables. In [Granger, 2004] they are called action X and reaction Y. The definition is as follows

1. The action X occurs before the reaction Y and
2. The action X contains explicit informations about the reaction Y, which cannot be found in any other (process-)variable.

It is a consequence of these statements, that the action-variable can contribute in forecasting the reaction-variable. With this in mind, causalities characterize the scope in which a process is causing another. Future values can be estimated superiorly by the use of past values and causalities, than only by the use of past values. The additional consideration of current values is called *instantaneous causality* and the generalized causal model for two variables is given as [Granger, 1969]:

$$\begin{aligned} X_t + b_0 Y_t &= \sum_{j=1}^{m} a_j X_{t-j} + \sum_{j=1}^{m} b_j Y_{t-j} + \epsilon_t, \\ Y_t + c_0 X_t &= \sum_{j=1}^{m} c_j X_{t-j} + \sum_{j=1}^{m} d_j Y_{t-j} + \eta_t \end{aligned} \tag{3.7}$$

where ϵ_t, η_t are taken to be two uncorrelated white noise series. This is also the underlying linear regression model of the standard GC-Test. It is criticized

not to characterize the relationship between cause and effect, since only mere statistical correlations between two variables are tested [Maziarz, 2015]. In general this model is well suited for finding dependencies among variables in time series, which eventually lead to good predictive models. However, statements about causal relationships can not be made.

[Simon and Rescher, 1966] showed that a causal relationship is a functional relation rather than a relation between values of variables. They stated

1. *The asymmetry of the causal relation is unrelated to the asymmetry of any mode of implication that contraposes.*
2. *A causal relation is not a relation between values of variables, but a function of one variable (the cause) on to another (the effect).*

The second statement is of importance. For the desired detection solution this means that a process variable Y, utilizing the physical dependencies within the process and the use of the causal processes variable X, can be estimated assuming causal dependencies.

The model used in equation 3.7 is only one example of a causal model. If a linear regression model is sufficient to represent complex physical processes with multiple nonlinearities has to be discussed elsewhere. [Pearl, 2000] defined a causal model as follows:

> *A **causal structure** of a set of variables V is a directed acyclic graph (DAG) in which each node corresponds to a distinct element of V, and each link represents direct functional relationship among the corresponding variable. A **causal model** is a pair $M = (D, \Theta_D)$ consisting of a causal structure D and a set of parameters Θ_D compatible with D. The parameters Θ_D assign a function*
>
> $$x_i = f_i(pa_i, u_i),\ i = 1, ..., n \tag{3.8}$$
>
> *to each $X_i \in V$ and a probability measure $P(u_i)$ to each u_i, where PA_i are the parents of X_i in D and where each U_i is a random disturbance distributed according to $P(U_i)$, independently of all other u. Once a causal model M is formed, it defines a joint probability distribution $P(M)$ over the variables in the system. This distribution reflects some features of the causal structure [..] Nature then permits the scientist to inspect a select subset $0 \subseteq V$ of "observed" variables and to ask questions about $P_{[O]}$, the probability distribution over the observables, but it hides the underlying causal model as well as the causal structure.[..] Therefore, with no restriction on the type of models considered, the scientist is unable to make any meaningful assertions about the structure underlying the phenomena.*

With this in mind the function that connects two variables defines the causal relationship. Automated processes run in a cyclic and predictable manner. Most relations are implemented in and calculated by the automation software itself, which mostly contains a model of the process to control itself. This advantage

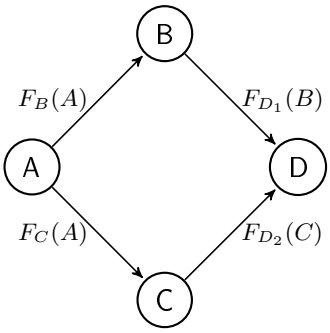

Figure 3.4.: Transfer functions added to a causal model

to conventional causal modeling can be used and should also be reflected by the causal model, which can be built using this already available information. The type of model itself may be chosen specifically for each application.

Definition 1. Process causality refers to causal relationships between variables of an automated process. The associated causal model is represented by direct functional relationships between the corresponding variables.

For the desired application the causal model can be built using causal relationships between the process variables (as reflected by the DAG) and its respective transfer functions (ref. to formula 3.8). Fundamental physical relations and control strategies can be used as transfer functions to cross-validate data between process variables that cause each other to detect anomalies. Figure 3.4 and example 3.4.1 illustrate this.

Example 3.4.1. A water reservoir has an inflow Q_i and a drain Q_d and the volume V is caused by both variables $Q_i \rightarrow V \leftarrow Q_d$. Using this causal relationship alongside with the physical transfer function for the volume V of the water reservoir, which is a function over time:

$$V(t) = V_0 + \int_0^t \Delta Q(t) dt \tag{3.9}$$

with starting volume $V(0) = V_0$ and $\Delta Q(t) = |Q_i - Q_d|$, the aforementioned estimation (ref. to figure 3.3) for the volume $\tilde{V}$ can be made. From this a manipulation of the effect value V can be detected by calculating $\delta(t)$ (ref. to formula 3.3).

4. Methodology and concept design

"Though human ingenuity may make various inventions which, by the help of various machines answering the same end, it will never devise any inventions more beautiful, nor more simple, nor more to the purpose than Nature does; because in her inventions nothing is wanting, and nothing is superfluous, and she needs no counterpoise when she makes limbs proper for motion in the bodies of animals." – Leonardo da Vinci, *Codex Arundel* (1478-1518)

The lack of a widely recognized methodology to develop a concept for securing specific Automation Technology Infrastructures (ATIs) (ref. to chapter 3) leads to the threat situation as outlined in chapter 1. To develop a methodology to derive a comprehensive and application specific concept to detect anomalous values within the cyber-layer of complex and distributed automation processes based on real scenarios and requirements existing methods have to be analyzed. Based on this analysis a synthesis can be conducted to derive the required methodology. Subsequently it can be utilized to develop a concept for real use cases. This chapter provides these steps to develop and prototypically use this methodology and its respective concept to secure ATIs against cyber-threats. First existing methodologies get presented and a novel one synthesized from these. The prototypical usage derives a general concept based on existing application scenarios.

4.1. Methodology

Since the construction of the first hand Axe [Semenov and Mitja, 1981] methodologies are used to develop and construct goods. These methodologies were historically often just used and not written down. To address the research goal of this thesis (ref. to section 3.3) related methodologies have to be analyzed to derive a novel method to create a concept to secure ATIs. This includes the scientific method, as commonly used by scientists, and other methods from system or software development, best practice based on international standards and specific application-oriented methodologies as used in research projects like STEUERUNG [Horn et al., 2014], pICASSO [Vick et al., 2015] and RetroNet [Horn and Krüger, 2016a].

4.1.1. Scientific method

The *scientific method* is defined in [Oxford, 2004] as:

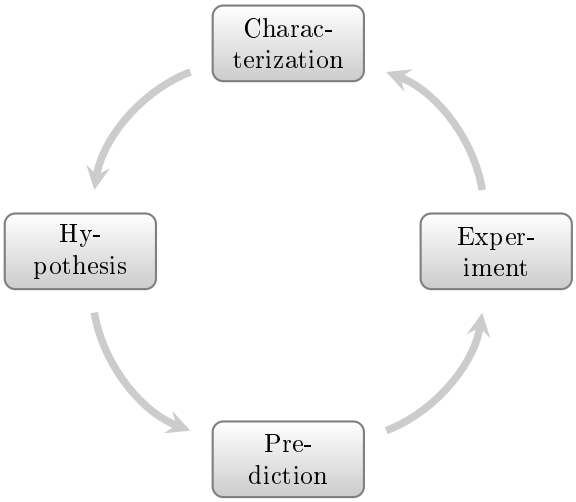

Figure 4.1.: General scientific work-flow

> *A method of procedure that has characterized natural science since the 17th century, consisting in systematic observation, measurement, and experiment, and the formulation, testing, and modification of hypotheses.*

This is also referred to as the *hypothetico-deductive method* [Gower, 1997]. This general process consists basically of the elements shown in figure 4.1. **Characterization** is the first step of the method. It contains observations, definitions, measurements and analysis of the research subject. With this a **Hypothesis** can be formulated, which is "*A supposition or proposed explanation made on the basis of limited evidence as a starting point for further investigation.*" [Oxford, 2004]. This enables to make a **prediction** which is an assumption about the outcome of an **experiment** based on the hypothesis. The measure for the hypothesis test is crucial and should be taken into account already formulating the hypothesis. The outcome of the experiment is then used for a discussion and enables further characterization.

4.1.2. System or software development methodologies

Several different methodologies especially for constructing large software systems have been developed over the last decades and comprehensive overviews are available [Office of Information Services, 2005] [Cohen et al., 2003] [Ruparelia, 2010].

Sequential methods like the Waterfall model [Royce, 1970] or its later developed extension to the V-Model [Forsberg and Mooz, 1991] divided the process of development into different sequential phases. Each phase is based on the outcome of the preceding one. This methodology relies on stable requirements and is inappropriate for flexible development under fast changing conditions [Office of Information Services, 2005].

Iterative methods like prototyping or Rapid Application Development (RAD) [Ruparelia, 2010] attempt to reduce risks by dividing the desired system into smaller segments to react more flexibly to changing conditions during the development process. This flexibility is achieved at the cost of a well defined design and robust implementation. This approach is especially not recommended to develop safety critical or computationally complex systems [Office of Information Services, 2005].

Incremental methods like the spiral method [Boehm, 1986] combine sequential and iterative approaches by dividing the process of development into different sequential phases and the desired system into smaller segments. This system segments cycle though the different phases incrementally. "*Each cycle involves a progression through the same sequence of steps, for each portion of the product and for each of its levels of elaboration, from an overall concept-of-operation document down to the coding of each individual program.*" [Boehm, 1986] This approach focuses on risk avoidance and is best used for systems that need a high degree of accuracy during development due to safety critical applications of the desired system.

Agile methods like Scrum [Cohen et al., 2003] are based on the values of the Agile Manifesto [Beck et al., 2001]. They focus on an evolving development process under changing requirements through daily briefing and review. Teams are self-organizing and consist of members with different functionalities. Iterative changes are frequently released with fixed schedule. This approach is used for timely development of complex software systems under changing conditions.

Lean methods like Kanban [Ahmad et al., 2013] were developed originally as a scheduling system for lean and just-in-time manufacturing for the car production industry. In software development lean methods focus on iterationless development and the lean principles [Anderson, 2008]. Lean methods are often considered agile methods.

4.1.3. Best practice based on International Standards

International Standards like the ISO/IEC 27000 and ISA-99/IEC 62443 families define best practice recommendations on information security management. Especially the standard 62443 recommends measures to improve digital security and safety within production and SCADA environments. It is recently derived from the ISO/IEC 27000 standard family with focus on Industrial Control Systems environments.

While both standards families offer a wide range of recommendations, the methodology described is a cyclic process. Figure 4.2 shows the Plan-Do-Check-Act (PDCA) cycle that is the foundation for implementing recommendations like ISMSs. The following descriptions are adopted from [ISO27004, 2009]:

Plan is similar to a combined analysis and conception phase. Here measurements, assessments and objectives are determined and selected.

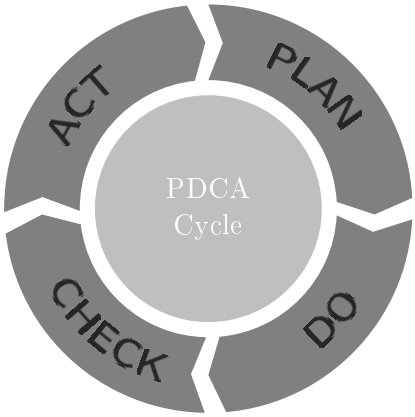

Figure 4.2.: PDCA cycle according to [ISO27004, 2009]

Do is similar to an implementation phase. The aforementioned selections are put into practice.

Check equals an evaluation phase. Measurements are reviewed and effectiveness determined. Therefore possible improvements arise.

Act means to implement these improvements.

4.1.4. Specific application-oriented methodologies

Within the research project STEUERUNG [Horn et al., 2014] the goal of the sub-project "*process-security*" was the development of methods for automatic detection of anomalies in the entire process of a critical infrastructure. In order to achieve this, the different use cases were used to combine the characteristics and causalities describing the respective process into a meaningful pattern, the "process fingerprint". A challenge is complexity of processes in Critical Infrastructures, therefore an application specific methodology was used implicitly with respect to the possibilities and limitations of uninterrupted operations within supply infrastructures. The methodological steps conducted during the research project were as follows:

1. collect all possible information concerning the hardware, networking and software systems used,
2. select a relevant partial aspect of the whole infrastructure to reduce overall complexity,
3. determine the relevant process parameters,
4. create TechMaps,
5. collect small samples of initial real data from critical points (i.e. central network hubs etc.),
6. analyze the data concerning requirements on methods of anomaly detection,
7. collect relevant parts (i.e. over several days) of real data including meta-

data for labeling,

8. labeling and methods development,
9. evaluation in simulation environment using different scenarios based on collected data.

Since the focus of the project STEUERUNG was not to develop a methodology, this practice-oriented approach emerged. Following that this was refined within the research project RetroNet [Horn and Krüger, 2016c], where the focus was explicitly on developing a methodology. Especially relevant international standards (ref. to table 2.2) were analyzed to improve the methodology. This resulted in the following modified steps:

1. protection requirements analysis,
2. requirements engineering for concept,
3. risk assessment (identification, analysis, evaluation),
4. create TechMaps,
5. vulnerability and weak points analysis,
6. attack scenario development based on worst-case scenarios,
7. create attack scenario based TechMaps,
8. effects analysis (business indicators, financial ratios and need for action),
9. selection and implementation of security tools,
10. evaluation.

Layer	Protection requirements	Concept requirements	Risks	Effects	Counter-measures
channel	medium, protocols	authenticity, availability	disturbance, manipulation		
platform	access, configuration	availability, accountability	disturbance, manipulation		
software	binaries, source	integrity, confidentiality	manipulation, eavesdropping		
data	process-variables	integrity, confidentiality	manipulation, eavesdropping		
systems	physical access	availability, authenticity	disturbance, destruction		
human	knowledge, accounts	authenticity, confidentiality	manipulation, eavesdropping		

Table 4.1.: Example for a methodological matrix (intentionally incomplete)

These steps were performed for different layers like communication channel, cloud-/ service-platform, application-software, data, physical systems and operators as human factor. The layers itself were also developed application-specific. This resulted in a matrix structure, that can be developed application-oriented

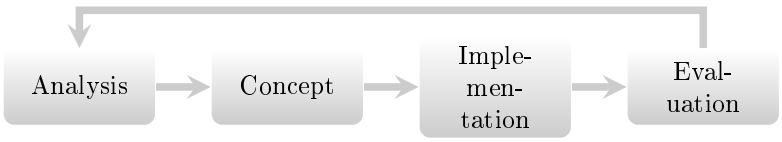

Figure 4.3.: Methodological development steps for an entity

cooperatively by application partner and security consultant. An incomplete example is given in table 4.1, since this methodological matrix cannot be abstracted from a specific use-case. Focus of the research project RetroNet was not on developing new security tools, but using existing ones instead, the methodology can also not be used directly for this work.

4.1.5. Synthesis of methodologies

All of these methods have in common that an analysis has to be done as the first step for determining requirements of the desired solution. With these a basic concept can be developed to implement a prototype. The evaluation of the prototype is then the foundation for a new analysis, or if the implemented system meets all desired requirements, the development process is done. The development is followed by integration and maintenance. Finally a best practice based on relevant international standards, as outlined in table 2.2, has to be followed by the organization using the developed system.

The development steps shown in figure 4.3 are present in all the aforementioned methodologies. They can be shortened or extended, processed sequentially or in parallel, as well as repeated multiple times similarly to the spiral method. An incremental approach has the advantage that each time the prototype gets improved and the requirements revised.

4.2. System analysis

The analysis of large infrastructure systems requires methodological approaches that take the complexity into account [Hempel et al., 2015]. Infrastructure systems can be studied in their respective structural and technological design. Researchers can rely on different methods to gain deeper insights into the functionality as well as vulnerability and risk status. This approach can be described as *System Analysis*. It is the foundation of all subsequent work. Its extend and quality define the applicability, usefulness, power and accuracy of any developed solution based on it.

Figure 4.4 shows the methodological steps that were performed during this work. The sequence starts with information collection, which is also the first phase of an attackers approach, as shown in figure 2.1 and described in section 2.1.2. The information collected can then be used to create TechMaps, which are the foundation for an extensive analysis and subsequent data capture at relevant

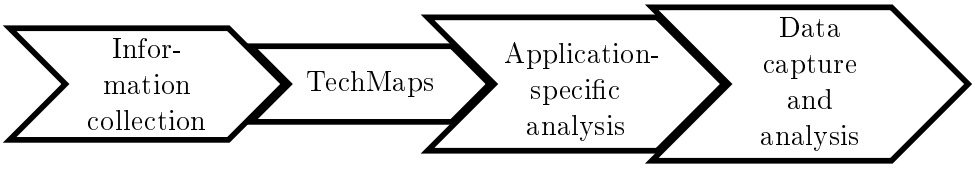

Figure 4.4.: Methodological steps for system analysis

locations. The examination of captured and labeled data concludes the system analysis and enables the development of a detection concept.

4.2.1. Information collection

Like an attacker the information collection is fundamental for successful further advancement in developing a functional solution. Referring to postulate 3 the defender should be able to get a greater extend, more detailed and more exact information more easily compared to an attacker. Several different tools are available to both and include:

Documentation screening of blue-prints, planning sheets, technical papers, manuals, user policy handouts, topological and structural documents of an infrastructure reveal substantial information. Usually the available documentation is very extensive but mostly not complete, since the establishment of a complex ATI takes place over a longer time and structure or elements also change over time. These deviations from planning are sometimes not documented sufficiently, which can be a disadvantage for an attacker that relies on the information as documented.

On-site inspection means the examination of a specific location and enables an overview of structure, interaction, integration, configuration and accessibility of all elements in this entity of an ATI. An example is one electric power transformer substation in an energy supply infrastructure and how its technological, infrastructural and human elements are composed. An on-site inspection can additionally give better insights than just studying the documentation itself.

Workshops are a form of guide-lined and moderated meeting that is used to elaborate information details regarding a specific topic. All participants can benefit from this, i.e. if a security practitioner conducts a workshop with operating personnel regarding attack scenarios. The security practitioner will get better insights to the infrastructure itself and what needs to be protected the most (or is worth attacking), while the operators learn about how attackers proceed and how to protect themselves against social engineering.

Source code screening can be done only with full access to the source code. A detailed review reveals particular insights on how a process works and which calculations or measures are processed automatically. While an

attacker can use this knowledge to launch more devastating attacks, a defender can include this knowledge into a countermeasure, i.e. to detect attacks.

Expert interviews are used in empirical social research in particular to get specific and concentrated knowledge of selected persons to a limited query subject. A guideline is usually used to structure the conversation for content and sequence. The choice of a right interview partner is hereby crucial. Experts are competent persons who have specific knowledge of action and experience in the field under investigation. They represent certain organizations or institutions and have specialized internal organizational knowledge (business knowledge), which is usually not documented.

Questionaires are less laborious and time consuming than a specific expert interview, making it easier to examine a larger number of individuals across different organizations for statistically reliable statements. In this work a simple lists of questions was used for documentation or evaluation of answers in terms of content in contrast to standardized and test-methodically constructed questionnaires from social and psychological science.

Training is similar to a lecture and is intended to increase the basic knowledge for a specific application or use-case. An example is a security practitioner attending a regular training for operators of a specific ATI. This way he can not only get technical knowledge itself, but also adopt domain-specific language and behavior of operators.

Experimental games need a screenplay which contains scenarios based on prior analysis of the infrastructure and operation context. If available a technological simulation environment (testbed) can contribute to increased immersion and authenticity for the participants, since a necessary constraint is the provision of realistic communication paths. The coordination and execution of the game includes a wide range of complex tasks, including the spatial-technical setting, the training of game supervisors and the development of the screenplay itself. During the experimental game participants will be confronted with a disturbance along the preparatory scenario, observing how they can get a picture of the new situation and coordinate measures to eliminate the disturbance. Furthermore a training effect for the participants is present.

All these tools can be used for information gathering. Their sequence is hereby not fixed and a specific procedure has to be developed application-specific. Example 4.2.1 shows an approach used in the context of the research project STEUERUNG (ref. additionally to appendix A.1).

Example 4.2.1. To develop a security concept by an external security practitioner for natural gas supplier CI of a huge urban area, the first step was a workshop to gain basic information. This included topics from the regular training for dispatchers/ operators of the CI. Thus, an overview of the components and interfaces for the control system software was elaborated. A following documentation screening enabled the sketching of basic structural and techno-

logical context, which then could be discussed in a second workshop. Within that the overview of infrastructure was discussed and requirements were specified. Furthermore details of individual processes in possible sub-aspects of the overall structure were elaborated. With this information a first version of a TechMap could be developed. The subsequent detailed code analysis brought critical process parameters to light. It was carried out with external programmers (contractors), because the extensive code base contained closed source, which could not be analyzed without these contractors. During this analysis many questions could be answered. Based on this, a data capture was possible. However, the analysis of this data revealed that further data capturing was necessary due to the recorded period of time, which contained mostly maintenance works and therefore parts of the infrastructure were disabled.

In order to avoid unrealistic or even scenarios with too little complexity, a fundamental understanding is essential. The system analysis is based on information about the systems, their elements, as well as functional and informational relationships, ranging through the aforementioned range of techniques.

4.2.2. Technological Maps

A novel tool for a system analysis of large and complex infrastructures developed during this work is the Technological Map (TechMap) [Horn and Krüger, 2016b]. It contains a complete description of the technological elements of a complex information, communication and automation infrastructure, as well as their connection and interaction. This can be done individually or along a certain taxonomy, such as the automation pyramid. In the first step, the collected information is used to create graphical elements similar to a map. These are described in a second step from a technical-functional point of view and finally supplemented in the third step by further information on the general organizational context of the infrastructure.

In other words, a TechMap contains finely granulated representations of the entire[1] system and provides information in a comprehensive way about possible modes of operation that are executed by operations and process control system. To this extent a TechMap represents the respective state of a complex infrastructure at a specific time, although the descriptions in particular make it possible to identify the history of system configurations, which can further provide information on security gaps.

TechMaps consist of graphic elements, detailed technical descriptions and descriptions of general relationships. In figure 4.5 the graphic representation of a control center is shown as an example and how it is embedded in the overall network of the infrastructure. The general descriptions show how the architecture is embedded in the technical organizational structures of the entire system and the connection of different entities within the infrastructure. It is shown how, for

[1] According to the objective maybe only an excerpt from the overall system, such as the operations control center itself.

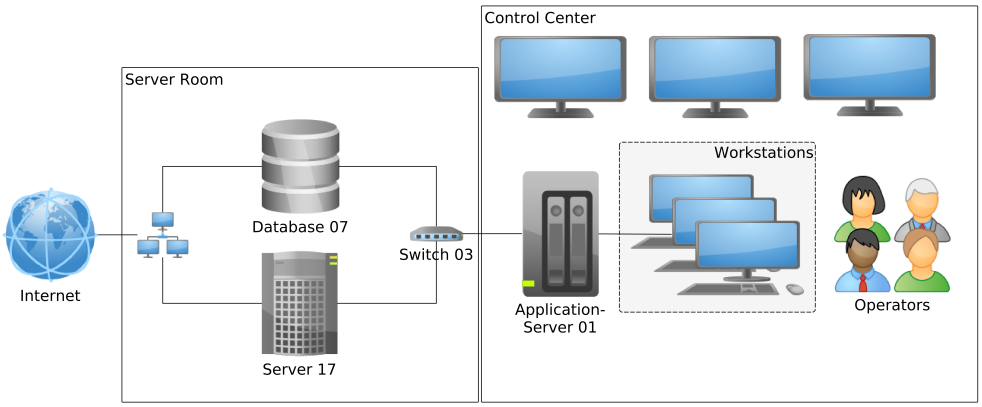

Figure 4.5.: Operations control center as exemplary graphical element of a TechMap [Horn and Krüger, 2016b]

example, the decentralized periphery and technical components are distributed in the service area and connected to one another and also to the respective control center. Decentralized peripherals and technological components include entire plants, tele-control substations, individual PLCs, machinery, sensors and actuators in the field and their connection channels and protocols. Detailed technical descriptions according to the example in figure 4.5 would reflect the precise characteristics of software and hardware components, communication channels and -protocols, e.g. "*Server17 using processor type XY and Z MB RAM and is running operating system A and application software B, C, D, and using communication protocols* σ, ι, τ *etc.*".

The ensemble of detailed descriptions for overall structures and individual elements enable superior analysis. In particular, interconnection and configuration of systems provide important information and insights for any security analysis. With a TechMap an image of the infrastructure is created which an attacker would make similarly to reach his target. He reads in system architecture like other people orient themselves in cities. Doing so, he may also use programs that spy out the technical landscape for him. The TechMap enables a defender to possess more detailed information than an attacker could usually get - and this is the actual message of the technical maps at this point. This is the foundation for a successful defense strategy. Especially the history of a technical landscape becomes available with different iterations of a TechMap over time. It provides information not only about the currently installed technology, but also about the security philosophy a system was created with. Decisions and contradictions emerge, which over time have led to changes in configurations and installations. The sketched landscapes are complex symbolic systems that provide detailed information about the organization.

A TechMap forms the basis of possible developments towards a realistic simulation environment (testbed), which can also be used for real simulation exercises and personnel training. With this testbed it is possible to conduct further studies on the development of security methods and their evaluation with regard to

the significance of individual parameters related to the overall process.

4.2.3. Application-specific analysis

The previous steps of data collection and preparation form the foundation for an analysis. Having a detailed and comprehensive TechMap at hand this analysis can be done superiorly. Further analysis can contain all or a subset of the following practices:

Research and Development goal definition should be done as foundation for any further analysis. The issuing principal organization should at least know, what needs arose from daily practice towards improving or securing processes and organizational context. These can be formulated as goals to a desired solution that has to be researched or developed.

Protection requirements analysis has to be done especially for securing an ATI. Critical assets have to be defined which are especially sensitive towards a necessity of protection. For example a company that has certain knowledge about a production process of a special product usually wants to keep this knowledge since it is its trade secret. Also the production machinery itself has to be protected, since a manipulation of the process parameters could lead to quality or financial loss.

Risk assesment is a "*systematic approach to information security risk management [..] regarding information security requirements [..]. Security efforts should address risks in an effective and timely manner where and when they are needed. [..] Information security risk management should be a continual process. [..] Risk management analyses what can happen and what the possible consequences can be, before deciding what should be done and when, to reduce the risk to an acceptable level. Information security risk management should contribute to the following: risks being identified, risks being assessed in terms of their consequences to the business and the likelihood of their occurrence, the likelihood and consequences of these risks being communicated and understood, priority order for risk treatment being established, priority for actions to reduce risks occurring, stakeholders being involved when risk management decisions are made and kept informed of the risk management status, effectiveness of risk treatment monitoring, risks and the risk management process being monitored and reviewed regularly, information being captured to improve the risk management approach and managers and staff being educated about the risks and the actions taken to mitigate them.*" [ISO27005, 2011]

Requirements engineering should be done to survey the technological, functional and organizational properties of the desired solution. International Standards like [ISO/IEC27001, 2013] should be used as starting point since it "*specifies the requirements for establishing, implementing, maintaining and continually improving an information security management system within the context of the organization. This International Standard also includes requirements for the assessment and treatment of information se-*

curity risks tailored to the needs of the organization. The requirements set out in this International Standard are generic and are intended to be applicable to all organizations, regardless of type, size or nature."

Vulnerability assesment means to identify and manage vulnerabilities within the infrastructure and is based on a risk assessment. According to [ISO/IEC27000, 2014] a vulnerability is "*a weakness of an asset or control that could potentially be exploited by one or more threats.*" [ISO/IEC27001, 2013] and [ISO/IEC27002, 2013] state " *Information about technical vulnerabilities of information systems being used should be obtained in a timely fashion, the organizations exposure to such vulnerabilities evaluated and appropriate measures taken to address the associated risk.*"

Attack scenario development is based on a vulnerability assessment and defines how different attacks on the infrastructure could look like and what consequences arise for the organization. This combined with an evaluation can provide information on how to improve efficiently the security with coordinated minimal efforts.

Depending on the research and development goal different application specific sequences can arise. Two examples are the research projects STEUERUNG and RetroNet, as detailed in section 4.1.4.

4.2.4. Data capture and analysis

To implement a solution for a specific research and development goal, the respective data has to be captured to gain an understanding of the technological language of the system itself. Especially for designing a monitoring system as part of an IDS all available data sources have to be analyzed and evaluated for later usage. The tools for doing so range from freely available software like Wireshark [Combs, 2018b] to specialized proprietary software. Figure 4.6 shows a specialized tool for capturing packets on an obsolete field-bus named Profibus, which is still present in some application scenarios. The field-bus is very different from usual Ethernet traffic and can only be captured or analyzed with special hardware and software. The following data is usually present and can be captured and analyzed:

Network data can usually be captured straightforward including different networks like real-time field-buses or standard Ethernet networks. The packets contain usually a packet header and data. The header gives additional information, while the data frame contains application data. Reconstructing the application data also usually requires the capture of handshakes and initial transmissions, as well as the correct time-line.

Process images are on-line representations of all variables, sensor readings and control commands present in memory of control devices like PLCs itself. The values are only valid within the cycle time of the device, i.e. $10 Milliseconds$. After that new readings are performed and these values possibly change. To capture these process images externally freely avail-

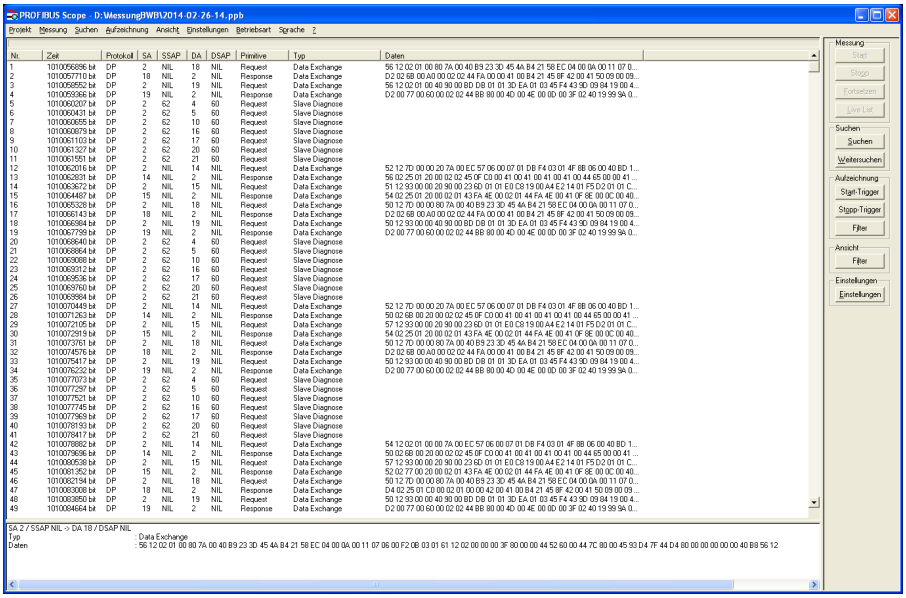

Figure 4.6.: Profibus network packets captured with specialized tool

able software libraries like Snap7 [Nardella, 2015] or ADS [Beckhoff, 2017] can be utilized to communicate with PLCs.

Process databases contain aggregated or preprocessed values close to the control systems itself. The data is usually only available for a specific amount of time (i.e. 24 hours), long term storage is done in another location. Timely snapshots during those intervals can provide valuable details.

Application databases usually contain information concerning operation control, business relevant data as basis for processes like billing and user context. Here the foundation for developing specialized attack vectors can be found.

Long-Term Archives usually contain shortened versions of application and alarm databases for long term storage. These archives can provide valuable information about a historic context and a development of the infrastructure.

Alarm- and messaging protocols contain event based data with additional information in case of alarms, warnings or other activities within the infrastructure. This data is especially valuable for labeling.

Other log data like operating system log files etc. can be used to enhance the contextual information of an event.

Side channel features like Central Processing Unit (CPU) load, memory allocation, heat signatures of computing elements, acoustic readings or power consumption rates can additionally be used (if captured) for developing patterns for specialized monitoring systems, where a live monitoring of the process values itself is not possible.

Time stamps and their synchronization is essential when capturing and correlating data from different sources.

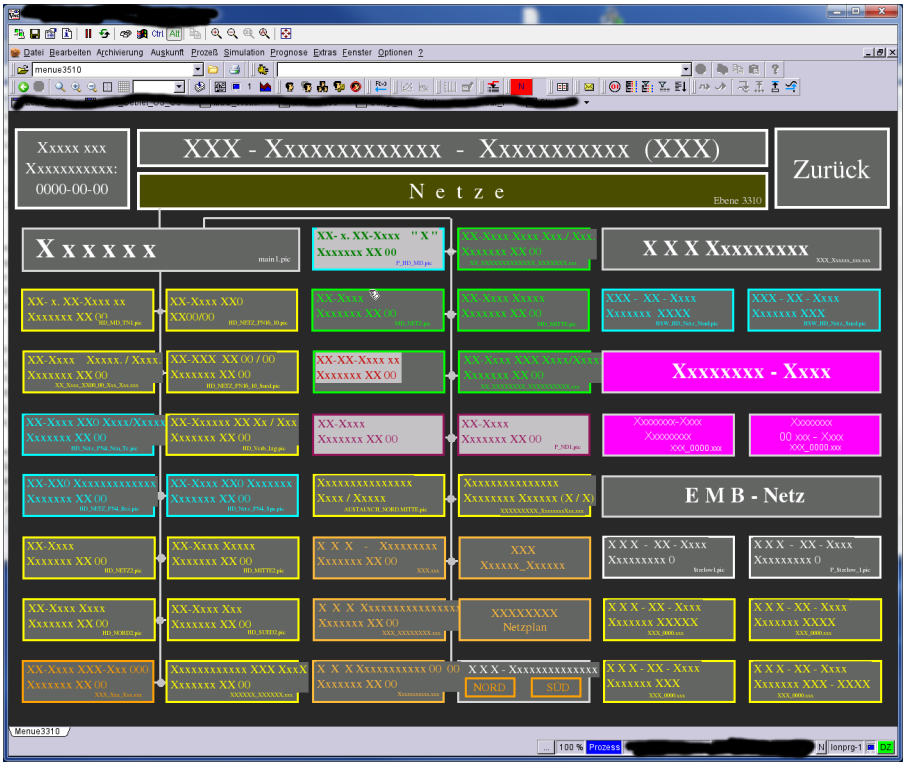

Figure 4.7.: Excessively anonymized operation control system

Several operating companies also possess different policies especially for capturing and analyzing all kinds of data. During information gathering phase not only technological, but also organizational challenges have to be identified and best eliminated or circumvented to reach the desired goal.

Figure 4.7 shows an example of an operations control system that was anonymized by the issuing company before submission to researchers. While these tried to study the system itself, correlations and interplay of data and operations control, this excessive anonymization rendered any further research impossible. The simple measure of replacing most of the character strings within the application with "XXX" combined with the lack of time and will to give further information or detail obliterated half a years work of preceding analysis. This shows that some challenges can arise, that make further research impossible, especially when it relies on detailed data, i.e. for training a classifier. The following challenges concerning data were met during this work:

- proprietary and closed interfaces prevent access to the data
- excessively anonymized data prevents analysis
- descriptive meta data (i.e. data models, characteristic curves and units) is not available which prevents an understanding of the system
- data capture is incomplete, relevant data is missing (i.e. third party involved)

- manual analysis of data is too complex
- automated analysis leads to false results or correlations
- relevant patterns cannot be observed or interpreted
- labeling of data is not possible due to the lack of expert knowledge

These challenges usually lead to another iteration of the whole system analysis, some parts of it or only further data captures. The lessons learned can then be utilized to gain better results incrementally.

4.3. Design of concept

This section provides information about application of the previously developed methodology to design a concept for detecting unknown attacks towards an Automation Technology Infrastructure (ATI). First generalized application scenarios are presented from the available eleven use cases from three different research projects. Second the requirements, risks and attack scenarios are shown. Based on this information, a detection concept is developed and presented. The detailed description of its elements concludes this chapter.

	STEUERUNG	pICASSO	RetroNet
Security research and development goal	Detection algorithm for CIs	Security architecture for Cloud-based control services	Security concept for retrofitting methodology
No. Use-Cases/ TechMaps	4/ 4	4/ x	4/ 2
Requirements	✓	✓	✓
Risks	✓	x	✓
Attack scenario	✓	✓	✓
Data capture from real process	✓	x	(✓)
Testbed	✓	✓	(✓)

✓ available (✓) partly available x not available

Table 4.2.: Availability of data from different research projects

The available data from different research projects is shown in table 4.2. The use cases include four distributed ATIs in CIs: Gas, water (2x) and power distribution and six ATIs from production use cases (huge, medium and small companies). Additionally, an academic use-case where an industrial robot gets directly connected to services on the Internet was examined. The academic use-case was present in both research projects, pICASSO and RetroNet, therefore is counted twice in table 4.2. Additionally, the captured data and testbed are only available from this academic use-case. Therefore the respective check-marks are in brackets, since the transferability can be questioned. Furthermore the risk

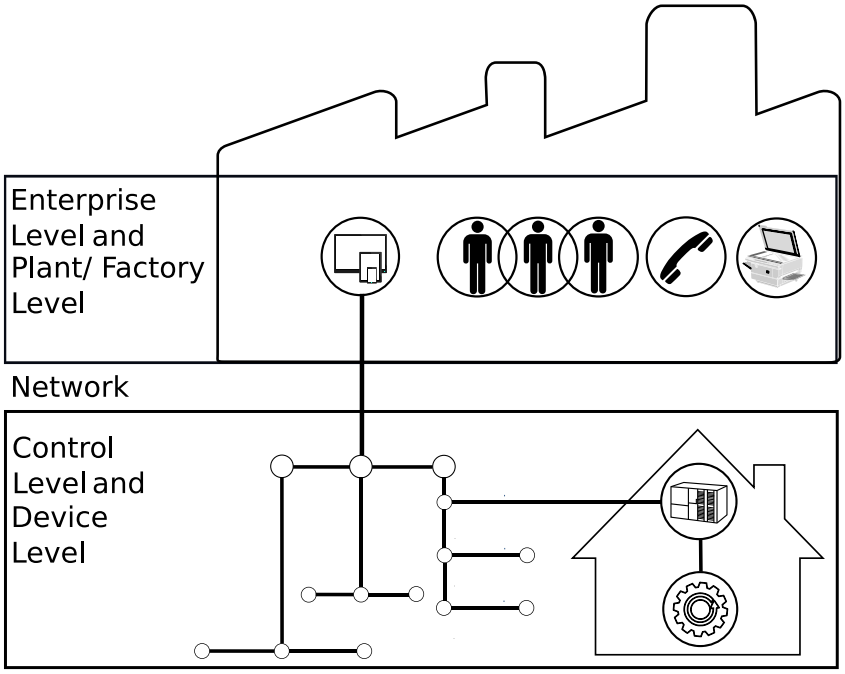

Figure 4.8.: Scenario A – distributed Automation Technology Infrastructure (ATI)

analysis and attack scenario development in pICASSO are available but done without participation of this author (check-marks also in brackets).

