Process-aware SCADA Traffic Monitoring: A Local Approach

Justyna Chromik

Process-aware SCADA traffic monitoring:
A local approach

Justyna Chromik

Graduation Committee:

Chairman:	prof. dr. J.N. Kok
Promoter:	prof. dr. ir. B. R. H. M. Haverkort
Promoter:	prof. dr. A. Remke

Members:

prof. dr. J.L. Hurink	University of Twente, The Netherlands
dr. C.E.W. Hesselman	University of Twente, The Netherlands
prof. dr. S. Nadjm-Tehrani	Linköping University, Sweden
prof. dr. S. Lehnhoff	University of Oldenburg, Germany
prof. dr. R. Sadre	University of Louvain, Belgium

Funding Sources:

"MOSES" – More Secure SCADA through Self-Awareness.
This research is supported by Dutch Organisation for Scientific Research (NWO) through project number 628.001.012

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

DSI Ph.D. thesis series No. 19-009
Digital Society Institute
P.O. Box 217, 7500 AE
Enschede, The Netherlands

ISBN 978-90-365-4801-4
ISSN 2589-7721 (DSI Ph.D. thesis series No. 19-009)
DOI 10.3990/1.9789036548014
https://doi.org/10.3990/1.9789036548014

Cover design by Esie Kamari.
Type set with LaTeX. Printed by IPSKAMP.

Process-aware SCADA traffic monitoring

A local approach

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. T.T.M. Palstra,
on account of the decision of the Doctorate Board,
to be publicly defended
on Friday the 12^{th} of July 2019 at 12.45

by

Justyna Joanna Chromik

born on the 30^{th} March 1989
in Pszczyna, Poland

This thesis has been approved by:

prof. dr. ir. Boudewijn R. Haverkort (promoter)
prof. dr. Anne Remke (promoter)

Acknowledgments & Podziękowania

Inspiration and support are the basic ingredients for completing a PhD thesis. Therefore, I would like to thank all of you, who were the source of inspiration and the source of support in my PhD life. Should I forget to mention someone, I sincerely apologise.

First of all, I would like to thank my supervisors, Anne and Boudewijn, for many encouraging meetings. You believed in me when I had doubts, and motivated me when I was sceptical. Boudewijn, thank you for all your insights, which not only were focused on research but also on getting in touch with the right people. Anne, thank you for being both a supervisor and a friend. I enjoyed all our coffee breaks discussions mixing the subject of my research with talks about horses and horse riding, a hobby that we both share.

Managing tricky agendas of my supervisors would of course not be possible without Jeanette. Thank you, Jeanette, for always being there to help arrange meetings, trips, and brighten some gloomy days with your contagious laughter.

I could not imagine a better pair of paranymphs, than Bernd and Jair. Bernd has been the best assignment companion during my Master degree at the University of Twente. As a colleague, Bernd was always there to walk to the Subway or complain about the soup during the lunch break. Jair was my supervisor for my Master thesis, and he encouraged me to write my first conference proceeding. He is the person to talk to if you feel like getting inspired, or if you need help, as he is always there for you.

My time spent at DACS was great thanks to also other colleagues, including, of course, the office "roommates". Hamed, Mozhdeh, Björn, Baver, and especially Anne, whose promotion date so close to mine made us share a common hobby of getting stressed about the final process. I would also like to thank all the students that have collaborated with me during my PhD time, especially Robert, Max, Ben and Alpha.

My industrial knowledge and experience would be much more limited if it weren't for the help from people working with critical infrastructures every day. Gerard (from Coteq), Ronald (from DataWatt), Joey and Pascal (from Stedin), and Rafael and Maarten (from ENCS) helped me to gain a more realistic view on the current state of security in power distribution systems. Financial support cannot be underestimated. I would therefore like to thank the partners of the

MOSES project: Coteq Netbeheer, European Network for Cyber Security and TNO, for sponsoring this NWO project.

Serdecznie dziękuję wszystkim którzy mnie wspierali na odległość. Mieszkanie za granicą ma swoje wypełnione nostalgią uroki, których nie da się obejść. Dziękuję, Dziewior, za kawy na Skypie, pełne wsparcia i wspólnego narzekania na obczyznę. Marysi i Madzi za wizyty, które w jakiś sposób przybliżyły Pszczynę do Utrechtu. Martynie za cieprliwość przy moich pytaniach o to jak po angielsku napisać to czy owo.

Mij PhD-tijd ging niet alleen over onderzoek, maar ook over integratie met de Nederlandse maatschappij. Deze integratie was moeilijker geweest zonder mijn Nederlandse familie. Ik ben superblij met alle ondersteuning van mijn lieve familie: Ellen, Paul, Sarah en Leon.

Mojej siostrze, Paulinie, dziękuję za bycie inspiracją w życiu prywatnym i zawodowym. Nie znam lepszego przykładu do "co Cię nie zabije to Cię wzmocni" niż ona. Babci i dziadkowi, dziękuję za nauczenie się obsługi Skypa (niestety, z czasem też zapomnienie). Oczywiście chciałam też podziękować moim rodzicom, Bogusi i Darkowi, którzy wielokrotnie stali w korkach gdzieś na autostradzie w Niemczech, byle przywieźć Polskie smakołyki swoim córkom w Holandii. Bez Waszego wsparcia nie miałabym jak iść na studia, a co dopiero przeprowadzić się do Holandii. Mimo że nie każde "sto lat" mogliśmy śpiewać sobie na żywo, to i tak dzwoniliśmy i śpiewaliśmy. Nie ma nic piękniejszego od kochającej rodziny, więc dziękuję Wam całym sercem za Wasze serca, za Waszą miłość, za Waszą cierpliwość i upór. Kocham Was z całego serca!

And Jurriën, my Honey. Thank you for your unlimited support and love. Thank you for being there for me, listening to my complaints about NS or otherwise picking me up every time the train got cancelled. Thank you for encouraging me to try out different methods for solving my problems along the way. Finally, thank you for being *you*.

Stysia
June 2019

Abstract

Supervisory Control and Data Acquisition (SCADA) systems are used to monitor and control large physical infrastructures, such as electricity transmission and distribution systems. For years they have operated as isolated systems, using proprietary protocols, and keeping the exchanged information only within the system, which was designed in a centralized architecture.

Nowadays, however, SCADA systems are closely connected to the Internet in order to provide remote control capabilities. This makes them vulnerable to adversaries, which aim at disrupting the controlled process. Cyber security of SCADA systems has only recently started to pave its way up the companies' agendas after discovering the disastrous physical consequences of the Stuxnet malware in 2010. It was the first registered case where cyber commands resulted in physical damage of a system. This incident has made the operators more aware of the possibilities that malicious parties have, once they have entered a SCADA system.

Monitoring SCADA systems is a popular way to keep track of activities that are happening inside such systems. Unfortunately, approaches that are successful in regular IT systems are, however, not always applicable in a critical infrastructure environment, where SCADA systems are often used. Many existing approaches rely on the assumption that traffic within SCADA systems is quite stable and predictive, and identify hosts that are allowed to communicate within the system by creating so-called whitelists. Other techniques, such as deep packet inspection, require the capability to read and interpret protocol-specific information from captured packets in real-time. Based on extracted information, adequate measures are taken, for example, an alert can be raised when a specific host sends a message that has not been authorised. However, real-life incidents show that disruptive commands can originate at authorised, legitimate hosts, leading to undesired consequences, such as a blackout. Unfortunately, most of the proposed approaches do not investigate the effect of the analysed packets on the underlying, physical system.

In contrast, this thesis focuses on enhancing the traffic monitoring by proposing a local and process-aware monitoring tool for power distribution systems, that detects when the physical process is in an unsafe state. Introducing such a monitoring tool at each local substation is feasible by maintaining a model of

the substation and of the sensors and actuators that are directly accessible from this substation.

As a result, this thesis proposes a new and generic modelling formalism that can describe (a part of) a power distribution system, combined with a new local monitoring algorithm that can validate a set of physical constraints and safety requirements that are required to hold in the power distribution system. The proposed formalism and algorithm have been tested in a co-simulation testbed, and have also been implemented as a Self-Aware Monitor (SAM) tool. The SAM tool automatically generates the appropriate set of rules, based on the description of the topology of the local substation, and on the configuration of the controlling Remote Terminal Unit. Finally, a case study conducted at a substation of a Dutch distribution system operator has brought important insights about the feasibility of process-aware monitoring.

For several scenarios simulated in the testbed, our proposed new algorithm has been able to correctly identify unsafe states of the physical system upon sensor readings, as well predicting unsafe future states, in case of commands. The detected bad readings and malicious commands would not have been detected by a centralized system. Furthermore, the automatic generation of rules based on system topology and device configuration used in the SAM tool emphasized the necessity of keeping information about the system up to date. The tool reported problems that arose from outdated information. We conclude that the future of process-aware methods depends highly on the quality, freshness and availability of the process information. Current-day systems might not be ready for process-aware methods, as they are unable to provide the necessary information.

Samenvatting

SCADA-systemen sturen kritieke infrastructuren, zoals elektriciteitsnetwerken, aan en worden gebruikt om deze grootschalig te monitoren. Jarenlang werkten deze systemen met eigen protocollen, in een afgesloten en gecentraliseerde architectuur, zonder informatieverlies naar buiten.

Tegenwoordig worden SCADA-systemen in toenemende mate gekoppeld aan het internet om deze op afstand aan te sturen. De verbinding met het internet maakt het echter mogelijk om in te breken op deze netwerken en de infrastructuren die zij beheren te verstoren. Het beveiligen van deze SCADA-systemen heeft pas sinds kort prioriteit in het bedrijfsleven. Aanleiding hiervoor zijn incidenten zoals de aanval met Stuxnet-malware in 2010 met desastreuse gevolgen voor de getroffen kritieke infrastructuren. Dit was het eerst bekende incident waarbij een cyberaanval heeft geleid tot fysieke schade aan een systeem. Hierdoor zijn beheerders van SCADA-systemen bewuster geworden van de mogelijkheden die een aanvaller heeft zodra deze zich toegang heeft verschaft tot deze systemen.

Het monitoren van systemen is een gangbare manier om de activiteiten die op het systeem plaatsvinden in de gaten te houden. Beveiligingsmethoden voor gewone IT-netwerken zijn vaak niet geschikt voor SCADA-systemen. Monitoring van SCADA-systemen gaat er vaak vanuit dat het verkeer binnen het systeem stabiel en voorspelbaar is en werkt bijvoorbeeld op basis van zogenaamde whitelists. Hierbij wordt enkel communicatie tussen vertrouwde entiteiten binnen het systeem toegestaan. Andere technieken, zoals deep packet inspection, analyseren de inhoud van pakketten in realtime. Op basis hiervan kunnen maatregelen, zoals het sturen van een melding, genomen worden wanneer ongeautoriseerde berichten worden gedetecteerd. In de praktijk blijkt echter dat deze technieken niet effectief zijn: aanvallers zijn in staat kwaadaardige berichten vanuit een legitieme entiteit binnen het systeem te versturen. Daarmee blijven deze berichten onopgemerkt en kunnen bijvoorbeeld een black-out veroorzaken. Traditionele technieken laten het effect van de geïnspecteerde pakketten op het onderliggende fysieke systeem buiten beschouwing.

Dit proefschrift richt zich op het verbeteren van de beveiliging van SCADA-systemen voor elektriciteitsnetwerken. Dit wordt lokaal gerealiseerd door middel van een monitoring-tool die zich bewust is van het fysieke proces en een onveilige staat van het systeem kan detecteren. Door een model van het verdeelstation en

de staat van sensoren en aandrijvers bij te houden, kan de tool op verschillende verdeelstations worden ingezet.

Een nieuw en generiek modelformalisme wordt geïntroduceerd welke geschikt is om een (deel van een) elektriciteitsdistributiesysteem te beschrijven. Op basis van dit model is een algoritme ontwikkeld, welke in staat is een aantal fysieke wetten en veiligheidsvoorwaarden te toetsen die te allen tijde van toepassing dienen te zijn binnen het elektriciteitsnetwerk. De combinatie van formalisme en algoritme zijn geïmplementeerd in de zogenaamde SAM-tool (Self-Aware Monitor) en gevalideerd in een co-simulatie-framework. Op basis van de topologie van het verdeelstation en de configuratie van de gebruikte RTU, genereert de SAM-tool automatisch een geschikte set aan regels. Tot slot is een praktijkstudie uitgevoerd in een verdeelstation van een Nederlandse netbeheerder. Deze heeft tot belangrijke inzichten rondom de haalbaarheid van proces-bewust monitoren geleid.

Tijdens de simulatie van de verschillende scenario's is gebleken dat de voorgestelde aanpak in staat is om een onveilige staat van het fysieke systeem te identificeren op basis van gemeten en gecommuniceerde sensordata. Tevens is het mogelijk om een toekomstige onveilige staat te voorspellen op basis van commando's. De gedetecteerde onjuiste sensordata en kwaadaardige opdrachten zouden normaliter niet gedetecteerd worden bij een gecentraliseerd systeem. De praktijkstudie benadrukt het belang van actuele informatie: veel van de problemen die door de SAM-tool gerapporteerd zijn, bleken veelal te wijten aan het gebruik van gedateerde informatie. De toekomst van procesbewuste methoden is afhankelijk van de kwaliteit, actualiteit en beschikbaarheid van procesinformatie. Huidige systemen bieden vaak niet de benodigde informatie, waardoor deze mogelijkerwijs niet in staat zijn om te werken met procesbewuste methoden.

Contents

CHAPTER 1

Introduction

Safe and reliable critical infrastructures are the core of our modern society. In order to ensure their stable operation, they are continuously managed by dedicated control systems. Geographically distributed critical infrastructures, such as electricity distribution and transmission systems, are monitored and controlled by Supervisory Control and Data Acquisition (SCADA) systems. By allowing remote control of such systems, operators save time and companies save money when managing the power grid. However, at the same time, this introduces new opportunities for malicious parties to disrupt the controlled physical process.

In the past, SCADA systems used to operate in isolated networks, and implemented proprietary communication protocols and supervisory software solutions. Over the last decades, however, this has been changing. Firstly, standardized protocols are becoming more popular in order to ensure interoperability between different vendors. Secondly, commercial *off-the-shelf* solutions are replacing the proprietary software solutions in order to reduce the costs of maintenance and improve the reliability and quality of the operation. Finally, companies often use data collected by SCADA systems in their corporate network, however, the connection between the control and the corporate network is not always established in a secure way. Also, misconfigurations may result in devices being accessible to anyone on the Internet. In the Netherlands almost one thousand SCADA devices are accessible from the Internet, and many of them are susceptible to known, remotely exploitable vulnerabilities [127]. These vulnerabilities pose a significant threat to the security of the entire supply chain of an organization using such susceptible devices.

Even when the systems are isolated from the public Internet, adversary parties can use other techniques, such as spam campaigns, in order to infiltrate SCADA systems. Once inside the control network, an adversary can gain knowledge about that network [139] and the process [79]. Once hackers understand the controlled process, they can design attacks that disrupt its operation. Recent events confirm this: in 2010 the Stuxnet malware disrupted the operation of Iranian nuclear centrifuges [64], while in 2015 hackers in Ukraine were able to

disconnect houses of more than 225 000 customers from the power grid [4]. Reports show that breaches in the energy domain account for 20% of the reported cyber security incidents in 2016[1] [62]. Moreover, new hacking tools are being developed with the energy sector in mind [38, 61]. For example, CrashOverride [38] abuses vulnerabilities of protocols used in the energy sector. Since disruptions in delivering electric power directly affect other critical infrastructures, such as gas distribution and water treatment facilities [122], it is of utmost importance to improve the security of the power distribution and transmission systems.

Keeping SCADA systems secure is complicated. Security standards [66, 97, 109, 138] list best practices and guidelines for establishing a secure system. The basic objectives include restricting logical and physical access to the control network and assets, and protecting individual devices from exploitation. Moreover, a defence-in-depth strategy is recommended by implementing several layers of security mechanisms. These include developing security policies, encrypting the communication and stored data, disabling unused ports and services on devices, and restricting user privileges [138]. In this way, if one mechanism fails, another might be able to stop or detect the adversary.

Monitoring SCADA traffic provides insight into activities happening in the system. Modelling properties of the traffic during normal operation of the network, as well as generating signatures of known abuse patterns, are two main approaches for detecting intrusions in the network. Intrusion Detection Systems (IDSs) together with effective security policies are one of the recommended security strategies for SCADA systems [138]. However, a network IDS implementing, for example, whitelisting [11] or log-analysis [56] will not detect an attack performed by sending legitimate commands, from a legitimate source, that possibly results in disrupting the physical process. Another way to protect the inherently unsecure SCADA protocols is to encrypt the communication between SCADA sites [67, 106]. However, it has been shown that encrypting connections would not help most of the recorded attacks [43], as hackers usually compromise one of the communication endpoints. Therefore, it is important to secure these endpoints, and to investigate the network traffic at each side of the connection.

Power grid operators control the power grid by means of so-called Energy Management Systems (EMSs). An EMS uses information gathered by the SCADA system in order to calculate the current state of the managed power grid. This process, called State Estimation (SE), uses sensor measurements and physical properties of system components to estimate values of current and voltage on all power lines and buses, that is, also on the ones that are not directly monitored by sensors. To perform SE correctly, EMS relies on accurate data

[1] More recent ICS-CERT "Year in review" reports do not provide this information.

collected by the SCADA system, and false information can result in incorrect control decisions performed by the supervisory software or the operator. Various approaches have been reported that evaluate the sensor data on the central site in power transmission and distribution systems [58, 96, 121], possibly protecting the central SCADA system from false information from remote stations. However, the remote field stations in the power distribution system can also receive malicious information from the central SCADA server, as it happened in Ukraine in 2015 [4]. Currently, to the best of our knowledge, no protection mechanisms exist to avoid this.

The direct impact of the SCADA system on the physical process requires additional security methods specifically tailored to the safety requirements and physical constraints of the process. The concept of *process-aware* traffic monitoring has recently been investigated in the context of industrial control systems and electricity transmission and distribution systems [19, 47, 57, 93, 110]. Process-aware IDS techniques distinguish between learning-based [20, 57] and specification-based [7, 84, 94, 100, 111, 144] approaches. The latter either use static rules (for example, [111]) or dynamic rules (for example, [94, 144]) for detecting and/or preventing malicious commands. The specification-based approaches are closely related to the approach presented in this thesis. However, they can either *not* be used in the field stations [94], are able to detect but *not* prevent malicious commands [111, 144], or do *not* implement a dynamic policy depending on the system state [84]. These methods usually involve deep-packet inspection and processing of that content, for example, by comparing process variables to predefined thresholds ensuring system safety.

This thesis investigates the *concept of local process-aware SCADA traffic monitoring for power distribution systems* and discusses the feasibility of including physical process information into the detection method.

1.1 Goal and Research Questions

The goal of this thesis is to improve the safety of SCADA systems controlling electric power distribution by incorporating information about the physical system state in the detection process. At the same time, we do not advocate against using traditional network IDS. On the contrary, we believe that a process-aware system can be an excellent solution complementing the state-of-the-art conventional network monitoring methods. We therefore assume that the investigated systems *do implement* regular network IDS, and that the process-aware method investigated here serves as a defence-in-depth control. The proposed monitoring system has to be able to detect attacks that aim to disrupt the operation of the physical system.

Our objective therefore is to:

> *Design a process-aware monitoring system for power distribution, that detects when the physical process is in an unsafe state.*

This is achieved by addressing the following three research questions. First, we investigate the current state of security in power distribution systems, as follows:

> ***RQ1*** *– Where in the power distribution system is an extra layer of security needed? How can it be designed and implemented?*

To address this research question, we describe related work on power distribution and SCADA systems. In order to understand how to protect this critical infrastructure from process attacks, we analyse the known and reported incidents that happened in SCADA systems, and disrupted or aimed at disturbing the operation of the electrical power grid.

With knowledge of reported attacks on SCADA systems controlling power distribution infrastructure, we investigate which features of a physical system should be modelled in a process-aware monitoring tool. This brings about our second research question:

> ***RQ2*** *– Which aspects of the physical system state should be modelled in a local process-aware monitoring system?*

Not all attacks can be detected by analysing only features of communication exchange, as done by a conventional IDS. Also, not all process-aware methods are applicable in a local monitoring approach, for example, because they require knowledge of the entire system in order to perform calculations of the system state. Finally, we evaluate the practicality of the process-aware local monitoring, as follows:

> ***RQ3*** *– Is a local process-aware monitoring solution feasible to protect power distribution systems?*

The feasibility of the proposed local process-aware monitoring could be investigated in a dedicated testbed or in a real-life case study. When working with critical infrastructures, the latter is usually not possible, as failures during the tests can be devastating. For example, in power distribution systems, they can result in disconnecting part of a neighbourhood. The former testing approach has to be capable of demonstrating the operation of both a power distribution system and the network controlling that system, and of showing the interaction between these two. By modelling this interplay, it will be possible to understand the result of the cyber commands on the physical system.

1.2 Contributions

The main contribution of this thesis is the design and development of a local monitoring approach for process-aware intrusion detection for SCADA systems. To achieve this, the required and developed components are:

- A **modelling formalism** which can be used to formally describe a model of (a part of) a power distribution system. We study the limitations of currently used models and propose a new formalism that can be used to describe a locally controlled system.
- A **local monitoring** method that tests a set of physical constraints and safety requirements. The physical constraints are established by analysing physical laws that apply in a power distribution setting. The safety requirements are derived from standards and physical capacities or properties of the components of the system. If all rules in this defined set are met, the tested system is considered safe.
- A **testbed** co-simulating the operation of a power distribution system controlled by a SCADA system implementing the Modbus protocol [105]. The co-simulation testbed builds on top of the Mosaik framework [113] and allows for testing the proposed local monitoring approach on different topologies.
- The **SAM (Self-Aware Monitor) tool** that supports two popular SCADA protocols: IEC-104 [69] and Modbus. Given a topology, this tool automatically matches physical constraints and safety requirements that have to be tested in order to validate whether the system is in a safe state.
- A **parser for the IEC-104** protocol. Using the Spicy framework [136], we provide a parser for this popular protocol used in the electricity distribution sector. This parser can be used with the Zeek network monitor[2] [118] to implement security policies for IEC-104.
- **Real-life implementation insights**. A case study conducted at a substation of a Dutch operator of a power distribution system allowed us to test the SAM tool using a real traffic capture. This experiment brought important insights on data freshness and availability, which are relevant for future development of local monitoring methods.

1.3 Organization of the Thesis

This thesis is organized as illustrated in Figure 1.1. The three research questions, defined in Section 1.1, are addressed in different parts of this thesis. First,

[2]Zeek is the new name of the Bro network monitor since October 2018.

background and related work on monitoring electric power distribution systems is provided. To address the second research question, we study modelling formalisms that are capable of detecting process attacks. Finally, we investigate whether a local process-aware monitoring for SCADA systems is feasible in real infrastructures. Most chapters are based on previously published work, as indicated in the summary below.

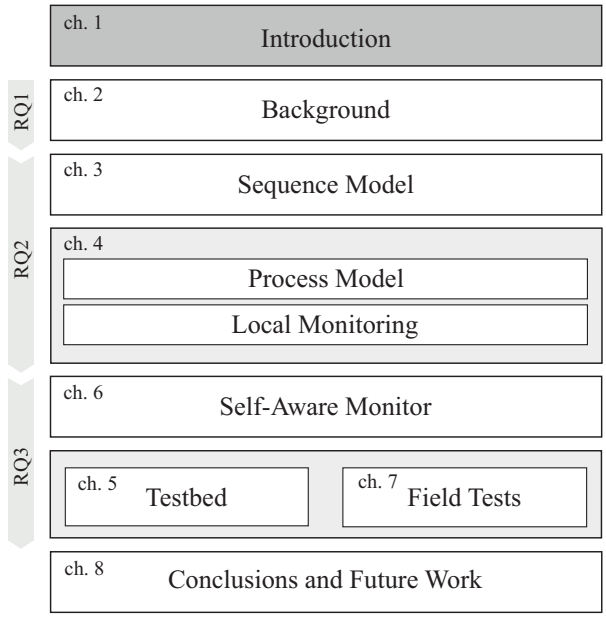


Figure 1.1: Organization of the thesis.

Chapter 2 describes background information necessary to understand the monitoring of the power distribution grid. Moreover, we investigate what attacks on the physical process are currently threatening power distribution systems. Finally, the state-of-the-art in process-aware intrusion detection methods is studied, to learn which threats to SCADA systems are currently possible to detect. Parts of this chapter are based on [28] and [32].

Chapter 3 investigates sequence attacks on the processes within power distribution systems. We model the communication exchange between devices in a SCADA system as discrete-time Markov chains and propose two methods for generating smaller models, that can still be used to detect attacks. However,

sequence attacks are only a small portion of possible process attacks, therefore, another modelling formalism is needed to detect them, as will be addressed in the remainder of this thesis. Chapter 3 is based on [44].