4.3.1. Application scenarios

In the research projects STEUERUNG [Horn et al., 2014], pICASSO [Vick et al., 2015] and RetroNet [Horn and Krüger, 2016c] several end-users contributed a total of eleven different ATI application scenarios. While the detailed information is protected under the respective Non-Disclosure Agreements (NDAs), a generalization is made here leading to two different scenarios.

Scenario A: Distributed Automation Technology Infrastructure

In scenario A, as shown in figure 4.8, the ATI is distributed over a large area. The company operating the infrastructure usually supplies an urban area with water, oil, gas or power and is grouped into two levels. The headquarter on the enterprise and plant level contains management, administration and operations within the same complex. Operations take place in a separated network and secured control rooms with specifically trained operations personnel. The operations control system is designed redundantly and two functionally identical control rooms exist in different locations.

The operations high level control system consists of several logical modules that can be distributed on different servers and workstations, so several operators can carry out different tasks in the control room. Proper storage

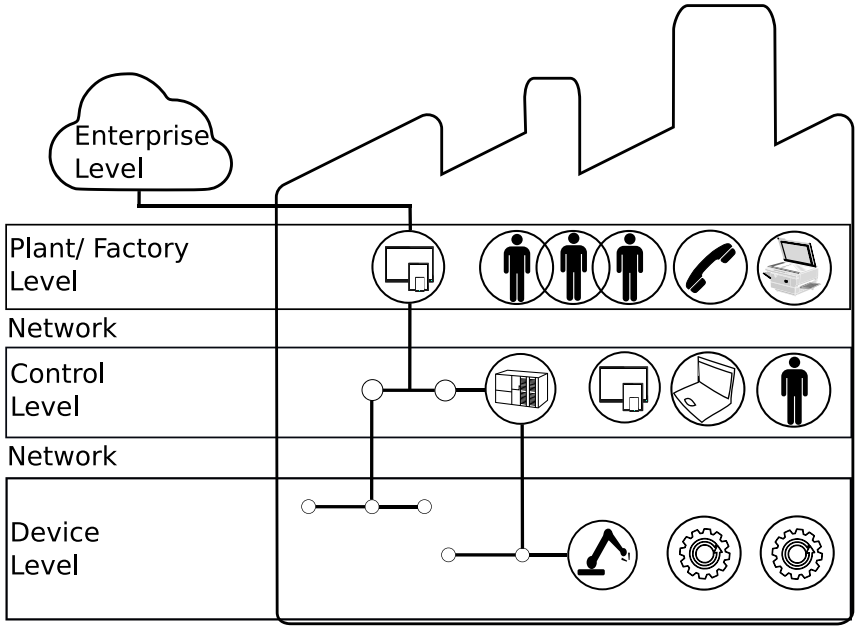

Figure 4.9.: Scenario B – localized Automation Technology Infrastructure (ATI)

is enabled through a redundant database, usually named by operators as historic archive . The high level control system connects to the tele-control master station (head), which connects to the remote stations, transmits commands and receives process values from them. The tele-control coupling is done via tele-control heads, which use the standardized protocols IEC 60870-5-101 and IEC 60870-5-104, where the latter requires a TCP/IP connection.

On the logical lower control and device level the network is composed of sensors and actuators, which are connected to control devices (usually PLCs) in the remote stations. The control devices are connected to tele-control terminal stations. Inside the remote stations basic testing and processing steps are carried out to enable critical consistency checks of process variables to prevent physical damage to the system.

Scenario B: Localized Automation Technology Infrastructure

Scenario B, shown in figure 4.9, represents an ATI within a local area, usually a factory or plant that produces a product – ranging from bent-pipes or full engines to pharmaceuticals. All lower levels of the Automation pyramid structure are present in the factory or plant location, only the enterprise level services like Manufacturing Execution System (MES) and Enterprise Resource Planning (ERP) are present in a distant (and central) location. The factory itself can be one of many within the enterprise.

The local complex usually consists of an office part (i.e. higher floors or separate building) connected to machinery halls.

The IT department is centralized and manages enterprise-wide policies, data and services. Locally synchronized data like hourly production plans for individual factory assembly lines are available in case of network downtime, but all data is stored centrally. The different parts of the factory are reflected within the networking topology in terms of segmentation and perimeter defenses. The central enterprise network is connected to the office network, which is connected to multiple shop-floor networks for the assembly lines. Internet access is usually routed through the enterprise network by policy, but many other access points are usually available (i.e. remote terminals via cell phone networks operated by machine vendors). Usually the shop-floor network contains wireless networks and its protection level is the lowest of all, since the usage of security tools on that level could jeopardize production timings. On the shop floor sensors, actuators, control devices, machinery, terminal stations and IoT-devices are present. The focus is set on functionality and availability. The goal is to maximize production capacity and minimize costs to do so. Therefore all entities that do not directly contribute to these goals are omitted.

4.3.2. Requirements

Based on qualitative expert interviews [Mayring, 2010] as well as an analysis of the TechMaps, requirements for eleven different real ATIs were identified (ref. to appendix A.1). These include four distributed CIs in application scenario A: gas, water and power distribution and seven production use cases in application scenario B: a huge, a medium, a small company and an academic robotic use case. The requirements identified are as follows:

Direct costs include all expenses that are related to acquire, develop, implement and integrate the solution. In simple terms: to get it up and running.

Scalability means the capacity of a desired solution to be adjusted for size or scale. Especially for ATIs this is an important requirement, since the use cases range from very small to large scale infrastructures.

Ressource efficiency is related to the amount of resources like computing power, memory footprint or even personnel capacity is used by the solution.

Operating expenses include all costs for running the solution over time, i.e maintenance costs, license fees and the like.

Ease of use determines the level of knowledge that is necessary for the end-user, if a technical layman can use it or an expert is needed. It also contains the usability itself.

Portability describes how well the solution can be ported to different runtime environments like operating systems, service platforms or system architectures.

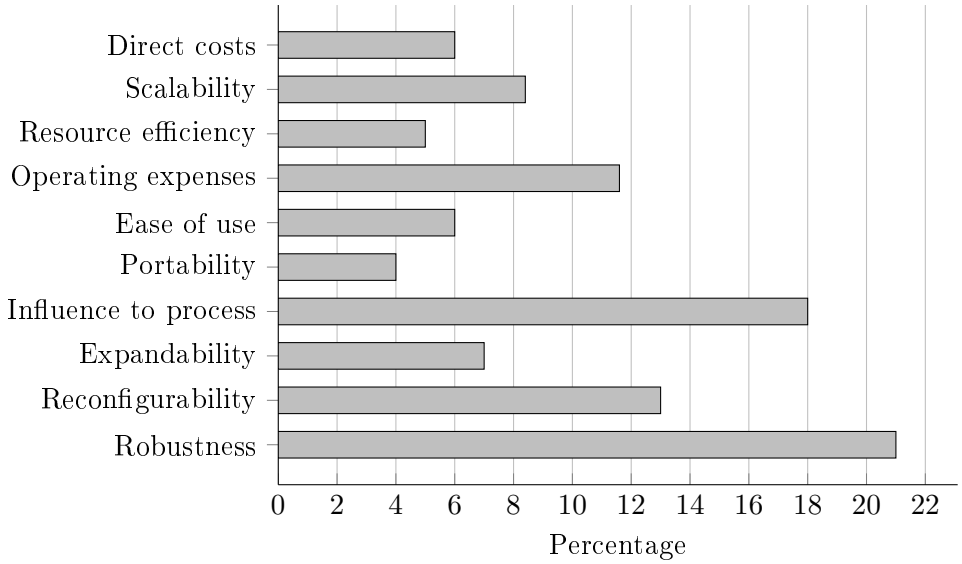

Figure 4.10.: Requirements weighted by qualitative expert interviews

Influence to process contains possible feedback effects to the process the solution will be integrated into.

Expandability means how well the solution can be updated to more recent developments.

Reconfigurability means how well can the solution be adapted to the specific application context, for example to different system layers.

Robustness of the solution even against changing conditions. For developing a detection framework this means e.g. a reliable detection rate.

Following that questionnaires were used to weight the requirements according to their relevance in practice. Several experts from different organizations contributed to the result shown in figure 4.10, where the average of all answers was calculated.

4.3.3. Risks

According to [ISO27005, 2011] risk management includes three activities:

1. Risk identification
   "*The purpose of risk identification is to determine what could happen to cause a potential loss, and to gain insight into how, where and why the loss might happen. [..] Risk identification should include risks whether or not their source is under the control of the organization, even though the risk source or cause may not be evident.*"
2. Risk analysis
   "*Risk analysis may be undertaken in varying degrees of detail depending on the criticality of assets, extent of vulnerabilities known, and prior in-*

cidents involving in the organization. A risk analysis methodology may be qualitative or quantitative, or a combination of these, depending on the circumstances. In practice, qualitative analysis is often used first to obtain a general indication of the level of risk and to reveal the major risks. Later it may be necessary to undertake more specific or quantitative analysis on the major risks because it is usually less complex and less expensive to perform qualitative than quantitative analysis. The form of analysis should be consistent with the risk evaluation criteria developed as part of establishing the context. "

3. Risk evaluation
   "*The nature of the decisions pertaining to risk evaluation and risk evaluation criteria that will be used to make those decisions would have been decided when establishing the context. These decisions and the context should be revisited in more detail at this stage when more is known about the particular risks identified. To evaluate risks, organizations should compare the estimated risks with the risk evaluation criteria defined during the context establishment. Risk evaluation criteria used to make decisions should be consistent with the defined external and internal information security risk management context and take into account the objectives of the organization and stakeholder views etc. Decisions as taken in the risk evaluation activity are mainly based on the acceptable level of risk. However, consequences, likelihood, and the degree of confidence in the risk identification and analysis should be considered as well. Aggregation of multiple low or medium risks may result in much higher overall risks and need to be addressed accordingly.*"

Based on workshops, questionnaires and expert interviews several risks for an ATI could be identified in different categories. The resulting mind map is shown in figure 4.11. The different categories can be roughly classified to technological and non-technological risks. Not every risk can be countered by technological measures, i.e a cautious and thoughtful user cannot be replaced by a careless user using several technological measures. The following detailed risk categories were identified:

Process risks include everything that compromises the business value creation itself. For a factory of engine parts this means an outage stops production process or quality parameter manipulation could lead to defective parts, i.e. financial loss. For a supply infrastructure this involves all risks that could compromise stability of supply.

Network category includes all risks that arise by utilizing networking technology. Especially eavesdropping and manipulation are severe risks in this category.

Personnel can be deceived, tricked, bribed, threatened or even blackmailed. Human personnel is prone to errors, diseases and fatigue. All risks concerning human personnel are summarized in this category.

Software can contain bugs or a canny attacker can utilize its features in a not

Figure 4.11.: Risk identification mind map

intended way. The interplay of different software frameworks can also lead to unintended behavior or open new loopholes. Risks like erasure, manipulation or even replacement of the software itself arise from these possible attack vectors.

Hardware can be destroyed, stolen or manipulated. Risks in this category usually require physical access for an attacker. Other possible risks arise by technical malfunction itself, since no hardware is reliable to 100%.

Organization or the lack of it leads to risks. In this category especially missing policies or guidelines and a change of focus for the entire organization could be identified.

This listing is not complete, but other risks arise application specific. With this identified risks at hand a risk analysis can be done. Table 4.3 shows a matrix that can be used to determine a simplified risk factor. To do so, the respective risks have to be assessed by their likelihood of occurrence an impact to the organization. This process can be very complex and cumbersome, in any case experts are needed to do so and must be done application specific.

Likelihood \ Impact	Very Low	Low	Medium	High	Very High
Very High	4	5	6	7	8
High	3	4	5	6	7
Medium	2	3	4	5	6
Low	1	2	3	4	5
Very Low	0	1	2	3	4

Table 4.3.: Risk analysis matrix [ISO27005, 2011]

The last step, namely risk evaluation, can also only be done application specific, since a qualitative or quantitative evaluation criteria has to be developed. A simple example to outline this step is developed here using a quantitative evaluation criteria called weighted financial loss W_{FL}, which is the amount of financial loss F_L an organization would have, if the risk occurs, weighted by the normalized risk factor $R_f = \frac{R}{MAX(R)}$, where R is taken from Table 4.3:

$$W_{FL} = F_L \times R_f \tag{4.1}$$

An example would be a company having a financial loss of €1.000.000 if a specific risk occurs and a simplified risk factor of 6. The weighted financial loss would be

$$\begin{aligned} W_{FL} &= 1.000.000 \times \frac{6}{8} \\ &= 750.000 \end{aligned}$$

With this done for all risks, the need for action can be determined for critical assets. The company can calculate in detail where it would be necessary to invest a budget to increase security. Based on this identification of critical assets, attack scenarios can be developed.

4.3.4. Attack scenarios

In literature especially for attacks on ATIs [Klick et al., 2014] describes a specific attack scenario and [Green et al., 2017] describes several attack scenarios on ICSs and how important it is to be a skilled attacker or not. Furthermore [Amin et al., 2013a] describes stealthy deception attacks against canal systems (ref. to [Amin et al., 2013b]).

To achieve a higher degree of relevance in practice, the technological risks (ref. to section 4.3.3) can be addressed in different attack scenarios with respect to

the goal of this work (ref. to section 3.3). This includes risk categories process, network, software and hardware. Furthermore specific attack scenarios from the context of research projects STEUERUNG, pICASSO and RetroNet contribute to attack scenarios developed here. There several group discussions with different experts produces different application specific attack scenarios. Three scenarios are composed on this basis extracting the core ideas. Furthermore the attack vectors itself are irrelevant (ref. to postulate 1) and only included here for the sake of completeness. Please note that the scenarios are, despite that they are based on real companies and scenarios, completely fictional and similarities with real persons, companies, organizations or events are entirely coincidental and unintended.

Attack scenario 1: Eavesdropping/ Denial of Service

Attack vector. Attackers use a Zero-Day-Exploit on a perimeter defense firewall to gain access to a companies network. Since they control the firewall system itself they can stay unnoticed by intrusion detection systems. They use the office network located behind the firewall to get access to the production network.
Target. Network components are attacked to gain command and control of them. This includes standard network hardware (i.e. hubs, switches etc.) and IoT bridging components (i.e. connector technologies).
Goal. Attackers are able to eavesdrop any information on the network. After getting the desired information they start a DoS attack to distract from eavesdropping and possibly delete important information.
Effects. Network or process outage, financial and information loss, also possible transfer of business secrets to competitors.
Description. A huge company, global market leader for electric engines, produces all kinds of electric and electronic goods. Several factories exist in different locations which compete against each other for manufacturing of product lines. In one factory that produces electric engines, similar to application scenario B, new IoT bridging technology has been introduced on the shop floor to gain better cost efficiency. This advantage to other factories gets noticed by a competitor, which contracts a group of hackers to get the prototype firmware of this new device. Equipped with basic information of the factory landscape, the hackers follow a similar procedure as outlined in section 2.1.2. The reconnaissance includes stealthy network exploration and port scanning techniques, where a Zero-Day vulnerability in the firewall that protects the factory gets discovered. The attackers exploit the vulnerability to gain control of the firewall system. From this point on they are able to access the office network where they find several vulnerabilities in network switches. These get further used to stride forward to the shop-floor network. Due to the lack of security solutions on this layer, the firmware of the respective device gets downloaded easily and exported through a stealthy timing channel. For covering tracks a simple overwriting of all network management configuration ends up in a complete DoS for the entire factory. The recovery takes up to two weeks of 24/7 work of the factory

personnel accompanied by a huge financial loss..

Attack scenario 2: PLC Code manipulation

Attack vector. An attacker gets access (i.e. malicious insider or external intruder using disguise and social engineering) to a maintenance workstation and attacks the plants master PLC through coupling PLC (i.e. from power subsystem), which is less well protected.
Target. The binary code on the master PLC gets modified by the attackers own version, which alters the process values slowly and it is undetected by monitoring agents. [Klick et al., 2014]
Goal. If performed well (using experts knowledge) it is possible that destruction or malfunction of physical components happens over time while the modified code remains undetected. With this even the replacement of failing machinery without replacing that malicious code could lead to a unnoticed long term attack.
Effects. Process values get changed without operators realizing that. Those react according to their experience to the gradually changing conditions possibly initiating damaging effects themselves. Machinery and other physical devices get damaged or destroyed, which jeopardizes security of supply for the time until recovery. Furthermore the replacement of machinery means an unplanned huge financial investment.
Description. The initial situation takes place in a water distribution infrastructure similar to application scenario A (ref. to section 4.3.1) in summer, on a weekend, more specifically a Sunday at 10 pm. The number of operating personnel throughout the whole infrastructure is at minimum or on standby. A countdown ticks down, created by a disgruntled employee (alternatively someone externally that wants to disrupt water supply for the urban area). The attacker has already replaced the Code on the master PLC with its own version. As the countdown times out, the modified software adds 10% to the demand of supply (measured through a drop of pressure in the pipes), while removing 10% of water pump engine speed. The original values are not shown to the operators, only the modified ones. Furthermore the valves directly behind the water pumps get closed, but the value shown to the operators remains in an open state. The medium circulates within the pump housing and heats up, caused by the friction of impeller and slide bearing. Within minutes the heat reaches a critical level and plastic parts of the pump start melting, which destroys the pump[2]. If this is done in a plant which is a focal point of the infrastructure, no water will be available in the supply area until the destroyed pumps get replaced. the operators realize nothing until further notice (i.e. external phone call or other systems reacting to a cascade, too late for any case).

[2]Pump refers here to an engine that spans several meters and costs several 10K €.

Attack scenario 3: Jam/ Shield/ Modify/ Replay

Attack vector. A barely protected cellular network (i.e. Global System for Mobile communications (GSM)) used for tele-control (using IEC 60870-5-104) gets attacked to gain access to the organizations ATI network, which controls and monitors an oil pipeline.
Target. The tele-control head software gets attacked and modified. This enables the attackers to intercept and alter any command received from the tele-control master station (head) at operations control headquarters and also modify any process values sent to it.
Goal. Messages get incepted and recorded by an attacker, real values get jammed and shielded during attack while recorded messages get replayed.
Effects. Substantial amounts of oil can be stolen using a previously recorded replay attack while shielding the tele-control substation from the network. Better effects can be achieved if the tele-control master head can be attacked, so the whole pipeline can be shielded from operators, which prevents detection through other measuring points along the pipeline.
Description. An unmanned remote pumping station of an oil pipeline transmits and receives tele-control operations and values through cellular networks, since the location is poorly accessible as constrained by the terrain itself. A group of attackers try to steal a respective amount of oil for selling on the black market, which needs a specific duration of time to get it out of the pipeline. First the tele-control head of the substation is attacked via hacking the cellar network credentials. Messages get recorded for a longer timespan to enable replay. By altering the software on the tele-control head attackers are able to jam and shield the substation from the whole network. During that time the recorded messages get replayed, possibly modified to avoid detection (i.e. insert new sequence numbers). The physical pipeline gets opened up by drilling and substantial amounts of oil get drained from the pipeline.

4.3.5. Data sources

It is critical for a successful and robust detection, which and how many of the available data sources are used by the detection system. Additionally the placement of the sensors itself is an extensive topic [Schaelicke et al., 2003]. The following data sources are available and have been considered and analyzed, as shown in figure 1.8:

Network traffic consists of packets that contain a header and data section. Both parts can be analyzed to detect deviations from normal behavior. This has already been extensively done in literature, as shown in section 2. Network traffic is comparatively easy to access, since most recent network hardware offers mirroring of traffic. With respect to industrial protocols the entire communication has to be captured since most systems work incrementally. Furthermore industrial protocols are often not supported by conventional network analyzers.

Behavior means certain activity patterns that each computing entity, i.e. algorithm, service, process, task or thread follows as defined in its source code or protocol and based on the current conditions. Depending on the communication abilities of the entity itself the behavior can be captured through different features, ranging from register values of the computing nodes CPU to network activity patterns, i.e. when and how packets are exchanged with a communication partner. Another possibility is the analysis of the source itself. Either way a normal behavior of an entity can be trained to an algorithm and further be used to detect deviations.

Physical process values means measured physical values of the process to be controlled. Examples are temperature, pressure, volumetric flow rate, engine revolutions, current or voltage. Usually a sensor detects these values using a specialized method to capture a raw value. This value is then digitalized and further processed by a computing entity like a PLC. Other process values contain control values like desired engine revolutions or valve opening angles. These manipulate the physical environment directly after being transformed from a digital value to current or force. These values can be found in memory, databases and network packet data. Correlations and causal dependencies can be used to verify these values.

Soure and binary code is available at two points as data source: the source form can be obtained directly from the programmer before integration and the binary form is available on the devices at all times. Attackers usually try to modify the binary according to their needs. This possible modification has to be subject to a successful and robust detection.

Side-channel values are indirectly generated by the hardware or other surrounding systems that are necessary to make the whole system work. An example is a managed CPU fan which revolutions usually get controlled according to the heat produced by the CPU. This way the CPU-load can be guessed. Side channels such as CPU load, temperature, fan activity, memory cache activity and disk drive status can be used to create a normality pattern for a relatively constant software process. The essential foundation to use these is access through open platforms.

Log files are usually used by conventional IDS systems to monitor e.g. user activities or operating system activities. In the context of SCADA and ATI logging and alarming is performed on higher layers by operations control (as utilized by [Hadžiosmanović et al., 2012]). On the focused layers of this work, log files as data source are currently not available. Future architectures may include logging capabilities on these layers.

The data sources have to be chosen according to their availability, the possibilities of the application infrastructure, as well as the capabilities of the defending party. With respect to these limitations an application specific monitoring concept can be developed. After choosing distinct data sources and feature sets for the learning phase, normal patterns for each crucial point of the infrastructure can be trained.

Requirements \ Risks	Process	Network	Software	Hardware
Robustness	Separation/ Isolation	Out-of-band	Use well tested SW	Industry grade HW
Influence to process	Minimal interference	Minimal bandwidth	Passive detection	Avoid SPOF
Operating expenses	Minimal effort	Common interfaces	Use free software	Existing HW usage
Reconfigurability	Change adaptation	Open interfaces	Multi-Layer architecture	Simple integration
Scalability	Segmented architecture	Re-scalable architecture	Modular architecture	Relocatable services

Table 4.4.: Risks categories, major requirements and resulting development requirements

4.3.6. Data fusion

Data fusion is an important step if confronted with multiple data sources. In general three different approaches exist [Li et al., 2018]:

Data level fusion: *"it is also called low level fusion, which combines several different raw data sources to produce refined data that is expected to be more informative and synthetic."*

Feature level fusion: *"it combines many data features and is also known as intermediate level fusion. The objective of feature fusion is to extract or select a limited number of important features for subsequent data analysis through feature reduction methods, which can reduce computation and memory resources."*

Decision level fusion: *"it is also called high level fusion, which fuses decisions coming from multiple detectors. Each detector completes basic detection locally including preprocess, feature reduction, and identification to establish preliminary inferences on observed objectives. And then these inferences are fused into a comprehensive and accurate decision through the decision fusion techniques."*

Each approach defines a field of research and many algorithms can be found accordingly. Once more an application specific data fusion has to be applied.

4.4. Detection concept

A detection concept was derived from the requirements, risks and attack scenarios previously developed from real use-cases. Table 4.4 shows the five major requirements along the decisive risk categories with resulting development requirements. Some of the development requirements shown in table 4.4 conflict

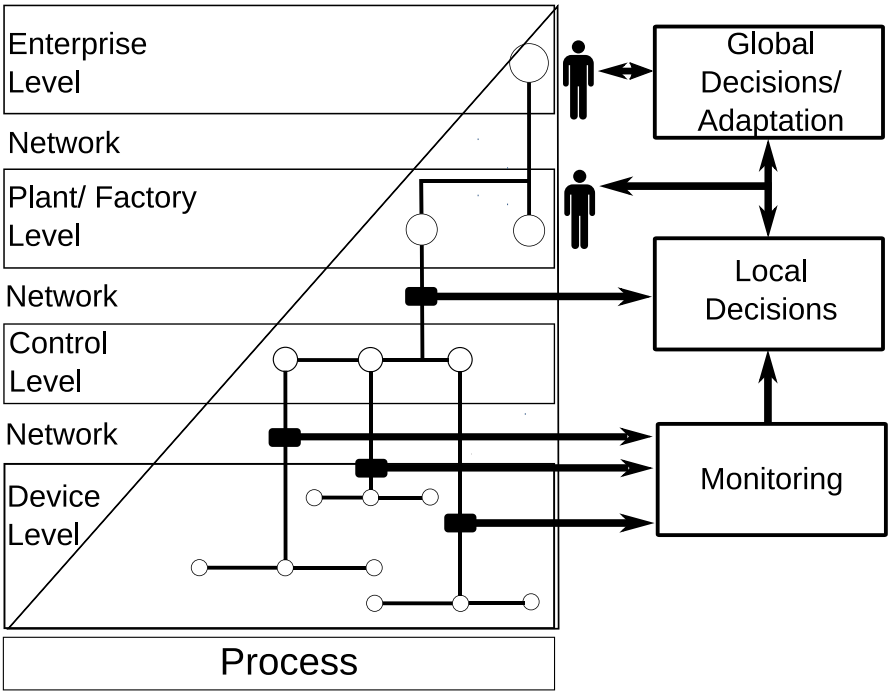

Figure 4.12.: Segmented modular multi-layer service-oriented detection architecture

with each other, for example minimal interference into the process (Influence to process/ Process) conflicts to the usage of existing hardware (Operating expenses/ Hardware). Since a minimal influence to the process was ranked higher by all experts, this requirement has more impact and wins this conflict. Other conflicts were resolved in the same manner.

4.4.1. Architecture

The resulting architecture is shown in figure 4.12. The following properties have been adopted from the development methodology (ref. to section 4.3):

Segmentation means to break the whole system (horizontally) into smaller segments to achieve a higher grade of security. This principle is known from memory or network segmentation. This is especially important for monitoring, since each network segment in the target infrastructure can be addressed using segmentation.

Modularity describes the division of functionalities into encapsulated modules using exactly and well described interfaces. [Scholz, 2005] This is usually used in software design as a basic principle. This way if one module gets compromised, other modules can possibly still perform their task.

Multiple layers are formed to logically, locally or functionally divide an architecture vertically into different elements. This principle is used in the automation pyramid (ref. to figure 1.4 and [ISO/IEC62264, 2013]) as well as within novel cloud-based production environments [Horn and Krüger, 2016c]. This way even complex structures stay manageable and understandable.

Service-orientation "*..is a way of thinking in terms of services and service-based development and the outcomes of services. A service: Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, consolidate drilling reports), Is self-contained, May be composed of other services, Is a "black box" to consumers of the service.*" [The Open Group, 2009]

Multiple data sources or even all available ones should be monitored by the detection system. Relying on only one data source like network traffic neglects the usage of all possible information. The opportunity of correlation or identification of causalities is only enabled by using different sources of information.

Separation has to be done from process and between detection services itself to avoid feedback effects. This principle is well known in many scientific domains, i.e. computer security (separation of privilege) [Saltzer and Schroeder, 1975] or control theory (separation principle - separates feedback from controller and observer) [Lunze, 2008b] and is used to increase robustness. Additionally side-effects can be avoided.

Hardware abstraction means that all functional software modules should not be bound to a specific hardware by design. This way the software stays relocatable and scalability can be achieved much easier. The relocation of a service to a more powerful hardware system during runtime is made possible too.

The resulting architecture was mirrored along the Automation Pyramid (ref. to figure 1.4 to allow and easy integration. The desired system runs parallel to the process which should be monitored and contains the layers monitoring, local decisions and global decisions. In terms of data fusion the requirements segmentation, modularity and multiple layers point directly to a decision fusion, which can be done locally as well as globally.

Monitoring

On this layer different data sources in individual segments can be monitored. The collected data can be preprocessed prior to transmission or fast safe automatic decisions can be made. A service running on this layer should have

minimal hardware requirements being able to run on micro-controllers or single board machines.

Local decisions

The services running on this layer aggregate the collected data and process it into information. Different segments intersect locally and cross-validation is possible to superiorly classify the collected data. Local decisions can be made in a timely manner based on binary classifiers and pre-compiled executables. Furthermore this reduces the amount of data transferred between local and global layer significantly, which increases scalability [Thomopoulos et al., 1987]. For possibly unknown states the local information gets transmitted to the global layer for further processing. Generally, the decisions get transmitted to the global layer. A minimal interface exists to communicate with factory personnel (i.e. a traffic signal on a industry panel PC) for fast human reactions in case of emergency. The hardware available on this layer ranges from industrial panel computers to local factory servers, which enables the services to utilize moderate computing power.

Global decisions and adaptation

On this layer the gathered local information can be processed in two ways: first, a landscape of the local decisions can be generated to cross validate local segments along physical conditions. For example if one segment on a pipeline reports an abnormal condition, other segments should follow in a timely fashion along the pipeline. This way it would be possible to differentiate technical malfunction from manipulation. Second, an adaptation of the services algorithms can be done with strong involvement of operators for safeguarding purposes. This way automatic error generation and propagation can be avoided. The services on this layer usually need stronger computing power like powerful server clusters, e.g. for training classifiers with new sets of data to adapt an algorithm of a service. Also timely responding here means non-real-time behavior.

4.4.2. Detection services

Suitable countermeasures for the previously identified technological categories (ref. to section 4.3.3) process, network, software and hardware, based on Table 2.1, include monitoring and analysis. The following questions arise for each application infrastructure that the previously presented methodology gets applied to:

- What should be monitored?
- How should this data be analyzed?
- What causal dependencies exist?
- Which critical points exist in the infrastructure?

Attack phase	Target information	Countermeasure
Target identification	Public available information	Acting fair/ careful
Reconnaissance	Organizational/ social data	User training/ regulations
Exploitation	Network traffic, host or service	Network traffic/ service validation
Back-Door installation	Host or service	Service validation
Objective pursuit	Source/ binary code	Code validation
Accomplishment	Network traffic/ covert channel	Network traffic validation
Cover-up	File manipulation/ erasure	Host file system validation
Aftermath	Process values	Process value/ behavior validation

Table 4.5.: Attack phases and defense possibilities for an APT

The TechMap of the infrastructure can help to answer these questions. Furthermore the sequence of actions an attack consists of as described in section 2.1.2 can be used to define general basic services here. Further individual services can arise application specific. Table 4.5 shows the required information and possible detection approach for general basic services in an ATI. Furthermore the attack scenarios in section 4.3.4 contribute to the development of these basic general services.

The following respective general basic services were developed: Network Traffic Validation Service, Code Validation Service, Service Validation Service, Process Value Validation Service and Behavior Validation Service. These reflect directly the generally available data sources as described in section 4.3.5 with exception of Side Channel Values. This is substantiated due to the fact that access to these values is usually not available. Closed platforms like commercially available industrial PLCs do not offer access to internal variables and algorithms. Furthermore other side channel like sound recording of a fan's noise requires additional hardware, which contradicts to the requirement minimal interference into the process. Approaches like [Stone et al., 2015] or [Dunlap et al., 2016] are not feasible. Nevertheless due to the modular architecture, not included data sources can be utilized later by implementing a respective service.

Network Traffic Validation

The goal of this service is to monitor and validate network traffic for abnormal behavior. Common NIDSs [Mukherjee et al., 1994] typically contain the ability to detect deviations in communication patterns and network traffic protocols.

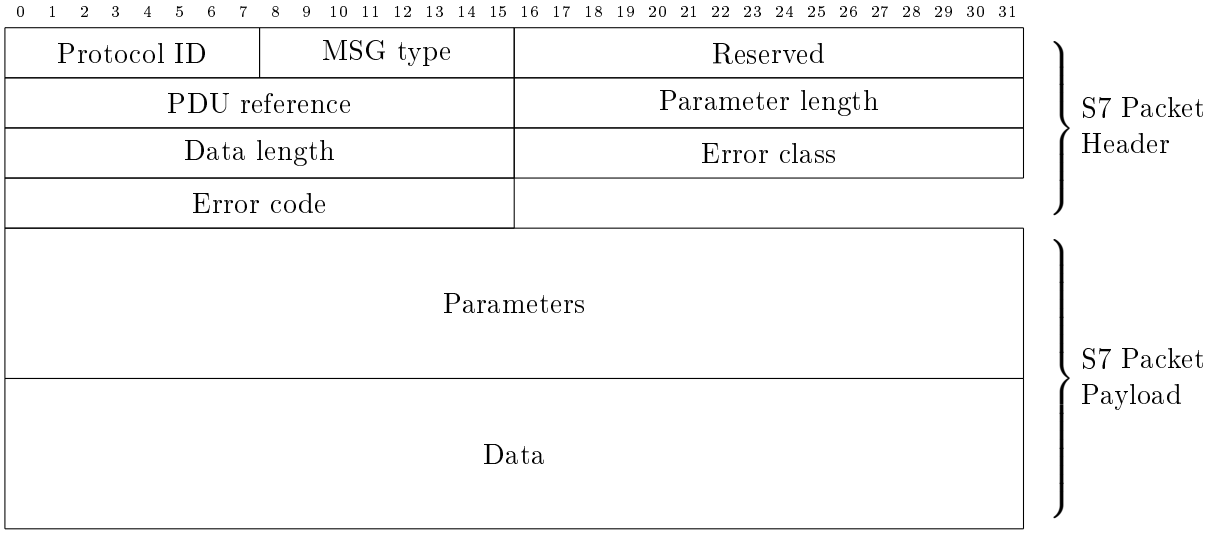

Figure 4.13.: SIEMENS S7 protocol packet frame [Nardella, 2015]

Most available systems focus on Ethernet traffic but industrial protocols can also be monitored (ref. to section 2.2.2).

Network traffic consists of packages that contain a header and payload section. The former usually contains meta-information about the packet according to the network protocol used, like source and destination address. Figure 4.13 shows an example for a SIEMENS S7 packet (ref. to [Nardella, 2015]). The latter contains application specific data or parameters, for PLCs usually the desired physical process values of the ATI. Using the meta-data of the header section for traffic verification requires either a model of characteristic communication patterns or the protocol itself, for ATIs especially a model of industrial field-bus protocols like SIEMENS S7 communication protocol. The model itself can be built using different approaches, examples are presented for Modbus/TCP in [Cheung et al., 2007], which utilize Markov chains and for SIEMENS S7 traffic in [Kleinmann and Wool, 2014] which utilize DFA.

The information of the header section of a network packet is always accessible, since network infrastructure device nodes also process this information to forward the packet to the right destination, or in case of an application specific protocol to the right application. Utilizing the payload of the packet is called *deep packet inspection*. This information is not always easily accessible, since it can be encrypted. Industrial network traffic is usually not encrypted [Fauri et al., 2017], but recent developments especially in the context of connecting SCADA networks to WANs like the Internet point into this direction [Alves et al., 2017]. Therefore encryption has to be kept in mind here. Furthermore the processing of the payload section is very challenging, e.g. packages using the SIEMENS S7 network protocol can be incremental to already sent packages which requires complete capture of all packages. At the same time the model necessary to verify the process values does not reflect network traffic patterns, but the underlying process itself. Therefore packet payload can also be used for

the *Process Value Validation Service*.

Code Validation

The goal of this service is to validate the code that will be executed on automation nodes itself. To do so the source or binary form can be examined. The source form can be obtained directly from the programmer and deviations can be found using simple diff-tools [MacKenzie et al., 1997] regarding text-based sources. The binary form is available on the device itself during runtime and therefore it is more likely prone to manipulation.

A HIDS usually contains a database of static properties like checksums of binaries and dynamic properties like resource allocation patterns to check the integrity of entities against unauthorized modification. For binary files this is usually done using a hash function [Carlson and Scharlott, 2006].

The hash function $h(x)$ and its resulting checksum y can represent the unmodified binary object. Any deviation results in a changed y' which can be detected easily using formula 3.3 leaving δ unequal to zero.

Service Validation

As HIDS-like complement to the Code Validation, which monitors static properties, the goal of this *Service Validation* is to monitor dynamic changes in resource allocation to verify computing processes or services of automation nodes during runtime. Services are programs running on a process node itself like the interface services on a PLC using ISO on TCP at Port 102 [Rose and Cass, 1987]. Several tools exist to actively (nmap, [Lyon, 2009]) or passively (p0f, [Zalewski, 2016]) fingerprint a host concerning its services from a distant location. This is also referred to as OS fingerprinting.