Chapter 4 introduces two components needed for local process-aware monitoring. First, a modelling formalism used to describe a part of a power distribution system is proposed. Next, a local monitoring algorithm interpreting the content of measurements and commands exchanged in a SCADA system is presented. Finally, examples that illustrate the use of this monitoring approach are provided. This chapter contains the main theoretical contribution and is based on [27] and [28].

Chapter 5 describes a co-simulation testbed that is used to validate the monitoring approach proposed in Chapter 4. The testbed builds on top of the Mosaik framework, and it integrates the simulation of a power distribution system, with a small SCADA system built of a single control server and a single supervised station. The server communicates with the station using the Modbus/TCP protocol. This chapter is based on [30], [31] and [32].

Chapter 6 presents the Self-Aware Monitor (SAM) - a tool which, given a topology of the system and a configuration of the controller supervising a substation, automatically matches the physical constraints and safety requirements that have to be evaluated in order to monitor the safety of that substation. This chapter contains the main practical contribution and is based on [45].

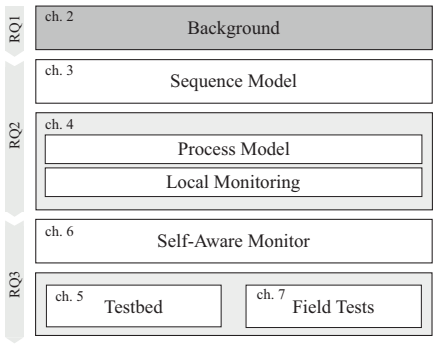
Chapter 7 analyses a case study which deploys the SAM tool in a real-life scenario. First, we provide a parser, developed using the Spicy framework. Then, we present four scenarios, where the system operator performed changes in the real system, by sending commands to the station. We provide an evaluation of those commands performed by the SAM tool. Figure 1.1 depicts this chapter together with Chapter 5 in a combined box, as these chapters both address the validation of the proposed monitoring approach. This chapter is based on [33] and [45].

Chapter 8 provides the overall conclusions from the research presented in this thesis. We also revisit the research questions and the goal stated in Section 1.1. Finally, we suggest possible directions for future work.

CHAPTER 2

Monitoring the Power Distribution

This chapter provides the general background and related work on the topic of power systems and on networks used for monitoring and control. The operation of power distribution systems is explained in detail, with a special focus on the interaction between the physical process and the control network. Security of SCADA systems is at the core of this thesis, hence, we outline security practices currently implemented in SCADA systems and discuss real-life incidents that have happened in power distribution systems. We review existing intrusion detection techniques that have been proposed for SCADA systems, to understand if they were capable of preventing the mentioned incidents. Developing dedicated intrusion detection methods for SCADA systems requires a good understanding of different concepts in place, as well as their interplay. Hence, this chapter aims at introducing the power distribution system, SCADA systems, threats to these systems and network traffic monitoring.



This chapter is organised as follows:

- *Section 2.1 explains the energy transition currently happening to electric power generation and distribution.*
- *Section 2.2 discusses components, the architecture, and characteristics of SCADA systems.*
- *Section 2.3 elaborates on the threats to SCADA systems, emphasizing what this means to the power distribution.*
- *Section 2.4 presents related work on Intrusion Detection Systems, with a focus on process-aware methods.*

Parts of this chapter have been based on [28, 32].

2.1 Electric Power Distribution

This section provides an informal description of the operation of electric power distribution systems. Section 2.1.1 describes the energy transition, which the modern power grid is facing before Section 2.1.2 explains the operation goals and priorities of distributing the electricity. Finally, Section 2.1.3 describes the mechanisms present to control the distribution of the electric power.

2.1.1 The Energy Transition

The main goal of the power grid is to ensure that generated electric power reaches its consumers and is of a good quality [146]. This is not a trivial task as today's power grid is built of complex systems of power lines, transformers, generators, switching and safety equipment that relies on a complex structure of embedded networks, sensors, optimization, communication and computation [101].

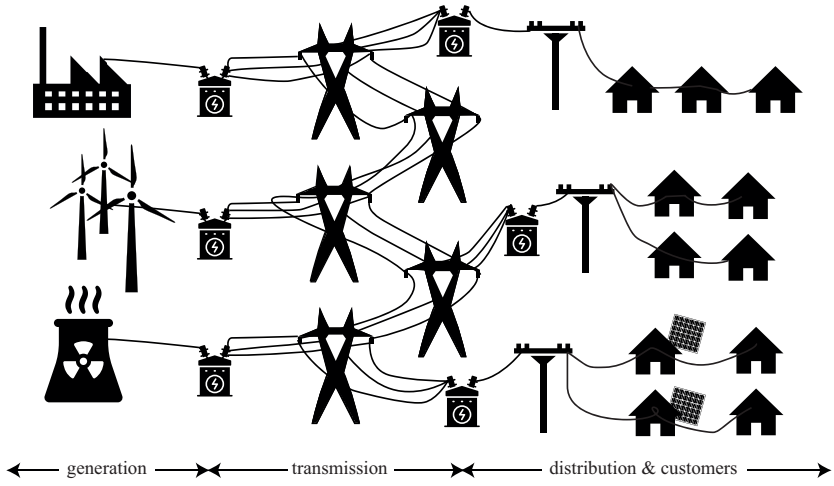


Figure 2.1: Electricity delivery stages in a traditional power grid

Traditionally, the power grid was functionally divided into three stages: generation, transmission, as well as distribution and customers [137], as illustrated in Figure 2.1. Power plants generate electric power from primary sources, such as coal and wind, at the **generation stage**, shown on the left side of the figure. The backbone of the power grid is the **transmission grid** built of large transmission poles that deliver electric power over long distances to so-called distribution substations. Step-up transformers are used to change the voltage

value from low to high, in order to reduce the power loss in the long-distance transmission. The transmission grid is maintained and managed by so-called Transmission System Operators (TSOs). At the **distribution stage**, illustrated on the right side of Figure 2.1, voltage is transformed from high to lower values using step-down transformers located in distribution substations. Moreover, these substations contain switching and control equipment that can be used to dis-/connect power lines. At the last stage, the electric power is transformed to the target voltage level and delivered to customers [146]. These final transformers are located in small so-called field stations. The distribution grid is maintained and managed by so-called Distribution System Operators (DSOs).

We are, however, currently facing a rapid change in the power grid. The integration of renewable Distributed Energy Resources (DERs) is a major target of the European Union's energy and climate policy objectives for 2020 and beyond [40]. The number of the DERs, such as photovoltaic panels, used in households, is growing. Just in the Netherlands, the total amount of electricity produced by photovoltaic panels has increased from 37 GWh/year to 1559 GWh/year in only 10 years [22]. This growth is affecting the whole power grid infrastructure: the traditional hierarchical, one-way flow is replaced by the distributed, two-way flow of electricity sketched in Figure 2.2.

To enable this change, and at the same time still be able to provide the required quality of the electric power and control the grid, TSOs and DSOs are busy modernising and automatizing the distribution substations [146].

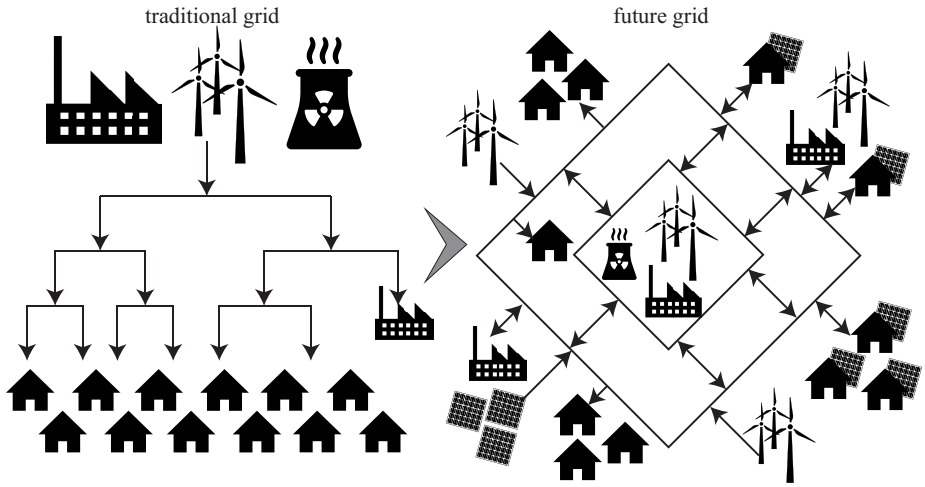


Figure 2.2: Electricity delivery stages in the future power grid, based on [88]

Especially the electric power distribution connecting the neighbourhoods of consumers to the transmission grid is affected by the mentioned changes. An example of a power grid distribution system is shown in Figure 2.3. It is a system built of power lines, bus bars, switching equipment, safety equipment, transformers, and power source(s) and consumers. Section 4.3 provides a more formal and more detailed definition of the power distribution system and the components used. Formerly, the distribution and field stations operating at medium and low voltage were not monitored by the DSOs, and all switching in these stations was performed manually. Currently, with the transition of the power grid, automating the process of switching and monitoring these stations is necessary [12, 34, 98]. Remote control at the distribution and field station allows for many of the (future) smart grid's principles: using renewable DER, self-healing, enabling participation by consumers, protection against physical and cyber-attacks, power quality, adapting all generation and storage options, enabling new products, services and markets, as well as performance optimisation [65].

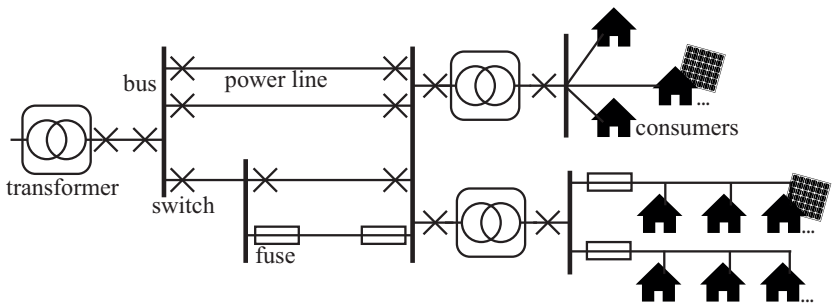


Figure 2.3: Example of a power distribution system topology

Modernisation of the medium and low voltage substations at the distribution stage allows for more detailed monitoring and remote control of the connected devices by means of a control network. Examples of decentral control mechanisms in the distribution grid are: switching power lines for maintenance; voltage regulation using Load Ratio Control Transformers (LRTs) [72]; or reducing the peak demand with Conservation Voltage Reduction (CVR) [135]. Moreover, as the generation is becoming more popular in the distribution grid, the control mechanisms of the generation, for example, Automatic Voltage Regulator, a control loop used to regulate voltage, will also appear in the distribution in the future.

2.1.2 Quality, Availability and Safety

The delivered power must meet some specific requirements, for example, the generated power must equal the power consumed at any time; in Europe, the voltage level at the customer side has to be equal to $230V \pm 10\%$ [24]; the *mains frequency* has to stay within $50Hz \pm 0.4\%$. All entities controlling the power grid (for example, TSOs and DSOs) also ensure that a certain power **quality**, defined by, for example, voltage fluctuations such as the "flicker", frequency variations and the waveform, is provided. The quality of the supplied voltage may vary due to phase shifts, variations in voltage and/or current magnitude, and voltage unbalance [14].

Availability is the probability that customers are energized in the power distribution, that is, that they are connected and provided with electric energy [15]. Depending on the annual total time of lack of power, availability is defined as the total uptime per year divided by 8760h (one year). Because of sensitive equipment, many manufacturing plants require an availability of 99.9999%, which translates to only 31.5 seconds of downtime per year. The availability is influenced by outages, faults, open circuits, and customer interruptions. In normal operating conditions, all the customers and the power distribution equipment are energized (unless redundant). Events disrupting that state may lead to outages and interruptions.

Safety is defined as maintaining and/or achieving a safe state of a process is done by means of a safety system [125]. A safety system performs specific control functions that ensure a safe operation of a process, and acts when some of the predefined conditions are violated. The goal of such a system is that the process does not endanger people's lives or harm the environment. SCADA systems (explained in Section 2.2) often provide functionalities of a safety system.

2.1.3 Energy Management System

In order to fulfil the principles of the future smart grid, for example, self-healing, active participation by consumers, power quality, and performance optimization, process operators require a good real-time overview of the physical system. Such an overview helps them in their decision making, provides real-time performance optimization, quick outage/restoration management, gives numerical evidence on the basis of which they can perform any actions and be warned about emergency situations.

Some corrections of the electric power quality are done locally by various control loops, for example, at the generation side, the Automatic Voltage Regulator regulates the amount of reactive power that is injected into or absorbed from the system. Moreover, a centrally-located Energy Management System (EMS) performs the optimisation functions and based on their outcome it sends proper

commands to the controlled grid elements. The EMS processes sensor measurements from field stations and performs State Estimation (SE) [96, 137, 155]: using a model of the power system and knowledge about the physical process, the EMS is able to calculate the state that the power system is in. This state is defined by the values of voltage magnitudes and relative phase angles at the system nodes. Moreover, the EMS optimises, supervises and controls the power grid, and with a sufficient amount of data it can detect if a sensor is faulty using Bad Data Detection (BDD) [96, 140]. The measurements from the sensors can deviate from the truth due to errors, noise or cyber attacks [58]. BDD algorithms, performed at the end of the state estimation process, can detect such outliers in the measurement data [141]. For these algorithms to work, high redundancy of measurements of the system is needed. Additionally, using SE the EMS performs Contingency Analysis (CA), which predicts the most severe consequences of a system breakdown, given the current state of the system.

The functionality of the EMS and the control network (see Section 2.2) are overlapping: the control and monitoring parts are the same, but the EMS also has analysis and optimisation capabilities [155]. Note that, an EMS uses the control network to obtain the data, but performs all the calculations *centrally* in the control room.

2.2 SCADA Systems

Supervisory Control and Data Acquisition (SCADA) systems are a type of Industrial Control Systems (ICS) that monitor and control geographically distributed physical processes in a timely manner. They are widely utilized in critical infrastructures, for example, in gas distribution, water treatment and distribution, and energy transmission and distribution, but also in industrial and facility processes, for example, in manufacturing, process control and building automation systems [138]. Section 2.2.1 describes the SCADA architecture and components before Section 2.2.2 compares the SCADA and regular IT networks. Section 2.2.3 provides an overview on SCADA protocols. Finally, Section 2.2.4 highlights the security issues of SCADA systems controlling the power distribution grid.

2.2.1 SCADA Components and Architecture

As SCADA systems were organically growing and changing over the years and no substantial changes were done to these system, there is no typical architecture of a SCADA network [132, 138]. A conceptual picture of a SCADA deployment is shown in Figure 2.4. The SCADA system itself consists of a control network, communication link(s) and field stations, shown in the bottom part of Figure 2.4.

In the upper part of the figure, the SCADA system is connected to the corporate network, which most often is also connected to the Internet. The connection between the SCADA system and the corporate IT systems is becoming more common for multiple reasons: (i) the demand for remote access is increasing, (ii) there is a need to monitor the system outside of the control network, and (iii) operators of a company, who usually reside in the corporate network, need to obtain critical data from the control network on a regular basis [138].

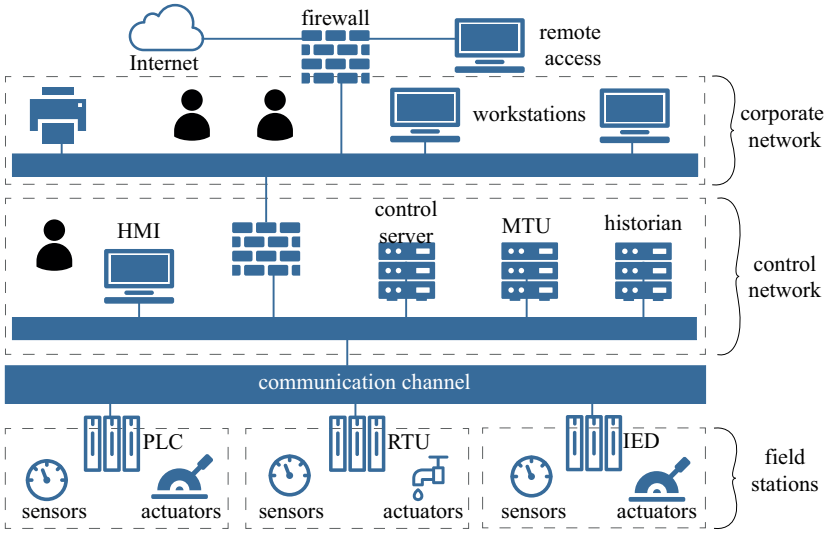


Figure 2.4: Basic topology of a SCADA system

SCADA makes use of some specific hardware. In the **control room** the following control components can be found:

- A **control server** (data acquisition server) hosts the supervisory control software, such as the EMS discussed in Section 2.1.3, that communicates with lower-level control devices.
- The **Master Terminal Unit** (MTU) is the master device in a SCADA system that polls the remote devices, such as RTUs and PLCs located in the field stations, for information, and transmits the control signals.
- A **Human Machine Interface** (HMI) is a system that provides the human operators with an overview of the current state of the controlled process [46]. The HMI also allows to modify control settings of the system, manually override the control operations, for example, in case of an emergency, and to configure set points and control algorithms of the elements of the system. Moreover, the HMI provides historical data about

the process, generates reports, and displays current information, to the authorized users. The form of HMI can vary: on the one hand it can be accessible only in the control room, on the other hand it can be accessible over a web browser on any system with Internet connection [138].
- The **historian** is a centralized database that logs the process information of the entire system. This information can be accessed to perform, for example, optimization and statistical analyses.

In the **field stations** the following components can be found:
- A **Remote Terminal Unit** (RTU) is a data acquisition and control unit, that communicates with the MTU: it transmits the local measurement information to the master unit, and it controls the locally connected objects by executing the commands it received from the master unit.
- A **Programmable Logic Controller** (PLC) is an industrial computer that performs internal logic and control functions that is later executed by electrical hardware, such as switches. Sometimes, in the field stations, a PLC serves as an RTU, that is, it can be polled by the master unit for measurements, and it also executes the commands coming from the master unit. Modern PLCs are capable of controlling complex processes.
- An **Intelligent Electronic Device** (IED) is an advanced controller that contains sensors, actuators, and the intelligence to acquire data and perform local processing and control. Also, it can communicate with other devices, for example, to notify about actions taken or share measurements. In power distribution, examples of IEDs are protective relays that protect the power lines from overcurrent and "*On-Load Tap Changer*" controllers that change the tap switch position of a transformer, if the secondary voltage leaves its bounds.

Field stations are connected with the control room via communication channels, for example, via leased telephone lines, cellular networks, WAN, or radio.

2.2.2 Difference from IT Systems

Although technologically SCADA systems are becoming more similar to regular IT systems [55, Section 2.2], there are many differences between the two. Table 2.1 summarises the differences between SCADA and IT systems.

The main distinction is the fact that regular IT systems operate with plenty of human-generated data and their priority is ensuring that the information is accessed and/or modified by the authorized users. SCADA systems, on the other hand, operate with machine-generated traffic and have a safe, functioning process as their priority. The lifetime of the components in SCADA is usually 15 to 20 years. This is significantly longer than for regular IT systems, where the lifetime of a component equals 3 to 5 years. SCADA components are much

Table 2.1: Summary of the differences between SCADA and IT systems

Aspect	IT	SCADA
Component lifetime	3-5 years	15-20 years
Main priority	Authorized information processing	Safety and functionality of the process
Data source	Human-generated	Machine-generated
Information requirements	Information confidentiality and integrity	Information availability
Time-criticality	Low, delay is acceptable	High, delay is not acceptable
Interaction with physical process	No	Yes

more difficult to maintain: any updates to the software have to be scheduled long in advance and should happen rarely, while IT components are updated on a regular basis. Policies in regular IT networks are designed according to the CIA triad: Confidentiality (providing information to authorized users) and Integrity (ensuring that the information is altered only by authorized users) of the system are the main concerns, while requirements for Availability (keeping information available to users) is less strict [138]. In contrast, in a SCADA system, the Availability of data and commands is crucial, as it ensures the proper operation and ensures the safety and functionality of the system [2]. SCADA systems are usually time-critical, therefore, such a system has rigid delay and jitter constraints. A delayed reaction could result in bringing the system to an unsafe state and even threaten human lives [159]. On the other hand, regular IT networks usually tolerate some levels of delay.

One paramount difference between SCADA amd IT systems is the fact that SCADA systems interact with the physical process. A regular IT system does not interact with physical assets, while executing the decisions of a control server of a SCADA system has a direct impact on the physical system [159]. It interacts with the physical infrastructure using:

- **sensors**, for example, voltage meters, pressure meters, thermometers,
- **controllers**, for example, a logical program, either local (IED) or central (EMS),
- **actuators**, for example, switches, valves, executing the requested changes.

The three elements listed above interact as follows. Sensors provide the information about the state of the physical system to controllers. The controllers process that information and, if necessary, send a command to change something in the physical system. The change is executed by the actuators.

2.2.3 SCADA Communication and Protocols

The SCADA system monitors and controls premises that can be geographically distributed, as depicted in Figure 2.4. The control room exchanges information with the field stations via communication channel(s), which can implement various technologies: wire, radio, fiber optic, cellular networks, etc. [138]. Most of the SCADA communication relies on the control message exchange between the master and slave devices [70].

For the SCADA elements to communicate, the devices need to use a communication protocol. Note that, SCADA protocols are not intrinsically secure. When first developing these protocols, the goal was to provide functionality and good performance, while network security was hardly ever a concern [70]. In the past decades, SCADA systems were using proprietary protocols, which made it difficult to integrate them with other systems. Next to that, this obscureness also gave a (false) sense of security, as the protocols were not publicly known. Also, as the SCADA and IT systems were not physically connected to each other, this so-called "air gap" was considered to be a sufficient security precaution. Therefore, SCADA communication protocols were not developed with security measures in mind. Today, protocols are open and standardized in order to enable easier and more efficient communication between various equipment vendors and operators. This standardization eliminates the sense of "security by obscurity" [108]. Below, the most common protocols used in the power system domain are discussed.

Modbus/TCP

One of the widely-used protocols to connect the remote RTUs to a central supervisory computer is Modbus/TCP [80]. Although Modbus is a generally accepted industrial process standard, especially popular in the oil and gas sector, it also plays an important role in power distribution [16, 77]. It is a master/slave type of protocol, where only one of the communicating devices, called master (or "client"), can initiate the communication. The slave (or "server") continuously listens for incoming connections on TCP port 502. Modbus stores either 1 bit values (so-called *coils*) or 1 byte values (so-called *registers*). Both coils and registers can be either read-only values (*discrete inputs* and *input registers*, respectively) or read/write values (*coils* or *holding registers*, respectively). In order to allow for, for example, floating point variables, some vendors allow for combining registers to hold 32-bit and 64-bit values [57].

Security extensions for Modbus/TCP protocol have been proposed [41, 46, 134], which, however, do require changes on the protocol level of operating devices. This is expected to be difficult as companies are reluctant to such changes and global standardization. Without a uniform standard, the proposed

approaches may be incompatible with existing systems. Recently, a Modbus Security standard was proposed [106], which introduces Transport Layer Security (TLS) to the traditional Modbus protocol. TLS encapsulates Modbus packets to provide authentication and message integrity. Detailed information about the Modbus protocol can be found in Appendix B.1.

IEC-60870-5-104

IEC-60870-5-104 is one of the most common protocols in the domain of electrical engineering and power systems in Europe and North Africa [35]. Among others, it is the protocol used in the Netherlands for communication between the distribution (field) stations and the control room in the power distribution. IEC-60870-5-104 was developed by the International Electrotechnical Commission (IEC) as part of the "IEC 870 Telecontrol equipment and systems" standard. It describes a set of open transmission protocols for SCADA systems in the domain of electric engineering. The first companion standard IEC-60870-5-101 defines all functionality and data objects that are necessary for telecontrol applications over wide areas, such as communication between electrical control station and substation systems. IEC-60870-5-104 extends this standard to be used over TCP/IP.

Every IEC-60870-5-104 packet, a so-called Application Protocol Data Unit (APDU), contains a header called Application Protocol Control Information (APCI). S-frames (for numbered supervisory functions) and U-frames (for unnumbered control functions) are only built from the APCI. I-frames (used for information transfer), consist additionally of Application Service Data Units (ASDUs). ASDUs determine what kind of function (the so-called Type ID) they carry, and they can contain up to 127 Information Objects (IOs), referring to different addresses on the RTU that is being controlled.

As opposed to Modbus, IEC-60870-5-104 has more modes of operation. The first one, as in Modbus, is the request/response mode. In the second mode, the field devices send some information *periodically* to the control server, without being polled. The third mode of operation works *asynchronously*: the field devices send information once a certain condition is met, for example, if some process variable changes its value significantly, a notification is sent to the control server immediately. More detailed information about IEC-60870-5-104 can be found in Appendix B.2.