A Service Validation Service can utilize these technologies. Depending on the possibilities of devices within the ATI, software agents can monitor features like processor load or RAM consumption and report these to the service to enhance the detection (ref. to infrastructure monitoring tools like Nagios [Galstad, 2017]). For open platforms like Linux-based devices a HIDS like Open Source Tripwire [Tripwire-Community, 2017] can additionally be used.

Especially for SIEMENS PLC devices features like CPU-load or memory consumption cannot be accessed by third-party security tools. A solution to access these features has to be laboriously developed and implemented manually. The necessary information can be found in [Siemens AG, 2010] and [Nardella, 2015].

Process Value Validation

Process values are the most crucial aspect in an ATI. Their correctness is essential for functionality and safety reasons, as several expert interviews indicated. Furthermore no reliable method is currently available for operators to validate

Requirements	Approach: Meta-Data	Simulation	Redundancy
Robustness	✓	✓	(✓)
Influence to process	×	✓	×
Operating expenses	✓	×	×
Reconfigura-bility	(✓)	✓	×
Scalability	✓	✓	(✓)

✓ available (✓) partly available x not available

Table 4.6.: Requirements compliance of approaches for process value validation

process values. They simply have to trust that everything is correct. A service that provides assistance in validating these values is strongly requested (ref. to appendix A.1).

Apart from simply trusting the correctness of process values their validation can be performed theoretically using three different approaches:

Meta-data generation refers to create additional information that describes the object itself. One example is the creation of a signature of each value for subsequent matching to a canonical signature for similar values. Alongside with strong encryption this approach can perform well, but requires significant changes to current ATIs.

Parallel simulation can reach from applying mathematical models for estimating values to creation of a complex testbed including Hardware-in-the-Loop (HIL) simulation and full virtualization of the process itself. This way the estimation can be compared to the current process value for validation (ref. to formula 3.3). The effort to create this solution is quite high, but no interference in the running process is necessary.

Redundant hardware layout means to duplicate each automation node in the ATI, i.e. use two sensors for measuring one value and use two PLCs running the same program. This approach is usually used for fail-safe operation. The impact to increase the security level seems to be quite low, since the attackers can simply attack both nodes.

The mere physical protection of systems using i.e. security personnel, burglar alarm, fences with barbed wire, surveillance cameras as well as isolation of systems from networks is not taken into consideration due to the focus of this work (ref. to section 3.3).

Table 4.6 shows non evaluated hypothetic compliance to the requirements from development methodology. As shown in figure 3.3 the simulation was chosen for

the following important reasons:

- It can be built and run completely without touching the ATI, all other approaches require significant changes to the process infrastructure itself. The operating companies prohibit that strictly, therefore it is considered unfeasible.
- A huge amount of algorithms and approaches exist in simulation, ranging from simple statistical models to complex virtualized environments. New and promising approaches like process causality (ref. to section 3.4.2) can be evaluated here too. The convenient ones can be used in ensemble and prospective superior ones can be integrated easily.

Within the area of simulation the choice of model itself is essential for developing a solution to protect the process values as most critical assets of an ATI. Especially the modeling of physical processes can be a challenging task. Especially for nonlinear systems the dimensionality of the model itself can lead to unfeasible simulations. "*When, however, the dimension is "large"(..), or infinite, the "curse of dimensionality" prevents the direct use of general methods.*" [Bellman, 1969].

Since the model itself is highly dependent on the specific application context, it has to be decoupled from the service. General solutions from state of science and technology like [Hadžiosmanović et al., 2014] or [Horn and Krüger, 2014] have to be evaluated for a reference implementation alongside a new approach, based on Process Causality (ref. to section 3.4.2) [Horn and Klein, 2017].

Behavior Validation

The goal of this service is to validate the behavior of individual PLC-nodes with closed internal platforms. The behavior is usually determined by the application program, which is available in IEC 61131-3 conform programming languages. Several different technologies are available for a Behavior Validation Service, that can process complex PLC Code (ref. to EN 61131-3:2014). For example the nodes can be monitored using a concurrent simulation of the application software in a virtual environment (ref. to [Horn and Krüger, 2016a]) or using a state space model of the code (ref. to appendix A.3 and [Horn and Krüger, 2014]). Furthermore a decoupling of the security service from the monitored system can be achieved easily by running the simulation on a different system.

Honeypot systems are valuable data sources if they can attract attackers and capture their actions. For ICS simple systems [MushMush-Foundation, 2015] are available to mimic industrial communication protocols. More advanced attackers may not be fooled by these systems. The developments for the Behavior Validation Service can also be used standalone as Honeypot. Using this approach not only the communication protocols, but also a user scenario written in PLC Code can be used to attract advanced attackers and capture their actions. The attack can be analyzed and delayed.

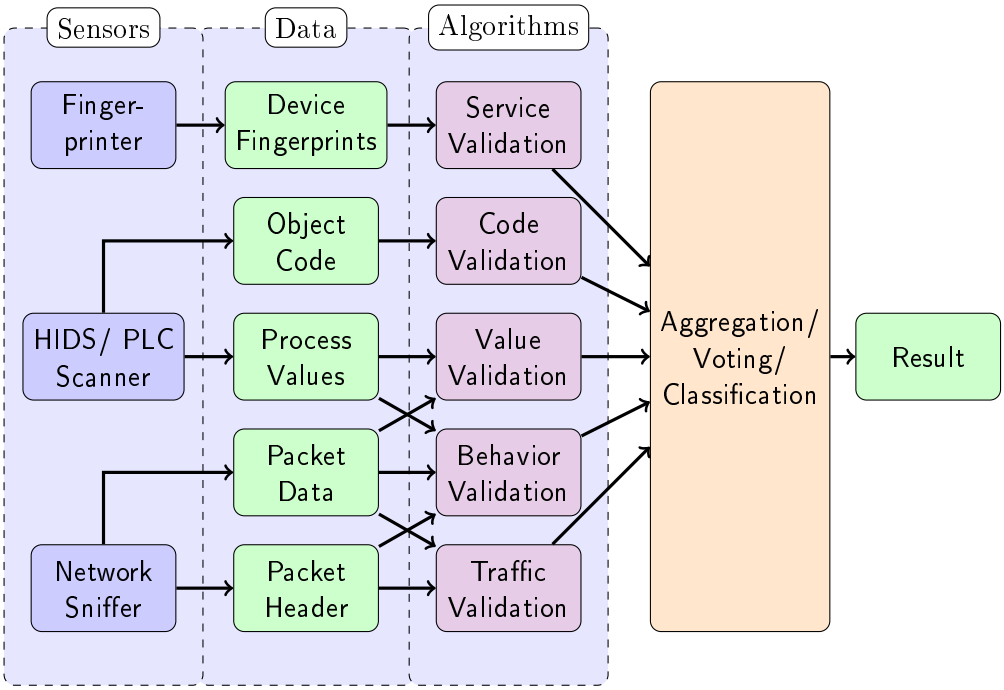

Figure 4.14.: Overview of detection concept

4.5. Concept synopsis

Figure 4.14 shows the elements of the detection concept in detail. Similar to an Intrusion Detection System it is divided into sensors and the respective analysis back-end (ref. to figure 2.3). The difference is that here multiple different algorithms can be used for analysis, the detection services. This modular concept ensures future-proof applicability due to high reconfigurability for the application context, since any new data source, as well as analysis algorithm can be added or obsolete ones can be omitted. A subsequent aggregation, voting or classification algorithm ensures hereby the fusion of different algorithmic results. The resulting decision can be used for alarming operators and long term storage for archive and adaption purposes.

This concept enables furthermore interaction with the user at different points. An example is the direct alarming of one very accurate and fast computing algorithm close to a machine. Here this specific algorithm could not only transfer its result to the fusion, but also directly alarm a machine-operator. Furthermore the data input could also be enhanced by user interaction, defining it as additional data source.

5. Implementation and evaluation

"There is no substitute for hard work." – Thomas A. Edison, *The New Yorker Magazine, Volume 41, Part 2* (1925)

In this chapter the previously developed concept to secure Automation Technology Infrastructures is prototypically implemented and then evaluated along different testing scenarios. These are derived from attack scenarios previously mentioned in section 4.3.4.

5.1. Implementation concept

Implementation follows the concept to detect anomalous values within ATI as developed in section 4.3. As previously mentioned the architecture is designed segmented, modular, multi-layer, service-oriented, hardware abstracted and separated from the automation process itself, as shown in figure 4.12. The three different layers of the architecture, namely monitoring, local decisions and global decisions/ adaptation are designed to run independently from hardware, but as previously described their requirements for resource allocation (CPU-load, memory consumption or the like) differ. For that reason the layers are deployed to three different systems, as shown in figure 5.1:

Embedded node refers to a low computing power entity like a micro-controllers or single board machine.

Local computing node refers to a moderate computing power entity like industrial rack mounted panel computers or small local factory servers.

Global computing node refers to a high power computing entity like private, community or public cloud environments.

The communication between these nodes takes place in an encrypted manner, utilizing the Secure Shell (SSH) [Campbell et al., 2018] cryptographic protocol. It offers state-of-the-art communication security that is transparent towards applications and services. Furthermore orchestration suites like Ansible [DeHaan, 2018], that need "*no agents to install on remote systems*" can be used over these connections to provide command and control, as well as updates and bug fixes. The nodes themselves should be well secured using state-of-the-art countermeasures, as pointed out in section 2.2. Especially hardened systems [Granberg et al., 2016] [The Debian Project, 2018] should be used to run the proposed detection services and components.

Furthermore each node contains besides services and applications additionally

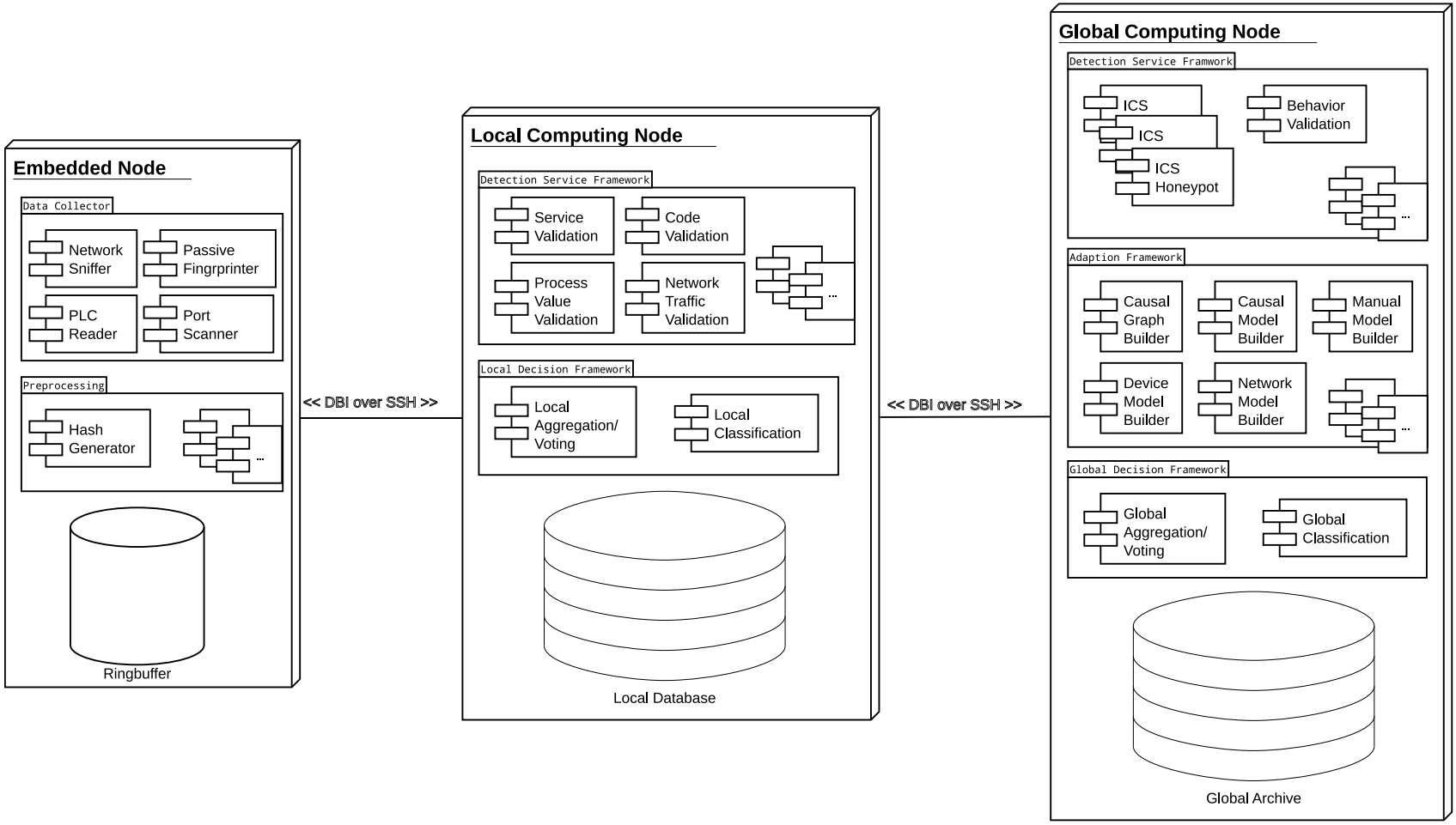

Figure 5.1.: Deployment diagram for implementation concept

individual databases. Each storage reflects directly the requirements of the layer and is fitted to the nodes specifications as follows:

Ring buffer is a temporary database residing in-memory on the embedded node (e.g. [SQLite Consortium, 2018]). It acts as cyclic buffer to compensate asynchronous cycle times between data acquisition and processing. The timing-window stored contains a multiple of process cycle times for each value to monitor, depending on the memory capabilities.

Local database is a storage residing on the local computing node that contains aggregated data blocks from the embedded node(s), return values of detection services and snapshots of the ring buffer(s) for adaptation purposes.

Global archive contains snapshots and aggregations of the local database(s) for long term storage.

Detection patterns should not be held in the databases to avoid easy tampering of the detection system. Instead every service should maintain its models and patterns for detection within its isolated runtime environment, container or virtual machine.

The data flow for the entire concept is shown in figure 5.2 per detection service and data source. First raw data values get received by the monitoring application from the data source (see section 4.3.5 for details). Depending on it tools like network sniffers [Combs, 2018b], passive traffic fingerprinting mechanisms [Zalewski, 2016], port scanner [Lyon, 2009], Host based Intrusion Detection System (if available) or new implemented tools based on available libraries like Beckhoff ADS [Beckhoff, 2017] or Snap 7 [Nardella, 2015] can deliver the necessary data values. The monitor adds a time-stamp and puts these timed data values into the ring buffer, where the associated detection service fetches a whole frame containing several consecutive data values including their timestamps. This asynchronous communication is done to reduce load on the network, being able to scale the concept to large scale architectures. The detection service uses the data frame to find anomalies and submits its result to the local decision service. As shown in figure 4.14 this includes aggregation and voting algorithms to handle results of multiple detection services as well as a classification of the scoring result. Data frame and scoring result get saved to the local database of the device and an alarm can be triggered. A global decision service, which handles several segments of the whole infrastructure, gets the event data frame from the local database, which contains scoring result and corresponding data frames from multiple detection services. In case of inconsistent scoring or new authenticated data (like new PLC Source Code uploaded by an authorized programmer) the adaption services get triggered, which modify the basic detection models. The global event frames, which contain all event data and decisions, get saved to a global archive to enable possible error analysis. An interface to other infrastructure monitoring applications (e.g. Nagios [Galstad, 2017]) is also part of the concept to enable easy integration into existing infrastructures.

Figure 5.3 shows the mock-up for the system containing an embedded and local

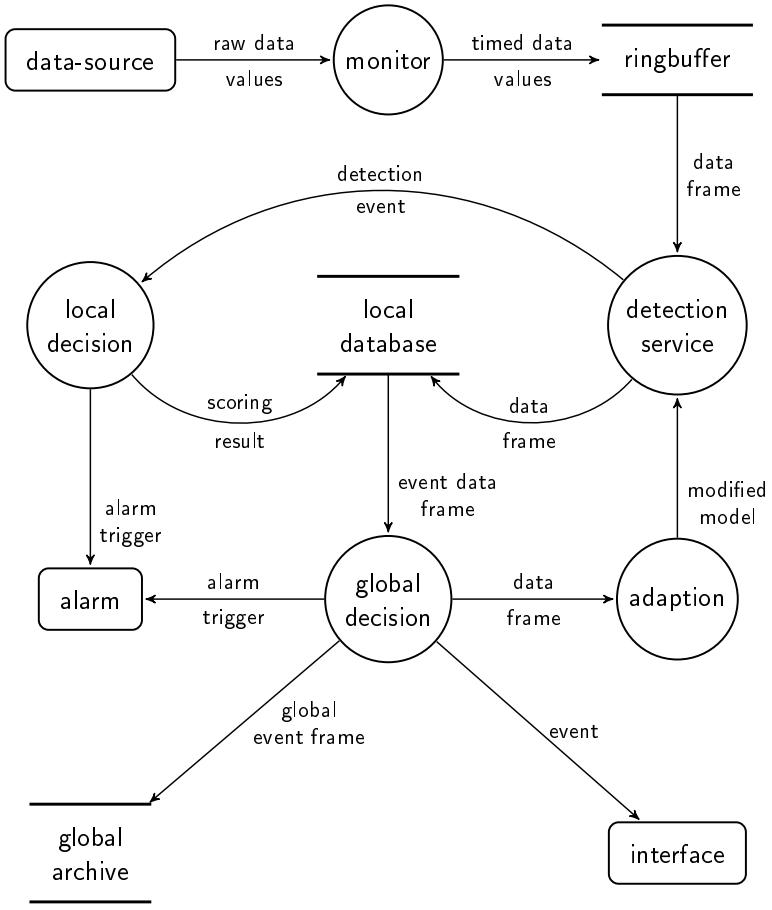

Figure 5.2.: Data flow diagram [Rumbaugh et al., 2004] of implementation concept

computing node. Both have HMI interfaces in terms of touch screens. The monitoring on the embedded node shows hereby the live captured data and the local computing node has an interface containing live information from the detection services on the shop floor. Configuration options can be included on this level too. The global computing node is connected to the mock-up through Ethernet and consists of a server running Debian Linux [The Debian Project, 2018] and KVM/qemu [Bellard, 2018]. The application software can therefore be implemented in a virtual machine running any operation system.

5.1.1. Detection services

The detection services are based on different algorithms and methods to detect anomalies. This is necessary since the data sources for each detection service are quite heterogeneous.

Figure 5.3.: Mock-up platform for reference implementation

Network Traffic Validation

As stated in section 4.3.5 the network traffic can be analyzed for protocol or data anomalies. Since the detection of data anomalies is the heart of the *Process Value Validation Service* a complement is the monitoring of network protocol patterns. Especially for industrial network traffic, some approaches are available (ref. to section 3.2), like [Cheung et al., 2007], [Linda et al., 2009], [Hahn and Govindarasu, 2013], [Goldenberg and Wool, 2013], [Genge et al., 2014] or [Cruz et al., 2016].

For a reference implementation of the service using the S7 protocol only the approach presented in [Goldenberg and Wool, 2013] with the extension to S7 in [Kleinmann and Wool, 2014] is available. It uses modified Deterministic Finite Automata (DFA) models. "*Because SCADA systems have clear communication patterns, each HMI-PLC channel can be modeled as a deterministic finite automaton (DFA). A classical DFA is a five-tuple* $(Q, \Sigma, \delta, q_0, F)$ *comprising a finite set of states* Q*, a finite set of input symbols called the alphabet* Σ*, a transition function* $\delta : Q \times \Sigma \rightarrow Q$*, a start state* $q_0 \in Q$ *and a set of accept states* $F \subseteq Q$*. Two adjustments are made in order to use a DFA to model Modbus data: (1) No accept states are required because the intrusion detection system continuously monitors an endless repetitive stream. Instead, a Moore DFA, which associates an action with every state transition in* δ*, is employed. Any deviation from the predicted pattern [..] potentially raises an intrusion detection system alert depending on the severity of the deviation. (2) The Modbus features that identify a symbol in the alphabet* Σ *must be selected.*" [Goldenberg

and Wool, 2013].

Algorithm 5.1 Pattern modeling algorithm [Goldenberg and Wool, 2013]

```
Pattern_Length ← 2
DFA ← DataLearning(Pattern_Length)
performance_value ← ModelValidation(DFA)
while performance_value > Threshold and Pattern_Length <
Learning_Window_Size do
  Pattern_Length ← Pattern_Length + 2
  DFA ← DataLearning(Pattern_Length)
  performance_value ← ModelValidation(DFA)
  if Pattern_Length > Learning_Window_Size then
    "FAILED"
  else
    return DFA
  end if
end while
```

First the approach uses a learning phase to generate a model of the traffic per channel. The respective algorithm can be found in algorithm 5.1, further details in [Goldenberg and Wool, 2013]. Initially made for Modbus/TCP traffic, an extension for the S7 protocol can be found in [Kleinmann and Wool, 2014]. The on-line '*enforcement*" phase [Garcia et al., 2016] checks the incoming network packets against the learned patterns. The associated monitoring can obtain packets from the network using *tshark* [Combs, 2018a].

Service Validation

The validation of dynamic resources on an target machine or host can be achieved using a HIDS, if the platform allows access. Otherwise the host has to be fingerprinted according to the visible services from distant locations with tools like p0f [Zalewski, 2016] or nmap [Lyon, 2009]. For commercially available industrial PLCs only the second approach seems feasible, since access to the platform itself is mostly not permitted by market leading vendors like SIEMENS [SIEMENS, 2016]. Furthermore the second top ranked requirement (ref. to figure 4.10), namely a low *influence to process* has to be kept in mind. A careless implementation of a HIDS on the control device itself without the support of a vendor could lead to unforeseen side effects, since the PLC needs to compute in a timely and predictable fashion.

The aforementioned tools are "*(passive) operating system (OS) fingerprinting tools that attempts to determine the OS of a system based on the [..] traffic it generates. [..] The efficacy [..] is dependent on an up-to-date signatures set.* " [Zalewski, 2016]. Usually fingerprints for industrial grade PLCs are not available, as shown in listings 2 for p0f and listing 3 for nmap. Furthermore [Caselli et al., 2013] hypothesized that these tools would not work at all. Instead tools like PLCscan [Efanov, 2012] are mentioned to work in ICS environments. The

```
<Fri Apr  6 14:33:31 2018> 10.0.2.20:33242 - UNKNOWN [S20:64:1:60:M1460,S,T,N,W7:.:?:?] (up: 1778
     hrs)
-> 10.0.2.20:80 (link: ethernet/modem)
<Fri Apr  6 14:33:31 2018> 10.0.2.20:44336 - UNKNOWN [S20:64:1:60:M1460,S,T,N,W7:.:?:?] (up: 1778
     hrs)
-> 10.0.2.20:443 (link: ethernet/modem)
<Fri Apr  6 14:33:32 2018> 10.0.2.70:60416 - UNKNOWN [S20:64:1:60:M1460,S,T,N,W7:.:?:?] (up: 1778
     hrs)
-> 10.0.2.20:135 (link: ethernet/modem)
<Fri Apr  6 14:33:32 2018> 10.0.2.70:53852 - UNKNOWN [S20:64:1:60:M1460,S,T,N,W7:.:?:?] (up: 1778
     hrs)
```

Listing 2: Output of p0f while scanning traffic from/to a SIEMENS S7-416 PLC

reference implementation of this service utilizes different tools for evaluation.

```
Starting Nmap 7.01 ( https://nmap.org ) at 2018-04-06 15:45 CEST
Initiating ARP Ping Scan at 15:45
Scanning 10.0.2.20 [1 port]
[...]
Nmap scan report for 10.0.2.20
Host is up (0.00069s latency).
Not shown: 65533 closed ports
PORT    STATE SERVICE
80/tcp  open  http
102/tcp open  iso-tsap
MAC Address: 00:1B:1B:A0:F0:BB (Siemens AG,)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ )
     .
TCP/IP fingerprint:
OS:SCAN(V=7.01%E=4%D=4/6%OT=80%CT=1%CU=40223%PV=Y%DS=1%DC=D%G=Y%M=001B1B%TM
OS:=5AC7942F%P=x86_64-pc-linux-gnu)SEQ(SP=105%GCD=2%ISR=107%TI=RD%CI=RI%TS=
OS:U)OPS(O1=M5B4%O2=M5B4%O3=M5B4%O4=M5B4%O5=M5B4%O6=M5B4)WIN(W1=B68%W2=AF0%
OS:W3=A00%W4=800%W5=860%W6=848)ECN(R=Y%DF=Y%T=3C%W=B68%O=M5B4%CC=N%Q=)T1(R=
OS:Y%DF=Y%T=3C%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=Y%DF=Y%T=3C%W=848%S=O%A=S+
OS:%F=AS%O=M5B4%RD=0%Q=)T4(R=Y%DF=Y%T=3C%W=800%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=
OS:Y%DF=Y%T=3C%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=3C%W=0%S=A%A=Z%F=
OS:R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=3C%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=
OS:3C%IPL=38%UN=0%RIPL=134%RID=G%RIPCK=I%RUCK=G%RUD=G)IE(R=N)

Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=247 (Good luck!)
IP ID Sequence Generation: Randomized

Nmap done: 1 IP address (1 host up) scanned in 6701.49 seconds
Raw packets sent: 85603 (3.778MB) | Rcvd: 83721 (3.352MB)
```

Listing 3: Output of nmap while trying to fingerprint SIEMENS S7-416 PLC

As shown in listings 2 and 3 the tools try to guess the underlying platform or operating system based on their fingerprints database. Here the protocol-stack specific implementations per visible service port get analyzed and compared to the generic entries in the database. To validate industrial devices in an ATI an own database of devices within the infrastructure has to be build up (learning phase). Using the fingerprint of a specific device to recognize the same device removes the guessing part of the aforementioned tools. As shown in listing 3, a device-specific fingerprint is already delivered by some tools themselves, other fingerprints should reflect specific characteristics as outlined in [Caselli et al., 2013]. As stated earlier, passive technologies should be preferred.

The procedure to build up the database is shown in algorithm 5.2. After building up the database containing device-specific fingerprints of the ATI, these tools can be used for the validation service.

Algorithm 5.2 Device fingerprinting algorithm

```
database ← empty
deviceList ← getDevices(maxNum)
while deviceList > 0 do
  deviceFingerprint ← parseOutput(toolOutput)
  database ← addFingerprint(deviceFingerprint)
  deviceList ← deviceList − 1
end while
```

Code Validation

To validate the code running on control devices during runtime the binary form of the code can be monitored. This is usually done by a HIDS on the device itself, but for closed platforms like commercially sold industrial PLCs this has to be done in conjunction with available interfaces.

Programming tools of the vendors communicate with these closed platforms and their interfaces provide also access to binaries available on the devices themselves. Examples are Snap7 [Nardella, 2015] for SIEMENS PLCs or ADS [Beckhoff, 2017] libraries. The necessary applications using these have to be implemented accordingly. Listing 4 shows the output from a SIEMENS PLC device scan of such an application. Here, the index shows all binary blocks for code (OB, FB, FC, SFB, SFC) and data (DB, SDB) on the device as well as block numbers in each category. The binary data can be accessed too.

The reference implementation in C++ is able to get this information from SIEMENS PLCs. Indexes and binary blocks can be obtained and used to fingerprint the device. A new code block, as implemented in the attack scenario of [Klick et al., 2014] can be seen in the respective index, as well as the modification of blocks in their binary form. Algorithm 5.3 shows the learning phase similar to the *Service Validation* approach.

Process Value Validation

As discussed in section 4.4.2 a simulation based approach fulfills the necessary requirements, where the choice of model is essential and application specific. This means that for each application the model has to be chosen and trained specifically. To develop a solution for a wider application context, a decoupling of the model from the detection service is necessary.

For a reference implementation more general solutions like linear autoregressive models [Hadžiosmanović et al., 2014], discrete event models [Horn and Krüger, 2014] or new approaches like Process Causality based models [Horn and Klein, 2017] can be evaluated (ref. to appendix A.4).

As shown in figure 5.4 and algorithm 5.4 the service simulates a trained model and its output is matched to the real process variable. A simple binary classifier then determines deviations of δ (ref. to postulate 4).

```
+-------------------------------------------------------
| UNIT Connection
+-------------------------------------------------------
Connected to   : 10.0.2.20 (Rack=0, Slot=2)
PDU Requested  : 480 bytes
PDU Negotiated : 480 bytes
+-------------------------------------------------------
| List all Blocks in AG
+-------------------------------------------------------
OBCount  : 13
FBCount  : 36
FCCount  : 57
SFBCount : 29
SFCCount : 86
DBCount  : 148
SDBCount : 20
+-------------------------------------------------------
| OB Block List in AG
+-------------------------------------------------------
BLock  : 1
BLock  : 35
BLock  : 80
BLock  : 82
BLock  : 83
BLock  : 84
BLock  : 85
BLock  : 86
BLock  : 87
BLock  : 100
BLock  : 102
BLock  : 121
BLock  : 122
+-------------------------------------------------------
| List OB 1 Block in AG
+-------------------------------------------------------
SubBlkType: 0x8
BlkNumber : 1
BlkLang   : 3
BlkFlags  : 1
MC7Size   : 1050
LoadSize  : 1186
LocalData : 26
SBBLength : 28
CheckSum  : 48096
Version   : 1
+-------------------------------------------------------
| Block Upload
+-------------------------------------------------------
Dump of Block Type OB No.1 (1050 bytes) :
0x0000: 10 01 41 60 00 14 3d 03 70 0b 00 02 10 02 10 01  ..A`..=.p.......
0x0010: 41 60 00 14 3d 01 70 0b 00 02 10 02 10 03 41 60  A`..=.p.......A`
0x0020: 00 18 fb 7c fb 79 00 1c fe 6f 00 14 85 04 41 50  ..?|?y..?o....AP
0x0030: 00 00 fe 0b 84 00 00 00 75 1c fe 6b 00 14 fb 7c  ..?.....u.?k..?|
[...]
0x0410: 70 0b 00 02 10 02 00 00 65 00                    p.......e.
```

Listing 4: Sample output of Snap7 application for SIEMENS S7-416 PLC

Algorithm 5.3 Code Block hashing algorithm using Snap7

$database \leftarrow empty$
$blockList \leftarrow getAllBlocks(maxNum)$
while $blockList > 0$ **do**
  $blockHash \leftarrow parseOutput(snap7app)$
  $database \leftarrow addBlockHash(blockHash)$
  $blockList \leftarrow blockList - 1$
end while

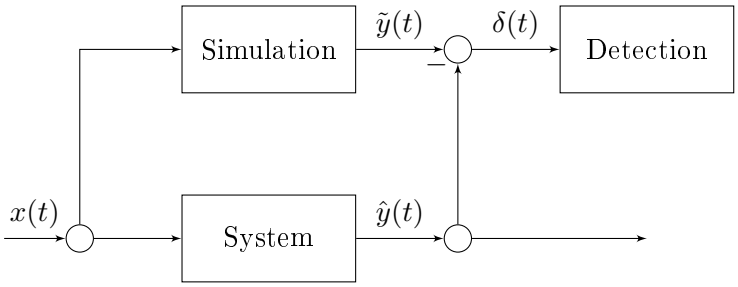

Figure 5.4.: Simulation based attack detection to validate process values

Algorithm 5.4 Process value validation for generic simulation models

Require: $windowSize, x(t), \hat{y}$
Ensure: $detectionEvent$
$X \leftarrow getDataFrame(x(t), windowSize)$
$\tilde{y} \leftarrow simulateModel(X)$
$\delta \leftarrow |\tilde{y} - \hat{y}|$
$detectionEvent \leftarrow classify(\delta)$

Behavior Validation

The behavior is defined by the application program, for typical PLC devices in an IEC61131 conform language. Therefore to simulate the behavior of a PLC this application program can be used directly or to derive models from the source, since it usually contains all states of the device. For a reference implementation this can be achieved using two different approaches:

Virtualization which means decoupling of the software from hardware. For PLC devices this can be achieved by running the application program directly in a virtual Software Programmable Logic Controller (vSoftPLC) as utilized in [Horn and Krüger, 2016a]. A variety of Software PLCs were modified according to purpose and evaluated in a minimal hardware setup (ref. to appendix A.2). Following that they were ported to virtual environments. Based on that an interface to communicate with other PLCs in the field was developed and implemented [Kittmann, 2017]. The resulting setup can mimic the behavior of the original PLC in detail, since it runs the exact program.

Modeling means here to transfer the logic of the program into a model using states that change based on discrete events. A variety of theories and methods can be found to do so, but especially for PLC-devices, this was done by [Hanisch et al., 1997] and [Heiner and Menzel, 1998], which are the foundation of the work done in [Horn and Krüger, 2014]. Here the IEC61131-3 conform source code in Instruction List (IL) form is utilized to automatically generate a discrete event dynamic system model in the colored Petri-Net form (ref. to appendix A.3).

Both approaches are subject to evaluation. The validation of correct behavior

is shown in algorithm 5.5, which is similar to 5.4, but uses multiple trajectories of input variables to generate output from the simulated PLC. Predicted trajectories are then compared to the real output values to detect deviance.

Algorithm 5.5 Behavior validation for generic PLC models

Require: $windowSize, X_i(t), \hat{Y}_o$
Ensure: $detectionEvent$
$X \leftarrow getDataFrame(X_i(t), windowSize)$
$\tilde{Y}_o \leftarrow simulatePLC(X)$
$\Delta \leftarrow |\tilde{Y}_o - \hat{Y}_o|$
$detectionEvent \leftarrow classify(\Delta)$

Decision fusion

The aforementioned reference implementation of detection services submit each a binary output signal. To get an aggregated result for all possible detection services, a fusion scheme has to be applied where a metric has to be chosen for weighting the classification results.

[Powers, 2007] summarizes the most typical metrics to compare pattern recognition algorithms. These are based on true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) and defined as shown in definition 2. Other metrics like Receiver Operating Characteristic (ROC)-curve analysis or the F-measure exist and "*debiased*" versions are also available, where it is necessary to have knowledge of the bias [Powers, 2007]. Especially for evaluation of the presented detection services "*there is no globally acceptable standard/ metric for evaluating an intrusion detection system. Although the Receiver Operating Characteristic (ROC) curve has been widely used to evaluate the accuracy of intrusion detection systems and analyze the trade-off between the false positives rate and the detection rate, evaluations based on the ROC curve are often misleading and/or incomplete*" [Patcha and Park, 2007].

This diversity of approaches for evaluating algorithms in literature is also reflected for decision fusion. Some schemes use the accuracy of all algorithms to call a vote [Moreno-Seco et al., 2006], others rely on different measures [Li et al., 2018] [Tidriri et al., 2018]. If an alert based on the vote is subject to a human interface, fundamental research in ergonomics [BLISS et al., 1995] showed:

> *..if consistent responding is desired, alarm reliability (lack of false alarms) is of the utmost importance..*

Methods using the Neyman-Pearson test [Thomopoulos et al., 1987], Bayesian estimation, Dempster-Shafer evidence theory or Neural Networks [Li et al., 2018] can utilize multiple metrics.

Definition 2. Typical metrics for pattern recognition evaluation

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{Fraction correct (accuracy)}$$

$$PPV = \frac{TP}{TP + FP} \quad \text{Positive predictive value (precission)}$$

$$P(TN) = \frac{TN}{TN + FP} \quad \text{True negative rate (specificity)}$$

$$P(TP) = \frac{TP}{TP + FN} \quad \text{Probability of detection (sensivity, recall)}$$

$$P(FP) = \frac{FP}{FP + TN} \quad \text{Probability of false alarm (fall-out)}$$

$$F = 2 \cdot \frac{PPV \cdot P(TP)}{PPV + P(TP)} \quad \text{F-measure}$$

The applicability of aforementioned fusion approaches is questioned for the use cases and the defined detection concept throughout this work. The reason is a heterogeneous nature of the different data sources. The approaches available in literature focus on data sources of the same kind, i.e. network traffic from different IDS sensors, or even the same dataset examined by different detection algorithms. Also complete synthetic data is used for evaluation [Tidriri et al., 2018] [Downs and Vogel, 1993]. The fusion schemes in literature would lead to an effect, where a lower accuracy rate is fabricated by the voting itself. Example 5.1.1 illustrates this.

Example 5.1.1. An attacker tries to change the code on a PLC to achieve malicious behavior. He accesses the network through an engineering terminal, which is usually inactive during normal operation. The network validation service would detect unusual traffic and signal positive detection. Other services would signal normal operation. In an ensemble of five, as presented in section 4.4.2, where all detection services have similar performance metrics, the detected attack would be obfuscated by a fusion algorithm using weights. The probability of an alert would be reduced significantly, leading to undetected attacks by fusion.

For that reason in a reference implementation all detection services should have equal rights and their binary output can be logically linked by an "or" function.

The reference implementation of a signaling service is shown in figure 5.5. It was initially developed within the context of the research project STEUERUNG and later modified for this work. Every detection service signals its state towards that, respectively using the states OK (green), Warning (Yellow) and Anomaly (red). For detecting attacks congesting the network, like DoS, or if the detection services itself get targeted, timeouts are implemented using a multiple of respective process time constants. During that time each detection service has to signal its status, otherwise the component gets flagged with warning-state.

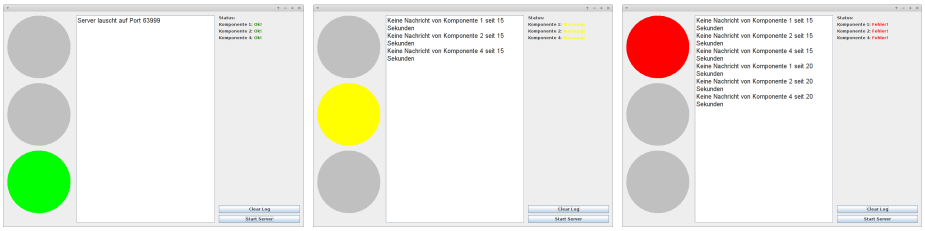

Figure 5.5.: Signaling service in form of a traffic light

If the timeout gets exceeded the respective component has to be considered as failed. The signaling services reacts accordingly marking the service with Anomaly-state.

5.2. Evaluation concept

Related algorithms and methods shown in section 3.2 are usually individually evaluated using simulations, testbeds, synthetic data or real-world data [Urbina et al., 2016]. A comparison based on these is difficult, since not only the evaluation methodology, but also the metrics to measure the effectiveness of each approach differ. Since "*there is no globally acceptable standard/ metric for evaluating an intrusion detection system*" [Patcha and Park, 2007] each approach focuses naturally on its strengths during evaluation. Furthermore there are few real datasets or test-beds available (e.g. [Mathur et al., 2016]), which represent specific application domains.

The application specific methodology developed within this work demands additionally an evaluation according to practical requirements (ref. to section 4.3.2). Since the developed concept is highly modular, this has to be done for each detection service individually. Furthermore the complete set-up has to be evaluated under realistic conditions to allow a prediction towards its applicability in practice.

The following section shows in detail the evaluation methodology, the test-bed for attack data generation and live evaluation purposes and the datasets captured from real application scenarios. The evaluation procedure follows the following step: first each detection service gets individually evaluated using specifically generated attack data or using live tests to evaluate the applicability in practice. Second live experiments show the performance of the complete set-up with all implemented detection services. Hence the performance of the evaluation methodology can be discussed.

5.2.1. Evaluation methodology

The methodology for evaluation is derived from the Analytic Hierarchy Process (AHP) [Saaty, 1987] in the context of finding a scoring model towards a utility analysis [Schneeweiss, 1990]. This analysis contains the following steps:

$$w_m = \begin{bmatrix} \text{Direct costs} \\ \text{Scalability} \\ \text{Resource efficiency} \\ \text{Operating expenses} \\ \text{Ease of use} \\ \text{Portability} \\ \text{Influence to process} \\ \text{Expandability} \\ \text{Reconfigurability} \\ \text{Robustness} \end{bmatrix} = \begin{bmatrix} 0.06 \\ 0.084 \\ 0.05 \\ 0.116 \\ 0.06 \\ 0.04 \\ 0.18 \\ 0.07 \\ 0.13 \\ 0.21 \end{bmatrix}$$

Figure 5.6.: Weighted requirements vector (ref. to section 4.3.2)

1. find criteria for decision-making
2. define weights for each criterion
3. define evaluation scales
4. asses alternatives and calculate key index
5. documentation of results

Evaluation criteria should usually be aligned with development requirements. ‘‘*Requirements are qualitative and / or quantitative definitions of properties or conditions for a product. Different weights can be defined* [..]” [VDI, 1993]. The requirements from practice (ref. to section 4.3.2) can be used to evaluate the individual solutions within the overall concept. The weights already obtained by qualitative expert interviews were rated according to AHP and can now be used to calculate an indicator k_p that represents the ratio for meeting the requirements from practice per solution.

$$k_p = \frac{w \cdot a}{P} \tag{5.1}$$

where w is a vector containing the weights, a is a vector containing the resulting points in each requirements category and P is the maximum number of points to normalize the result. The arithmetic mean of weights (ref. to figure 4.10) are shown in figure 5.6.

The scale of point values for evaluating the individual requirement per solution contains numbers from 1 to 5, where "1" is the worst and "5" the best value.

5.2.2. Test-bed

An evaluation of developed methods and reference implementations is not possible in a live system of end users. There are several reasons which prohibit that:

Interruption of operation can happen by testing a novel software due to unforeseen side-effects or bugs. This could lead in application scenario A (ref.

to section 4.3.1) to an interrupted supply of water, power, oil, telecommunication or similar basic urban infrastructures. In application scenario B (ref. to section 4.3.1) the interruption means directly an interruption of functionality in production systems, which usually leads to a direct financial loss of the company owning the factory. The example in [Duggan et al., 2005] showed, that this is not a fictional scenario.

Safety loss based on the interruption leads to direct hazard for personnel or other people interacting with the machinery. A suddenly occurring irrational movement can injure or kill a person within the working space of the respective machine.

Security issues can further arise by implementing and evaluating new methods. The interplay of technical systems and tools should usually be well planned and orchestrated. The unmanaged involvement of new methods which are subject to research and therefore not as well tested as a commercial product could lead to spontaneous security holes.

Objection is guaranteed by operating companies to integrate untested systems into their infrastructures. Without permission this cannot be done legally.

For these reasons, design and use of a test environment is a crucial aspect for the evaluation of novel methods. To achieve best performance, the test-bed should be designed and implemented very close to real conditions. Therefore the entire infrastructure must be included into consideration for design and concept of the simulation environment. Simulation of a single element of the entire ATI, for example only the PLC, is not sufficient. The evaluation of developed methods relies on data that is as realistic as possible, at best acquired in the live system. So the requirements for a test-bed are demanding. Changes in code of one element or in communication behavior between multiple elements alter the overall behavior and lead to changes in the resulting evaluation datasets. A test-bed simulating an entire waterworks plant resulting from the research project STEUERUNG (ref. to section 4.3) is available as evaluation use-case. A photo and the respective schematic representation of the test-bed is shown in figure 5.7. The components mirror hardware and software specifications strictly as used in the real use-case. Workstations and servers were cloned and run as virtual machines, network connections and PLC components have the exact specifications as used in the real system. The software running on the PLC is also an exact copy. The underlying I/O and process simulation generates the necessary data to run the PLC correctly and was developed by a third party using commercially available simulation software. Furthermore the datasets to initialize the simulation are captures from the real application scenario. The whole setup leads to a very realistic test-bed.

Using the test-bed it is possible to simulate different evaluation scenarios. Based on data captured within the real system trajectories of water consumption can be specified. Based on that the simulation generates process-values (sensors and actuators) at the input and output channels of the PLC. From the point of view of the controller software there is no difference controlling the simulation or a real plant, since was not altered. The following components are part of the

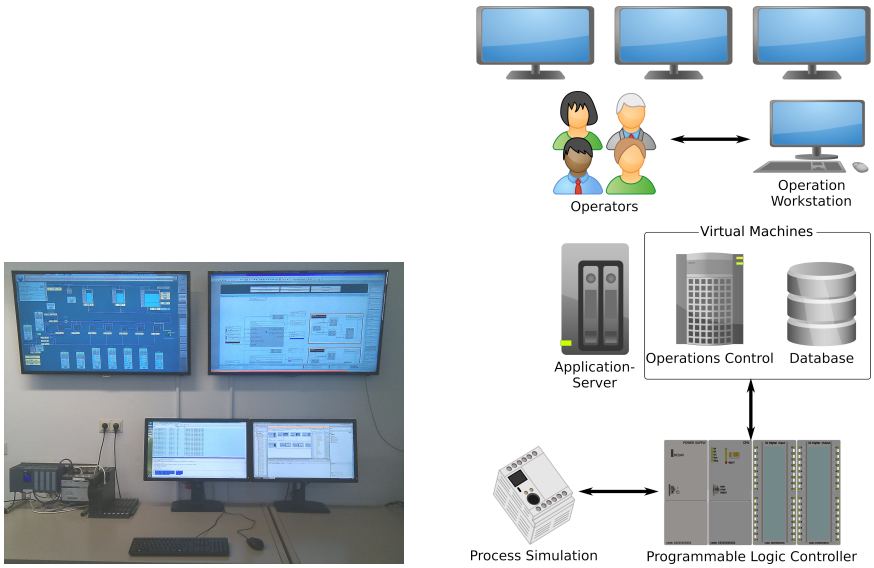

Figure 5.7.: Photo and schematics of simulation and test environment

test-bed:

Operations Control includes HMI and SCADA components as well as a database. Operator input to the test-bed is available using the real HMI this way. It is connected to operations control which exchanges data with the underlying PLC. The protocol used in the real system and therefore also in the test-bed is Ethernet based and uses the S7-protocol directly on top of the Ethernet frames (ISO transport protocol instead of IP-based communication).

PLC has the main purpose to control the physical process while incorporating commands from operators. It is linked to the underlying physical process through sensors and actuators, which are interconnected by the industrial standard field bus PROFIBUS. Even though this is a deprecated field bus it is still used in the real system.

Process Simulation is achieved using a HIL simulation to serve I/O signals to the PLC. Therefore the PLC software can be used without changes and it is connected to simulated sensors and actuators. The physical behavior of the plant is integrated as a process model directly into the HIL simulation application using a scripting interface. Demands of the end user of the critical infrastructure can be modeled this way, where real data captures form the basis for this. As shown in figure 5.8 simulated process parameters include volumetric flow rate $Q(t)$, pressure $p(t)$, pump engine revolutions $n(t)$ and tank water levels $V(t)$. Some physical values, such as pH-value and temperature, are not part of the simulation.

The existing test-bed simulating a waterworks can be used unmodified for evaluation purposes of detection services. The flow of water through the (real and simulated) plant is as follows: water is pumped out of the ground by nine

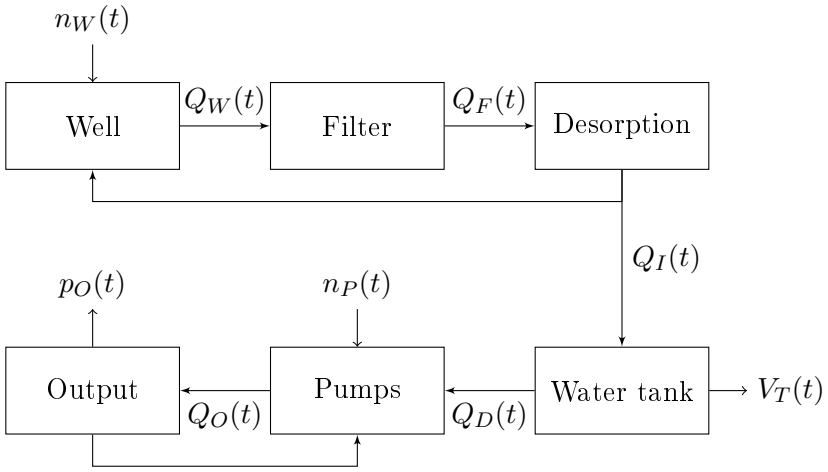

Figure 5.8.: Components and control feedback of the physical simulation

fountain-pumps. Subsequently it gets purified through filters and desorption units. The clean water flows into three water tanks. These reservoirs are used like a buffer which is needed in times of high water consumption. From there seven powerful clean-water-pumps generate the water-flow and pressure that is needed to supply all consumers. The controlled variables for water output are pressure $p(t)$ and Volumetric flow rate $Q(t)$. Both are established by the interplay of demand and reacting control of clean-water-pumps.

5.2.3. Datasets

To achieve a realistic simulation, the test-bed should be provided with data captured from real use cases. Figure 5.9 shows the controlled variables pressure $p(t)$ and Volumetric flow rate $Q(t)$ of three complete days of data captured in the real waterworks plant. One day represents a regular workday in February, the other two represent special occasions for water supply companies: Christmas eve and New Year's Eve. Despite these data sets show a different inhabitants' behavior in water consumption they still show similarities within the day and night cycle.

The complete set of data represents several different full days (00:00:00 until 23:59:29 hours) of district related water consumption and the respective plant wide traffic including Ethernet and field buses. The data was captured within the context of the aforementioned research project STEUERUNG (ref. to section 4.3) and is used to achieve realistic simulation results. It is also the foundation to implement attack scenarios and generate theoretically an unlimited amount of attack datasets.

Furthermore it is possible to generate synthetic datasets from existing ones (ref. to [Klein, 2017]). For example the daily consumption rates from the three days shown in figure 5.9 can be combined to generate those, as shown in [Klein, 2017]. Those synthetic datasets have been used for individual evaluation of some

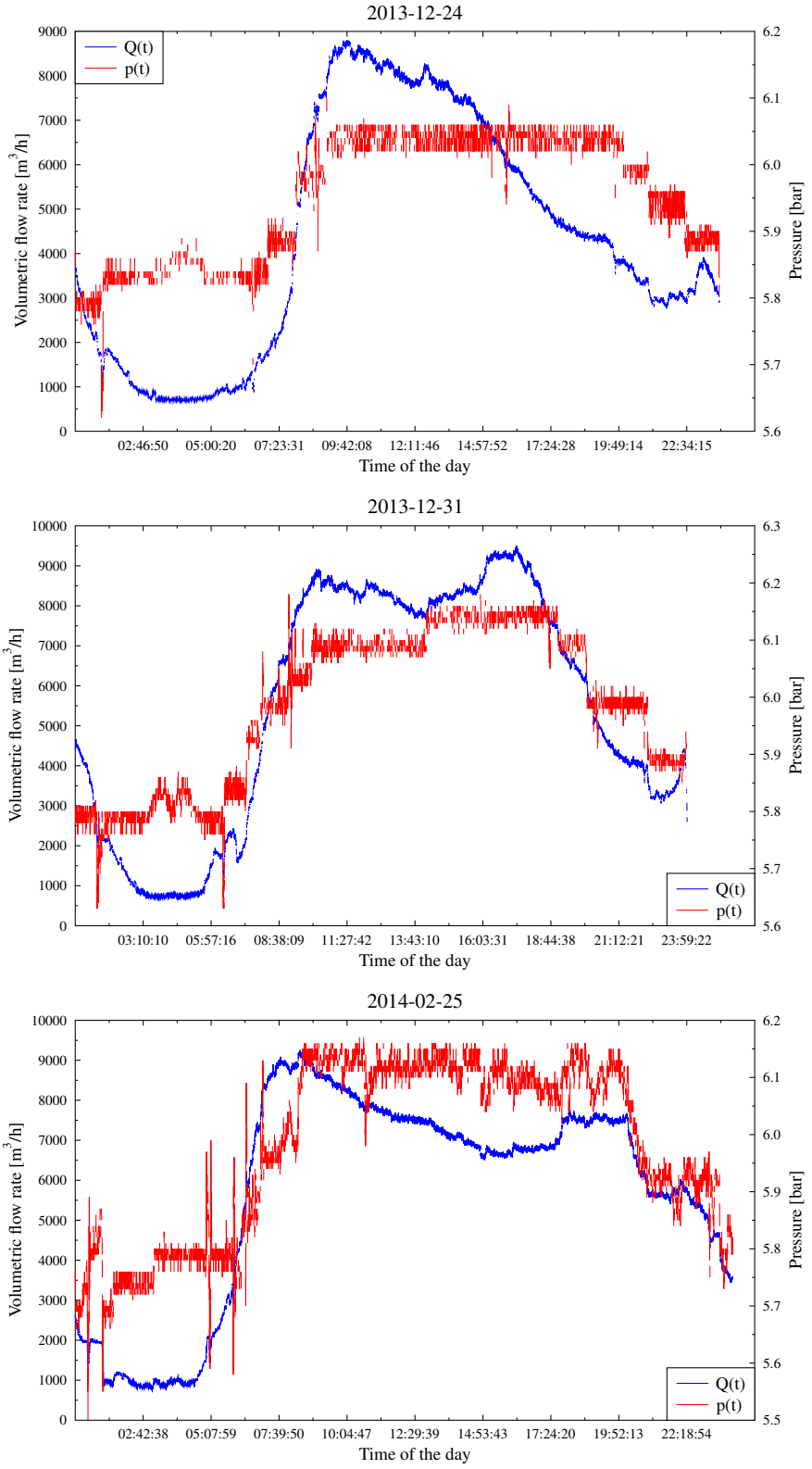

Figure 5.9.: Three datasets containing complete days from live captures

```
$ capinfos 20180406_capture_Tagessim.pcap
File name:           20180406_capture_Tagessim.pcap
File type:           Wireshark/... - pcapng
File encapsulation:  Ethernet
File timestamp precision:  microseconds (6)
Packet size limit:   file hdr: (not set)
Number of packets:   207 M
File size:           53 GB
Data size:           46 GB
Capture duration:    193741,036429 second
First packet time:   2018-04-06 15:30:10,696989
Last packet time:    2018-04-08 21:19:11,733418
Data byte rate:      241 kBps
Data bit rate:       1.933 kbps
Average packet size: 225,84 bytes
Average packet rate: 1.069 packets/s
SHA256:              4371a4f3237d70fd05cf6f1b83f9f873c736140f6c7fb6202c02cd412009fcc4
RIPEMD160:           275d133d38d05517edd02ec0512bf9427be8b0e1
SHA1:                db6e491302c6a2bdc15d0e8fc05f3c733264161b
Strict time order:   False
Capture oper-sys:    64-bit Windows 7 Service Pack 1, build 7601
Capture application: Dumpcap 1.10.8 (v1.10.8-2-g52a5244 from master-1.10)
Number of interfaces in file: 1
Interface #0 info:
Name = \Device\NPF_{A9EEDF20-576C-4AB1-961A-8634AE110CEE}
Encapsulation = Ethernet (1 - ether)
Capture length = 65535
Time precision = microseconds (6)
Time ticks per second = 1000000
Time resolution = 0x06
Operating system = 64-bit Windows 7 Service Pack 1, build 7601
Number of stat entries = 1
Number of packets = 207282678
```

Listing 5: Three days of network traffic captured in the test-bed

services. This is especially important for detection services which use a learning algorithm. By using synthetic data more input is available for testing and evaluation purposes. Here datasets containing process values similar to figure 5.9 as well as live network captures, as shown in listing 5, can be generated. These network datasets can be used for individual evaluation of services that monitor especially network traffic.

5.2.4. Attack scenario implementations

The three different attack scenarios presented in section 4.3.4 reflect realistic approaches from practice. Some of the attacks described in literature are also part of these, e.g. [Klick et al., 2014] also focuses on modification of PLC source code. Expert interviews (ref. to appendix A.1) especially for the real use case, which the test-bed is based on, showed the following feasible approaches:

Plant-wide disturbance of operation can be achieved relatively easily using DoS attacks within the network. An attack towards connections with PLC devices will disrupt operation in full leaving operators running the system "blind". This means no process values reach the PLC or HMI and no commands reach the underlying actuators manipulating the physical process. During the time of attack anything might happen. Attack scenario 1 in section 4.3.4 is reflected.

Fast destruction of pumps through manipulation of feeder gates and pump revolutions. Three different physical effects can be used, which all lead to melting and fusing of pump parts within minutes: hot running, dry run-

ning and cavitation (ref. to [Stauß, 2005]). To achieve hot running the valve in the pressure line needs to be closed completely without switching off the pump. The medium then circulates inside the pump housing and heats up. With dry running the valve in the suction line needs to be closed completely without switching off the pump. As a result there is too little liquid in the pump housing and the plain bearings heat up because they are insufficiently cooled. Cavitation is a local evaporation of liquid caused by a hydrodynamic pressure reduction. The vapor bubbles are carried along by the flow and collapse implosion-like as soon as the pressure rises above the vapor pressure again. This is accompanied by strong vibrations and high-frequency pressure surges, which destroy the components through erosion. Too high flow velocities lead to a negative pressure that causes the liquid to evaporate and hot-running also favors cavitation. This reflects attack scenario 2 in section 4.3.4.

Stealthy running dry all tanks can be done by reducing flow rate or shutting down[1] the well pumps, filter systems or increasing the pump revolutions of clean water pumps. Furthermore a manipulation of the values shown in the HMI is necessary for obfuscation. After the tanks run dry this will lead to an interrupted water supply for the supply area. Furthermore dry-running of pumps can be triggered. This reflects attack scenario 3 in section 4.3.4.

Life-threatening hazards like automatically discharging too much chlorine into the clean-water at the plants outlet valve. While this scenario is possible its feasibility is low due to practical circumstances. Chlorine itself gets used to react to possible bacterial contamination. Operators set this off when necessary, but no huge amounts of chlorine are stored within the facility. Just-in-time delivery gets used instead. Therefore an attacker cannot unleash dangerous concentrations within the clean water. Discharging other hazardous elements usually needs manual access to the clean-water tanks, which is not the focus of this work.

There are two ways to perform realistic attacks within the test-bed. Firstly process parameters can be forced by a consecutively carried out batch script (ref. to section 5.2.2), which mirrors an attack on the field device level. This way it is possible to generate several test-cases with arbitrary input data. Secondly the attack itself can be implemented and performed live while the simulation is running, which mirrors an attack on control level. Either way results in a realistic setup and the reaction of the test-bed software can be tested too. Furthermore it is possible to generate several attack datasets for evaluation purposes.

Each attack scenario consists of components which were implemented as follows:

[1] Shutting down a device is not stealthy, especially pumps have a size that shutting down can be felt in the local operations control room.

Denial of Service

The tools to implement this kind of attack are already available in the operating systems itself. For example the HMI system is Microsoft Windows-based, where a tool called "ping" is available. The simple script shown in listing 6 implements the whole attack. A nice side effect for attackers is that this script also uses the so called "Ping of Death" and will crash any device prone to that [Shekhar, 2016]. The script can be used in multiple instances congesting any network

```
:loop
ping <device ip here> -l 65500 -w 1 -n 1
goto loop
```

Listing 6: Simple DoS attack script

completely. This way it scales well towards larger infrastructures.

Code manipulation

The source project on the PLC device of the testbed, which is the unmodified original code from the real use case running in the plant, is relatively complex to analyze. As shown in listing 7 it contains 221 code blocks (OB, FB, FBC, FCC, SFB, SFC) and 168 data blocks (DB, SDB).

```
------------------------------------------------------
| UNIT Connection
Connected to   : 10.0.2.20 (Rack=0, Slot=2)
PDU Requested  : 480 bytes
PDU Negotiated : 480 bytes
OBCount  : 13
FBCount  : 36
FCCount  : 57
SFBCount : 29
SFCCount : 86
DBCount  : 148
SDBCount : 20
```

Listing 7: Code and data blocks list of test-bed PLC

After the cumbersome analysis a sub-function (FB) was chosen for manipulation. It is a function which calculates an average for a given amount of input values. It was chosen, because it is used in different places throughout the whole project and affects especially sensor readings. The idea is to show wrong values to the operators and therefore let them initialize the wrong measures. This is a similar approach to the computer-worm STUXNET [Falliere et al., 2011].

The resulting manipulation is shown in figure 5.10. The output value of the function is simply divided by 1.1, which equals returning around 90% of the correct value. A division was chosen for reasons of obfuscation, since only divisions are used in the original part of the code. The resulting binary forms can be seen in listings 8 and 9.

To upload the code to the PLC device during runtime a self implemented tool using the Snap7 [Nardella, 2015] library was used.

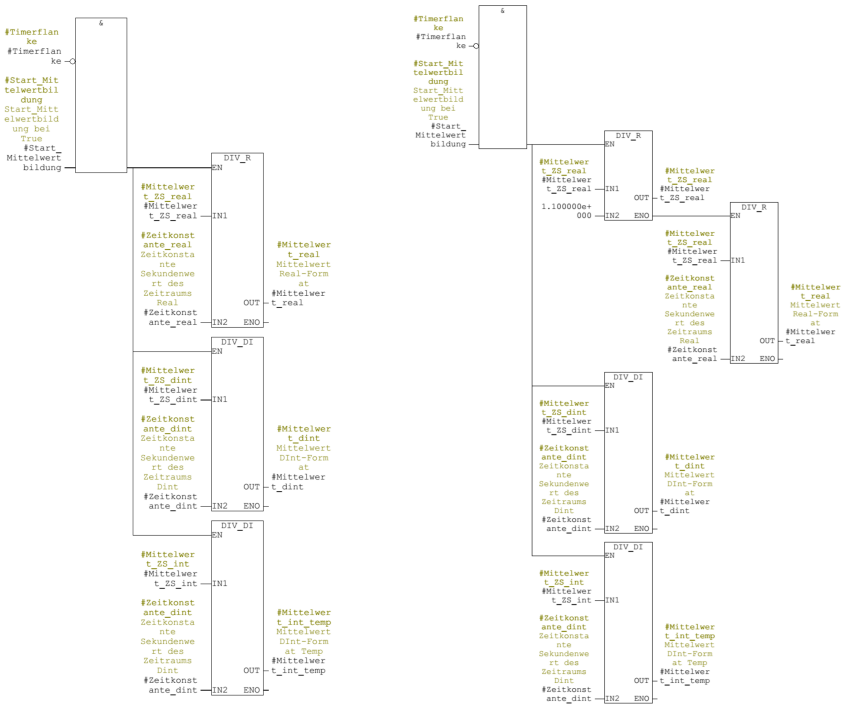

Figure 5.10.: Original (left) and manipulated (right) part of the IEC61131 code

```
Dump of Block Type 0x45 No.??? (714 bytes) :
0x0000: 79 58 00 01 79 58 00 00 79 dd 01 60 41 60 00 00 00 60 00 00 79 58 00 00
0x0018: ff 98 00 09 30 03 00 01 be 5a 01 50 79 00 be 5e 01 50 00 00 ba 00 be 5a
0x0030: 01 50 be 5a 00 10 21 a0 bf 00 79 dc 01 62 ff 98 00 06 30 03 00 00 be 5e
0x0048: 01 50 00 00 be 5a 00 30 68 0c 7e 66 00 10 30 07 00 0c be 5a 00 30 68 24
0x0060: 7e 66 00 12 7e 62 00 12 30 03 00 00 21 80 ff f8 00 17 7e 62 00 12 30 03
0x0078: 00 01 21 80 ff f8 00 19 7e 62 00 12 30 03 00 02 21 80 ff f8 00 1b 7e 62
0x0090: 00 12 30 03 00 03 21 80 ff f8 00 1d 7e 62 00 10 30 03 00 64 60 00 be 5e
0x00a8: 01 70 70 0b 00 1b 7e 62 00 10 30 03 00 0a 60 00 be 5e 01 70 70 0b 00 12
0x00c0: 7e 62 00 10 30 03 00 01 60 00 be 5e 01 70 70 0b 00 09 7e 62 00 10 30 03
0x00d8: 00 0a 60 04 be 5e 01 70 00 00 be 5a 01 70 be 5a 00 10 60 00 7e 66 00 02
0x00f0: 00 00 7e 62 00 02 68 1e 7e 67 00 04 00 00 7e 63 00 04 68 06 7e 67 00 08
0x0108: 00 00 be 5a 00 80 68 1e 7e 67 00 0c 00 00 ba 00 7e 67 00 14 70 02 7e 67
0x0120: 00 18 be 5a 00 20 7e 66 00 1c 7e 63 00 18 7e 63 00 14 bf 62 00 1c 79 5b
0x0138: 00 00 bf 00 79 dd 01 61 be 5a 00 30 7e 67 00 14 70 02 7e 67 00 18 be 5a
0x0150: 00 20 7e 66 00 1c 7e 63 00 18 7e 63 00 14 bf 6a 00 1c 00 00 00 00 7e 67
0x0168: 00 14 70 02 7e 67 00 18 be 5a 00 20 7e 66 00 1c 7e 63 00 18 7e 63 00 14
0x0180: bf 69 00 1c be 5e 00 e0 00 00 79 58 01 62 79 58 00 00 79 58 01 61 41 60
0x0198: 00 14 00 60 00 14 ff 98 00 09 be 5b 00 40 be 5b 00 f0 60 0f be 5f 00 f0
0x01b0: 00 00 00 60 00 14 ff 98 00 09 be 5b 00 60 be 5b 01 10 60 0d be 5f 01 10
0x01c8: 00 00 00 60 00 14 ff 98 00 09 7e 63 00 0c be 5b 01 30 60 0d be 5f 01 30
0x01e0: 00 00 79 59 01 61 79 58 00 00 41 60 00 14 00 60 00 14 ff 98 00 09 be 5b
0x01f8: 00 f0 7e 63 00 08 60 03 be 5f 00 90 00 00 00 60 00 14 ff 98 00 09 be 5b
0x0210: 01 10 7e 63 00 04 60 0e be 5f 00 b0 00 00 00 60 00 14 ff 98 00 09 be 5b
0x0228: 01 30 7e 63 00 04 60 0e be 5f 01 80 00 00 79 59 01 61 79 58 00 00 41 60
0x0240: 00 14 00 60 00 14 ff 98 00 07 38 01 00 00 00 00 be 5f 00 f0 00 00 00 60
0x0258: 00 14 ff 98 00 07 38 03 00 00 00 00 be 5f 01 10 00 00 00 60 00 14 ff 98
0x0270: 00 07 38 03 00 00 00 00 be 5f 01 30 00 00 ba 00 be 5b 01 80 38 03 00 00
0x0288: 7f ff 39 a0 bf 00 ff 98 00 07 38 03 00 00 7f ff be 5f 01 80 00 00 ba 00
0x02a0: be 5b 01 80 38 03 ff ff 80 00 39 c0 bf 00 ff 98 00 07 38 03 ff ff 80 00
0x02b8: be 5f 01 80 00 00 be 5b 01 80 be 5e 00 d0 00 00 65 00
```

Listing 8: Binary form of original code block (ref. to fig. 5.10)

```
Dump of Block Type 0x45 No.??? (746 bytes) :
0x0000: 79 58 00 01 79 58 00 00 79 dd 01 60 41 60 00 00 00 60 00 00 79 58 00 00
0x0018: ff 98 00 09 30 03 00 01 be 5a 01 50 79 00 be 5e 01 50 00 00 ba 00 be 5a
0x0030: 01 50 be 5a 00 10 21 a0 bf 00 79 dc 01 62 ff 98 00 06 30 03 00 00 be 5e
0x0048: 01 50 00 00 be 5a 00 30 68 0c 7e 66 00 10 30 07 00 0c be 5a 00 30 68 24
0x0060: 7e 66 00 12 7e 62 00 12 30 03 00 00 21 80 ff f8 00 17 7e 62 00 12 30 03
0x0078: 00 01 21 80 ff f8 00 19 7e 62 00 12 30 03 00 02 21 80 ff f8 00 1b 7e 62
0x0090: 00 12 30 03 00 03 21 80 ff f8 00 1d 7e 62 00 10 30 03 00 64 60 00 be 5e
0x00a8: 01 70 70 0b 00 1b 7e 62 00 10 30 03 00 0a 60 00 be 5e 01 70 70 0b 00 12
0x00c0: 7e 62 00 10 30 03 00 01 60 00 be 5e 01 70 70 0b 00 09 7e 62 00 10 30 03
0x00d8: 00 0a 60 04 be 5e 01 70 00 00 be 5a 01 70 be 5a 00 10 60 00 7e 66 00 02
0x00f0: 00 00 7e 62 00 02 68 1e 7e 67 00 04 00 00 7e 63 00 04 68 06 7e 67 00 08
0x0108: 00 00 be 5a 00 80 68 1e 7e 67 00 0c 00 00 ba 00 7e 67 00 14 70 02 7e 67
0x0120: 00 18 be 5a 00 20 7e 66 00 1c 7e 63 00 18 7e 63 00 14 bf 62 00 1c 79 5b
0x0138: 00 00 bf 00 79 dd 01 61 be 5a 00 30 7e 67 00 14 70 02 7e 67 00 18 be 5a
0x0150: 00 20 7e 66 00 1c 7e 63 00 18 7e 63 00 14 bf 6a 00 1c 00 00 00 00 7e 67
0x0168: 00 14 70 02 7e 67 00 18 be 5a 00 20 7e 66 00 1c 7e 63 00 18 7e 63 00 14
0x0180: bf 69 00 1c be 5e 00 e0 00 00 79 58 01 62 79 58 00 00 79 58 01 61 41 60
0x0198: 00 14 00 60 00 14 ff 98 00 09 be 5b 00 40 be 5b 00 f0 60 0f be 5f 00 f0
0x01b0: 00 00 00 60 00 14 ff 98 00 09 be 5b 00 60 be 5b 01 10 60 0d be 5f 01 10
0x01c8: 00 00 00 60 00 14 ff 98 00 09 7e 63 00 0c be 5b 01 30 60 0d be 5f 01 30
0x01e0: 00 00 79 59 01 61 79 58 00 00 41 60 00 14 ba 00 00 60 00 14 ff 98 00 0d
0x01f8: be 5b 00 f0 38 01 3f 8c cc cd 60 03 be 5f 00 f0 ff 11 68 2c 68 1c ff e0
0x0210: bf 00 ff 98 00 09 be 5b 00 f0 7e 63 00 08 60 03 be 5f 00 90 00 00 00 60
0x0228: 00 14 ff 98 00 09 be 5b 01 10 7e 63 00 04 60 0e be 5f 00 b0 00 00 00 60
0x0240: 00 14 ff 98 00 09 be 5b 01 30 7e 63 00 04 60 0e be 5f 01 80 00 00 79 59
0x0258: 01 61 79 58 00 00 41 60 00 14 00 60 00 14 ff 98 00 07 38 01 00 00 00 00
0x0270: be 5f 00 f0 00 00 00 60 00 14 ff 98 00 07 38 03 00 00 00 00 be 5f 01 10
0x0288: 00 00 00 60 00 14 ff 98 00 07 38 03 00 00 00 00 be 5f 01 30 00 00 ba 00
0x02a0: be 5b 01 80 38 03 00 00 7f ff 39 a0 bf 00 ff 98 00 07 38 03 00 00 7f ff
0x02b8: be 5f 01 80 00 00 ba 00 be 5b 01 80 38 03 ff ff 80 00 39 c0 bf 00 ff 98
0x02d0: 00 07 38 03 ff ff 80 00 be 5f 01 80 00 00 be 5b 01 80 be 5e 00 d0 00 00
0x02e8: 65 00
```

Listing 9: Binary form of manipulated code block (ref. to fig. 5.10)

Parameter manipulation

This part of an attack includes the manipulation of process parameters and values. The easiest way for an attacker to do so (apart from code manipulation as shown in the last section) is the capturing of existing network infrastructure or smart sensor/ actuator devices in the field. Also a placement of own hardware that manipulates the network traffic or signals directly to the PLC can be feasible.

The scripting interface of the process simulation software was used to simulate that kind of attacks. Here any simulated field device can be forced to deliver any desired value, in context of the overall simulation or even separately. Figure 5.11 shows the part of the physical process simulation for the clean water tanks. As shown, manipulations can even be done manually without the scripting interface using the slider buttons shown on the left. While this is an alternative for quick tests, it is neither a feasible approach to generate datasets spanning several days and nights, nor sophisticated attack procedures.

The manipulation of process values was carried out in a similar way to code manipulation attack. Selected variables were multiplied by a factor and labeled for later learning. The implemented manipulations are related to attack scenarios elaborated during this work (ref. to section 4.3.4) and details are implemented as follows:

- Reduction of the flow rate of filter system
- Reduction of the flow rate of wells
- Increasing the flow rate of clean water pumps

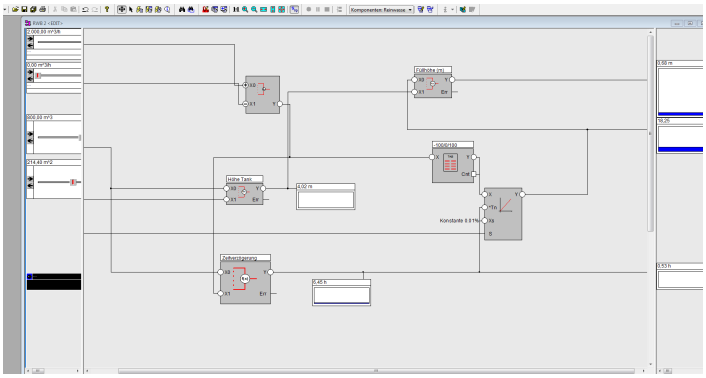

Figure 5.11.: Process simulation example of clean water tanks

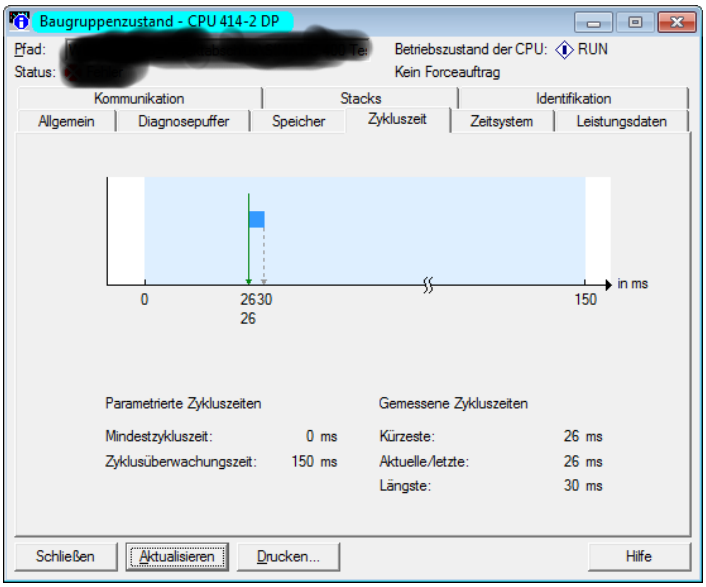

Figure 5.12.: Cycle time of the PLC in the test-bed

Manipulation offsets are within the ranges of 10%, 5% and 1%. Attack datasets generated this way contain complete days with manipulation durations of the time necessary to run a tank dry. This is usually achieved within the time-span of one hour.

The Fastest Wins

Especially using direct signaling of an additional device as stated in the last section, a special case of parameter manipulation was discovered during the implementation and testing phase for a vSoftPLC of the *behavior validation service* [Kittmann, 2017]. It became evident that it s possible to overwrite data blocks and registers of PLCs using an appropriate timing.

As shown in figure 5.12 the cycle time of the test-bed PLC ranges from 26 to 30 ms. according to the standard work-flow of a PLC, the input values are read

at the beginning of the cycle and the outputs are updated at the end. In between the implemented process logic gets calculated. Since the difference of cycle times is 4 ms the rate to overwrite process or data values would be according to the Nyquist frequency half of that, which is $2ms$. Therefore an attacking device that overwrites PLC registers and data blocks every 2 ms will be able to alter anything in that application specif process example. The drawback for the attacker would be a flooding of the network, which might be detected easily.

5.3. Evaluation results

The evaluation was conducted either directly within the test-bed or using generated datasets from it. First, each detection service was evaluated independently. Second, the ensemble was evaluated with live tests in the test-bed. This way all services get evaluated not only accordingly to the aforementioned evaluation methodology, but the complete setup gets tested under realistic conditions.