Other SCADA Protocols Used in Power Grids

Several other protocols are used in power distribution systems. **DNP3** (Distributed Network Protocol) is often used in the power systems domain in North

America, South America and Asia for communication between the SCADA control room and RTUs. In Europe, DNP3 is used for other critical infrastructures like oil, gas and water distribution and sewage treatment [35]. DNP3 was developed as an alternative to the IEC 60870-5 standards while they were still under development, and provide similar functionalities.

IEC 61850 is an international standard that defines protocols for substation automation, for example, for IEDs, such as protective relays in electrical substations. Examples of IEC 61850 implementation are MMS (Manufacturing Message Specification) or GOOSE (Generic Object Oriented Substation Event).

Open Platform Communications (**OPC**) is used across numerous ICS industries as a translator between various protocols. As ICS systems, even in a single substation, use different protocols implemented by different vendors, OPC unifies data in order to display this information on a dedicated OPC server and HMI.

Table 2.2 lists and compares the above listed protocols. Although security standard IEC 62351 proposes TLS encryption for, for example, IEC 61850 protocols (such as MMS and GOOSE), DNP3, and generally - any profiles including TCP/IP, the implementation of this may vary between vendors and might not always work. Therefore, we only consider "built-in" security of the discussed protocols in Table 2.2.

2.2.4 SCADA Security

Initially, SCADA systems were isolated and running proprietary protocols on specialized hardware. These systems were protected mostly against human errors and accidents, while the physical isolation and the earlier mentioned "security by obscurity" kept them relatively secure. Nowadays, using TCP/IP devices [46], commercial *of-the-shelf* solutions [70] and standardised protocols [158] make cyber attacks on such systems more feasible. SCADA systems are also implementing other IT solutions to provide remote access to the controlled assets, increasing the attack surface on such systems [70].

Traditionally, **cyber security** is considered only in an information security context [51]. Therefore, the classic definition of cyber security refers to the earlier mentioned CIA-triad: it is defined as protecting the Confidentiality, Integrity and Availability of the information exchanged in the network. This definition, however, does not consider the priorities of a SCADA system, which is the safety and the functionality of the process (see also Section 2.2.2). Therefore, the security objective of a SCADA system is *providing safe and reliable physical operations by assuring the correct and authorized control of physical and cyber assets* [51].

Table 2.2: Comparison of protocols used in SCADA systems

Protocol	Location	Function	Security	Operation
Modbus/ TCP	worldwide; different sectors (such as gas, power domain)	communication of devices on the same network, for example, between an RTU and a control server	not built-in	request-response, broadcast
IEC-60870	power systems in Europe and North Africa	communication between RTU and control server	not built-in	request-response, periodic updates, asynchronous updates
DNP3	power systems domain in America and Asia, other domains in Europe	communication between RTU and control server	not built-in	requests polled by the master, different frequencies for events and static data, asynchronous reports from the slaves, unsolicited integrity polls
IEC 61850	substation automation worldwide	communication protocols for IEDs in the electrical substations	not built-in[a]	request-response, asynchronous reporting, periodic reporting
OPC	translation protocol worldwide	OPC is an interoperability standard: OPC server converts SCADA protocols used, for example, by a PLC into the OPC protocol	n/a	request-response

[a]MMS has possibility of authentication, which is not widely supported, and the passwords are exchanged as plain text

2.3 SCADA Threats

In Section 2.1 we showed that the extensive deployment of IT assets in power transmission and distribution systems allows for remote control of often vulnerable devices [70, 97]. At the same time, as emphasized in Section 2.2, the integration of control networks with corporate networks potentially increases the accessibility of these vulnerable devices to anyone connected to the Internet. Section 2.3.1 provides an overview of the reported incidents in the power systems domain before the more general emerging threats to the power distribution are discussed in Section 2.3.2. Finally, Section 2.3.3 defines the threat model considered in this thesis.

2.3.1 Known Incidents on the Power Grid

Information about incidents in critical infrastructures, such as in the power systems domain, is not always shared, as it often contains sensitive data. Therefore, it is difficult to estimate the exact number of cyber incidents on the power grid. This section presents a set of chosen reported incidents of cyber attacks on the power grid.

Slammer at Davis-Besse

In 2003, a *slammer* worm infected over 75000 machines. This malware exploited a vulnerability of Microsoft SQL causing network outages. The most serious victim of this incident was the nuclear power plant in Davis-Besse in Ohio in the United States, where the worm bypassed the firewall between the corporate and the control network. The worm caused a Denial-of-Service of the safety-related system for almost 5 hours, and of the process computer of the plant for more than 6 hours [123]. Fortunately, the operators did not lose control over the power plant, since the plant control and protection functions were not affected.

Dragonfly/HAVEX

The Dragonfly campaign has been active since December 2015 [139]. It is an espionage campaign targeting multiple Industrial Control Systems in the United States, Turkey, and Switzerland. It is estimated that it affected over 2,000 sites, with a large emphasis on electric power systems and petrochemical systems [38, 139]. This campaign used the HAVEX malware that was distributed using phishing emails. HAVEX malware exploited the OPC protocol to map out all the devices in the ICS network.

Although the Dragonfly campaign was purely used for espionage and caused no physical harm, it provided information for designing future attacks.

BlackEnergy 1-3

The BlackEnergy malware has become one of the most sophisticated and modular malware targeting critical infrastructures [79]. BlackEnergy 2 targeted, among others, the Internet-connected HMIs of the ICS. It contained exploits specific for the HMI applications that use Siemens Simatic, GE Cimplicity, and Advantech WebAccess [38].

Although targeting HMIs does not directly cause physical damage, it is a useful espionage tool. The attackers are able to learn about the industrial process, and obtain a graphical representation of the SCADA system components.

Cyber-Attack Against Ukrainian Critical Infrastructure

On December 23, 2015, several Ukrainian power companies experienced unexpected and unscheduled power outages which affected more than 225,000 customers [63]. The attack was carefully planned and well coordinated. The BlackEnergy 3 malware was used to gain access to corporate networks of power companies and to connect to the SCADA networks [38]. The attackers were able to blind system dispatchers, make undesireable changes to the state of the power distribution system, and delay the restoration by wiping the SCADA servers controlling this power distribution system using KillDisk malware [4]. The distribution system operators were left without automated control for up to a year in some of the locations [38]. The attackers most likely had obtained legitimate credentials prior to performing any changes in the power system. At the last stage of the cyber attacks, the attackers either used remote ICS client software connected to the power distribution companies through a VPN, or used existing remote administration tools to remotely change the status of the circuit breakers and disconnect multiple power lines.

Crash Override (Industroyer) Malware and Kiev Incident

On December 17, 2016, almost one year after the previous ukrainian incident, the Crash Override malware was used to affect a single transmission level substation [38, 53, 157]. As a result, part of Kiev was left without electricity for about an hour. This malware was designed to learn and codify knowledge about the process at hand, in order to disrupt the physical process, similarly to the well-known Stuxnet malware [38, 42, 64]. It used techniques employed in the previous incidents, for example, the HAVEX malware OPC protocol mapping was used. As compared to the previous incident, the Industroyer malware campaign was done in an automated way, no manual connections were necessary.

Industroyer malware targets the ICS protocols IEC-60870-5-101, IEC-60870-5-104, and IEC 61850, that are, as mentioned in Section 2.2.3, popular in Eu-

rope [61]. However, this malware can easily be extended to also target other protocols [38]. The implementation used during the Kiev incident was capable of:

- Issuing valid commands to the RTUs over SCADA protocols.
- Denying service to local serial ports. This prevents the legitimate connections over serial connection to the affected devices.
- Scanning the SCADA environment and using the obtained knowledge to increase the success of other malware's tasks.
- Possibly exploiting known vulnerabilities, like the Siemens relay DoS.
- Wiping the Windows systems platform.

2.3.2 Threats to the Power Grid

The previous section lists reported incidents on cyber intrusions in critical infrastructures. From their descriptions it is clear that multiple threats constitute a single successful attack. This section systematizes these threats.

Even if deploying security standards, operators cannot protect field stations from malicious commands sent from the control room by, for example, a disgruntled employee [39, 50], or by accident [50]. This type of so-called *insider attacks* constitute the majority of targeted computer attacks reported in SCADA systems [18, 108]. For example, in 2000 in Maroochy Shire, Australia, a disgruntled ex-employee hacked into a water control system and flooded the nearby terrains with millions of liters of sewage [107].

SCADA systems are also abused by *outsiders* [50]. By hijacking a session, attackers are able to display a fake picture of the system state to the operator, or even reverse the semantic meaning of operator's actions, while presenting a consistent picture to the operators [82]. Stuxnet is a complex malware designed to change values of data sent and received by PLCs. It was most likely introduced to the target environment of Iranian's nuclear facility by an unaware insider or by a third party contractor [64]. By spreading malware within operators' networks, hackers are able to maintain a connection within these networks and take control over remotely accessible devices [63].

The increasing complexity of the power grid potentially introduces new vulnerabilities, increases exposure to attackers and creates new possibilities for unintentional errors [109]. There are many threats to the continuous operation of the power grid. An overview of cyber threats that are relevant in the power grid context is listed below (based on [97]). Figure 2.5 visualises the location of some of the threats in a controlled power distribution.

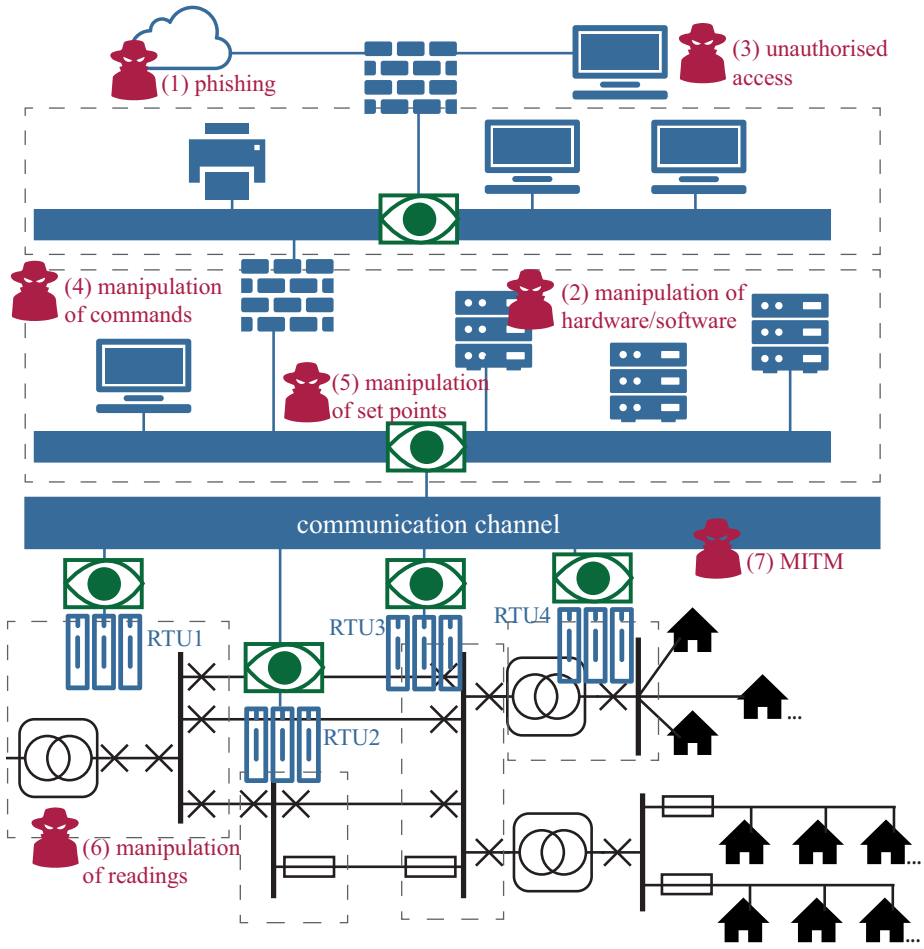


Figure 2.5: Visualisation of the threat location in power distribution. Black colour represents the (physical) power distribution system components, blue elements refer to the SCADA system, red colour illustrates the system threats, and green colour denotes the possible monitoring tool items.

- **Damage, Loss of IT assets.** Attacks of this type aim to *acquire sensitive information* from various power system control components, for example, about the energy consumption, access control data, credit card information.