A very important parameter for monitoring is the respective sample time. It is highly dependent on the application specific process cycle times. The following criteria have to be considered:

- the physical process time values (here $> 10s - Xh$),
- the executing controllers capabilities (here $> 10ms$),
- the conversion time of inputs and outputs (here $10 - 100ms$),
- the network protocol and hardware (here $< 1ms$), and
- the number and type of IEC61131 instructions used (here $26 - 30ms$).

The physical process times for a clean water pump can be a multiple of seconds, while the other times from this list are usually below 100 ms. Therefore a monitoring of physical process values requires a sample time depending on the physical process times, as for the application scenario the test-bed is based on. The real pumps used have an average start-up time of 10s, while the clean water tanks take up to several hours to fill up or run dry. The monitoring sample time to measure physical process values gets set according to these application specific circumstances to $1s$. This value is especially relevant for active parts of data acquisition, like applications based on the Snap7 library. Passive monitoring, eg. getting the process values from the data section of network packets, could be done as fast as possible, since no potential disturbance arises from that. Based on that the sliding window size or lag gets set to 20 ($\simeq 20s$) being able to capture relevant patterns of physical process values.

5.3.1. Individual service evaluation

The individual evaluation includes the evaluation according to the evaluation methodology for each service. The results for all services can be compared in table 5.1.

Requirement	NTVS	CVS	SVS	PVS	BVS
Direct costs	●●●●◐	●●●●●	●●●●●	●●●●○	●○○○○
Scalability	●●●○○	●●●○○	●●●●●	●●●●○	●○○○○
Resource efficiency	●●●●○	●●●●●	●●●●●	●●●○○	●○○○○
Operating expenses	●●●○○	●●●●●	●●●●●	●●●○○	●●○○○
Ease of use	●○○○○	●●○○○	●●○○○	●○○○○	●○○○○
Portability	●●●●○	●●●●●	●●●●●	●●●●●	●○○○○
Influence to process	●●●●●	●○○○○	●○○○○	●●●●●	●●●●●
Expandability	●○○○○	●●●●●	●●●●●	●●●●●	●○○○○
Reconfigurability	●●○○○	●●●●●	●●●●●	●●●●●	●○○○○
Robustness	●○○○○	●●●●●	●●●●●	●●●●●	●●●○○
Weighted Total	●●◐○○	●●●●○	●●●●○	●●●●○	●●○○○

●○○○○ poorly fulfilled ... ●●●●● well fulfilled

Table 5.1.: Comparison of weighted evaluation results for individual services

For each requirement the following considerations have been taken into account:

Direct costs can only be approximated for the application-specific solutions developed here. The time to acquire and integrate the solution is estimated by the author according to own experiences. For simplicity and better comparability an hourly wage is assumed with €100. Software purchase costs and equivalent costs are included upon availability of information. Other costs like hardware purchase or power consumption are considered equal for all solutions and therefore negligible. Low costs equal to 5 points while high costs equal 1 point, where a linear distribution from maximum to minimum is applied.

Scalability is approximated per solution according to the ability of the solution to process a greater amount or higher complexity of input data. If the solution still performs well with more and increasingly complex input data (i.e. five times more than the aforementioned sampling rate) it is rated 5 points, while a solution which does not scale is rated only one point. A linear interpolation from maximum to minimum is applied.

Resource efficiency is measured in terms of RAM consumption and, CPU load of the respective process for each solution. To allow a comparison they were executed on the same machine for all experiments. A low footprint is associated with 5 points and a high one rates only one point. A linear interpolation from maximum to minimum is applied.

Operating expenses include license fees and or possible maintenance costs. Low costs equal to 5 points while high costs equal 1 point, where a linear distribution from maximum to minimum is applied.

Ease of use is estimated by the complexity of the solutions interface and the knowledge of the user necessary. A complex parametrization and training of the algorithm that can only be done by an expert rates only one point,

where a one-click interface that can be integrated by a freshman would rate five points.

Portability is estimated in terms of relocatability of the solution between the deployment nodes (ref. to Fig. 5.1). This way it can be estimated how well the solution can be ported to different runtime environments even with different hardware specifications. A solution that runs on the embedded node and can get ported to other nodes easily equals 5 points, where a solution that can only be run in the global computing node and cannot be ported equals only one point.

Influence to process is determined by resource consumption within the ATI to monitor. If the solution does not generate any additional memory usage and network or computing load it rates five points. If the solution generates additional load in all these three categories it rates only 1 point.

Expandability means how well the solution can be updated to more recent developments. Each solution consists of different parts, e.g. model based detection and classification. If the solution itself can be updated with new algorithms that perform better or additional preprocessing steps easily, than it is rated five points. For the opposite behavior only one point applies. A linear interpolation from maximum to minimum is applied.

Reconfigurability means how well can the solution be adapted to the specific application context, for example to different system layers. A solution that is only able to model one specific network protocol and cannot be easily reconfigured is rated only one point. On the other hand a solution that can be integrated easily into another part or layer of the ATI rates five points.

Robustness is measured by detection rate for the individual solution, which is reflected by showing the accuracy in conjunction with false positives and false negatives. Further measures can arise solution specific. A solution with a high accuracy, no false positives and few false negatives would rate five points. A low detection rate including a high false positive rate would rate only one point. A linear interpolation from maximum to minimum is applied.

The following sections show the evaluation results in detail per service.

Network Traffic Validation

Protocol, communication or state-based patterns are derived from network traffic by the vast majority of all related approaches shown in section 3.2. The test-bed (ref. to section 5.2.2) uses the SIEMENS S7 protocol on Ethernet frames (not IP based), so approaches that have been designed for or at least proven to work with this protocol can be used. Only the approach presented in [Kleinmann and Wool, 2014] fulfills this requirement. Other approaches focus on Modbus/TCP, power transmission protocols like IEC61850 or rely on TCP/IP based communication.

The naive DFA modeling algorithm shown in algorithm 5.1 ([Goldenberg and Wool, 2013] and [Kleinmann and Wool, 2014]) was evaluated. It is roughly estimated that the time necessary to train the algorithm is $40hours$, based on the network datasets described in section 5.2.3 and on the evaluation hardware mentioned earlier. This means an experienced data scientist collects the necessary symbol patterns in the (test-bed) network traffic and constructs the DFA from that. Since free implementations exist, development costs do not count. Following that the algorithm can be used straight away for detection. Its memory footprint is below $1GB$ and the CPU-load was below 10% during the experiments. Since the network traffic is captured it works without using any resources from the ATI, therefore an influence to the process is not existent. To evaluate the robustness traffic captures of the test-bed were used. These include complete days of scenarios while attack datasets were generated (ref. to section 5.2.3 and 5.2.4).

	NTVS	**[Kleinmann and Wool, 2017]**
Accuracy (%)	54.41	96.6
False positives (%)	24.76	5.3
False negatives (%)	20.83	N/A

Table 5.2.: Detection rates for network traffic validation

The average detection rates for all experiments are shown in table 5.2. For reasons of comparability the results of the original papers are given. Please note that the original authors did not test any attack scenarios at all, they just presented results for "*benign*" traffic. Furthermore detection rates under varying conditions are not given, it is just said that this "*results in very high anomaly rates*" (for normal traffic without real anomalies).

Code Validation

For the reference implementation a fingerprint for code validation equals the hash value of a binary code or data block of a given PLC. The monitoring of this service is one part of the ensemble that depends strictly on active monitoring, since it makes use of the Snap7 library. A reference database has to be established, as shown in algorithm 5.3. This can be done automatically, the average skilled user only has to pick the respective PLC and binary blocks that are to monitor. This estimated time required is $8hours$. Likewise other services, free implementations exist for large parts of this service, so development costs do not count. Its memory footprint is below $1GB$ and the CPU-load was below 1% during the experiments. To evaluate the robustness, the code manipulation example in figure 5.10 shows the idea. Several code blocks were monitored and their live generated hash value compared to the respective one in the database. Each block was tested 15 times, where 5 manipulations were included per block. Since hashing algorithms showed outstanding performance,

another 10 experiments manipulating random bits in the code and data blocks were conducted.

	MD5	**SHA1**	**SHA256**	**SHA512**
Accuracy (%)	100	100	100	100
False positives (%)	0	0	0	0
False negatives (%)	0	0	0	0
Collision Resistance	No	No	Yes	Yes
Preimage Resistance	No	No	Yes	Yes
Key length (Bit)	128	160	256	512

Table 5.3.: Hash function evaluation values

The average results for all experiments are shown in table 5.3. The respective attacks for collision resistance can be found in [Xie et al., 2010] for MD5 and [Stevens et al., 2017] for SHA1. The attacks on preimage resistance were shown accordingly in [Sasaki and Aoki, 2009] for MD5 and [Stevens, 2012] for SHA1. Based on these results the SHA256-algorithm is chosen at this point for this service, since it offers reliable results and a smaller memory footprint (key length) than SHA512.

Service Validation

Four different fingerprinting algorithms were evaluated for a reference implementation. Three of these use active scanners, only one uses passive scanning techniques[Caselli et al., 2013]. Each algorithm and its respective tool was first used to generate the fingerprint, as shown in listings 3 and 2. These fingerprints were then used to create a reference database for each tool. Following that experiments were conducted automatically. The estimated time to integrate the fingerprint database per tool is $8hours$ and can be done by an average skilled user. Likewise other services, free implementations exist for large parts of this service, so development costs do not count. Each memory footprint is below $1GB$ and each CPU-load was below 1% during the experiments for any of these tools. Real attack scenarios were not implemented, but the web-server running on the S7 PLC was turned on and off to create a different situation for all tools that had to be detected during experiments. Each tool was tested 15 times, where 5 "attacks" were included.

The average results for all experiments can be found in table 5.4. Neither nmap nor p0f were able to even recognize the same device. The observation made in [Caselli et al., 2013] can be confirmed: "*..the tool was not able to recognize the same device. Further analyses showed that the field "icmp echo tos bits" was filled in randomly by the PLC invalidating the signature enough to be discarded..*". For that reason nmap could not recognize the device anymore. Experiments with p0f confirmed the assumption of [Caselli et al., 2013]: "*In*

	nmap	**p0f**	**PLCscan**	**s7scan**	**SVS**
Accuracy (%)	33.33	N/A	66.67	66.67	100
False positives (%)	60.00	N/A	0	0	0
False negatives (%)	6.67	N/A	33.33	33.33	0
Scanning technique	active	passive	active	active	active

Table 5.4.: Detection rates for service validation

ICS environments, there is no guarantee to see any information useful to exploit standard TCP/IP signatures due to long TCP sessions and consequently few useful packets.". The tool was not able to deliver any results due to lack of input data[2]. With help of the tools PLCscan and s7scan it was possible to recognize the device, but no attack [Klick et al., 2014] was recognized, since these tools read the configuration.

```
Host is up (0.00074s latency).
Not shown: 65533 closed ports
PORT    STATE SERVICE
80/tcp  open  http
102/tcp open  iso-tsap
```

Listing 10: Simple port scan of S7-416

Therefore a combination of simple port scanning (refer to listing 10) and s7scan is used as reference implementation of the SVS. This approach is able to additionally detect running services, making it able to detect malicious ones that should not be there.

Process Value Validation

Huge effort has been done to implement and evaluate an approach based on process causality (see definition 1) called Process Causality based Anomaly Detection (PCBAD) (refer to appendix A.4). Here two approaches are evaluated which differ in how the causalities are identified: manually ($PCBAD_M$) and automatically from process data ($PCBAD_D$). Other approaches to validate process data itself presented in [Hadžiosmanović et al., 2014] and [Inoue et al., 2017] were also evaluated. The closest rival presented in [Hadžiosmanović et al., 2014] was implemented using the prediction of an autoregressive model and detecting deviations between prediction error variance and residual variance. Here most of it was already available using existing libraries and frameworks. For that reason the effort for individual approaches differ. If the development costs do not count as for the aforementioned services, the integration effort to identify process semantics and train the respective model is estimated with $80 hours$ per approach. The only exception is the approach based on DNNs presented in

[2]The traffic was additionally captured. An analysis revealed that only 0.1% of packets were relevant HMI-PLC connections.

[Inoue et al., 2017]. Here only the"*Training takes about two weeks for the DNN with 100 dimensions of hidden layers with 58 training epochs*", which is the small DNN tested. The effort for set-up and preliminary data analysis would count additionally for DNNs. Two experts are necessary to integrate either approach: a data scientist and an experienced operator of the ATI. For the running detection each memory footprint is below $2GB$ and each CPU-load was below 10% during the experiments for reference implementations. Portability is only limited here due to the fact that the R framework [R Foundation, 2018] was used. Since it is also available on the embedded node used here (Raspberry Pi Model 3b), it was possible to run reference implementations there. Influence to the process is limited, since the process values can be taken from an interface to passively read network traffic. Training of the algorithms was done using the datasets based on real days (ref. to section 5.2.3) and evaluation based on the synthetic datasets generated from that including attack scenarios (ref. to section 5.2.4).

	AR	**PCBAD**$_M$	**PCBAD**$_D$	**OCSVM**	**DNN**
Accuracy (%)	47.13	99.87	94.37	N/A	N/A
False positives (%)	29.71	0	8.133	N/A	N/A
False negatives (%)	23.16	0.13	3.133	N/A	N/A
Precision	0.4058	0.9974	0.9687	0.9250	0.98295
Recall	0.4670	1	0.9225	0.69901	0.67847
F measure	0.4343	0.9987	0.9451	0.79628	0.80281

Table 5.5.: PVVS classification rates for application datasets and rivals

The classification results for all experiments can be found in table 5.5. Furthermore the evaluation results presented in [Inoue et al., 2017] based on approaches using Deep Neural Networks (DNNs) and One-Class Support Vector Machines (OCSVMs) are also shown. Based on these results the approach using PCBAD with manually identified causalities is chosen for the final concept.

Behavior Validation

Three different approaches to simulate a PLC device have been evaluated as simulator core for this service. Huge effort has been done to develop an approach based on Discrete Event Models (DEMs) (ref. to appendix A.3 and [Horn and Krüger, 2014]). Furthermore virtual Software Programmable Logic Controllers (vSoftPLCs) and respective interfaces have been integrated, a representative of Open Source with Awlsim (ref. to [Büsch, 2018]) and a representative of commercially available standard products, the Siemens SIMATIC WINAC (for further details oh the topic vSoftPLC ref. to [Horn and Krüger, 2016a]), for evaluation. All approaches need access to the source code of the respective PLC. The approach based on DEMs transforms it into a CPN, while the other ones process the code directly. The process to transfer the source from physical PLC

to vSoftPLC is not trivial. Extensive modifications have to be made to the code to get it running in the virtual environment [Kittmann, 2017]. It has not been possible to transfer the highly complex code from the real application scenario, as used in the test-bed, to any of these approaches. Therefore another evaluation platform including a simple hardware setup has been used to proof the concept, where the complexity of source is manageable (ref. to appendix A.2). The code porting has to be done by an average skilled user for the DEM approach and an expert in IEC61131-3 code languages for the vSoftPLCs. The estimated time is approximately $360 hours$ per approach. Likewise other services, development efforts do not count. Its memory footprint is below $15 GB$ and the CPU-load was below 50% during experiments. To evaluate the robustness, simulations were triggered with process values from the simple hardware setup and the results were compared to the respective simulation results. The time window was set to 20 minutes per experiment.

	DEM	**WINAC**	**Awlsim**
Simulation accuracy (%)	93.4	100	100
Simulation time (s)	18,000 - 96,000	1,200	1,200
Real time (s)	1,200	1,200	1,200

Table 5.6.: Simulator core results for behavior validation

According to evaluation results as shown in table 5.6 the approach utilizing Awlsim [Büsch, 2018] is favored for this detection service, since its code is available for modification purposes and no license fees apply.

5.3.2. Live experiments in test-bed

To get an impression of a possible performance in practice, the overall ensemble of chosen approaches was evaluated during live tests within the test-bed. Here attack scenarios 1 to 3 (ref. to sections 4.3.4 and 5.2.4) were performed alongside tests with normal operation within the live running testbed. An additional attack scenario 4 was included to show the relevance of the *Service Validation Service*. The days of normal operation are based on synthetic datasets, since existing real world datasets were used for training of algorithms (ref. to section 5.2.3).

Attack scenarios 1 to 3 were automated using scripts and cron-jobs while the test-bed was running 24 hours per test unsupervised. For better subsequent analysis the attacks started at 12:00.00 hours (43,200 seconds of the day) until 23:59.59 hours (86,400 seconds of the day). Attack scenario 4 lasts the whole day. The following attacks were tested:

Attack scenario 1 refers to a plant-wide disturbance of operation and started after 43,200 seconds of the day. It refers to sections 4.3.4 and 5.2.4 using the DoS attack on the network connecting automation level 2 with level 3 [ISO/IEC62264, 2013], respectively the PLC with the HMI.

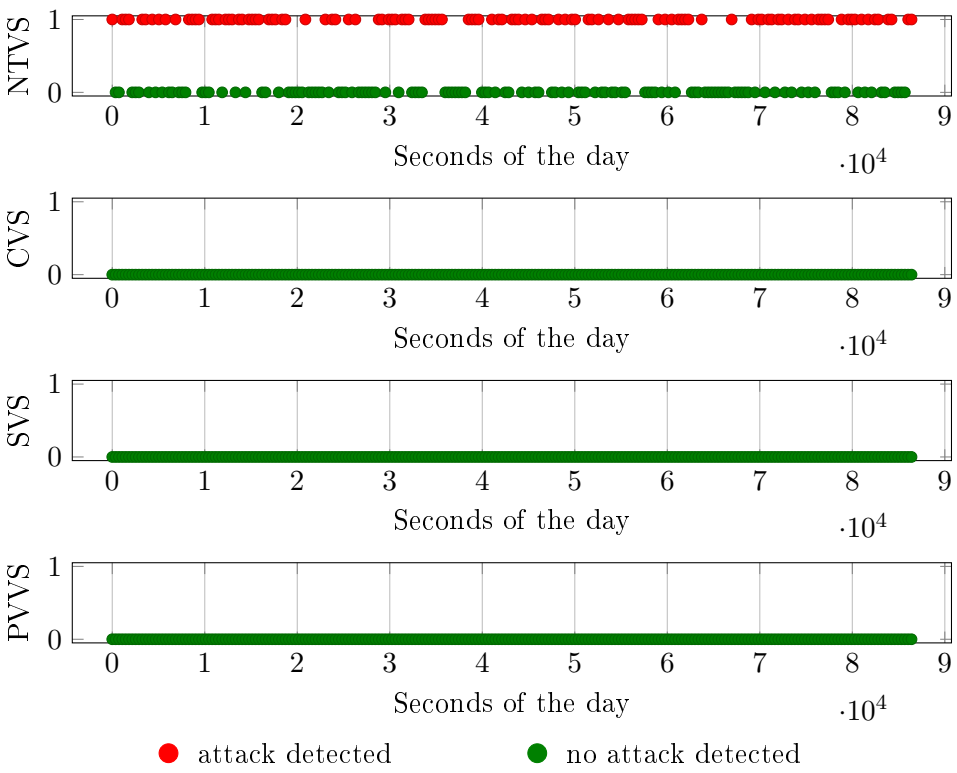

Figure 5.13.: Detection results for a normal operation day

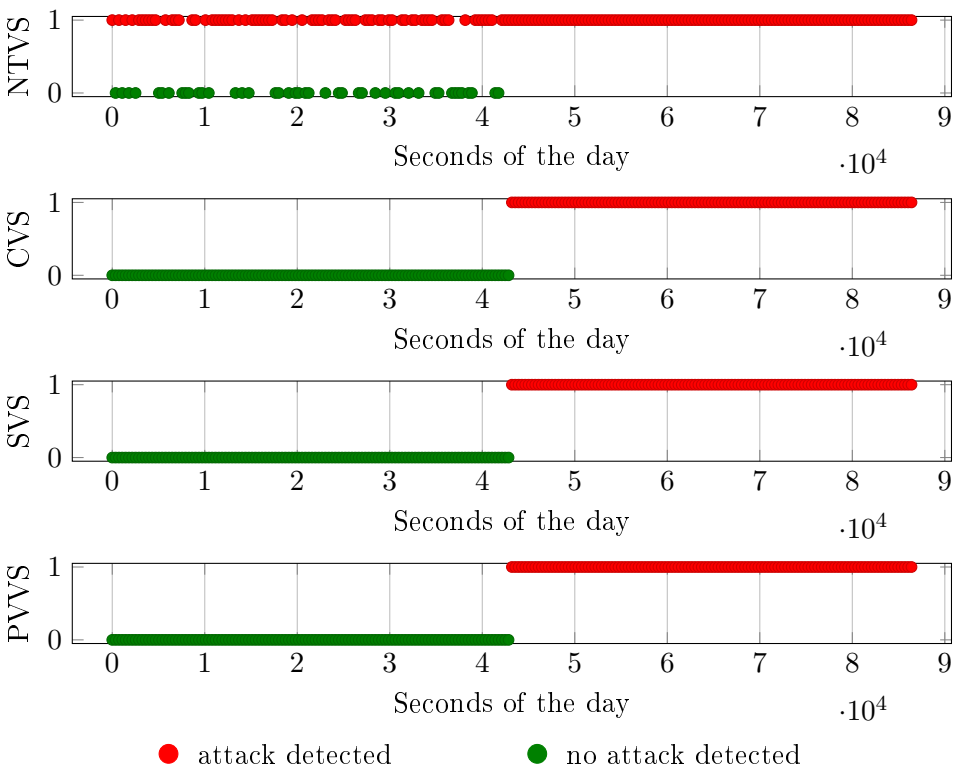

Figure 5.14.: Detection results for attack scenario 1

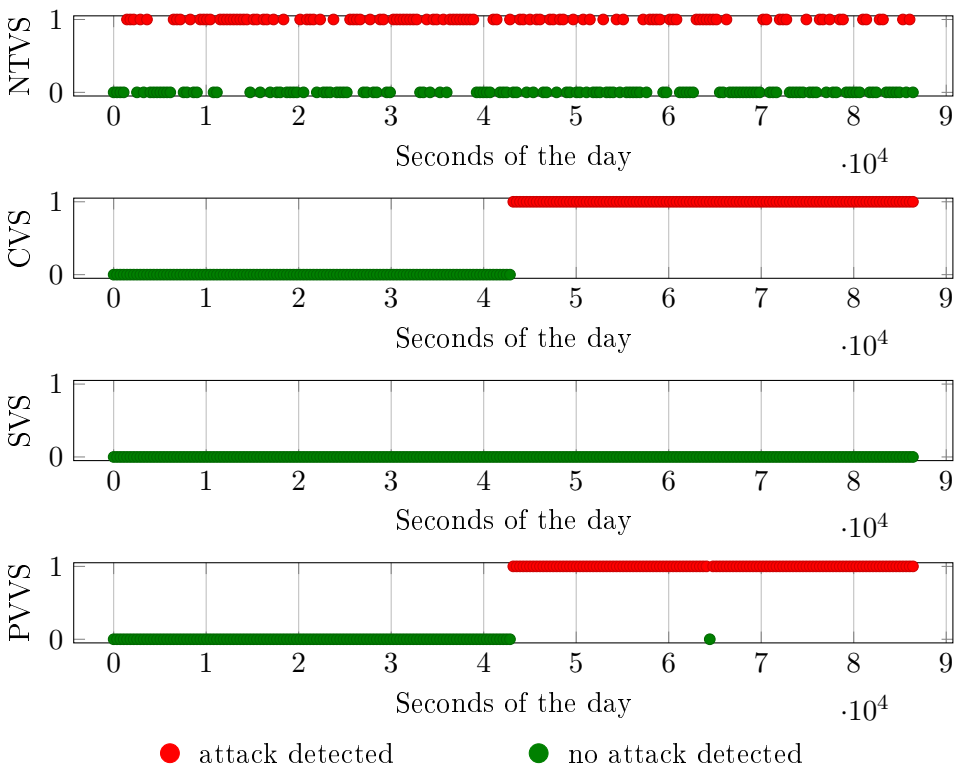

Figure 5.15.: Detection results for attack scenario 2

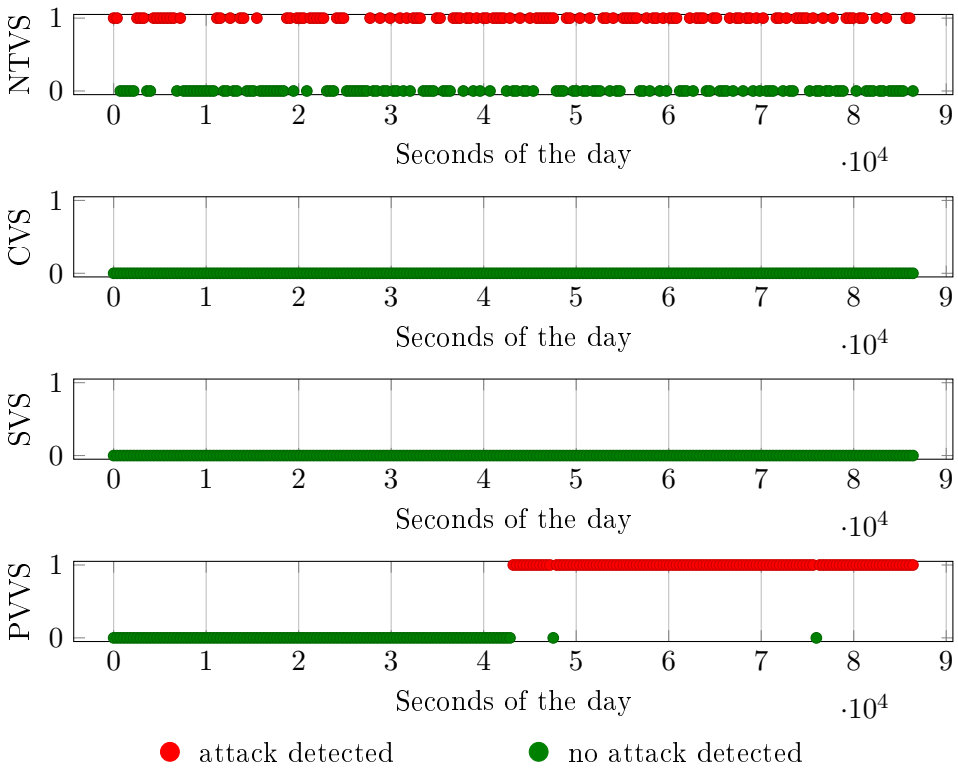

Figure 5.16.: Detection results for attack scenario 3

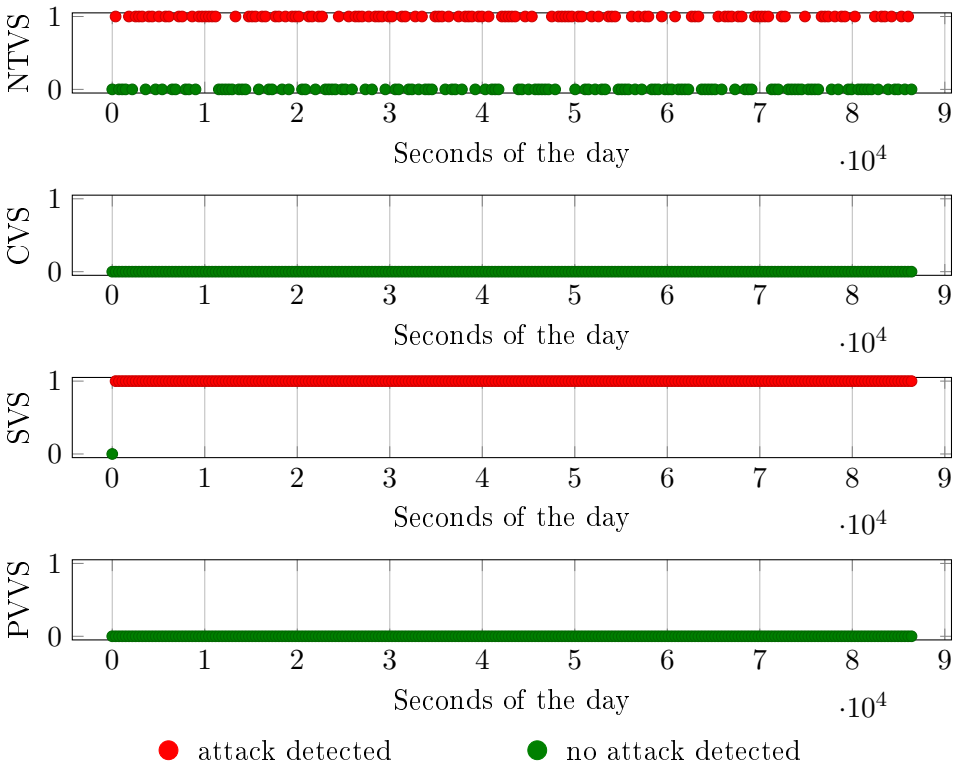

Figure 5.17.: Detection results for attack scenario 4

Attack scenario 2 uses code manipulation in a simulated effort to achieve e.g. fast destruction of pumps. It refers to sections 4.3.4 and 5.2.4. Here a timed script starts after 43,200 seconds of the day to upload malicious code to the PLC. Afterwards the (virtual) operator can see only manipulated values, the real values of the physical simulation are hidden. Furthermore control values get increased to achieve destructive measures virtually.

Attack scenario 3 is an effort to run the tanks dry stealthy. It refers to sections 4.3.4 and 5.2.4 using the scripting interface of the physical simulation to manipulate the tank water level shown to the operator as well as reducing the speed of well pumps. The manipulation also starts at 12.

Attack scenario 4 was included in the live experiments to address another data source with an attack. The idea is that a malicious attacker alters the firmware (not the IEC61131 program) of a PLC during a scheduled maintenance operation spawns an additional service for remote command and control. This scenario is described in [Dunlap et al., 2016] and the focus is on firmware and application program. The implementation was done by turning on the web-server of a PLC device, which is shipped with the product by the vendor, before the test to "spawn" this additional service.

All services of the ensemble reported their resulting decisions to the decision fusion service (ref. to section 5.1.1), where each decision of all detection services were logged, as shown in figures 5.13, 5.14, 5.15, and 5.16. The aggregation and

presentation of results follows [Fisch et al., 2017].

	NTVS	CVS	SVS	PVVS	BVS	Fusion
Normal operation	red	green	green	green	N/A	red
Attack scenario 1	red	red	red	red	N/A	red
Attack scenario 2	red	red	green	red	N/A	red
Attack scenario 3	red	green	green	red	N/A	red
Attack scenario 4	red	green	red	green	N/A	red

Table 5.7.: Aggregated results of complete service set during live evaluation

Fully aggregated results including the resulting decision fusion can be found in table 5.7. The normal operation day shown in figure 5.13 represents hereby the first day of six different available days of normal operation. The other days 2 to 6 are not shown, since they look very similar.

	CVS	SVS	PVVS	Fusion
Normal operation	green	green	green	green
Attack scenario 1	red	red	red	red
Attack scenario 2	red	green	red	red
Attack scenario 3	green	green	red	red
Attack scenario 4	green	red	green	red

Table 5.8.: Aggregated results of reduced service set (CSP) during live evaluation

NTVS and BVS have been removed in table 5.8 according to weak or no results at all during live experiments. The decision fusion based only on CVS, SVS and PVVS (CSP) shows more reliable results. This mirrors also the results of the individual evaluation that is shown in table 5.1. Performance measures calculated over all live-experiments are shown in table 5.9.

	NTVS	CVS	SVS	PVVS	Fusion	Fusion CSP
ACC	0.5360	0.7777	0.7768	0.8865	0.7113	0.9982
FNR	0.1762	0.2223	0.2232	0.1135	0.0009	0.0018
FPR	0.2878	0.0000	0.0000	0.0000	0.2878	0.0000
Precision	0.4834	1.0000	1.0000	1.0000	0.6071	1.0000
Recall	0.6046	0.5010	0.4990	0.7453	0.9979	0.9959
F-measure	0.5373	0.6676	0.6657	0.8541	0.7549	0.9979

Table 5.9.: Performance results of live evaluation

6. Discussion and conclusion

Learning is a good thing, but more often it leads to mistakes.
– Tsunetomo Yamamoto, Hagakure

In 2018, seven years after the ground-shaking discovery of the STUXNET-worm [Falliere et al., 2011] operators of Automation Technology Infrastructures (ATIs) still face difficulties in securing their infrastructures (ref.to appendix A.1). The increasing interconnection and integration of new technologies for the automation world such as virtualization or consumer technologies including tablets and smart-phones promise efficiency, cost and competitive advantages. On the other hand, this also increases potential risks regarding attacks, manipulation, operating errors or technical faults. This is due to the increased complexity of the overall systems, the heterogeneity in the technical systems and protocols themselves, as well as an broadened attack surface with new attack vectors. In addition, the effort and know-how required for a potential attacker is reduced by many new freely available, well documented and easy to use attack tools such as the Metasploit framework. The execution of a successful attack nowadays no longer requires a computer expert, any layman would be able to do it after a short training. The developed methodology and template concept (ref. to section 4.3) within this work can help by enhancing detection rates of attacks within these ATIs.

6.1. Discussion

The detection of anomalous values within distributed, complex and heterogeneous Automation Technology Infrastructures (ATIs) is a challenging goal. Based on research presented in chapter 2 the following postulates have been made in chapter 3:

Postulate 1: An attacker will get access.

Postulate 2: To achieve practical relevant results a detection concept has to be developed with respect to practical requirements.

Postulate 3: A defender of an infrastructure possesses more knowledge about it than an attacker.

Postulate 4: Using a δ from well estimated $\tilde{y}(t)$ and respective live measured values $\hat{y}(t)$ in an ATI, it will notably deviate between manipulated values and non manipulated values.

Based on these assumptions and a system theoretical approach this work is based on (ref. to section 3.4) the focus of this work was especially set on

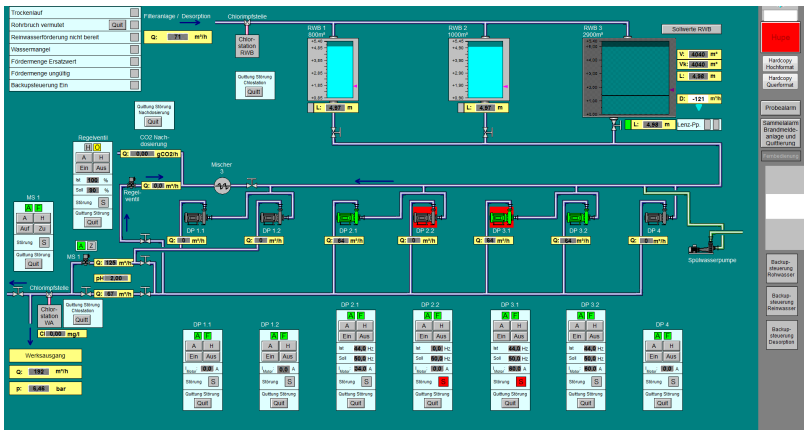

Figure 6.1.: HMI display during DoS attack

development and prototypical usage of a methodology to derive a comprehensive and application specific concept to detect anomalous values within the cyber-layer of complex and distributed automation processes based on real scenarios and requirements. Also the concept was demanded to have the possibility to include different data sources and algorithms, since there is no perfect detection algorithm per se. The combination of different strengths could lead to a more robust detection, which was expressed in the following hypothesis:

Hypothesis 1: A detection concept with heterogeneous elements using different data sources and algorithms can lead to a more robust detection compared to an individual solution.

It contains the variables a) different data sources and algorithms as well as the respective dependent variable: more robust detection, which was defined as better detection performance earlier in this thesis. There are no evaluation metrics that are globally accepted, as discussed in [Axelsson, 2000] [Gilmore and Haydaman, 2016] [Powers, 2007] [Patcha and Park, 2007] and [Urbina et al., 2016]. Therefore especially accuracy, false negatives and false positives based metrics were used.

Following the methodology developed in chapter 4 a generalized concept and its respective segmented, modular, multi-layer, service-oriented detection architecture using different detection services based on different data sources and algorithms was evaluated along a test scenario. The concept itself focuses on detection as part of covering technological security issues. In general a defender of an organization should also keep organizational, personnel, hardware and process relevant risks in mind, since an attacker is able to alternate between these different layers to achieve its objective (i.e using social engineering). Additional concepts to the one developed here have to be applied to achieve a certain level of security for the whole infrastructure.

The performance measures shown in table 5.9 can be used to verify hypothesis 1. **The numbers clearly show that the fusion of elements leads to better results than respectively for each element separately.** The detection

elements were chosen on the basis of developed (evaluation) methodology as summarized in table 5.1. Validity is ensured, since each reference implementation for each detection service contained the best available approach within the application context to detect anomalies at the respective data source. The evaluation with live experiments using different attack scenarios on various data sources provide thoroughness. It is clearly shown in figures 5.13 to 5.17 that each service reacted differently to diverse attack scenarios:

Attack scenario 1 consists of a Denial of Service (DoS) attempt using network flooding. [Niedermaier et al., 2018] showed that PLCs of most vendors are prone to network flooding attacks. This kind of attack is easy to detect, even without an IDS, since the disruption of normal operation appears immediately, as shown in figure 6.1. Detection was done during live experiments by using the time-out implemented at the fusion service (ref. to section 5.1.1).

Attack scenario 2 is difficult to detect by conventional tools, as shown in chapters 1 and 2. The manipulated binary code was easily detected by the CVS and the resulting process value changes were additionally detected by the PVVS. Furthermore a well performing NTVS should be able to detect the upload process itself, which was not the case during live experiments.

Attack scenario 3 reflects the equivalent to a manipulation of the sensor values directly (i.e. manipulated firmware). This can only be detected using a validation of these values and therefore the results come as no surprise: only PVVS detected the attack.

Attack scenario 4 refers to a back-door installation as shown in figure 2.1. This can only be detected by monitoring and analyzing dynamic resource allocation of a device. Therefore the effort of SVS to do that under the given restrictions detects these kind of attacks. Other services were not able to detect this scenario.

The combination of data sources and detection algorithms leads especially to a better detection performance in terms of higher accuracy (ACC), lower false negatives (FN), higher precision, recall and F-measure. False positives (FP) of the fusion reflect hereby the highest (worst) value of all chosen approaches. With the statement[1] of [BLISS et al., 1995] kept in mind the choice of any algorithm itself should not be threated lightly, since it has an impact on the overall performance. Additionally the applied fusion scheme is simple and the possible usage of more complex schemes opens further research questions. Detection services are complimentary and specialized on detecting anomalies in their sources: *Process Value Validation* detects anomalies in process values using causal structures, *Code Validation* detects anomalies in binary application code, *Service Validation* detects anomalies in dynamic service configurations, *Network Traffic Validation* detects anomalies in protocol or communication patterns and *Behavior Validation* detects anomalies in the behavior of a device.

[1] "*..if consistent responding is desired, alarm reliability (lack of false alarms) is of the utmost importance..*"

6.1.1. Network Traffic Validation Service

The individual service evaluation in section 5.3.1 showed already the weak performance of the reference implementation. The results from live experiments shown in figures 5.13 to 5.17 for this service and the respective performance metrics over all experiments shown in table 5.9 are weak too. The detection performance looks like an erratic process. The reason for this is that the approach using DFA [Goldenberg and Wool, 2013] utilized here suffers from different drawbacks. "*The authors rely on the assumption that Modbus traffic is highly periodic and they argue that tests performed on real traffic show a low-rate of false positive. It is worth noting that [..] our tests show that Modbus traffic is periodic only if we are able to filter out random delays*" [Caselli et al., 2015]. The approach for S7 traffic [Kleinmann and Wool, 2014] suffers from the same random delays as stated: " *We call these distinct periods phases. A phase transition is a change in the control system that results in a transition from one phase to another phase. Using the DFA learned in one phase to enforce the model during another phase results in very high anomaly rates*" [Kleinmann and Wool, 2014]. Additionally, active scanning techniques utilized in CVS and SVS may have an impact on the detection performance of this approach. The reference implementation (ref. to section 4.4.2) also does not apply any optimizations of subsequent work by the same authors [Kleinmann and Wool, 2017].

Most of the related work presented in section 2.2.2 utilizes network packets and respective protocol-, communication- or state patterns to detect anomalies. Furthermore there are already some tools commercially available to substitute the aforementioned simple implementation efforts during this work. Examples are [Rhebo GmbH, 2018]: "*Rhebo Industrial Protector continuously analyzes the communication in the ICS and evaluates it for anomalies.*" or [Symantec, 2018] "*Anomaly Detection for Industrial Control Systems solution which detects IoT devices and their network traffic. By identifying IoT devices and analyzing their network traffic, the solution creates a detailed asset map. Using Deep Packet Inspection (DPI) it proactively identifies attacks against ICS networks and flags any suspicious activities.*"

Any of these can replace the approach utilized in the reference implementation, which would not have made it into practice anyway regarding the resulting points from practical requirements during individual evaluation. Further experiments have to be done with different approaches to find the best suitable in practice. The methodology developed during this work can be iteratively applied to find a more suitable approach for this service.

6.1.2. Service Validation Service

Performance rates (ref. to table 5.9) for complementary detection of specific attacks (attack scenario 4) are high and therefore monitoring and analysis of this service are a well chosen addition to detection within the concept. The individual service evaluation revealed the major drawback of this reference implementation: active scanning. It is not optimal and should be avoided in gen-

eral, since it has a direct influence to the process. The lack of access to closed platforms [SIEMENS, 2016] of PLCs and missing security by design made this necessary. An alternative could be the usage of passive scanning while keep using this service as it is. This would be enabled by porting basic techniques of tools like p0f [Zalewski, 2016] to the monitoring part, where the network traffic gets only listened to. The analysis would be additionally more complex and vendor specific protocols have to be implemented to be able to monitor configuration changes and all communications from and to this device. This would be very complex and resource demanding since still not all attacks could be detected, but the active scanning could be avoided.

The task of monitoring dynamic resource allocation in general (CPU load, memory consumption, open ports, etc.) should be preferably done by a HIDS on the device itself, like proposed in [Reeves et al., 2012] and [Garcia et al., 2016]. Future generations of control devices may have this option and better security mechanisms available.

6.1.3. Code Validation Service

This service can detect additional attacks, that might not be detectable by the other service. Individual service evaluation, figure 5.15 and the respective attack scenario 2 show that this service is able to detect code manipulations reliably. Table 5.9 shows the overall performance where notably no false positives arise. The underlying use of hash functions and the simplicity of this approach are the foundation for that. Likewise the SVS, the active scanning part is the major drawback here. This could be avoided by implementing this service as HIDS on the device itself. Like already stated, this would be the static part of a HIDS which monitors binary checksums or hashes. For the existing setup other approaches could be used, but at the cost of lower accuracy. This can be done using passive monitoring of all network connections for vendor protocol specific code changes made to the device [Senyondo et al., 2015] [Malchow et al., 2015].

One very important issue that arose during this research is the choice of code blocks to monitor. A higher amount of code blocks to monitor increases the impact to the process in terms of network bandwidth consumption, CPU load and memory usage on the PLC device. A very complex code project (ref. to section 5.2.2) with a huge number of code and data blocks would make the monitoring very resource demanding, if all blocks were monitored. The choice of monitoring only some code blocks lowers the possibility to detect a manipulation. An integration of the service into the PLC as HIDS would perform better. Nevertheless further research questions arise.

Commercially available change management solutions like [MDT Software, 2018] "*detect changes made outside the change management system. AutoSave compares the latest program copy on file in AutoSave with the program running in each device to identify any differences. If differences are found, the appropriate people are notified with an email highlighting the differences.*" These tools could also be utilized, if the performance is similar to the reference implementation

based on Snap7 library [Nardella, 2015] and hashing tools.

6.1.4. Process Value Validation Service

The validation of process values is a very challenging goal, but it is able to detect the results of several attacks (see figures 5.13 to 5.17) combined with a low amount of false positives. The approach using Process Causalities was developed during this work due to the confrontation with time-series based process data from a complex and distributed ATI, where big amounts of data apply every second. Existing approaches suffer mostly from the *curse of dimensionality* [Bellman, 1969] and the general IID assumption [Nouretdinov et al., 2001]. Furthermore the challenge of concept drift has to be addressed. Using a causal based approach takes these challenges into account, since "*causal analysis goes one step further; its aim is to infer not only beliefs or probabilities under static conditions, but also the dynamics of beliefs under changing conditions, for example, changes induced by treatments or external interventions*" [Pearl, 2009]. This outperforms approaches based on autoregressive-moving-average (ARMA) models [Hadžiosmanović et al., 2014], as shown during individual service evaluation. Further drawbacks of [Hadžiosmanović et al., 2014] are outlined in [Caselli et al., 2015]: "*The approach shows promising results but faces several difficulties in modeling specific classes of variables (e.g. floating points). This is due to the way ICS applications deal with variables. When the size of a process variable exceeds the memory unit used by an application (e.g., 16 bits in Modbus) the value is split and saved within two or more memory locations. The intrusion detection system is unable to follow such mapping and treats every memory location independently from the others. This ultimately causes the usage of erroneous models.* "

The process to generate the causal dependencies among the process values (ref. to appendix A.4) has an impact on the performance of the algorithm, as shown in table 5.5. This comes as no surprise, since it is strongly dependent on the quality of the causal model. Further research questions arise (ref. to section 6.2.2).

6.1.5. Behavior Validation Service

The idea to use a digital copy of a PLC device to verify the application code was utilized in [Senyondo et al., 2015] to check the code for malicious content. Different approaches have been evaluated to utilize the IEC61131 code of a PLC to validate its behavior (ref. to table 5.6). Any of these approaches needs extensive modification of the code itself to run in these virtual environments (vSoftPLC) or a complete parser for all possible commands and vendor specific flavors (transformation to CPN, ref. to appendix A.3). Due to code complexity, lack of access to original programmers and protected code blocks by authoring companies none of the approaches made it into the live evaluation in the end. Furthermore the performance during individual service evaluation drops this approach anyway.

This approach could be adopted by a PLC vendor, since it possesses the required resources and know-how to put it into practice. Additionally, the proposed idea to create an ICS honeypot that processes application code [Leder, 2016] could be adopted. This can be done by utilizing the new developed interface to AWLSIM [Kittmann, 2017] to combine it with conpot [MushMush-Foundation, 2015]. Further research questions arise.

6.2. Conclusion

In order to align new technological possibilities and the increased demands on IT security, a minimum standard has to be created in the company, both in the technical requirements and in the security concepts. Classical security concepts for automation infrastructures, where isolation and "security by obscurity" are the focus, are no longer sufficient. These concepts date back to a time when devices and systems were hardwired directly to their controllers. The controllers themselves were then only networked in local and isolated environments with special and non-open protocols and at specific nodes. At that time, the connection of automation systems and networks to wide area networks such as the Internet was not taken into account.

In general, companies rely on well-known preventive IT-specific security measures such as firewalls, gateways, segmentation into security zones or perimeter defense. Measures to physically secure data communication such as encryption and data tunnels are also used. However, these measures are not sufficient. Further possibilities can be found in reactive security tools such as intrusion detection and crisis management. Other measures like risk analysis of individual systems is also insufficient to uncover all weak points, as a large proportion of existing security gaps arise through the interaction of complex systems. The possible feedbacks that particular security tools can have on an automation infrastructure should also not be underestimated. These can range from unintentionally triggered movements of a robot to complete breakdown of the entire production plant.

These challenges place new demands on protection requirements, especially in production and critical infrastructures. For this reason, in addition to IT-specific security measures, new innovative "system hardening" procedures are required to prevent and, in particular, to cope with IT and cyber attacks, which relate to increasing the robustness, quality and retro-activity of security products and the digital autonomy of automated systems. Only a methodical approach to establish information security tailored to the application can satisfy that.

Some amount of research funding was invested worldwide and lead to different research projects which developed several new approaches. The transferability to practice was usually not the major requirement regarded by scientists and a huge amount of publications exists where this can be questioned at all. An important drawback in developing methods and tools to protect ATIs that can be used in practice within those years was the lack of standards and regulations and more importantly the lack of datasets and simulation environments. There-

fore more reference platforms for evaluating novel approaches for ATI like the test-bed in section 5.2.2 are needed.

This research thesis aimed to answer the general question: *How can a previously unknown attack on a distributed Industrial Information and Automation Technology Infrastructure be detected?* Chapter 2 showed the available countermeasures for securing Information Technology (IT) infrastructures and especially its counterparts in ATIs. Based on analyzing this research the focus was formulated in chapter 3, where a general answer can be given: **Anomaly Detection**. The related work presented in the same chapter showed the challenges for operators to secure their infrastructures in practice. Detailed research questions arose:

- *What would a sound and practice-oriented concept for detecting unknown attacks look like?*
  In chapter 4 such a concept was developed using the methodology elaborated within the same chapter. Following requirements from practice this concept can be utilized there.
- *What methodical steps are necessary to develop such a concept?*
  Figures 4.3 and 4.4 show the answer to that question: system analysis, concept design, implementation and evaluation. Especially the system analysis requires detailed steps like information collection, Technological Map (TechMap) generation, application specific analysis and data capture/ analysis.
- *Which limitations and constraints exist and how do they alter a possible concept?*
  The limits are set by practice and the respective application context. For example the availability of data sources can lead to different application specific manifestations of the template concept shown in figure 4.14. The developed architecture as shown in figure 4.12 also resulted from requirements of practice, since these infrastructures are segmented. A monitoring concept has to be tailored to these requirements.
- *Which parameters influence the quality of detection and which are modifiable?*
  The usage of different data sources in conjunction with different algorithms leads to a better detection performance. By using different data sources the quality of detection can be enhanced (as shown in section 5.2), since some attacks can only be detected by using the respective data source. Furthermore detection algorithms influence the quality strongly. By utilizing a service-based concept these can be interchanged according to the needs of the application context.
- *How can a possible concept be integrated within the heterogeneous infrastructures?*
  The developed template concept was mirrored along the Automation Pyramid structure, which current ATIs are based on. Additionally the impact of real practical requirements to the concept like a small influence to the process, segmentation, modularization, service-orientation and multi-layer

preparedness ensure the applicability.

- *How to scale the possible concept depending on the need for each application?*
  The aforementioned arguments that guarantee an applicability to heterogeneous infrastructures also ensure the scalability. The deployment concept shown in figure 5.1 of the reference implementation shows that different services can be run on various computing entities. The architecture itself (see figure 4.12) and data-flow (see figure 5.2) enable the application for a broad variety of infrastructures. Bigger infrastructures with more segments require more embedded nodes and local computing nodes accordingly. The limits are set by existing network and computing technologies.

The developed concept is able to detect several different attack types which are unknown to the detection system, as shown in chapter 5 and is intended to work within for a broad variety of application contexts. The applicability is ensured by the combination of a template concept which can be flexibly tailored using the presented methodology. Future research that puts forth novel approaches, algorithms or even data sources can be included in an uncomplicated manner.

6.2.1. Contributions of this thesis

This work analyzed current state of basic principles, threats, attack models and countermeasures for Automation Technology Infrastructures. The **analysis** revealed that especially countermeasures regarding a detection of anomalies are necessary and related works were not developed having requirements from practical application contexts in mind. Therefore the need by operating companies for a **methodology** to develop a detection concept was addressed. This methodology was utilized on basis of requirements and data of eleven real use cases from four Critical Infrastructures (CIs), six manufacturing companies and one academic example. The resulting generalized **concept** can be used as template in conjunction with the methodology to generate application specific concepts much faster for new use-cases. A prototypical implementation was evaluated using requirements, attack scenarios taken from practice. The **evaluation** took place in a very realistic test-bed using real data captures from the respective application scenario. It was shown that a **decision fusion** of different **detection services and data sources** has a better detection performance compared to individual solutions.

Additionally, during experiments a new attack vector was found: **the fastest wins** (ref. to sec 5.2.4. This attack needs well calculated timing in conjunction with access to the subnet of the PLC. If aware of that, this scenario can not only be detected by the **PCBAD algorithm**, an easy way to detect it is monitoring of network parameters like typical bandwidth usage.

Furthermore the analysis, implementation and evaluation lead to several new findings, which lead to **new research questions**.

6.2.2. Future work

These results mirror a snapshot of current conditions. The arms race of attackers and defenders [Schneier, 2014] leads to changing requirements over time. Even if the presented methodology and the template concept were developed to be flexible and applicable for a broad variety of application contexts it cannot be seen as valid for all times. Similar to a security process new requirements and changing conditions have to be adapted. A concept has to be changed accordingly. The whole methodology of this work should be implemented into a process that can be repeated to react to changes iteratively. To apply the developed security concept of the test-scenario in practice an iterative approach for the present methodology could be: a) conduct another iteration of the implementation and evaluation for a better algorithmic basis, b) generate a prototype by implementing the best available algorithm for each service and c) apply it in a customers real-world scenario.

An example for the arms race is the so called *malicious learning*. [Ferrara et al., 2014] demonstrated how machines with trained algorithms can be outwitted. They morphed two faces into one picture and both people gained access with these fabricated passports at automated border control machines used in airports. This approach shows that especially algorithms using machine learning components can be fooled. It is just a matter of time until an attacker discovers a loophole.

During this work further research questions arose though observations. These can form the basis of future work. The following questions resulted:

- *Do the attacks as developed with operators reflect real attacks?*
  Attack scenarios this thesis is based on were developed with experts from operating companies. These scenarios demand highly skilled attackers with profound background knowledge. Is it like that in practice? To answer that question a well made *honey-process* emulating an entire infrastructure has to be set up to generate basic research data of real attacks. The works done for the BVVS could be a starting point.
- *Which algorithm performs best according to a specific data source?*
  A vast number of algorithms are available for pattern recognition purposes. Related work of this thesis only adopted a small amount of those compared to conventional Intrusion Detection Systems. Basic research towards this topic is enabled using the present methodology and test-bed. The porting and implementation of further algorithms for evaluation can lead to profound statements toward comparability and performance.
- *Which fusion scheme performs best?*
  Fusion can be done on data, feature and decision level. The usage of different algorithms indicates decision fusion, but the other schemes were not tested. Furthermore different fusion algorithms can be evaluated too.
- *Which further data sources are available?*
  Other application contests may contain additional data sources, that were not part of the eleven analyzed ones. Furthermore existing data sources

could be possibly split into further sub-categories. An example is network traffic data source. It could be split into packets (header, data), protocol, stats (bandwidth usage, etc.), communication patterns (which node communicates with another in which frequency,..).

- *Which services could be integrated into a PLC device?*
  CVS and SVS are parts of conventional HIDS. Due to lack of access to the closed platforms and additional performance and stability reasons these were developed to monitor PLC devices from distance. A vendor could integrate these services into the core functionality of these devices.
- *How can the auto-generation of causal models be enhanced?*
  The chosen approach for the PCBAD algorithm uses manual identification of causalities. This is cumbersome and requires experts knowledge. The automatic identification from process data shows promising, but still enhanceable results. Further research could lead to better models from auto-generation (ref. to appendix A.4).
- *How would a honey-process look like and perform?*
  The already mentioned approach to create a fake ATI that traps intruders for analysis purposes seem worth further researching. Resource efficiency, scalability and portability seem to be requirements that have to be confirmed.

A. Appendix

A.1. Expert interviews

During three research projects STEUERUNG [Horn et al., 2014], pICASSO [Vick et al., 2015] and RetroNet [Horn and Krüger, 2016c] several empirical research data were collected. Workshops, interviews, questionnaires and experimental games contributed to this research as described in section 4.2. Respective primary data[1] are available in terms of protocols, audio recordings, transcriptions, mind maps and further documents.

A.1.1. STEUERUNG

This research project ran from 2013-07-01 to 2015-06-30 and information collection took place at several locations in Berlin, Germany. Research challenges, methodology, requirements, attack scenarios, TechMaps, development of novel algorithms, a first evaluation of existing ones and the testbed resulting from the project contributed to this research.

Information collection followed these methodological steps to create Technological Maps (TechMaps) of hardware and software (ref. to section 4.2.2): (1) survey of the used hardware and software systems of the respective critical infrastructure, (2) selection of a partial aspect of the infrastructure, (3) data collection and (4) determination of relevant process parameters of the respective critical infrastructure. Especially the first and second step included several tools like technology audits, workshops, on-site inspections, interviews with experts from different Critical Infrastructures (CIs) and an analysis of source code and plant documentation.

In order to take the complexity of various application partners into account, four different working groups were formed: (a) water 1, (b) water 2, (c) gas and (d) power. Within these working groups different steps were undertaken with respect to the application specific characteristics. Table A.1 provides an overview.

Most relevant requirements include a very low influence to the process ("*The process and control devices must not be altered or touched.*" [Security engineer, Work group Water 2], [similar statements were made in all groups]), a very low false alarm rate ("*If the system produces any false alarm the applicability in practice will be obsolete within days, since the operator would lose trust and avoid to use it at all.*" [Operations manager, Work group Water 2], [similar statements

[1] Each protected under the respective NDA with these companies.

	Work group Water 1	Work group Water 2	Work group Gas	Work group Power
Methodological strategy	✓	✓	✓	✓
Workshops and interviews	✓	✓	✓	✓
Requirements engineering	✓	✓	✓	✓
On-site inspection	✓	✓	✓	✓
Analysis of documentation	✓	✓	✓	✓
Technology audits	✓	✓	✓	
Analysis of source code		✓		
Analysis of data models	✓	✓	✓	
Development of TechMap	✓	✓	✓	✓
Definition of critical parameters	✓	✓	✓	✓
Participatory observation	✓	✓	✓	✓
Elaboration of attack scenarios	✓	✓	✓	✓
Interfaces and live data capture	✓	✓	✓	
Simulation and test-bed	✓	✓	✓	
Experimental games		✓	✓	

Table A.1.: Information collection tools used in research project STEUERUNG

were made in all groups]), reliable detection of Advanced Persistent Threats (APTs) ("*This is what we consider the most serious threat at the moment which scares us most.*" [Operator, Work group Gas], [similar statements were made in all groups]) and user friendly reporting ("*Operators alertness relies on user friendly reporting taking the complex information situation within operations control into account. Any reporting has to work even at night or during tedious work shifts.*" [Department manager, Work group Power], [similar statements were made in all groups]). In general the protection of automation systems was considered a complex and difficult task, nevertheless due to missing tools and technologies. Especially the lack of methods and tools for a validation of process values was considered one of the major research challenges. Furthermore various application specific attack scenarios could be elaborated.

Anforderungen an die technischen Lösungen

Für wie wichtig halten Sie jeweils die nachfolgenden Anforderungen an eine technische Lösung? Vergeben Sie eine Wichtung in %, so dass die Summe aller Wichtungen der Anforderungen 100% ist. (0% bedeutet nicht notwendig und 100% bedeutet dann, dass nur diese eine Anforderung wichtig wäre. Vergeben Sie also 90% bei einer Anforderung, bleiben noch 10% zur Aufteilung auf alle anderen. Alle gleich wichtig bedeutet überall 10% eintragen (bei 10 Anforderungen), was aber niemandem weiterhilft) **Falls Ihnen noch Anforderungen fehlen, tragen Sie die bitte im letzten Freitextfeld dazu. (mit Wichtung)**	**Gesamt 100%**
Kosten (Anschaffungskosten, Lizenzgebühren etc.)	
Skalierbarkeit (bspw. bei Erweiterung des Prozesses, wie skaliert die Lösung)	
Ressourceneffizienz (läuft auf Raspberry Pi oder nur auf Server)	
Betriebsaufwand (Aufwand beim „Einbau" der Lösung in meinen Prozess und jeden Tag während des Betriebes)	
Nutzerfreundlichkeit (kann das jeder bedienen oder nur Experten)	
Portierbarkeit (auf versch. Betriebssysteme etc.)	
Einfluss auf den Prozess (Netzwerk, Systemstabilität etc.)	
Wartbarkeit/ Erweiterbarkeit (Updates etc.)	
Konfigurierbarkeit/ Flexibilität (wie gut lässt sich Lösung an eigenen Prozess anpassen bspw. auf unterschiedliche Systemebenen)	
Robustheit (der Lösung selbst gegen wechselnde Anwendungsbedingungen)	
Weitere Anforderungen aus Ihrer Sicht:	

Figure A.1.: Questionnaire to weight requirements (in German)

A.1.2. pICASSO

Meetings and workshops during this research project took place from 2013-09-01 to 2016-12-31 at various locations distributed over Germany. Three application contexts from industrial production including respective attack scenarios contributed to this research. Further requirements could be added to the ones elaborated for CIs in STEUERUNG by small, medium and large industrial pro-

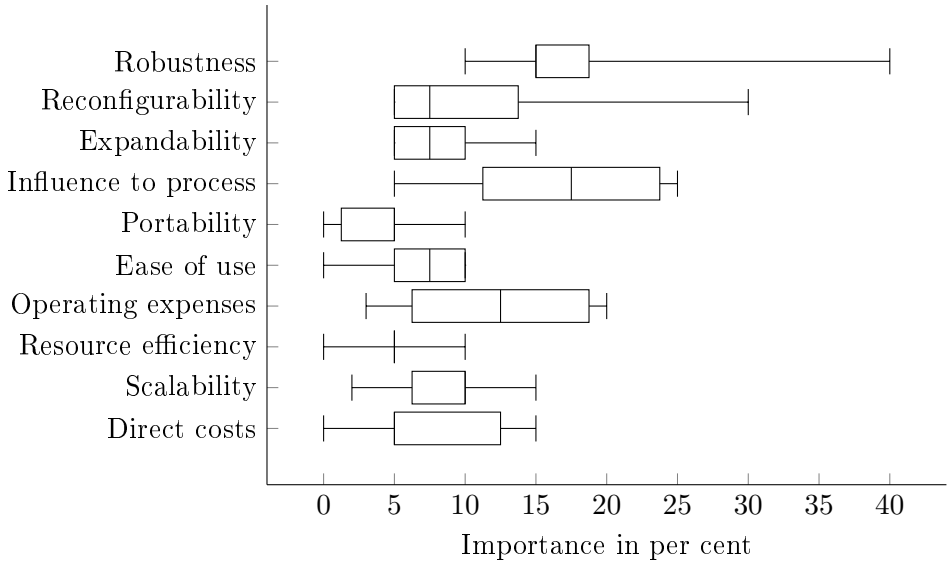

Figure A.2.: Box plot of all questionnaire answers

duction use cases. The factor costs plays a more significant role in terms of purchase price, operating expenses and training of factory staff. Furthermore future demands of customers and possible changing conditions result in further requirements: resource efficiency, scalability, portability, reconfigurability and expandability.

A.1.3. RetroNet

This research project was conducted from 2015-12-01 to 2018-11-30, meetings and workshops took place at various locations distributed over Germany. One focus was laid on developing a methodology to secure practice oriented approaches to connect legacy machinery lacking recent communication interfaces to distant computing pools to enable data processing using machine learning and pattern recognition. This methodology, weighted requirements from three industrial production use cases plus one academic example and further attack scenarios contributed to this research.

A description of the methodology can be found in chapter 4 and the developed methodological matrix is shown in table 4.1. Requirements from previous research were confirmed and weighted during this project. A part of the questionnaire related to weighting can be found in figure A.1 and all answers from several experts of six companies can be found in figure A.2.

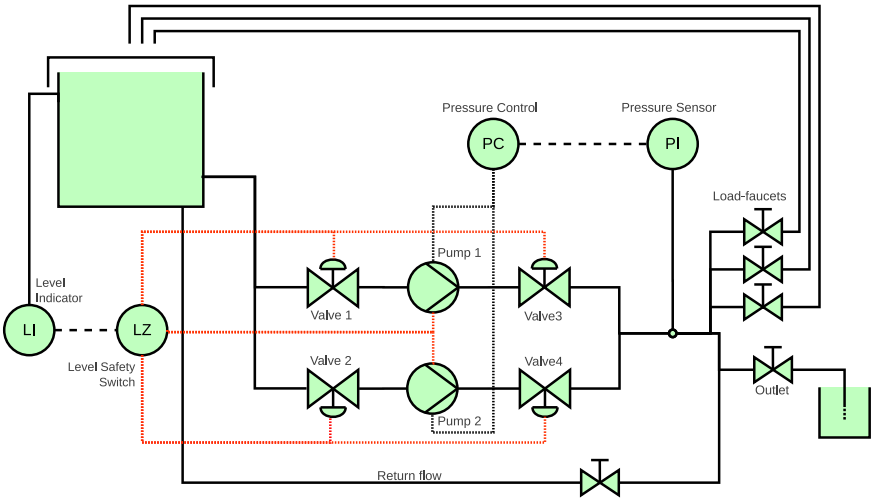

Figure A.3.: Piping and instrumentation alike diagram for miniature water-flow show-case [Kittmann, 2017]

A.2. Miniature water-flow show-case

The miniature water works plant model was developed within the framework of a teaching project[2] using fish tank components. It consists of a water reservoir (tank) with upstream and downstream delivery lines. These contain pumps and assigned solenoid valves (normally closed) to protect from dry-running. Downstream also contains a pressure sensor that supplies data to the controller (controlled variable). Figure A.3 shows the layout plan (ref. to [Kittmann, 2017]). It indicates the arrangement of water-bearing components and a protection of pumps from destruction by dry-running, hot-running or cavitation by two valves each. Position of the pressure sensor is behind merging of the two delivery lines and in front of the load faucets. Load faucets emulate water consumption of an urban area as a distortion to the controlled system and their number was set to three. All load valves plus an additional emergency tap allow the water to flow into a drain pan. In addition, a third pipe was integrated, which allows

[2] Authors: Mr. Tom Kittmann, Mr. Linus Lee and Mr. Clemens Briese; Automatisierungstechnisches Projekt (0536 L 110) , Sommersemester 2015, Supervisor: Christian Horn; IWF, Fachgebiet Industrielle Automatisierungstechnik, Head: Prof. Dr.-Ing. Jörg Krüger; TU Berlin

draining the water tank without running pumps. This line also enables testing scenarios, where the water circles continuously within the system. An additional integrated manual faucet prevents uncontrolled outlet during regular operation.

The set-up is hardwired to a PLC. Due to specification of analog outputs at the PLC, which can be operated at ±10 V with 20 mA, the pumps are supplied by an additional power supply unit. In order to implement control by the PLC, one hardware capacity controller per pump was included. Fuses additionally protect the analog outputs of the PLC in case of defects at power supply unit and/or power regulator. The valves can be operated directly via the digital outputs of the PLC. The pressure sensor is also directly supplied with power from the PLC and submits an analog data signal to the PLC. The water level safety device is activated by two electrodes integrated in the water tank which are supplied by the PLC. The according continuous current measurement emits a notable step signal as soon as the electrodes are no longer in contact with water.

A.3. Colored Petri Net generator

This work was published in [Horn and Krüger, 2014]. The transformation of PLC source code to a Colored Petri Net (CPN) (refer to [Hanisch et al., 1997] and [Heiner and Menzel, 1998]) forms the basis of this development. A tool was developed to convert PLC source code in the form of IEC 61131-3 standard Instruction List (IC) to Pseudo Code, which is parsed and transformed into a CPN (see figure A.4). The execution of the resulting CPN is based on the simulator of freely available CPN-Tools [Westergaard and Verbeek, 2018]. A novel interface to this simulator enables a simulation of the created CPNs using real parameters from live running PLCs.

[Jensen, 1981] and the extension in [Jensen, 1991] define a CPN as tuple $CPN = (\Sigma, P, T, A, N, C, G, E, IN)$ satisfying the following requirements:

(i) Σ is a finite set of types, called color set. Each color set must be finite and non-empty.

(ii) P is a finite set of places.

(iii) T is a finite set of transitions.

(iv) A is a finite set of arcs, such that $P \cap T = P \cap A = T \cap A = \emptyset$.

(v) N is a node function. It is defined from A into $P \times T \cup T \times P$.

(vi) C is a color function. It is defined from P into Σ.

(vii) G is a guard function. It is defined from T into expressions such that $\forall t \in T : [Type(G(t)) = Bool \wedge Type(Var(G(t))) \subseteq \Sigma]$.

(viii) E is an arc expression function. It is defined from A into expressions such that $\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$ where $P(a)$ is the place of $N(a)$.

(ix) IN is an initialization function. It is defined from P into expressions such that $\forall p \in P : [Type(IN(p)) = C(p))_{MS} \wedge Type(Var(IN(p))) \subseteq \emptyset]$.

Instruction	Short explanation
+I,-I,*I,/I,+D,-D,*D,/D	Arithmetic operations for (16Bit)
+R,-R,*R,/R	Integer, Double and Real operands
<=I,>=I,<I,>I,<>I,==I	Integer comparison commands
<=R,>=R,<R,>R,<>R,==R	Real comparison commands
BEA	Block end unconditional: continue in next/calling block
BLD	Program display instruction, ref. NOP
CALL	Execute a given function
CLR	Set the VKE bit to 0
DTR	Type conversion from Double to Real
ITD	Type conversion from Int to Double
L	Load a given value/variable
NOP	Null instruction
R	Reset given variable if the previous condition (VKE) is fulfilled
RND	Round (float) variable/value
S	Set the given variable if the previous condition (VKE) is fulfilled
SAVE	Save VKE bit to BIE Register
SE	Set an on-delay timer
SET	Set the VKE bit to 1
SPA, SPB	Unconditional jump/conditional jump
SPBN,SPBB,SPBNB	Conditional jump with side effects
T,=	Stores the current value to a given variable
TAK	Toggle accumulator 1 and 2
TAR1	Load address from register 1 to Accumulator 1
U,UN,O,ON,X	Logic operations: AND, NAND, OR, NOR, XOR
U(,O(	Bracket instructions
UD,OD	Bitwise and/or operation
FUNCTION, END_FUNCTION	Begin of a new function/end of correlate function
ORGANIZATION_BLOCK X	Begin of a new blog: Calls functions and basic instructions
END_ORGANIZATION_BLOCK	End of blog
NETWORK	Begin of a new network

Table A.2.: Currently recognized commands for automatic generation of CPNs

The transformation from IC to CPN is done in the sequence as the PLC would do. Vendor specific flavors have been taken into account too. Currently recognized instructions are shown in table A.2. The translation starts by parsing instructions from the beginning of the main routine. For every variable in the

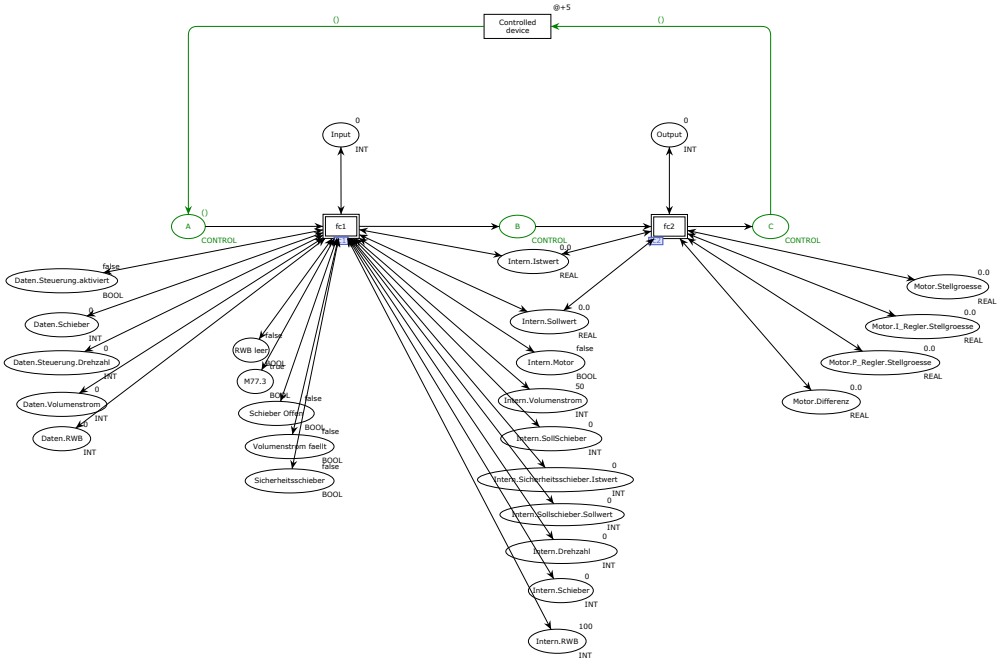

Figure A.4.: Example of resulting Colored Petri Network

PLC program, a node with the same data type and name is created in the net. The current value of the variable translates to its token value. During runtime a block of conjugate instructions mostly reads, modifies and writes values or checks a condition for jumping to other parts of the program. This behavior is modeled by creating a transition and combining it with the variable nodes that are addressed from these instructions. In addition there is a token that is passed from one transition (over a place) to another, to model the sequential behavior and the flow through the program. If the main routine calls another function or part of the program, the current transition and continue is saved from that one later when the end of that block is reached. A node with the name of the new function is labeled and another sub-net for the instructions belonging to that part is created. This allows jumping to these parts of the program, e.g. if the function is called again later in the program. At the end of the program a cycle in the model can be created to increase the simulation time counter, because the main function of a PLC is called periodically. This is finally modeled by inserting a transition from the end state back to the initial state.

A.4. Process Causality based Anomaly Detection

This work has been published in [Horn and Klein, 2017] and [Klein, 2017]. The process to generate a causal model containing transfer functions is shown in figure A.5. First causalities have to be identified among process values. From that a Directed Acyclic Graph (DAG) is created as basic model (ref. to section 3.4.1) and a transfer function is associated to each transition (ref. to figure

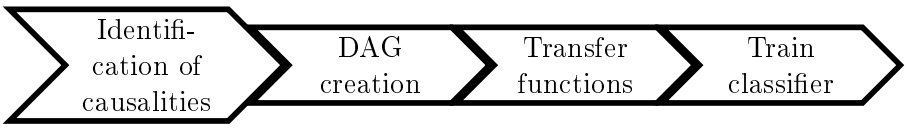

Figure A.5.: Methodological steps for Process Causality based Anomaly Detection

3.4).The resulting prediction model is used in conjunction with formula 3.3 and Postulate 4 to train a basic classifier. The identification of causalities is the crucial part, since the quality of detection relies on that (see table 5.5).

Three different approaches for identifying causalities among process values have been considered for evaluation:

Manual development using experts knowledge ($PCBAD_M$) uses a combination of a full system analysis (ref. to section 4.2) in combination with expert interviews to find causalities in an ATI. From this very detailed knowledge causal models can be made for specific nodes (e.g. waterworks plant), as well as a more abstract model for the overall infrastructure, where each specific node can represent a detailed model itself. An evaluation with regard to the significance of individual parameters related to the overall process was the foundation for identifying the structural points where real data from the live running processes is the part of the process. While this methodology used to generate a causal model is very time consuming, resulting models can be seen as canonical. This manually created models can also be used for benchmarking the automatic generated models.

Automatically derive causalities from process-data ($PCBAD_D$) using available algorithms. Several algorithms to detect causal relations in process data were evaluated in [Klein, 2017]. A wide scope of different applications that make use of causal inference lead to this numerous different algorithms that have been developed and improved over time. Especially the work of [Spirtes et al., 2000] needs to be mentioned, it introduced the SGS- and later the PC-Algorithm . These laid the foundation for many other algorithms and is often used as a benchmark. Examples for other works based on that are [Friedman et al., 1999], [Cheng et al., 2002] and [Tsamardinos et al., 2006]. Furthermore there are several tools available that implement a variety of different methods, for example [Scheines et al., 1998], [Kalisch et al., 2000] and [Statnikov et al., 2009]. A comprehensive overview of all algorithms can be found in [Pearl, 2009] and [Spirtes et al., 2000]. The PC-Algorithm [Spirtes and Glymour, 1991] [Kalisch et al., 2012] showed best results in the setup presented in [Klein, 2017]. Therefore it was further used.