- **Nefarious Activity, Abuse (IT assets).** These are attacks with the intention to deliberately interfere with the information systems:
  - *denial of service* [137] attacks aim to make the service unavailable to their intended users. For SCADA this is especially dangerous, as the operation relies on the availability of information.
  - *phishing emails* - personnel of power grid companies may receive an unsolicited email that can later infect their machines with malicious software. Phishing campaigns originate from outside of the company's network (see 1 in Figure 2.5) and aim at providing access to the corporate network of the company.
  - *malicious code/activity* attacks aim to disrupt or manipulate the operation of both IT and SCADA components. Once the malicious software is inside a company network (for example, brought on a USB stick or through phishing emails), it can distribute itself using the local network.
  - *manipulation of hardware/software* - an attacker can change the firmware of power grid components, for example, smart meters, field controllers [103], or devices within the control network (see 2 in Figure 2.5). Such a modified firmware could ignore the messages from the control room or send false information.
  - *unauthorized access to a network/system* can be performed using remote access to the internal systems, or poorly configured endpoints, for example, the customer AMI endpoint, or equipment located in the remote field stations (see 3 in Figure 2.5).
  - *manipulation of information* threat aims at adjusting data sent to/from various power grid components. For example, an attacker can send information to the control room that looks valid and harmless; information sent to an RTU located in the field station may affect its operation to perform malicious actions (for example, opening a power line when it should not be opened). The manipulated data can refer to commands, set points, and sensor readings, what in Figure 2.5 is illustrated with 4, 5 and 6, respectively.
- **Eavesdropping, Interception, Hijacking.** These types of attacks allow for undesired communication between the hacker and a device:
  - *man-in-the-middle (MITM) attacks*: the hacker gains control over the communication channel (see 7 in Figure 2.5) and is able to relay all the communication exchanged between two devices. While the

messages captured by the attacker can be altered, the communicating devices are convinced they communicate directly [102]. In these attacks, information is altered or new communication is injected, for example, like in *false data stealth attacks* [96, 140]. Depending on the topology of the grid, the attacker does not even need to have the knowledge of the entire system to perform this attack [121].
  - *interception of information* is possible to perform, for example, through side-channel attacks, or by intercepting messages from WiFi or ZigBee networks.
  - *message replay*, for example, sending false acknowledgement messages to the sender even when the recipient has not received a message.
  - *reconnaissance and information gathering* by the earlier mentioned MITM and interception attacks, also by performing port scanning of a system, an attacker is gathering information about the network, to later misuse.

- **Deliberate physical attacks, outages**, which result in physical damage to the system:
  - *damaging the equipment* by means of cyber commands, for example, damaging a generator by connecting and disconnecting it from the grid [156], or overloading power lines by disconnecting crucial power lines, and changing the data set points of devices [159].
  - *loss of electricity* by disconnecting part of the grid, for example, by sending an improper message causing unauthorized breaker operations [60]. An outage may be generated by switching or by damaging the devices. An example of the former is the Ukrainian grid hack [63].

Not all parts of the smart grid are susceptible to all types of threats, for example, it is more difficult to destroy the integrity of the information in a system with many sensors, as this can be detected by the Bad Data Detection. Moreover, since different operators and providers are on different levels of implementing smart grid ideas, they may be more or less susceptible to an attack.

2.3.3 Threat Model

The threats considered in this thesis are cyber-enabled physical attacks and outages. These attacks can originate either in the control room or in the communication channel, and aim to disrupt the process controlled by remote stations.

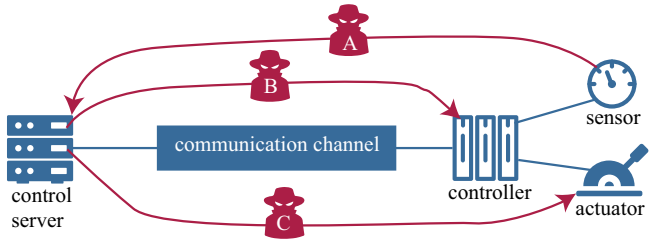


Figure 2.6: Threat model considered in this thesis.

Note that in order to perform these types of attacks, the hacker can use other threat types from this list: MITM or manipulation of hardware/software can be used in order to manipulate the information (readings, set points, commands), eventually bringing the system to an unsafe state. However, the proposed approach cannot detect a MITM attack that does not harm the physical process.

The considered attacks are based on the physical attacks described by Zhu and others [158]. They are depicted in Figure 2.6, marked with letters A-C:

A Manipulation of sensors readings (between the sensors in the field station and the control server);

B Manipulation of set point configuration of the devices in the field stations (between the control server and the controller in the field station);

C Manipulation of issued commands (between the control server and the actuator in the field station).

2.4 Intrusion Detection for SCADA

Monitoring and intrusion detection systems (IDS) are security services tracking and analysing system events in order to detect, and provide a warning about unauthorised access to system resources in real-time [66]. These systems are considered to provide the basis for security measures of a cyber-physical system [138]. Unfortunately, most commercial IDSs are not capable of monitoring SCADA protocols for suspicious behaviour [70]. Some basic functionality, signatures and patterns may exist for the most popular SCADA protocols, however, the less common ones are hard to address. This section provides an overview on the state of the art of intrusion detection in SCADA systems. Section 2.4.1 systematizes the taxonomy related to the IDS, Section 2.4.2 briefly mentions related work on classic SCADA IDS, and Section 2.4.3 provides an extensive literature review on related work in the area of process-aware intrusion detection.

2.4.1 IDS Classification

There are many approaches for intrusion detection. Depending on, for example, the detection technique, audited material, or the notion of state, the IDSs differ in their detection performance and ability to detect new attacks. In this section, we discuss the taxonomy of SCADA IDS.

Detection Techniques

We distinguish three classes of detection techniques, as follows:

- **Signature-based** (or knowledge-based) detection techniques aim at matching a pattern, that is *known* to represent a misuse [158]. For example, a certain byte sequence in the network traffic can represent malware instructions. Such a pattern can be included as a rule in an IDS and, regardless of the current system state and the current behaviour of the system, every time this pattern is matched, it is marked as an intrusion. Unfortunately, these types of attacks can only be detected if they have been observed before, and the pattern of misuse is included in the IDS. This approach, given a database with well-defined signatures of attacks, achieves a high detection rate and low false alarm rate. However, for new attacks, that have not been seen before, this system is of little use.

- **Anomaly-based** (or behaviour-based) detection techniques search for irregularities in the behaviour of the SCADA system as compared to the normal behaviour [95]. It builds a model based on the previously collected normal behaviour of the system, typically using machine-learning techniques. The behaviour is a description of the traffic, and could refer to the order of the packets, the list of the communicating pairs, the application-layer content of the packets, etc.. The IDS alerts about any behaviour that is "extremely unusual" compared to the learned behaviour [158], for example, a different order of packets, a new communicating pair or content that was not seen before. In this way, also yet unknown new attacks can be detected. However, if the network traffic is not predictive by default (for example, like in regular IT networks), it will result in a high rate of false positives. Moreover, the traffic used for training has to be attack-free.

- **Specification-based** detection techniques use the specification of the system, such as the documentation, to define what constitutes good behaviour of the system. Any deviations from that definition are marked as violations. Specification-based detection does not require training, unlike for

the anomaly-based IDS. It is more accurate than anomaly-based IDS, however, it is not as good at detecting new types of attack as the anomaly approach.

Table 2.3: Comparison of different intrusion detection techniques

Detection technique	Input	Detection performance	Attacks detected	New attacks
Signature-based	Signatures of the misuse	High detection rate, low false alarm rate	Known	No
Anomaly-based	Normal behaviour	Possible false positives	Deviations from normal behaviour	Yes
Specification-based	Model of the system	Good detection rate, low false alarm rate	Violations of the model	Possible

A combination of the above approaches is also possible, for example, specification-based anomaly detection [133]. As some attacks steer the system in such way, that its state stays within the description provided by the specification, additional monitoring for behaviour that is out of normal model could enhance the detection capabilities. Table 2.3 summarizes the characteristics of the detection techniques.

Audited Material, Granularity and Location

Regarding **audit material**, two approaches for collecting data exist: (i) host-based, and (ii) network-based [104]. In a **host-based** approach, analysis is based mostly on the logs of single devices. In a **network-based** approach, the network activities are studied in order to detect breaches. Network-based approaches can perform the analysis on a per-packet basis, or on a per-flow basis. The per-packet approaches also differ: one can only analyse the network-layer properties of the packets, such as the IP addresses of the source and destination, or one can perform deep packet inspection and analyse the application-level properties of the packets. The granularity of the approach has its trade-off: the more detailed the approach, the more overhead is introduced. However, not all attacks can be detected when the information is aggregated too much.

Along with the audited material type comes the location of the IDS. The monitoring system can be: (i) central, or (ii) local. A central IDS is deployed for example, in the control room, as a network-based IDS. Also, for example, EMS

monitors the process, controlled by the system, in a centralized way. However, the control room may be compromised as shown in the reported incidents, or the communication can be hijacked through a MITM attack. Once the communication between the central control room and the remote substations is compromised, it is crucial to check whether the commands sent to those substations are legitimate. Therefore, implementing **local** IDS at the routers/gateways in the field stations, is also desired [39].

State-awareness

State-aware detection mechanisms consider the currently observed events in reference to the most recent known system state(s). A state here can be defined by network-layer properties, application-layer properties [20], or process properties [19]. A state-aware IDS is able to detect malicious activities of functions that otherwise seem benign, when considered in isolation; it can identify malicious sequences of commands, for example, issuing the same commands repeatedly [129], or a specific sequence of commands [20]. Although promising, state-aware IDSs can be very resource-intensive due to high computation and memory usage [129].

Detection Performance

The detection performance of an Intrusion Detection System is described by the count of correct decisions in relation to the count of false decisions. A malicious event correctly classified as malicious is called a *true positive* [49]. A malicious event incorrectly classified as a benign event is called a *false positive*. A *true negative* is a benign event classified as benign, and a *false negative* is a benign event wrongly classified as malicious. The total number of the events described above is denoted as TP, FP, TN and FN respectively. The goal of any IDS is to maintain a high accuracy, defined as

$$accuracy = \frac{TP + TN}{P + N},$$

where P is the number of all malicious events and N is the number of all benign events, defined as:

$$P = TP + FP, \text{ and}$$

$$N = TN + FN.$$

Once an event is classified as an intrusion, the IDS notifies, for example, the network manager about the possible breach. However, the IDSs can also classify the traffic wrongly, as indicated above. The occurrence of a false positive means

that the operator is not informed about a malicious activity in the system, which could have quite severe consequences. On the other hand, a false negative means that an operator is warned about a benign activity. An operator can then investigate the nature of the alert and ignore it. However, if too many false negatives occur, the operators may ignore the warnings altogether. This means that also when a true positive occurs, the operator will not react. Therefore, in *detecting* intrusions, false positives are potentially dangerous to the system, while false negatives do not directly affect the system in a negative way.

Intrusion Prevention Systems

Intrusion Prevention Systems (IPSs) are systems that have the same detection capabilities as the IDS, but can also attempt to stop potential incidents [129]. Such a system can, for example, delay for example, the message detected as malicious, or discard it.

The action taken by the IPS influences the safety of the SCADA system. While a false positive, just like for IDS, is potentially dangerous to the system, a false negative introduces yet additional risk in the system. Consider a packet classified as malicious. If this packet is indeed malicious, it is a *true positive* and it is rightly discarded. In case it is benign traffic, it is a *false positive*, and this means that a valid command or a valid measurement from the field was discarded. As explained in Section 2.2.2, the availability of information (commands, measurements) is crucial for SCADA, therefore, any false positives are potentially dangerous for such a system.

Therefore, when implementing IPS in SCADA systems, one must remember that *false negatives*, that is, system packets classified as attacker's actions, are very dangerous. As SCADA relies on the availability of information, discarding valid information, can lead the system to an unsafe state.

2.4.2 Conventional SCADA IDS

Considering the previously discussed classification of IDS types, let us now review the related work on SCADA-oriented IDS. Traditional IDSs rely on the network and transport layer characteristics of communication or packets. Even when supporting SCADA protocols, the monitored process is not taken into account in the detection process.

Anomaly-based approaches model normal SCADA communication, which is learned from benign traffic and any deviations from that model are marked as intrusions [25, 95, 145, 150]. Whitelisting uses the knowledge of the source/destination host and ports [8, 26, 86]. It can additionally allow only certain SCADA protocols to communicate on specific networks [152]. Moreover, protocol-specific

communication patterns can be defined [52], or signatures of attacks on specific SCADA protocols can be simulated and learned [119, 151].

Vulnerabilities of SCADA protocols could either be intrinsic to the standard specification, or a result of improper implementation of the standard. Analysis of the existing protocols and understanding the vulnerabilities can be useful to define protocol-specific intrusion detection rules [70, 78]. As the current protocols are often well-established standards, it could take a long time before a vulnerability dicsovered in a standard is fixed. On the one hand, implementations of a protocol have to be carefully tested to check whether they implement the standard correctly, which is often not done [78]. On the other hand, stateful protocol analysis of the communication can be used to detect anomalies, such as packet injection, replay attacks and data manipulation [52, 81, 153]. Also, to detect an improper implementation of a standard, it is possible to define what the format/content of packets should look like according to the **protocol specification** [124]. The intrusion detection system can then alert when a parsed packet deviates from these definitions.

Other IDS techniques analyse **logged events** of remotely set commands, for example, login attempts, setting or changing passwords, updating firmware, copying or changing files. The analysis of these activities is compared to a reference behaviour (anomaly-based) or it can be matched with known attack patterns (signature-based). The analysis can be done both in a network-based [116] or in a host-based [60] manner.

IDSs focusing solely on characteristics of the communication exchange are important, however, by analysing only the properties of the packets, a system is not able to detect well-formatted legitimate packets which could nevertheless harm the underlying physical system. This can only be done if the IDS also takes the information of the process being controlled into account.

2.4.3 Process-aware IDS

Using the state of both the *control network* and the *the physical process* to improve security has been proposed under different names: [94, 147] discuss semantic-based security analysis, [7] describes a similar approach as behavior-based detection, and [85, 144] introduce physics-based attack detection. Hadziosmanovic and others [57] characterize the types of variables in the network traffic based on their behaviour over time and model the resulting regularity. This approach assumes that the process variables describing the physical process remain consistent over time. Moreover, this method does not predict the outcome of an incoming command, it rather detects whether process variables deviate from their normal value. The proposed technique has been shown to be 98% accurate in real-life traffic [57]. Aoudi and others [3] use raw sensor

measurements to model the process dynamics and detect attacks that would otherwise be hidden within the noise of the process. This approach is not dependant on the type of process, it learns the process dynamics from the measurements. Instead of predicting the future, this method only detects whether the currently observed measurements are diverging from previously observed behaviour.

Lin and others [92, 94] propose an intrusion detection system for SCADA systems controlling the power grid, targeting attacks that send commands that potentially harm the physical system but are hidden in a legitimate format. Although accurate, this approach heavily relies on the assumption that the proposed monitoring system is not compromised. This method uses a so-called *central Master IDS*, communicating with so-called *remote Slave IDSs*, over a communication link, which resembles the current topology of the MTUs and RTUs.

Urbina and others [144] study the detection of stealthy attacks in a system controlling the acidity level of fluid in a tank. Using real-time measurements from the tank and a physical model of the process being controlled allows detecting malicious behaviour if the observations are significantly different from the model-based predictions. The authors present both, a stateful and a stateless approach. Koutsandria and others [85] investigate the so-called "physics aware" Hybrid Control Network IDS (HC-NIDS), which checks a set of cyber-physical security policies on the communication traffic obtained from a network tap. This HC-NIDS is tailored to the protection of digital relays [84] and can also be used in automated power distribution systems when adjusting the rules accordingly [117].

Caselli and others [20] do not take process information into account, directly. However, they investigate the importance of *sequences of commands* in the ICS setting. The violation of predefined sequences of commands can directly impact the process negatively. Sequences of packets are modelled as a discrete-time Markov chain and compared to a precomputed reference model, which represents normal traffic behaviour. Any new states, transitions or changes in transition's frequency are then considered suspicious. Nivethan and Papa [110] propose a SCADA IDS framework that incorporates process semantics, by implementing extra warning notifications in case process variables exceed some threshold values. A system description language and a mapper for turning requirements into actual IDS policies is also provided. This approach is considered *static*, as it computes policies and thresholds, only once. This approach is not validated and to some extent duplicates the HMI in SCADA. Moreover, the authors in [111] analyse the use of open source firewalls in SCADA/ICS and propose to use `iptables` for filtering SCADA traffic. Using string matching they detect, for example, unauthorized write commands and test this approach on Modbus/TCP traffic. Bao and others [7] use rules obtained from physical

properties of the system, which are then translated into state machines. Based on measurements from the system, the state machines are updated continuously and when reaching a critical state a warning is issued to the operator. Mashima and others [100] propose to implement an *active command mediation* mechanism in the electrical substations. Their approach builds on the idea to actively inspect and preprocess the command sent to the remote station before executing it on the physical power system devices. The authors provide an example implementation of this mechanism, the so-called *command delaying mechanism*. In this mechanism, a command could be delayed by a number of proxies so the central system has the opportunity to cancel such a command.

Table 2.4 summarizes and compares the related work discussed above. The table indicates whether the used approach is specification-based or learned from the traffic. It mentions the sector to which the approach has been applied: PG indicates the Power Grid, while ICS indicates a more generic approach and refers to Industrial Control Systems in general. The validation method used in the literature is listed either as TB - physical TestBed, SIM - SIMulation, or RS - Real System (Real Traffic). An approach is capable of detecting attacks or can also prevent attacks, as indicated in the table. Moreover, the detection rules used in the approach are compared. They are either static (generated only once) or dynamically adapt to the current system state. The combination of a learned approach with static rules means that the approach investigates only one-time learning for the proposed mechanism. Finally, the location, which is the placement of the detection mechanism, is compared. It either uses local information and protects a single station, is distributed and relies on information from multiple controllers, or centrally works with information from the entire network, protecting the whole system.

Table 2.4 shows that most approaches tailored for the power grid are based on specifications of the power grid. Approaches that only detect but cannot prevent attacks mainly duplicate the work of the HMI, as operators are notified about values exceeding predefined thresholds. Furthermore, adapting models of the physical process during run-time is not done often to prevent attacks.

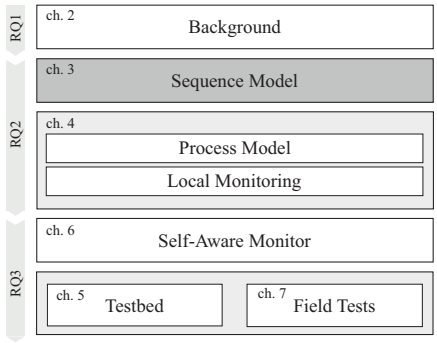
Table 2.4: Comparison of process-aware IDS techniques.

Work	Approach	Sector	Validation	Detection	Prevention	Rules	Location
Hadzismanovic et al. [57]	L	ICS	TB, RS	yes	no	static	local
Aoudi et al. [3]	L	ICS	SIM	yes	no	static	local
Lin et al. [92, 94]	S	PG	SIM	yes	yes	dynamic	central
Urbina et al. [144]	S	ICS, PG	SIM, TB, RS	yes	no	dynamic	local
Koutsandria et al. [84, 85, 117]	S	PG	TB	yes	yes	dynamic (not yet)	local
Caselli et al. [20]	L	ICS	RS	yes	no	static	local
Nivethan et al. [110, 111]	S	PG	none	yes	no	static	local
Bao et al. [7]	S	PG	SIM	yes	no	dynamic	distributed
Mashima et al. [100]	S	PG	SIM	yes	yes	dynamic	distributed delay, central detection

L - learned, S - specification, ICS - Industrial Control Systems, PG - Power Grid, SIM - Simulation, TB - TestBed, RS - Real System (or trace)

Traffic Sequence Model

This chapter investigates modelling the sequences of communication of a SCADA system. We first illustrate the importance of sequences of commands by explaining the concept of interlocks. We then present how both, sequences of normal communication and attacks, can be modelled using discrete-time Markov chains. We have noticed that when modelling a real-life traffic capture using this method, the obtained models were quite large, and often model redundant information. For example, measurement information is often sent not necessarily obeying any sequence. We therefore propose to generate smaller models, which still should be able to detect sequence attacks.



This chapter is organised as follows:

- *Section 3.1 provides related work on modelling SCADA traffic.*
- *Section 3.2 introduces an example of interlocks.*
- *Section 3.3 explains modelling sequences of commands and attacks using discrete-time Markov chains.*
- *Section 3.4 discusses the limitations of this technique and proposes two methods for generating smaller traffic models.*
- *Section 3.5 tests this approach and discusses the results.*
- *Section 3.6 concludes the chapter.*

This chapter has been based on [44].

3.1 Related Work

Intrusion detection mechanisms have been compared on high-level basis in the previous chapter. Sections 2.4.2 and 2.4.3 provided an overview of methods used in related work. In the following, we take a closer look at the modelled information, in order to understand what attacks can be detected with these approaches.

The conventional IDSs often use the assumption that SCADA traffic is stable and periodic [9, 10], therefore, it is relatively easy to learn the normal behaviour of a system. For example, flow-based whitelisting learns "normal" communication pairs (defined by the source and destination IP address and port) in the network, and notifies about any communicating pairs not matching the ones learned before [11]. Flow-based whitelisting approaches are preferred over per-packet approaches, because the latter can be overwhelming in larger networks with a lot of network traffic [99].

Whitelisting approaches are a good way to filter out new devices in the network, however, they fail to detect legitimate devices performing illegitimate functions. This problem was partially addressed by more comprehensive solutions, where protocol-specific (application-layer) whitelists [73, 152, 154], other deep packet inspection methods [75], or timing aspects of the traffic [143] are defined. With these methods, it is possible to detect whether an authorized host is performing controlling functions, analyse the application-layer packet content follows the specification, and test whether the communication exchange is done in a periodic way, respectively.

Other approaches do not focus on modelling *hosts* communicating in the network, but on *periodicity* of the messages. Approaches proposed by Udd and others [143] and Lin and others [90] model the periodicity of the exchanged traffic in order to detect such attacks as denial of service (flooding), and packet injection attacks. These approaches achieve high precision rates for the request-response events, however, it is difficult to detect single messages injected in traffic with spontaneous communication. For asynchronous communication the timing patterns can be modeled using Probabilistic Suffix Trees [89], however, it is unclear whether this approach would detect a single maliciously injected message. Moreover, the timing aspect does not incorporate the notion of a process into the detection method.

A more semantic approach to model SCADA traffic was proposed by Caselli and others [20, 21], where the *order* of the packets sent is taken into account. In Section 3.2 we show that in SCADA systems, the sequence in which commands are executed is often very important.

3.2 Sequence Attacks on the Process

Sequence attacks are specific to industrial control systems and can potentially harm a system by sending valid messages or commands, which are misplaced or out-of-order [21]. To take control of the process, an attacker can either reprogram a PLC or directly control the process from the network, for example, by taking control over the communication channel (see Section 2.3.3). When controlling the process, an attacker sends commands in an order or timing inappropriate for the process. Such potentially harmful sequences of commands are called *sequence attacks* [21].

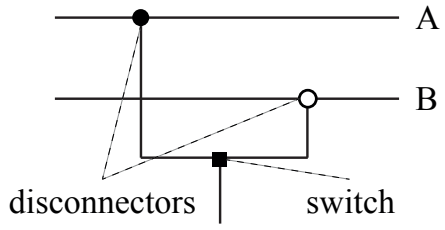


Figure 3.1: Scheme of a rail switching system

The concept of *interlocks* describes constraints on the execution of commands. In power distribution, this applies, for example, to the order in which power switches and disconnectors are used. According to IEC 60947-3, whenever a power line has to be disconnected, first, the power switch disconnects, which turns off the current on the power line. Only then, the disconnector is used to physically isolate the power line. Otherwise, a potentially dangerous electric arc is created. In order to connect a power line, first, the disconnector has to be connected, before the switch is closed.

Figure 3.1 shows two rails: A and B, which are used interchangeably depending on the switch and disconnector setting. The disconnector on rail A and the switch are closed (what is marked with filled figures), while the disconnector on rail B is open (indicated by the circle hollow inside). Hence, the power line at the bottom is connected to rail A. To switch from rail A to B, first the switch opens, then the disconnector on rail A opens. Next, the disconnector on rail B closes and then the switch closes. Figure 3.2 shows three disconnectors connected to three power lines at a Dutch power substation. When open, the rings in the figure are physically separated from the respective power line.