Automatically derive them from PLC source-code which represents the behavior of the system. The idea was to use existing knowledge already formulated in machine readable form by an engineer. The parser presented in appendix A.3 was to be used with light modifications, where only the vari-

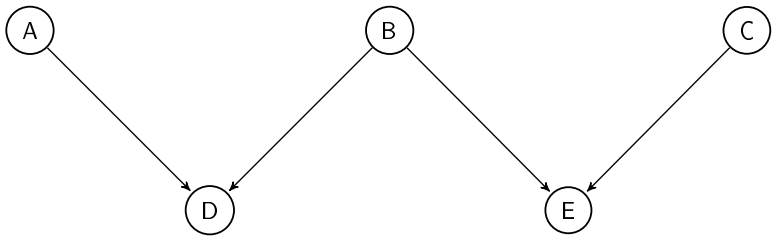

Figure A.6.: Example of causal dependencies among variables A,B.C,D,E

ables and their dependencies get modeled. Since the approach failed to deliver results, this approach was not available for evaluation.

Results of these approaches can be visualized in a DAG. An example of a DAG is given in figure A.6. The shown graph consists of five nodes. Each node represents one process value inside an ATI. An interpretation of this example DAG is as follows

- A has no given dependencies, A influences D
- B has no given dependencies, B influences D and E
- C has no given dependencies, C influences E
- D depends on the values A and B
- E depends on the values B and C

The causal dependencies do not reveal any information on how the variables correlate to each other. A further step is required that includes adding a transfer function to each effect variable (D, E in figure A.6), as sketched in figure 3.4. The transfer function can be non-linear and its output should be a resulting value (regression). [Russell and Norvig, 2012] states:"*the function represented by the network can be highly nonlinear [.., therefore..] neuronal networks can be a tool for nonlinear regression.*" In order to handle possible nonlinearities within these transfer-functions and avoid the a-priori assumption of a model dimension, simple neural networks can be utilized.

$$y(x, w) = f\left(\sum_{j=1}^{M} w_j \phi_j(x)\right) \tag{A.1}$$

[Bishop, 2006] states: "*Neural networks use basis functions that follow the same form as [formula A.1], so that each basis function is itself a nonlinear function of a linear combination of the inputs, where the coefficients in the linear combination are adaptive parameters. [The] goal is to extend this model by making the basis functions $\phi_j(x)$ depend on parameters and then to allow these parameters to be adjusted, along with the coefficients w_j, during training.*" The resulting

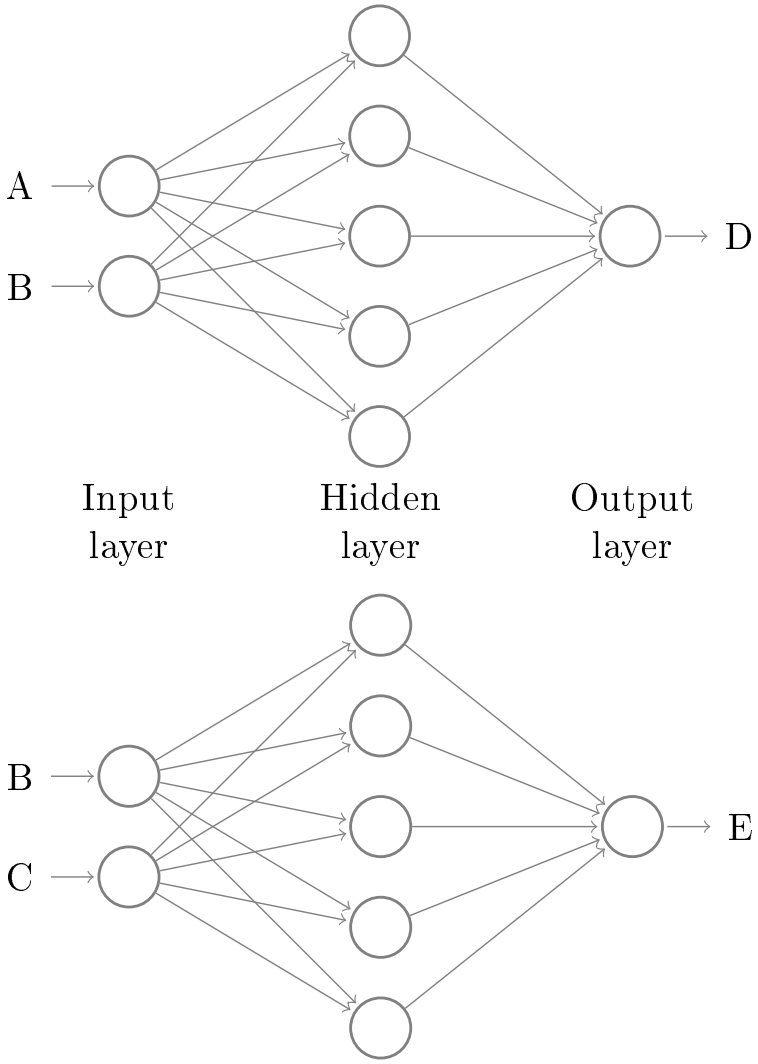

Figure A.7.: Neural networks derived from found causal dependencies

overall network function takes the form

$$y_k(x,w) = \sigma\Bigg(\underbrace{\sum_{j=1}^{M} w_{kj}^{(2)}\, \underbrace{h\Bigg(\overbrace{\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}}^{a_j}\Bigg)}_{z_i} + w_{k0}^{(2)}}_{a_k}\Bigg) \tag{A.2}$$

where h contains the activation function, z_j refer to hidden units, a_j refer to input activation with weights $w_{ji}^{(1)}$ and biases $w_{j0}^{(1)}$, a_k refer to output activation with weights $w_{kj}^{(2)}$ and biases $w_{k0}^{(2)}$. The given example DAG shown in figure A.6 leads to two neural networks, which are presented in figure A.7. To train the neural networks the same dataset that was used to detect the causal dependen-

cies can be reused. The resulting prediction model is used in conjunction with formula 3.3 and Postulate 4 to train a basic classifier as shown in [Horn and Klein, 2017].

Bibliography

[Adamczyk et al., 2015] Adamczyk, H., Bettenhausen, K. D., Daum, W., Dirzus, D., Figalist, H., Heim, M., Jumar, U., Leonhardt, S., Roos, E., Urbas, L., and Winterhalter, C. (2015). Automation 2025 - Thesen und Handlungsfelder. Technical report, VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik.

[Ahmad et al., 2014] Ahmad, A., Maynard, S. B., and Park, S. (2014). Information security strategies: towards an organizational multi-strategy perspective. *Journal of Intelligent Manufacturing*, 25(2):357–370.

[Ahmad et al., 2013] Ahmad, M. O., Markkula, J., and Oivo, M. (2013). Kanban in software development: A systematic literature review. In *2013 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 9–16. IEEE.

[Alcaraz et al., 2014] Alcaraz, C., Cazorla, L., and Fernandez, G. (2014). Context-awareness using anomaly-based detectors for smart grid domains. In *International Conference on Risks and Security of Internet and Systems*, pages 17–34. Springer.

[Alserhani et al., 2010] Alserhani, F., Akhlaq, M., Awan, I. U., Cullen, A. J., and Mirchandani, P. (2010). Mars: multi-stage attack recognition system. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 753–759. IEEE.

[Alves et al., 2017] Alves, T., Morris, T., and Yoo, S.-M. (2017). Securing scada applications using openplc with end-to-end encryption. In *Proceedings of the 3rd Annual Industrial Control System Security Workshop*, ICSS 2017, pages 1–6, New York, NY, USA. ACM.

[Amin et al., 2013a] Amin, S., Litrico, X., Sastry, S., and Bayen, A. M. (2013a). Cyber security of water scada systems—part i: Analysis and experimentation of stealthy deception attacks. *IEEE Transactions on Control Systems Technology*, 21(5):1963–1970.

[Amin et al., 2013b] Amin, S., Litrico, X., Sastry, S., and Bayen, A. M. (2013b). Cyber security of water scada systems—part ii: Attack detection using enhanced hydrodynamic models. *IEEE Transactions on Control Systems Technology*, 22(5):1679–1693.

[Anderson, 2008] Anderson, D. J. (2008). Kanban creating a kaizen culture and evolving lean software engineering solutions. In *QCon London 2008*.

[Anderson, 1980] Anderson, J. P. (1980). Computer security threat monitoring and surveillance. Technical report.

[Andreeva et al., 2016a] Andreeva, O., Gordeychik, S., Gritsai, G., Kochetova, O., Potseluevskaya, E., Sidorov, S. I., and Timorin, A. A. (2016a). Industrial control systems and their online availability. Technical report, Kaspersky Labs.

[Andreeva et al., 2016b] Andreeva, O., Gordeychik, S., Gritsai, G., Kochetova, O., Potseluevskaya, E., Sidorov, S. I., and Timorin, A. A. (2016b). Industrial control systems vulnerabilities statistics. Technical report, Kaspersky Labs.

[Arkin et al., 2005] Arkin, B., Stender, S., and McGraw, G. (2005). Software penetration testing. *IEEE Security & Privacy*, 3(1):84–87.

[Auchard and Finkle, 2016] Auchard, E. and Finkle, J. (2016). Ukraine utility cyber attack wider than reported. `https://www.reuters.com/article/us-ukraine-crisis-malware/ukraine-utility-cyber-attack-wider-than-reported-experts-idUSKBN0UI23S20160104`. Link visited last in 2018.

[Axelsson, 2000] Axelsson, S. (2000). The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):186–205.

[Axelsson et al., 2000] Axelsson, S., of Computer, D., and Engineering (2000). Intrusion detection systems: A survey and taxonomy. Technical report, Chalmers University of Technology.

[Bai and Gupta, 2014] Bai, C.-Z. and Gupta, V. (2014). On kalman filtering in the presence of a compromised sensor: Fundamental performance bounds. In *2014 American Control Conference (ACC) June 4-6, 2014. Portland, Oregon, USA*.

[Bai et al., 2015] Bai, C.-Z., Pasqualetti, F., and Gupta, V. (2015). Security in stochastic control systems: Fundamental limitations and performance bounds. In *American Control Conference (ACC)*, pages 195–200. IEEE.

[Barbosa, 2014] Barbosa, R. R. R. (2014). *Anomaly detection in SCADA systems: a network based approach*. PhD thesis.

[Barbosa et al., 2016] Barbosa, R. R. R., Sadre, R., and Pras, A. (2016). Exploiting traffic periodicity in industrial control networks. *International Journal of Critical Infrastructure Protection*, 13:56–62.

[Bass, 2000] Bass, T. (2000). Intrusion detection systems and multisensor data fusion. *Communications of the ACM*, 43(4):99–105.

[Bass and Robichaux, 2001] Bass, T. and Robichaux, R. (2001). Defense-in-depth revisited: Qualitative risk analysis methodology for complex network-centric operations. In *In Proceedings of the IEEE Military Communications Conference. Communications for Network-Centric Operations: Creating the Information Force.*

[Beck et al., 2001] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). Manifesto for agile software development. `http:`

//agilemanifesto.org/. Link visited last in 2018.

[Becker et al., 2012] Becker, T., Bartels, M., Hahne, M., Hempel, L., and Lieb, R. (2012). Cascading effects and interorganisational crisis management of critical infrastructure operators. findings of a research project. In *Proceedings of the 8th International Conference on Geo-information for Disaster Management – Best Practices.*

[Beckhoff, 2017] Beckhoff (2017). The automation device specification (ADS). https://infosys.beckhoff.com/english.php?content=../content/1033/tcadscommon/html/tcadscommon_intro.htm&id=. Link visited last in 2017.

[Bellard, 2018] Bellard, F. (2018). Qemu the fast! processor emulator v3.1.0. https://www.qemu.org/. Link visited last in 2018.

[Bellman, 1969] Bellman, R. (1969). A new and type of approximation and leading to reduction and of dimensionality and in control and processes. *Journal of Mathematical Analysis and Applications*, 27(2):454–459.

[Bellovin and Cheswick, 1994] Bellovin, S. M. and Cheswick, W. R. (1994). Network firewalls. *IEEE Communications Magazine*, 32(9):50 – 57.

[Berghel, 2007] Berghel, H. (2007). Better-than-nothing security practices security for general audiences. *COMMUNICATIONS OF THE ACM*, 50(8):15–18.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

[Bishop, 2007] Bishop, M. (2007). About penetration testing. *IEEE Security Privacy*, 5(6):84–87.

[BLISS et al., 1995] BLISS, J. P., GILSON, R. D., and DEATON, J. E. (1995). Human probability matching behaviour in response to alarms of varying reliability. *Ergonomics*, 38(11):2300–2312.

[Bobba et al., 2010] Bobba, R. B., Rogers, K. M., Wang, Q., Khurana, H., Nahrstedt, K., and Overbye, T. J. (2010). Detecting false data injection attacks on dc state estimation. In *Preprints of the First Workshop on Secure Control Systems, CPSWEEK.*

[Boehm, 1986] Boehm, B. W. (1986). A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes*, 11(4):14–24.

[Borchert and Spinczyk, 2016] Borchert, C. and Spinczyk, O. (2016). Hardening an l4 microkernel against soft errors by aspect-oriented programming and whole-program analysis. *ACM SIGOPS Operating Systems Review*, 49(2):37–43.

[Breivold and Sandstrom, 2015] Breivold, H. P. and Sandstrom, K. (2015). Internet of things for industrial automation – challenges and technical solutions. *2015 IEEE International Conference on Data Science and Data Intensive Systems.*

[Bryson, 1996] Bryson, A. (1996). Optimal control-1950 to 1985. *IEEE Control Systems*, 16(3):26–33.

[BSI, 2014] BSI (2014). Die Lage der IT-Sicherheit in Deutschland. `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2014.pdf?__blob=publicationFile`.

[BSI, 2016] BSI (2016). Die Lage der IT-Sicherheit in Deutschland.

[BSI, 2016a] BSI (2016a). Industrial control system security top 10 threats and countermeasures 2016. Technical report, Bundesamt für Sicherheit in de rInformationstechnik.

[BSI, 2016b] BSI (2016b). Sichere inter-netzwerk architektur sina. Technical report, Bundesamt für Sicherheit in der Informationstechnik (BSI).

[BSI, 2017] BSI (2017). Die Lage der IT-Sicherheit in Deutschland. `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2017.pdf?__blob=publicationFile&v=4`. Link visited last in 2017.

[Buchanan et al., 2008] Buchanan, E., Roemer, R., Shacham, H., and Savage, S. (2008). When good instructions go bad: Generalizing return-oriented programming to risc. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 27–38.

[Buza et al., 2014] Buza, D. I., Juhász, F., Miru, G., Félegyházi, M., and Holczer, T. (2014). Cryplh: Protecting smart energy systems from targeted attacks with a plc honeypot. In *International Workshop on Smart Grid Security*, pages 181–192. Springer.

[Büsch, 2018] Büsch, M. (2018). Awlsim: S7 compatible plc / sps, v. 0.67.2. `https://bues.ch/cms/automation/awlsim.html`. Link visited last in 2018.

[Campbell et al., 2018] Campbell, A., Beck, B., Friedl, M., Provos, N., de Raadt, T., and Song, D. (2018). The openssh suite 7.8. `https://www.openssh.com/`. Link visited last in 2018.

[Carcano et al., 2011] Carcano, A., Coletta, A., Guglielmi, M., Masera, M., Fovino, I. N., and Trombetta, A. (2011). A multidimensional critical state analysis for detecting intrusions in scada systems. *IEEE Transactions on Industrial Informatics*, 7(2):179 – 186.

[Cardenas et al., 2009] Cardenas, A. A., Amin, S., Sinopoli, B., Giani, A., Perrig, A., and Sastry, S. (2009). Challenges for securing cyber physical systems. In *Workshop on Future Directions in Cyber-physical Systems Security*.

[Carlson and Scharlott, 2006] Carlson, M. and Scharlott, A. (2006). Intrusion detection and prevention systems. `https://www.semanticscholar.org/paper/Intrusion-Detection-and-Prevention-Systems-Carlson-Scharlott/5bb422980cdcf1781cc34bef9980b04a2f763ed4`. Link visited last in 2018.

[Caselli et al., 2013] Caselli, M., Hadžiosmanović, D., Zambon, E., and Kargl, F. (2013). On the feasibility of device fingerprinting in industrial control systems. In *Critical Information Infrastructures Security - 8th International Workshop, CRITIS 2013*.

[Caselli et al., 2015] Caselli, M., Zambon, E., and Kargl, F. (2015). Sequence-aware intrusion detection in industrial control systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security - CPSS '15*. Association for Computing Machinery (ACM).

[Cavoukian and Dixon, 2013] Cavoukian, A. and Dixon, M. (2013). *Privacy and security by design: An enterprise architecture approach*. Information and Privacy Commissioner of Ontario, Canada.

[Chandola et al., 2009] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58.

[Chandola et al., 2012] Chandola, V., Banerjee, A., and Kumar, V. (2012). Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839.

[Chen et al., 2014] Chen, P., Desmet, L., and Huygens, C. (2014). A study on advanced persistent threats. In De, B., Decker, and Zúquete, A., editors, *15th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security (CMS 2014)*, volume 8735 of *LNCS*, page 63–72. Springer.

[Cheng et al., 2002] Cheng, J., Greiner, R., Kelly, J., Bell, D., and Liu, W. (2002). Learning bayesian and networks from data : An information-theory based approach.

[Cherdantseva and Hilton, 2013] Cherdantseva, Y. and Hilton, J. (2013). A reference model of information assurance & security. In *Eighth International Conference on Availability, Reliability and Security (ARES)*, pages 546–555. IEEE.

[Cheswick, 1992] Cheswick, B. (1992). An evening with berferd in which a cracker is lured, endured, and studied. In *Proc. Winter USENIX Conference, San Francisco*, pages 20–24.

[Cheung et al., 2007] Cheung, S., Dutertre, B., Fong, M., Lindqvist, U., Skinner, K., and Valdes, A. (2007). Using model-based intrusion detection for scada networks. In *Proceedings of the SCADA Security Scientific Symposium*.

[Cohen et al., 2003] Cohen, D., Lindvall, M., and Costa, P. (2003). Agile software development. *DACS SOAR Report*, 11:38.

[Combs, 2018a] Combs, G. (2018a). tshark - the wireshark network analyzer 2.6.3. `https://www.wireshark.org/docs/man-pages/tshark.html`. Link visited last in 2018.

[Combs, 2018b] Combs, G. (2018b). Wireshark network protocol analyzer. (v.2.6.0). `https://www.wireshark.org/`. Link visited last in 2018.

[Cruz et al., 2016] Cruz, T., Rosa, L., Proença, J., Maglaras, L., Aubigny, M., Lev, L., Jiang, J., and Simões, P. (2016). A cybersecurity detection framework for supervisory control and data acquisition systems. *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, 12(6):2236 – 2246.

[Cui et al., 2012] Cui, S., Han, Z., Kar, S., Kim, T. T., Poor, H. V., and Tajer, A. (2012). Coordinated data-injection attack and detection in the smart grid:

A detailed look at enriching detection solutions. *IEEE Signal Processing Magazine*, 29(5):106–115.

[Cárdenas et al., 2011] Cárdenas, A. A., Amin, S., Lin, Z.-S., Huang, Y.-L., Huang, C.-Y., and Sastry, S. (2011). Attacks against process control systems: Risk assessment, detection, and response. pages 355–366.

[Daneels and Salter, 1999] Daneels, A. and Salter, W. (1999). What is SCADA? In *International Conference on Accelerator and Large Experimental Physics Control Systems*, page 339–343.

[Davis et al., 2012] Davis, K. R., Morrow, K. L., Bobba, R., and Heine, E. (2012). Power flow cyber attacks and perturbation-based defense. In *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, pages 342–347. IEEE.

[DeHaan, 2018] DeHaan, M. (2018). Ansible v2.7.0.a1. `https://github.com/ansible/ansible`. Link visited last in 2018.

[Denning, 1987] Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering - Special issue on computer security and privacy*, 13(2):222–232.

[Digital Bond, 2010] Digital Bond (2010). Scada honeynet. `http://www.digitalbond.com/tools/scada-honeynet/`.

[Ding et al., 2018a] Ding, C., Zhai, J., and Dai, Y. (2018a). *An Improved ICS Honeypot Based on SNAP7 and IMUNES*.

[Ding et al., 2018b] Ding, D., Han, Q.-L., Xiang, Y., Ge, X., and Zhang, X.-M. (2018b). A survey on security control and attack detection for industrial cyber-physical systems. *Neurocomputing*, 275:1674–1683.

[DO et al., 2014] DO, V. L., Fillatre, L., and Nikiforov, I. (2014). A statistical method for detecting cyber/physical attacks on scada systems. In *Control Applications (CCA), 2014 IEEE Conference on*, pages 364–369. IEEE.

[Downs and Vogel, 1993] Downs, J. J. and Vogel, E. F. (1993). A plant-wide industrial process control problem. *Computers & chemical engineering*, 17(3):245–255.

[Dregier, 2017] Dregier, L. (2017). Penetration testing and ethical hacking. `https://www.cybrary.it/course/ethical-hacking/`. Link visited last in 2017.

[Drias et al., 2015] Drias, Z., Serhrouchni, A., and Vogel, O. (2015). Analysis of cyber security for industrial control systems. In *2015 International Conference on Cyber Security of Smart cities, Industrial Control System and Communications (SSIC)*. IEEE.

[Duggan et al., 2005] Duggan, D., Berg, M., Dillinger, J., and Stamp, J. (2005). Penetration testing of industrial control systems. Technical report, Sandia National Laboratories.

[Dunlap et al., 2016] Dunlap, S., Butts, J., Lopez, J., Rice, M., and Mullins, B. (2016). Using timing-based side channels for anomaly detection in industrial

control systems. *International Journal of Critical Infrastructure Protection*, 15:12–26.

[EC-Council, 2017] EC-Council (2017). Certified ethical hacker. `https://www.eccouncil.org/programs/certified-ethical-hacker-ceh/`. Link visited last in 2017.

[Efanov, 2012] Efanov, D. (2012). plcscan. `https://code.google.com/archive/p/plcscan/`. Link visited last in 2018.

[emergingthreats, 2018] emergingthreats (2018). Emerging threats open rulesets. `https://rules.emergingthreats.net/`. Link visited last in 2018.

[ENISA, 2011] ENISA (2011). Protecting industrial control systems - recommendations for europe and member states. Technical report, European Network and Information Security Agency (ENISA).

[ENISA, 2013] ENISA (2013). Annual incident reports 2013. `https://www.enisa.europa.eu/publications/annual-incident-reports-2013/at_download/fullReport`.

[ENISA, 2014] ENISA (2014). Annual incident reports 2014. `https://www.enisa.europa.eu/publications/annual-incident-reports-2014/at_download/fullReport`.

[ENISA, 2015] ENISA (2015). Annual incident reports 2015. `https://www.enisa.europa.eu/publications/annual-incident-reports-2015/at_download/fullReport`.

[ENISA, 2016] ENISA (2016). Annual incident reports 2016. `https://www.enisa.europa.eu/publications/annual-incident-reports-2016/at_download/fullReport`.

[ENISA, 2017] ENISA (2017). Baseline security recommendations for iot in the context of critical information infrastructures. Technical report, European Union Agency For Network And Information Security.

[Erez and Wool, 2015] Erez, N. and Wool, A. (2015). Control variable classification, modeling and anomaly detection in modbus/tcp scada systems. *International Journal of Critical Infrastructure Protection*, 10:59 – 70.

[Esler, 2017] Esler, B. (2017). Homag launching tapio platform to connect old, new machines at ligna 2017. `https://www.woodworkingnetwork.com/technology/homag-launching-tapio-platform-connect-old-new-machines-ligna-2017`. Link visited last in 2018.

[Eyisi and Koutsoukos, 2014] Eyisi, E. and Koutsoukos, X. (2014). Energy-based attack detection in networked control systems. In *Proceedings of the 3rd international conference on High confidence networked systems*, pages 115–124. ACM.

[F-Secure, 2014] F-Secure (2014). Havex hunts for ics/scada-systems. `https://www.f-secure.com/weblog/archives/00002718.html`.

[Falliere et al., 2011] Falliere, N., Murchu, L. O., and Chien, E. (2011). Symantec security response. `http://www.symantec.com/content/en/`

us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf. Version 1.4.

[Fauri et al., 2017] Fauri, D., de Wijs, B., den Hartog, J., Costante, E., Etalle, S., and Zambon, E. (2017). Encryption in ics networks: a blessing or a curse. Technical report, Technical report, Eindhoven Technical University (2017 to appear) Google Scholar.

[Ferrara et al., 2014] Ferrara, M., Franco, A., and Maltoni, D. (2014). The magic passport. In *IEEE International Joint Conference on Biometrics*.

[Fisch et al., 2017] Fisch, J., Rossdeutscher, M., and Diedrich, C. (2017). Anwendung datenbasierter methoden auf werkzeugmaschinendaten zur abweichungserkennung vom normalbetrieb. In *28. VDI-Fachtagung Technische Zuverlässigkeit 2017 - Entwicklung und Betrieb zuverlässiger Produkte*, VDI-Berichte. VDI Verlag GmbH.

[Forsberg and Mooz, 1991] Forsberg, K. and Mooz, H. (1991). The relationship of system engineering to the project cycle. In *INCOSE International Symposium*, volume 1, pages 57–65. Wiley Online Library.

[Foschepoth, 2009] Foschepoth, J. (2009). Postzensur und telefonüberwachung in der bundesrepublik deutschland (1949–1968). *Zeitschrift für Geschichtswissenschaft*, 57(5):413–426.

[Fraser et al., 1999] Fraser, T., Badger, L., and Feldman, M. (1999). Hardening cots software with generic software wrappers. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No.99CB36344)*, pages 2–16.

[Friedman et al., 1999] Friedman, N., lftach Nachman, and Peer, D. (1999). Learning bayesian and network structure and from massive and datasets: The "sparse and candidate" algorithm.

[Galloway and Hancke, 2013] Galloway, B. and Hancke, G. P. (2013). Introduction to industrial control networks. *IEEE Communications Surveys & Tutorials*, 15(2):860 – 880.

[Galstad, 2017] Galstad, E. (2017). The nagios it management software suite v4.3.4. https://www.nagios.org/. Link visited last in 2018.

[Garcia et al., 2016] Garcia, L., Zonouz, S., Wei, D., and de Aguiar, L. P. (2016). Detecting plc control corruption via on-device runtime verification. In *2016 Resilience Week (RWS)*. IEEE.

[Geer and Harthorne, 2002] Geer, D. and Harthorne, J. (2002). Penetration testing: A duet. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 185–195. IEEE.

[Genge et al., 2015] Genge, B., Haller, P., and Kiss, I. (2015). Cyber-security-aware network design of industrial control systems. *IEEE Systems Journal*, PP(99):1–12.

[Genge et al., 2014] Genge, B., Rusu, D. A., and Haller, P. (2014). A connection pattern-based approach to detect network traffic anomalies in critical infrastructures. In *Proceedings of the Seventh European Workshop on System*

Security. Association for Computing Machinery (ACM).

[Genge et al., 2013] Genge, B., Siaterlis, C., and Karopoulos, G. (2013). Data fusion-based anomaly detection in networned critical infrastructures, 43th ieee. In *IFIP International Conference on Dependable Systems and NetworNs (DSN 2013), WorNshop on Reliability and Security Data Analysis (RSDA 2013), Budapest, Hungary*.

[Ghaeini and Tippenhauer, 2016] Ghaeini, H. R. and Tippenhauer, N. O. (2016). Hamids: Hierarchical monitoring intrusion detection system for industrial control systems. In *CPS-SPC '16: Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, pages 103–111.

[Giani et al., 2011] Giani, A., Bitar, E., Garcia, M., McQueen, M., Khargonekar, P., and Poolla, K. (2011). Smart grid data integrity attacks: characterizations and countermeasures π. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 232–237. IEEE.

[Gilmore and Haydaman, 2016] Gilmore, C. and Haydaman, J. (2016). Anomaly detection and machine learning methods for network intrusion detection: an industrially focused literature review. In *Proceedings of the International Conference on Security and Management (SAM)*, page 292. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[Goh et al., 2017] Goh, J., Adepu, S., Junejo, K. N., and Mathur, A. (2017). A dataset to support research in the design of secure water treatment systems. In *Critical Information Infrastructures Security. CRITIS 2016. Lecture Notes in Computer Science.*, volume 10242. Springer.

[Goldenberg and Wool, 2013] Goldenberg, N. and Wool, A. (2013). Accurate modeling of modbus/tcp for intrusion detection in scada systems. *International Journal of Critical Infrastructure Protection*, 6:63–75.

[Gower, 1997] Gower, B. (1997). *Scientific Method An historical and philosophical introduction*. Routledge.

[Granberg et al., 2016] Granberg, M., Basile, A. G., Vroon, T., Riera, F. B. I., Zaman, J., Thode, M., Summers, M., and Vermeulen, S. (2016). Gentoo hardened. `https://wiki.gentoo.org/wiki/Project:Hardened`. Link visited last in 2018.

[Granger, 1969] Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438.

[Granger, 2004] Granger, C. W. J. (2004). Time series analysis, cointegration, and applications. *The American Economic Review*, 94(3):421–425.

[Green et al., 2017] Green, B., Krotofil, M., and Abbasi, A. (2017). On the significance and of process and comprehension for conducting and targeted ics and attacks. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 57–67.

[Grimm and Grimm, 1812] Grimm, J. and Grimm, W. (1812). *Kinder- und Haus-Märchen. Gesammelt durch die Brüder Grimm.* Realschulbuchhandlung.

[Guyon et al., 2008] Guyon, I., Janzing, D., and Schölkopf, B. (2008). Causality: Objectives and assessment. In Lawrence, N., editor, *JMLR Workshop and Conference Proceedings*, volume NIPS 2008 workshop on causality, page 1–38.

[Hadžiosmanović et al., 2012] Hadžiosmanović, D., Bolzoni, D., and Hartel, P. H. (2012). A log mining approach for process monitoring in scada. *International Journal of Information Security*, 11(4):231–251.

[Hadžiosmanović et al., 2014] Hadžiosmanović, D., Sommer, R., Zambon, E., and Hartel, P. H. (2014). Through the eye of the PLC: semantic security monitoring for industrial processes. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 126–135. Association for Computing Machinery (ACM).

[Hahn and Govindarasu, 2013] Hahn, A. and Govindarasu, M. (2013). Model-based intrustion detection for the smart grid (minds). In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*. ISBN: 978-1-4503-1687-3.

[Han et al., 2014] Han, S., Xie, M., Chen, H.-H., and Ling, Y. (2014). Intrusion detection in cyber-physical systems: Techniques and challenges. *IEEE systems journal*, 8(4):1052–1062.

[Hanisch et al., 1997] Hanisch, H.-M., Thieme, J., Luder, A., and Wienhold, O. (1997). Modeling of plc and behavior by means of timed net condition/event systems. In *In Proc. IEEE International Conference on Emerging Technologies and Factory Automation Proceedings (ETFA)*, pages 391–396. IEEE.

[Hawking, 2016] Hawking, S. W. (2016). Does god play dice? `http://www.hawking.org.uk/does-god-play-dice.html`. Link visited last in 2016.

[Heiner and Menzel, 1998] Heiner, M. and Menzel, T. (1998). A petri and net semantics and for the plc and language instruction and list. In *Proc. of the Fourth Workshop on Discrete Event Systems (WODES)*.

[Heisenberg, 1927] Heisenberg, W. (1927). Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. `https://web.archive.org/web/20130510070844/http://osulibrary.oregonstate.edu/specialcollections/coll/pauling/bond/papers/corr155.1.html`. Link visited last in 2018.

[Hempel et al., 2015] Hempel, L., Bartels, M., Büching, C., Horn, C., and Chemnitz, M. (2015). Handbuch kritische infrastruktur - hacker-angriffe auf infrastrukturen? zur sicherheit von versorgungsinfrastrukturen in unsicherer umgebung. Technical report, TU Berlin and Fraunhofer IPK.

[Hilt, 2016] Hilt, S. (2016). Gaspot. `https://github.com/sjhilt/GasPot`. Link visited last in 2018.

[Hlaváčková-Schindler et al., 2007] Hlaváčková-Schindler, K., Paluš, M., Vejmelka, M., and Bhattacharya, J. (2007). Causality detection based on information-theoretic approaches in time series analysis. *Physics Reports*, 1(441):1–46.

[Horn et al., 2014] Horn, C., Hempel, L., Chemnitz, M., Stewin, P., and Krüger, J. (2014). Steuerung: Advanced information security for critical infrastructures. In Klaus Thoma, Ivo Häring, T. L., editor, *9th Future Security SECURITY RESEARCH CONFERENCE*, pages 116–124, ISBN 978-3-8396-0778-7. Fraunhofer VVS, Fraunhofer Verlag.

[Horn and Klein, 2017] Horn, C. and Klein, M. (2017). Detektion von advanced persistent threats durch kausalitätsbasierte erkennung anomalen verhaltens in prozessen verteilter automatisierungsnetzwerke. In *Tagungsband zum 15. Deutschen IT-Sicherheitskongress*.

[Horn and Krüger, 2013] Horn, C. and Krüger, J. (2013). Enhanced infrastructure security through inter-level anomaly detection. In *32nd IEEE International Performance Computing and Communications Conference (IPCCC)*, pages 1 – 2. IEEE.

[Horn and Krüger, 2014] Horn, C. and Krüger, J. (2014). Proceed: Process state prediction for critis using process inherent causal data and discrete event models. In *Critical Information Infrastructure Security, 9th International Workshop, CRITIS 2014, Limassol, Cyprus, September 2014*, Lecture Notes in Computer Science.

[Horn and Krüger, 2015] Horn, C. and Krüger, J. (2015). Neuartige it-sicherheits-werkzeuge für industrie 4.0. In *Tagungsband zum Kongress Innosecure 2015*, pages 173–181. Apprimus Verlag, Aachen, 2015. ISBN 978-3-86359-299-8.

[Horn and Krüger, 2016a] Horn, C. and Krüger, J. (2016a). Feasibility of connecting machinery and robots to industrial control services in the cloud. In *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*. Electronic ISBN: 978-1-5090-1314-2 USB ISBN: 978-1-5090-1313-5 Print on Demand(PoD) ISBN: 978-1-5090-1315-9.

[Horn and Krüger, 2016b] Horn, C. and Krüger, J. (2016b). Herausforderungen der it-sicherheit an die industrie 4.0. In *Industrie 4.0 - Visionen der Steuerungstechnik/ im MES-Umfeld*, number 694, pages 38–48, Düsseldorf. VDI Verlag. ISBN 978-3-18-369402-0.

[Horn and Krüger, 2016c] Horn, C. and Krüger, J. (2016c). A retrofitting concept for integration of machinery with legacy interfaces into cloud manufacturing architectures. In *2016 16th International Conference on Control, Automation and Systems (ICCAS 2016) Oct. 16–19, 2016 in HICO, Gyeongju, Korea*.

[Hou et al., 2015] Hou, F., Pang, Z., Zhou, Y., and Sun, D. (2015). False data injection attacks for a class of output tracking control systems. In *Control and Decision Conference (CCDC), 2015 27th Chinese*, pages 3319–3323. IEEE.

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

[Hume, 1748] Hume, D. (1748). An enquiry concerning human understanding.

[Höne and Eloff, 2002] Höne, K. and Eloff, J. (2002). Information security policy — what do international information security standards say? *Computers & Security*, 21(5):402 – 409.

[Idika and Mathur, 2007] Idika, N. and Mathur, A. P. (2007). A survey of malware detection techniques. *Purdue University*, 48.

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

[Inoue et al., 2017] Inoue, J., Yamagata, Y., Chen, Y., Poskitt, C. M., and Sun, J. (2017). Anomaly detection for a water treatment system using unsupervised machine learning. In *In Proceesings IEEE International Conference on Data Mining Workshops (ICDMW 2017): Data Mining for Cyberphysical and Industrial Systems (DMCIS 2017)*.

[ISO12100, 2010] ISO12100 (2010). Safety of machinery - General principles for design - Risk assessment and risk reduction.

[ISO27004, 2009] ISO27004 (2009). Iso/iec 27004:2009 information technology — security techniques — information security management — measurement.

[ISO27005, 2011] ISO27005 (2011). Iso/iec 27005:2011 information technology - security techniques - information security risk management.

[ISO/IEC27000, 2014] ISO/IEC27000 (2014). Information technology - Security techniques - Information security management systems - Overview and vocabulary. International standard ISO/IEC 27000. Third edition.

[ISO/IEC27001, 2013] ISO/IEC27001 (2013). Information technology - Security techniques - Information security management systems -Requirements. International standard ISO/IEC 27001. Second edition.

[ISO/IEC27002, 2013] ISO/IEC27002 (2013). Information technology - Security techniques - Code of practice for information security controls. International standard ISO/IEC 27002. Second edition.

[ISO/IEC62264, 2013] ISO/IEC62264 (2013). Enterprise-control system integration. International standard ISO/IEC 62264. Third edition.

[Janicke et al., 2015] Janicke, H., Nicholson, A., Webber, S., and Cau, A. (2015). Runtime-monitoring for industrial control systems. *Electronics*, (4):995–1017.

[Jensen, 1981] Jensen, K. (1981). Coloured petri nets and the invariant method. *Theoretical Computer Science*, (14):317–336.

[Jensen, 1991] Jensen, K. (1991). Coloured petri nets: A high level language for system design and analysis. In *High-Level Petri Nets*, pages 44–119. Springer.

[Johnson, 2010] Johnson, R. E. (2010). Survey of scada security challenges and

potential attack vectors. In *International Conference for Internet Technology and Secured Transactions (ICITST)*, volume 1, pages 8–11.

[Kalisch et al., 2000] Kalisch, M., Mächler, M., Colombo, D., Hauser, A., Maathuis, M. H., and Bühlmann, P. (2000). More causal inference with graphical models in r package pcalg.

[Kalisch et al., 2012] Kalisch, M., Mächler, M., Colombo, D., Maathuis, M. H., and Bühlmann, P. (2012). Causal inference and using graphical and models with the r package pcalg. *JSS Journal of Statistical Software*, 47(11).

[Kaspersky Lab, 2016] Kaspersky Lab (2016). Threat landscape for industrial automation systems in the second half of 2016. Technical report, Kaspersky Lab Industrial Control Systems Cyber Emergency Response Team (Kaspersky Lab ICS CERT).

[Kaspersky Lab, 2017] Kaspersky Lab (2017). Threat landscape for industrial automation systems in h1 2017 kaspersky lab ics cert. `https://ics-cert.kaspersky.com/reports/2017/09/28/threat-landscape-for-industrial-automation-systems-in-h1-2017/`.

[Kemmerer and Vigna, 2002] Kemmerer, R. A. and Vigna, G. (2002). Intrusion detection: A brief history and overview. *IEEE Computer*, 35(4):27–30.

[Kim and Poor, 2011] Kim, T. T. and Poor, H. V. (2011). Strategic protection against data injection attacks on power grids. *IEEE Transactions on Smart Grid*, 2(2):326–333.

[King et al., 2004] King, S. T., Mao, Z. M., and Chen, P. M. (2004). Cids: Causality-based intrusion detection system. *Cse-tr-493-04, University of Michigan.*

[Kiss et al., 2015] Kiss, I., Genge, B., and Haller, P. (2015). A clustering-based approach to detect cyber attacks in process control systems. In *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*, pages 142–148. IEEE.

[Kittmann, 2017] Kittmann, T. (2017). Implementierung und evaluation der steuerung eines vorhandenen modell-wasserwerkes mit einer virtuellen open-source soft-sps. Master's thesis, Technische Universität Berlin.

[Klein, 2017] Klein, M. (2017). Entwicklung und implementierung eines konzeptes zur automatischen identifikation von kausalitäten in prozessdaten. Master's thesis, Technische Universität Berlin.

[Kleinmann and Wool, 2014] Kleinmann, A. and Wool, A. (2014). Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensics. *Journal of Digital Forensics, Security and Law*, 9(2).

[Kleinmann and Wool, 2016] Kleinmann, A. and Wool, A. (2016). Automatic construction of statechart-based anomaly detection models for multi-threaded scada via spectral analysis. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy.*

[Kleinmann and Wool, 2017] Kleinmann, A. and Wool, A. (2017). Automatic

construction of statechart-based anomaly detection models for multi-threaded industrial control systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(4).

[Klick et al., 2014] Klick, J., Lau, S., Marzin, D., Malchow, J.-O., and Roth, V. (2014). Internet-facing plcs - a new back orifice. Technical report.

[Kocher et al., 2018] Kocher, P., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., and Yarom, Y. (2018). Spectre attacks: Exploiting speculative execution. `https://spectreattack.com/spectre.pdf`. Link visited last in 2018.

[Kocher, 1996] Kocher, P. C. (1996). Timing attacks on implementations of diffe-hellman, rsa, dss, and other systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, pages 104–113.

[Kosut et al., 2010] Kosut, O., Jia, L., Thomas, R. J., and Tong, L. (2010). Malicious data attacks on smart grid state estimation: Attack strategies and countermeasures. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 220–225. IEEE.

[Krotofil et al., 2015] Krotofil, M., Larsen, J., and Gollmann, D. (2015). The process matters: Ensuring data veracity in cyber-physical systems. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 133–144. ACM.

[Kuipers and Fabro, 2006] Kuipers, D. and Fabro, M. (2006). Control systems cyber security: Defense in depth strategies. Technical report, Idaho National Laboratory (INL), U.S. Department of Energy National Laboratory.

[KUKA, 2018] KUKA (2018). Kuka connect - managing your automation process. `https://connect.kuka.com/`. Link visited last in 2018.

[Lahza et al., 2018] Lahza, H., Radke, K., and Foo, E. (2018). Applying domain-specific knowledge to construct features for detecting distributed denial-of-service attacks on the goose and mms protocols. *International Journal of Critical Infrastructure Protection*.

[Landry and Shamir, 2016] Landry, J. and Shamir, U. (2016). Malware discovered – sfg: Furtim malware analysis. `https://www.sentinelone.com/blog/sfg-furtims-parent/`. Link visited last in 2018.

[Langmann and Meyer, 2014] Langmann, R. and Meyer, L. (2014). Automation services from the cloud – new trends in process control technology. In *11th International Conference on Remote Engineering and Virtual Instrumentation*. IEEE.

[Laplace, 1814] Laplace, P. S. (1814). A philosophical essay on probabilities.

[Lau et al., 2000] Lau, F., Rubin, S. H., Smith, M. H., and Trajković, L. (2000). Distributed denial of service attacks. In *IEEE International Conference on Systems, Man, and Cybernetics*.

[Lau et al., 2016] Lau, S., Klick, J., Arndt, S., and Roth, V. (2016). Poster:

Towards highly interactive honeypots for industrial control systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1823–1825. ACM.

[Leder, 2016] Leder, F. (2016). Gsoc 2016 project ideas. `https://www.honeynet.org/gsoc2016/ideas`. Link visited last in 2018.

[Leibniz, 1686] Leibniz, G. W. (1686). Discourse on metaphysics.

[Li et al., 2010] Li, B.-H., Zhang, L., Wang, S.-L., Tao, F., Cao, J., Jiang, X., Song, X., and Chai, X. (2010). Cloud manufacturing: a new service-oriented networked manufacturing model. *Computer integrated manufacturing systems*, 16(1):1–7.

[Li et al., 2018] Li, G., Yan, Z., Fu, Y., and Chen, H. (2018). Data fusion for network intrusion detection: A review. *Security and Communication Networks*, 2018(8210614):1–16.

[Liang et al., 2014] Liang, J., Kosut, O., and Sankar, L. (2014). Cyber attacks on ac state estimation: Unobservability and physical consequences. In *PES General Meeting| Conference & Exposition, 2014 IEEE*, pages 1–5. IEEE.

[Lin et al., 2012a] Lin, H., Kalbarczyk, Z., and Iyer, R. K. (2012a). Adapting bro into scada: Building specification-based intrusion detection system for dnp3 protocol. Technical report, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.

[Lin et al., 2012b] Lin, H., Slagell, A., Di Martino, C., Kalbarczyk, Z., and Iyer, R. K. (2012b). Adapting bro into scada: building a specification-based intrusion detection system for the dnp3 protocol. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, page 5. ACM.

[Lin et al., 2013] Lin, H., Slagell, A., Kalbarczyk, Z., Sauer, P. W., and Iyer, R. K. (2013). Semantic security analysis of scada networks to detect malicious control commands in power grids. In *Proceedings of the first ACM workshop on Smart energy grid security*, pages 29–34. ACM.

[Linda et al., 2009] Linda, O., Vollmer, T., and Manic, M. (2009). Neural network based intrusion detection system for critical infrastructures. In *Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009*. IEEE.

[Lipovsky et al., 2017] Lipovsky, R., Cherepanov, A., Slowik, J., Miller, B., and Lee, R. (2017). Industroyer/crashoverride: Zero things cool about a threat group targeting the power grid. `https://www.blackhat.com/docs/us-17/wednesday/us-17-Lee-Industroyer-Crashoverride-Zero-Things-Cool-About-A-Threat-Group-Targeting-The-Power-Grid.pdf`. Link visited last in 2018.

[Lipp et al., 2018] Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., and Hamburg, M. (2018). Meltdown. `https://meltdownattack.com/meltdown.pdf`. Link visited last in 2018.

[Liu et al., 2001] Liu, S., Sullivan, J., and Ormaner, J. (2001). A practical approach to enterprise it security. *IEEE IT Professional*, 3(5):35 – 42.

[Liu et al., 2011] Liu, Y., Ning, P., and Reiter, M. K. (2011). False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13.

[Lunze, 2008a] Lunze, J. (2008a). *Regelungstechnik 1 Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen*. Springer.

[Lunze, 2008b] Lunze, J. (2008b). *Regelungstechnik 2 Mehrgrößensysteme, Digitale Regelung*. Springer.

[Lyon, 2009] Lyon, G. (2009). *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Nmap Project. ISBN 978-0-9799587-1-7.

[MacKenzie et al., 1997] MacKenzie, D., Eggert, P., and Stallman, R. (1997). *Comparing and Merging Files with GNU Diff and Patch*. Network Theory Ltd. ISBN: 978-0954161750.

[MacKinnon et al., 2013] MacKinnon, L., Bacon, L., Gan, D., Loukas, G., Chadwick, D., and Frangiskatos, D. (2013). *Strategic Intelligence Management*, chapter 20 - Cyber Security Countermeasures to Combat Cyber Terrorism, pages 234–261. Butterworth-Heineman, 1st edition edition.

[Maggi et al., 2017] Maggi, F., Quarta, D., Pogliani, M., Polino, M., Zanchettin, A. M., and Zanero, S. (2017). Rogue robots: Testing the limits of an industrial robot's security. Technical report, Trend Micro Incorporated.

[Maglaras and Jiang, 2014] Maglaras, L. A. and Jiang, J. (2014). A real time ocsvm intrusion detection module with low overhead for scada systems. *International Journal of Advanced Research in Artificial Intelligence*, (The).

[Malchow et al., 2015] Malchow, J.-O., Marzin, D., Klick, J., Kovacs, R., and Roth, V. (2015). Plc guard: A practical defense against attacks on cyber-physical systems. In *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 326–334.

[Markman et al., 2017] Markman, C., Wool, A., and Cardenas, A. A. (2017). A new burst-dfa model for scada anomaly detection. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 1–12.

[Mashima and Cárdenas, 2012] Mashima, D. and Cárdenas, A. A. (2012). Evaluating electricity theft detectors in smart grid networks. In *International Workshop on Recent Advances in Intrusion Detection*, pages 210–229. Springer.

[Massey, 1988] Massey, J. L. (1988). An introduction to contemporary cryptology. *Proceedings of the IEEE*, 76(5):533–549.

[Matherly, 2009] Matherly, J. (2009). The shodan computer search engine. `https://www.shodan.io/`. Link visited last in 2017.

[Mathur et al., 2016] Mathur, A. P., Ole, N., and Tippenhauer (2016). Swat: A water treatment testbed for research and training on ics security. In *In-*

ternational Workshop on Cyber-physical Systems for Smart Water Networks (CySWater). IEEE.

[Mayring, 2010] Mayring, P. (2010). *Qualitative Inhaltsanalyse*, volume Handbuch Qualitative Forschung in der Psychologie, chapter 4, pages 601–613. Günter Mey and Katja Mruck.

[Maziarz, 2015] Maziarz, M. (2015). A review of the granger-causality fallacy. *The Journal of Philosophical Economics*, VIII(2):86–105.

[McClean et al., 2013] McClean, J., Stull, C., Farrar, C., and Mascareñas, D. (2013). A preliminary cyber-physical security assessment of the robot operating system (ros). In *Unmanned Systems Technology XV*, volume 8741, page 874110. International Society for Optics and Photonics.

[McLaughlin, 2013] McLaughlin, S. (2013). Cps:stateful policy enforcement for control system device usage. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 109–118. ACM.

[McLaughlin et al., 2010] McLaughlin, S., Podkuiko, D., Miadzvezhanka, S., Delozier, A., and McDaniel, P. (2010). Multi-vendor penetration testing in the advanced metering infrastructure. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 107–116. ACM.

[McMillen and Li, 2017] McMillen, D. and Li, B. (2017). Security attacks on industrial control systems – how technology advances create risks for industrial organizations. Technical report, IBM Managed Security Services.

[MDT Software, 2018] MDT Software (2018). Mdt autosave protection and recovery solutions. `https://www.mdt-software.com/autosave-cybersecurity-solutions/`. Link visited last in 2018.

[Mercuri and Neumann, 2003] Mercuri, R. T. and Neumann, P. G. (2003). Inside risks: Security by obscurity. In *In Proceedings of the ACM Risks Forum*, volume 46, page 160.

[Miao et al., 2014] Miao, F., Zhu, Q., Pajic, M., and Pappas., G. J. (2014). Coding sensor outputs for injection attacks detection. In *53rd IEEE Conference on Decision and Control December 15-17, 2014. Los Angeles, California, USA*. IEEE.

[Minorsky, 1922] Minorsky, N. (1922). Directional stability of automatically steered bodies. *Journal of the American Society for Naval Engineers*, 34(2):280–309.

[Mitchell and Chen, 2014] Mitchell, R. and Chen, R. (2014). A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys (CSUR)*, 46(4).

[Mo et al., 2014] Mo, Y., Chabukswar, R., and Sinopoli, B. (2014). Detecting integrity attacks on scada systems. *IEEE Transactions on Control Systems Technology*, 22(4).

[Mo et al., 2015] Mo, Y., Weerakkody, S., and Sinopoli, B. (2015). Physical authentication of control systems: Designing watermarked control inputs to

detect counterfeit sensor outputs. *IEEE Control Systems*, 35(1):93–109.

[Moreno-Seco et al., 2006] Moreno-Seco, F., In˜esta, J. M., de Leo´n, P. J. P., Luisa, and Mico´ (2006). Comparison of classifier fusion methods for classification in pattern recognition tasks. In Yeung, D.-Y., Kwok, J. T., Fred, A., Roli, F., and de Ridder, D., editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 705–713. Springer Berlin Heidelberg.

[Morrow et al., 2012] Morrow, K. L., Heine, E., Rogers, K. M., Bobba, R. B., and Overbye, T. J. (2012). Topology perturbation for detecting malicious data injection. In *2012 45th Hawaii International Conference on System Sciences*, pages 2104–2113. IEEE.

[Mukherjee et al., 1994] Mukherjee, B., Heberlein, L., and Levitt, K. (1994). Network intrusion detection. *IEEE Network*, 8(3):26 – 41.

[MushMush-Foundation, 2015] MushMush-Foundation (2015). Conpot v0.5.1. `https://github.com/mushorg/conpot`. Link visited last in 2018.

[Nardella, 2015] Nardella, D. (2015). The snap7 suite 1.4.0. `http://snap7.sourceforge.net/`. Link visited last in 2017.

[NCCIC/ICS-CERT, 2013] NCCIC/ICS-CERT (2013). Year in review 2013. `https://ics-cert.us-cert.gov/sites/default/files/Annual_Reports/Year_In_Review_FY2013_Final.pdf`.

[NCCIC/ICS-CERT, 2014] NCCIC/ICS-CERT (2014). Year in review 2014. `https://ics-cert.us-cert.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2014_Final.pdf`.

[NCCIC/ICS-CERT, 2015] NCCIC/ICS-CERT (2015). Year in review 2015. `https://ics-cert.us-cert.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2015_Final_S508C.pdf`.

[NCCIC/ICS-CERT, 2016] NCCIC/ICS-CERT (2016). Year in review 2016. `https://ics-cert.us-cert.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2016_Final_S508C.pdf`.

[Nicholson et al., 2012] Nicholson, A., Webber, S., Dyer, S., Patel, T., and Janicke, H. (2012). SCADA security in the light of cyber-warfare. *Computers & Security*, 31(4):418–436.

[Niedermaier et al., 2018] Niedermaier, M., Malchow, J.-O., Fischer, F., Marzin, D., Merli, D., Roth, V., and von Bodisco, A. (2018). You snooze, you lose: Measuring plc cycle times under attacks. In *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*.

[Nivethan and Papa, 2016a] Nivethan, J. and Papa, M. (2016a). On the use of open-source firewalls in ics/scada systems. *Information Security Journal: A Global Perspective*.

[Nivethan and Papa, 2016b] Nivethan, J. and Papa, M. (2016b). A scada intrusion detection framework that incorporates process semantics. In *In Proceedings of the 11th Annual Cyber and Information Security Research Conference (CISRC '16)*.

[Nouretdinov et al., 2001] Nouretdinov, I., Vovk, V., Vyugin, M., and Gammerman, A. (2001). Pattern recognition and density estimation under the general iid assumption. In *International Conference on Computational Learning Theory*, pages 337–353. Springer.

[Office of Information Services, 2005] Office of Information Services (2005). Selecting a development approach. Technical report, Department of Health and Human Services - USA.

[OISF, 2017] OISF (2017). Suricata 4.0.3. `https://suricata-ids.org/`. Link visited last in 2018.

[Oxford, 2004] Oxford (2004). *The Oxford Thesaurus of English, Second Edition*. Casio EX-word, EW-S3000. Oxford University Press.

[Papa et al., 2012] Papa, S., Casper, W., and Nair, S. (2012). A transfer function based intrusion detection system for scada systems stephen papa, william casper suku nair. In *Proceedins of IEEE Conference on Technologies for Homeland Security (HST)*. IEEE.

[Parvania et al., 2014] Parvania, M., Koutsandria, G., Muthukumary, V., Peisert, S., McParland, C., and Scaglione, A. (2014). Hybrid control network intrusion detection systems for automated power distribution systems. In *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, pages 774–779. IEEE.

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

[Patcha and Park, 2007] Patcha, A. and Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470.

[Paxson, 1998] Paxson, V. (1998). Bro: A system for detecting network intruders in real-time. In *Proceedings of the 7th USENIX Security Symposium*.

[Paxson and Bro-Community, 2017] Paxson, V. and Bro-Community (2017). The bro network security monitor version 2.5.2. `https://www.bro.org/`. Link visited last in 2018.

[Pearl, 2000] Pearl, J. (2000). *CAUSALITY – Models, Reasoning, and Inference*. CAMBRIDGE UNIVERSITY PRESS.

[Pearl, 2009] Pearl, J. (2009). Causal inference in statistics: An overview. *Statistics Surveys*, 3(0):96–146.

[Peterson, 2004] Peterson, D. (2004). Intrusion detection and cyber security monitoring of scada and dcs networks. *ISA Automation West*.

[Poll, 1970] Poll, R. A. (1970). Approaches to system hardening. *IEEE Transactions on Nuclear Science*, 17(6):83–90.

[Pothamsetty and Franz, 2005] Pothamsetty, V. and Franz, M. (2005). Scada honeynet project: Building honeypots for industrial networks (v.0.3). `http://scadahoneynet.sourceforge.net/`. Link visited last in 2018.

[Powers, 2007] Powers, D. M. W. (2007). Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation. Technical Report SIE-07-001, Flinders University of South Australia.

[PwC, 2015] PwC (2015). 2015 information security breaches survey. Technical report, PricewaterhouseCoopers.

[Qin and Lee, 2004] Qin, X. and Lee, W. (2004). Attack plan recognition and prediction using causal networks. In *Computer Security Applications Conference, 2004. 20th Annual*, pages 370–379. IEEE.

[Qiu et al., 2012] Qiu, H., Liu, Y., Subrahmanya, N. A., and Li, W. (2012). Granger causality for time-series anomaly detection. In *IEEE 12th International Conference on Data Mining (ICDM)*.

[Quarantelli, 1988] Quarantelli, E. L. (1988). Disaster crisis management: A summary of research findings. *Journal of management studies*, 25(4):373–385.

[R Foundation, 2018] R Foundation (2018). The r project for statistical computing, v 3.5.1. `https://www.r-project.org/`. Link visited last in 2018.

[Radvanovsky, 2014] Radvanovsky, B. (2014). Project shine findings report. Technical report, Infracritical.

[Rapid7, 2014] Rapid7 (2014). The metasploit framework 4.10. `http://www.metasploit.com`. Link visited last in 2017.

[Redwood, 2015] Redwood, W. O. (2015). *Cyber Physical System Vulnerability Research*. PhD thesis.

[Reeves et al., 2012] Reeves, J., Ramaswamy, A., Locasto, M., Bratus, S., and Smith, S. (2012). Intrusion detection for resource-constrained embedded control systems in the power grid. *International Journal of Critical Infrastructure Protection*, 5(2):74–83.

[Rhebo GmbH, 2018] Rhebo GmbH (2018). Rhebo industrial protector. `https://rhebo.com/`. Link visited last in 2018.

[Rich, 1995] Rich, B. R. (1995). *Clarence Leonard (Kelly) Johnson 1910–1990: A Biographical Memoir*. National Academies Press.

[Roesch, 1999] Roesch, M. (1999). Snort - lightweight intrusion detection for networks. In *Proceedings of LISA '99: 13th Systems Administration Conference*.

[Rose and Cass, 1987] Rose, M. and Cass, D. (1987). Iso transport service on top of the tcp. `https://tools.ietf.org/html/rfc1006`.

[Royce, 1970] Royce, W. W. (1970). Managing the development of large software systems. In *Proceedings of IEEE WESCON*, pages 1–9.

[Rrushi and Campbell, 2008] Rrushi, J. and Campbell, R. (2008). Detecting cyber attacks on nuclear power plants. In *International Conference on Critical Infrastructure Protection*, pages 41–54. Springer.

[Rubio-Herna´n et al., 2016] Rubio-Herna´n, J., Cicco, L. D., and Garcı´a-

Alfaro, J. (2016). Revisiting a watermark-based detection scheme to handle cyber-physical attacks. In *In Proceedings 11th International Conference on Availability, Reliability and Security*. IEEE.

[Rumbaugh et al., 2004] Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified modeling language reference manual, the*. Pearson Higher Education.

[Ruparelia, 2010] Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, 35(3):8–13.

[Russell and Norvig, 2012] Russell, S. and Norvig, P. (2012). *Künstliche Intelligenz, Ein moderner Ansatz*. Pearson Deutschland GmbH.

[Saaty, 1987] Saaty, R. (1987). The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling*, 9(3-5):161–176.

[Sajjad et al., 2015] Sajjad, I., Dunn, D. D., Sharma, R., and Gerdes, R. (2015). Attack mitigation in adversarial platooning using detection-based sliding mode control. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, pages 43–53. ACM.

[Saltzer and Schroeder, 1975] Saltzer, J. H. and Schroeder, M. D. (1975). The protection of information in computer systems. *IEEE CSIT Newsletter*, 3(12).

[Sandberg et al., 2010] Sandberg, H., Teixeira, A., and Johansson, K. H. (2010). On security indices for state estimators in power networks. In *Preprints of the First Workshop on Secure Control Systems, CPSWEEK 2010*.

[Sasaki and Aoki, 2009] Sasaki, Y. and Aoki, K. (2009). Finding preimages in full md5 faster than exhaustive search. In *Advances in Cryptology - EUROCRYPT 2009*, pages 134–152. Springer.

[Schaelicke et al., 2003] Schaelicke, L., Slabach, T., Moore, B., and Freeland, C. (2003). Characterizing the performance of network intrusion detection sensors. In *International Workshop on Recent Advances in Intrusion Detection*, pages 155–172. Springer.

[Schauer, 2015] Schauer, T. (2015). Systemidentifikation und regelung in der medizin. `http://www.control.tu-berlin.de/Teaching:Systemidentifikation_und_Regelung_in_der_Medizin#Downloads`. Link visited last in 2018.

[Scheines et al., 1998] Scheines, R., Spirtes, P., Glymour, C., Meek, C., and Richardson, T. (1998). The tetrad project: Constraint based aids to causal model specification.

[Schneeweiss, 1990] Schneeweiss, C. (1990). Kostenwirksamkeitsanalyse, nutzwertanalyse und multi-attributive nutzentheorie. *Wirtschaftswissenschaftliches Studium*, 19(1):13–18.

[Schneider, 2000] Schneider, F. B. (2000). Enforceable security policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(1).

[Schneier, 2014] Schneier, B. (2014). The future of incident response. *IEEE Security & Privacy*, 12.

[Scholz, 2005] Scholz, P. (2005). *Softwareentwicklung eingebetteter Systeme - Grundlagen, Modellierung, Qualitätssicherung*. Springer.

[Schuster et al., 2015] Schuster, F., Paul, A., Rietz, R., and König, H. (2015). Potentials of using one-class svm for detecting protocol-specific anomalies in industrial networks. In *2015 IEEE Symposium Series on Computational Intelligence*. IEEE.

[Schwartau, 1998] Schwartau, W. (1998). Time-based security explained: Provable security models and formulas for the practitioner and vendor. *Computers & Security*, 17(8):693–714.

[Semenov and Mitja, 1981] Semenov, S. and Mitja, A. (1981). *Tecnología prehistórica: estudio de las herramientas y objetos antiguos a través de las huellas de uso*. Akal Universitaria. Akal.

[Senyondo et al., 2015] Senyondo, H., Sun, P., Berthier, R., and Zonouz, S. (2015). Plcloud: Comprehensive power grid plc security monitoring with zero safety disruption. In *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm): Architectures, Control and Operation for Smart Grids and Microgrids*. IEEE.

[Shamir, 2016] Shamir, U. (2016). Analyzing a new variant of blackenergy 3. Technical report, SentinelOne.

[Shekhar, 2016] Shekhar, A. (2016). How to perform ping of death attack using cmd and notepad (just for learning). `https://fossbytes.com/perform-ping-of-death-attack-using-cmd-just-for-learning/`. Link visited last in 2018.

[Shoukry et al., 2015] Shoukry, Y., Martin, P., Yona, Y., Diggavi, S., and Srivastava, M. (2015). Pycra: Physical challenge-response authentication for active sensors under spoofing attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1004–1015. ACM.

[SIEMENS, 2016] SIEMENS (2016). End user license agreement (eula) (splm) version 6.1. `https://www.plm.automation.siemens.com/de/products/eula/`. Link visited last in 2018.

[Siemens AG, 2010] Siemens AG (2010). System software for s7-300/400 system and standard functions. `https://support.industry.siemens.com/cs/document/44240604/system-software-for-s7-300-400-system-and-standard-functions-volume-1-and-volume-2?dti=0&lc=en-WW`. Link visited last in 2018.

[Simon and Rescher, 1966] Simon, H. A. and Rescher, N. (1966). Cause and counterfactual. *Philosophy of Science*, 33(4):323–340.

[sk4ld, 2015] sk4ld (2015). Gridpot. `http://www.gridpot.org/`. Link visited last in 2018.

[Smith, 2011] Smith, R. S. (2011). A decoupled feedback structure for covertly appropriating networked control systems. *IFAC Proceedings Vol-*

umes, 44(1):90–95.

[Sonneborn and Vleck, 1965] Sonneborn, L. M. and Vleck, F. S. V. (1965). The bang-bang principle for linear control systems. *Journal of the Society for Industrial and Applied Mathematics Series A Control*, 2(2):151–159.

[Spirtes, 2010] Spirtes, P. (2010). Introduction to causal inference. *Journal of Machine Learning Research*, (11):1643–1662.

[Spirtes and Glymour, 1991] Spirtes, P. and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72.

[Spirtes et al., 2000] Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search*. Carnegie Mellon University.

[Spitzner, 2003] Spitzner, L. (2003). The honeynet project: Trapping the hackers. *IEEE Security & Privacy*, 99.

[SQLite Consortium, 2018] SQLite Consortium (2018). Sqlite version3.24.0. `https://sqlite.org`. Link visited last in 2018.

[Sridhar and Govindarasu, 2014] Sridhar, S. and Govindarasu, M. (2014). Model-based attack detection and mitigation for automatic generation control. *IEEE Transactions on Smart Grid*, 5(2):580–591.

[Stallman, 2002] Stallman, R. M. (2002). What is free software. *Free Society: Selected Essays of*, 23.

[Staniford-Chen et al., 1996] Staniford-Chen, S., Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Wee, C., Yip, R., and Zerkle, D. (1996). Grids-a graph based intrusion detection system for large networks. In *Proceedings of the 19th national information systems security conference*, volume 1, pages 361–370.

[Statnikov et al., 2009] Statnikov, A., Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2009). Causal explorer: A matlab library of algorithms for causal discovery and variable and selection for classification.

[Stauß, 2005] Stauß, O. (2005). Würden pumpen doch nur richtig bedient.... `https://industrieanzeiger.industrie.de/allgemein/wuerden-pumpen-doch-nur-richtig-bedient/`. Link visited last in 2018.

[Stefanidis and Voyiatzis, 2016] Stefanidis, K. and Voyiatzis, A. G. (2016). An hmm-based anomaly detection approach for scada systems. In Foresti, S. and Lopez, J., editors, *WISTP 2016*, volume 9895 of *LNCS*, page 85–99. Springer.

[Stevens, 2012] Stevens, M. (2012). *Attacks on Hash Functions and Applications*. PhD thesis, Universiteit Leiden.

[Stevens et al., 2017] Stevens, M., Bursztein, E., Karpman, P., Albertini, A., and Markov, Y. (2017). The first collision for full sha. In *Annual International Cryptology Conference*, pages 570–596. Springer.

[Stewin and Seifert, 2010] Stewin, P. and Seifert, J.-P. (2010). In god we trust all others we monitor. In *Proceedings of the 17th ACM conference on Com-*

puter and communications security, pages 639–641.

[Stone et al., 2015] Stone, S. J., Temple, M. A., and Baldwin, R. O. (2015). Detecting anomalous programmable logic controller behavior using rf-based hilbert transform features and a correlation-based verification process. *International Journal of Critical Infrastructure Protection*.

[Stouffer et al., 2011] Stouffer, K., Falco, J., and Scarfone, K. (2011). Guide to industrial and control and systems (ics) and security - recommendations of the national institute of standards and technology. `http://csrc.nist.gov/publications/nistpubs/800-82/SP800-82-final.pdf`.

[Stouffer et al., 2015] Stouffer, K., Pillitteri, V., Abrams, M., and Hahn, A. (2015). Guide to industrial control systems (ics) security. Technical report, National Institute of Standards and Technology.

[Strackx and Piessens, 2012] Strackx, R. and Piessens, F. (2012). Fides: Selectively hardening software application components against kernel-level or process-level malware. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 2–13. ACM.

[Symantec, 2011] Symantec (2011). Advanced persistent and threats: A and symantec perspective. Technical report, Symantec Corporation.

[Symantec, 2014] Symantec (2014). Dragonfly: Cyberespionage attacks against energy suppliers (symantec security response). `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/Dragonfly_Threat_Against_Western_Energy_Suppliers.pdf`.

[Symantec, 2018] Symantec (2018). Symantec™ anomaly detection for industrial control systems. `https://support.symantec.com/en_US/article.DOC10938.html`. Link visited last in 2018.

[Sándor et al., 2017] Sándor, H., Genge, B., Haller, P., Duka, A.-V., and Crainicu, B. (2017). Cross-layer anomaly detection in industrial cyber-physical systems. In *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*.

[Tao et al., 2011] Tao, F., Zhang, L., Venkatesh, V. C., Luo, Y., and Cheng, Y. (2011). Cloud manufacturing: a computing and service-oriented manufacturing model. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 225(10):1969–1976.

[Teixeira et al., 2010] Teixeira, A., Amin, S., Sandberg, H., Johansson, K. H., and Sastry, S. S. (2010). Cyber security analysis of state estimators in electric power systems. In *49th IEEE Conference on Decision and Control (CDC)*., pages 5991–5998.

[Teixeira et al., 2012] Teixeira, A., Shames, I., Sandberg, H., and Johansson, K. H. (2012). Revealing stealthy attacks in control systems. In *50th Annual Allerton Conference on Communication, Control, and Computing, Allerton, IL, USA, October 01-05, 2012*, pages 1806–1813. IEEE conference proceedings.

[The Debian Project, 2018] The Debian Project (2018). Debian hardening. `https://wiki.debian.org/Hardening`. Link visited last in 2018.

[The Open Group, 2009] The Open Group (2009). Soa source book edition 7. `http://www.opengroup.org/soa/source-book/intro/`. Link visited last in 2018.

[Thomopoulos et al., 1987] Thomopoulos, S. C., Viswanathan, R., and Bougoulias, D. C. (1987). Optimal decision fusion in multiple sensor systems. *IEEE Transactions on Aerospace and Electronic Systems*, 23(5):644 – 653.

[Thompson, 2005] Thompson, H. H. (2005). Application penetration testing. *IEEE Security & Privacy*, 3(1):66–69.

[Tidriri et al., 2018] Tidriri, K., Tiplica, T., Chatti, N., and Verron, S. (2018). A generic framework for decision fusion in fault detection and diagnosis. *Engineering Applications of Artificial Intelligence*, 71:73–86.

[Tripwire-Community, 2017] Tripwire-Community (2017). Open source tripwire. `https://github.com/Tripwire/tripwire-open-source`. Link visited last in 2018.

[Tsai et al., 2009] Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., and Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000.

[Tsamardinos et al., 2006] Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78.

[United Nations, 2014] United Nations (2014). World urbanization prospects: The 2014 revision. Technical report, United Nations, Department of Economic and Social Affairs, Population Division.

[Urbina et al., 2016] Urbina, D. I., Giraldo, J., Cardenas, A. A., Valente, J., Faisal, M., Tippenhauer, N. O., Ruths, J., Candell, R., and Sandberg, H. (2016). Nist gcr 16-010: Survey and new directions for physics-based attack detection in control systems. Technical report, U.S. National Institute of Standards and Technology.

[U.S. Attorney's Office, 2016] U.S. Attorney's Office (2016). Manhattan U.S. Attorney Announces Charges Against Seven Iranians For Conducting Coordinated Campaign Of Cyber Attacks Against U.S. Financial Sector On Behalf Of Islamic Revolutionary Guard Corps-Sponsored Entities. `https://www.justice.gov/usao-sdny/pr/manhattan-us-attorney-announces-charges-against-seven-iranians-conducting-coordinated`. Link visited last in 2018.

[Vasilomanolakis et al., 2016] Vasilomanolakis, E., Srinivasa, S., Cordero, C. G., and Mtihlhauser, M. (2016). Multi-stage attack detection and signature generation with ics honeypots. In *IEEE/IFIP NOMS 2016 Workshop: 2nd Workshop on Security for Emerging Distributed Network Technologies (DISSECT)*. IEEE.

[VDI, 1993] VDI (1993). Methodik zum entwickeln und konstruieren technischer systeme und produkte.

[Verba and Milvich, 2008] Verba, J. and Milvich, M. (2008). Idaho national laboratory supervisory control and data acquisition intrusion detection system (scada ids). In *IEEE Conference on Technologies for Homeland Security*. IEEE.

[Vick et al., 2015] Vick, A., Horn, C., Rudorfer, M., and Krüger, J. (2015). Control of robots and machine tools with an extended factory cloud. In *Factory Communication Systems (WFCS), 2015 IEEE World Conference on*, pages 1–4. IEEE, IEEE.

[Virvilis and Gritzalis, 2013] Virvilis, N. and Gritzalis, D. (2013). The big four - what we did wrong in advanced persistent threat detection? In *Eighth International Conference on Availability, Reliability and Security*, pages 248–254. IEEE.

[Vollmer and Manic, 2014] Vollmer, T. and Manic, M. (2014). Cyber-physical system security with deceptive virtual hosts for industrial control networks. *IEEE Transactions on Industrial Informatics*, 10(2):1337–1347.

[Vuković and Dán, 2013] Vuković, O. and Dán, G. (2013). On the security of distributed power system state estimation under targeted attacks. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 666–672. ACM.

[Wang et al., 2014] Wang, Y., Xu, Z., Zhang, J., Xu, L., Wang, H., and Gu, G. (2014). Srid: State relation based intrusion detection for false data injection attacks in scada. In *European Symposium on Research in Computer Security*, pages 401–418. Springer.

[Westergaard and Verbeek, 2018] Westergaard, M. and Verbeek, H. (2018). Cpn tools v. 4.0. `http://cpntools.org/`. Link visited last in 2018.

[Whitman, 2003] Whitman, M. E. (2003). Enemy at the gate: threats to information security. *Communications of the ACM*, 46(8):91–95.

[Wilhoit, 2013a] Wilhoit, K. (2013a). The scada that didn't cry wolf – who's really attacking your ics equipment? (part 2). Technical report, Trend Micro Incorporated.

[Wilhoit, 2013b] Wilhoit, K. (2013b). Who's really attacking your ics equipment? Technical report, Trend Micro Incorporated.

[Wong et al., 2017] Wong, K., Dillabaugh, C., Seddigh, N., and Nandy, B. (2017). Enhancing suricata intrusion detection system for cyber security in scada networks. In *In Proc. IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE.

[Wrigstad et al., 2009] Wrigstad, T., Eugster, P., Field, J., Nystrom, N., and Vitek, J. (2009). Software hardening: a research agenda. In *Proceedings for the 1st workshop on Script to Program Evolution*, pages 58–70. ACM.

[Wu et al., 2013] Wu, D., Greer, M. J., Rosen, D. W., and Schaefer, D. (2013).

Cloud manufacturing: Strategic vision and state-of-the-art. *Journal of Manufacturing Systems*, 32(4):564–579.

[Xie et al., 2010] Xie, T., Liu, F., and Feng, D. (2010). Fast collision attack on md. Cryptology ePrint Archive, Report 2013/170. `https://eprint.iacr.org/2013/170`.

[Xu, 2012] Xu, X. (2012). From cloud computing to cloud manufacturing. *Journal of Robotics and Computer-Integrated Manufacturing*, 28(1):75–86.

[Yang et al., 2006] Yang, D., Usynin, A., and Hines, J. W. (2006). Anomaly-based intrusion detection for scada systems. In *Proceedings of the 5. International Topical Meeting on Nuclear Plant Instrumentation Controls, and Human Machine Interface Technology*.

[Yang et al., 2014] Yang, Y., McLaughlin, K., Sezer, S., Littler, T., Im, E. G., Pranggono, B., and Wang, H. F. (2014). Multiattribute scada-specific and intrusion and detection system and for power and networks. *IEEE Transactions on Power Delivery*, 29(3).

[Zalewski, 2016] Zalewski, M. (2016). p0f v3 (version 3.09b). `http://lcamtuf.coredump.cx/p0f3/`. Link visited last 2018.

[Zhang et al., 2011] Zhang, L., Lampe, M., and Wang, Z. (2011). A hybrid genetic algorithm to optimize device allocation in industrial ethernet networks with real-time constraints. *Journal of Zhejiang University-SCIENCE C*.

[Zhu and Sastry, 2010] Zhu, B. and Sastry, S. (2010). Scada-specific intrusion detection/prevention systems: A survey and taxonomy. In *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*, volume 11. IEEE.

,