On March 27th 2015, the Dutch province North Holland suffered from a large power outage [112]. It was a result of a technical error caused by the undesirable sequence of switching. In this incident, the disconnector did not

Figure 3.2: Three physical disconnectors at a medium voltage substation

entirely connect before the power switch was turned on. This caused a short circuit, which resulted in a power outage of several hours for more than a million households and disruptions at one of the major European airports [5]. Although the incident was caused by a technical error that was not noticed by the operator, a similar situation would arise by sending legitimate commands in the wrong order to the PLC controlling the switches and disconnector. Although most of the PLCs do check such interlocks internally, some control solutions perform this check at the central control room. Hence, if an attacker gains control over the communication channel to the remote PLC directly, the constraint check will not be performed. Even if interlocks are properly configured, any attempt to switch in a wrong order should be reported to the operator and a sequence-based IDS would fit the purpose.

3.3 Representing Traffic Sequences as DTMCs

This section provides the definition of discrete-time Markov chains in Section 3.3.1 before it explains how to model both a benign sequence of communication and a sequence attack using discrete-time Markov chains in Sections 3.3.2 and 3.3.3, respectively.

3.3.1 Discrete-Time Markov Chains

A discrete-time Markov chain (DTMC) is a stochastic process characterized by a discrete state space $S = \{0, 1, ...\}$ and the *Markov property*, stating that the probability of moving to the next state depends only on the current state (and not on the previous states). The definition of a DTMC is provided below [59, Section 3.3].

Definition 1. ***DTMC**: A labelled finite-state DTMC $\mathcal{M}$ is a tuple $(S, \mathbf{T})$, where:*

- *S is a finite and countable set of states,*
- *the transition relation $\mathbf{T} : S \times S \to [0, 1]$ is a stochastic matrix with $\sum_{s' \in S} \mathbf{T}(s, s') = 1$, for all $s \in S$.* □

This definition is used to describe the sequences of communication as shown in the next section.

3.3.2 Sequences of Normal Communication

Following the approach presented in [21], and the definition provided in the previous section, traffic is represented as a sequence of exchanges in terms of a discrete-time Markov chain (DTMC). In the following, a *sequence (l)* is a time-ordered list of application-level messages in SCADA traffic exchanged between two devices [21]. We use a finite-state DTMC, specified by $\mathcal{M} = (S, \mathbf{T})$. A state $s \in S$ in the DTMC reflects a message sent, that is, the content of a packet. Note that the DTMC state does not directly correspond to the physical state of the power distribution system, but with an event of a message that can affect that physical state. Hence, an event in the sequence, that is, the transmission of a packet, is associated with a state. Transitions model the choice between successor packets together with the respective probability of such a message occurring. The probability p_t of a transition $t = (s_i, s_j) \in \mathbf{T}$ taking place

Data: Sequence of Events
Result: DTMC $\mathcal{M} = (S, \mathbf{T})$ representing the sequence of events
1 **for all** $e_{t_n} \epsilon\ l$ **do**
2 $State_{DTMC} \leftarrow$ extractAttributes(e_{t_n});
3 **if** $State_{DTMC} \epsilon\ S$ **then**
4 update($State_{DTMC}$, $\mathcal{M}$);
5 **else**
6 add($State_{DTMC}$, S);
7 **end**
8 **if** $Transition_{previousState,State_{DTMC}} \epsilon\ \mathbf{T}$ **then**
9 update($Transition_{previousState,State_{DTMC}}$);
10 **else**
11 add($Transition_{previousState,State_{DTMC}}$, $\mathbf{T}$);
12 **end**
13 $previousState \leftarrow State_{DTMC}$
14 **end**

Algorithm 1: DTMC modeling of sequences based on [21]

between the states s_i and s_j is defined as the ratio of the number of jumps from s_i to s_j to the total number of jumps from s_i to any other state in S:

$$p_t = \frac{N_t}{\sum\limits_{\forall s_k \in S} N_{(s_i,s_k)}}, \quad \text{for } t = (s_i, s_j) \in \mathbf{T}, \tag{3.1}$$

where N_x is the number of times a transition x was taken.

We transform network traffic traces into time-ordered list of events $\{e_{t_n}\}$. An *event* e_{t_n}, which takes place at time point $t_n \in \mathbb{R}^+$ is defined as a triple <Direction, Address, Service>, which takes values from the IEC-104 specification (see Appendix B.2), as follows:

- **Direction** either takes the value 'request' or 'response',
- **Address** contains the 'ASDU-address' and the 'IOAs',
- and **Service** is the 'Type ID' that identifies a function.

A sequence (l) sorts events according to their time of occurrence from old to new. It is defined as a time ordered list of events e_{t_n}, such that $t_n < t_{n+1}$ for $n \in \mathbb{N}$. Algorithm 1 then builds a DTMC traffic model from a sequence of events, abstracting from possibly different inter-event times, in the following five steps:

S1 loops over all events in the sequence (see Algorithm 1 lines 1-14), processing all events.

S2 '*extractAttributes*' function extracts the attributes of an event and stores them in the variable '$State_{DTMC}$' (see line 2). The attributes are defined by: Request/Response, ASDU-Address, IOAs, and Type ID.

S3 checks whether that state is already present in the DTMC $\mathcal{M}$. If that is the case, the counter indicating how often that state has been visited is increased in the corresponding state (function '*update*'). Otherwise, a new state is added to the DTMC (function '*add*') in line 6.

S4 updates the transitions. If the transition from a '*previousState*' to '$State_{DTMC}$' is part of **T**, the transition counter, and, therefore, the transition probability, are updated (line 9). If the transition is new, line 11 adds a transition from '*previousState*' to '$State_{DTMC}$' to **T** with the counter set to 1.

S5 updates the variable '*previousState*' in line 13 with the '$State_{DTMC}$' variable created in line 2.

3.3.3 Attack Sequences

In order to investigate sequence attacks, reconsider the combination of switch and disconnector (see Section 3.2), as presented in Figure 3.3. Legal commands for the switch and disconnector are 'open' and 'close'. Let the initial state of the system be the command to open the switch (indicated with a green arrow in Figure 3.3).
The DTMC traffic model has four states: $S = \{s_1, s_2, s_3, s_4\}$, where

- s_1 models that a command to open the switch is sent,
- s_2 opens the disconnector,
- s_3 commands to close the disconnector, and
- s_4 requires to close the switch.

Those states and the corresponding exemplary transition probabilities are shown in Figure 3.3. As discussed before, a command to open the disconnector should always be preceded by the corresponding command to open the switch. Vice-versa, the disconnector should always be closed before the switch is closed. The only bidirectional transitions between states are between open and close disconnector and open and close switch. While it may occur that a switch is, for example, immediately closed after being opened, this will not occur often and hence has a low transition probability. Furthermore, each state is equipped with a self-loop that happens with a relatively low probability. This corresponds to the same command being sent multiple times, which can legally happen, for example, due to a packet retransmission. Recall that IEC-104 runs on top of

TCP, which can cause retransmissions due to preliminary timeouts or the loss of acknowledgements.

Detecting sequence violations is performed as shown in Figure 3.4. First, a benign traffic trace is used to create a *training* DTMC model $\mathcal{M}$ of the communication sequence, such as the one shown in Figure 3.3. Next, a parallel *testing* DTMC model $\mathcal{M}'$ is trained with the most recently observed traffic. The training and testing DTMC are then compared. Below we explain the idea of the comparison, while Section 3.5.2 provides details of this approach.

Three main differences can be distinguished between the two models, that correspond to three types of violations [21], as explained and depicted below. Note that the probabilities of the transitions mentioned in the following figures are not derived from an actual trace, but serve as example to describe sequence attacks.

1. **New transition violation** stems from a valid packet that is however not expected in the sequence of commands [21]. Consider the DTMC is in state s_4 of the switch example, that is, the last packet contained the command to close the switch. If the next command would request to open the disconnector a *new transition violation* is encountered, as the DTMC model does not contain a transition that corresponds to this sequence of commands, namely *close switch* succeeded by *open disconnector*. An alert would be issued to the operator in this case to warn about a potential intrusion. Figure 3.5 shows the violation as a red dashed line.

2. **New state violation** occurs when an unexpected command is sent to the controller. In our example, the switch could receive the command

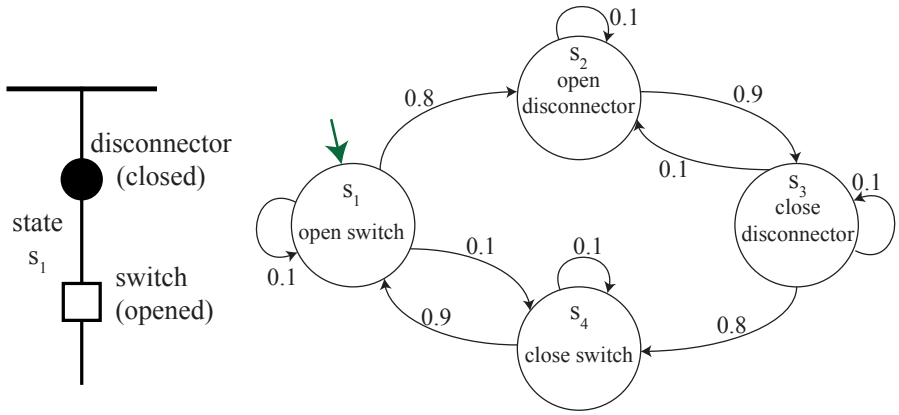


Figure 3.3: Scenario with a single switch and a disconnector

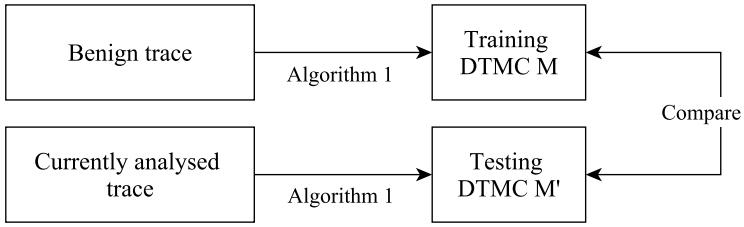


Figure 3.4: Comparing differences between two DTMCs

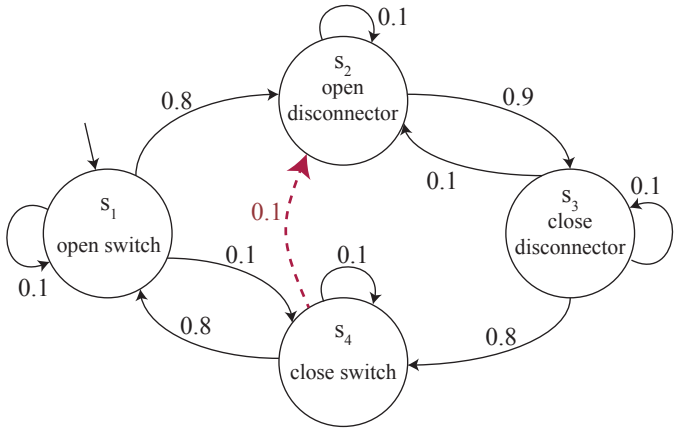


Figure 3.5: DTMC with a transition violation

to change some set point value. This is a legitimate command, which however does not occur often and has not been part of the traffic used to train the DTMC. Hence, there is no state that corresponds to this command. Figure 3.6 shows the violation as a red dashed state. Note that for every new state violation, a new transition violation has to also occur, depicted as the red dashed transition.

3. **Anomalous transition frequency**, a so-called timing violation occurs when a single transition is used too often. The commands arrive in an expected order, however they occur with a probability that deviates from the transition probability more than a certain predefined threshold. For example, if the switch is opened and closed repeatedly, the transition probability between *open switch* and *close switch* will grow to exceed the transition probability of 0.1 in the originally trained DTMC (depicted in

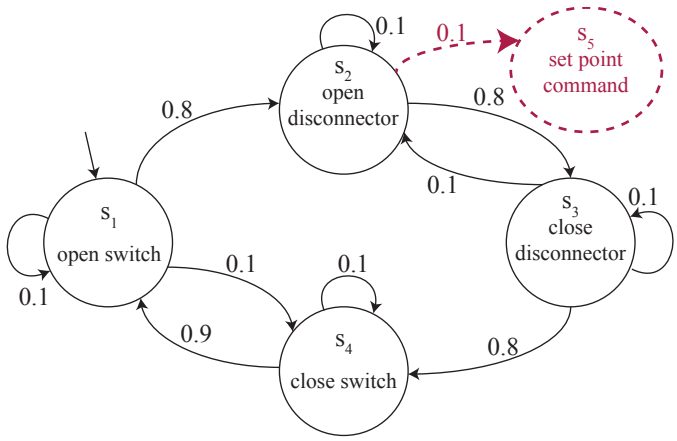


Figure 3.6: DTMC with a state violation

Figure 3.3). Figure 3.7 shows an example DTMC trained from anomalous traffic trace. The transition probabilities that differ from the original DTMC are indicated as red dashed lines.

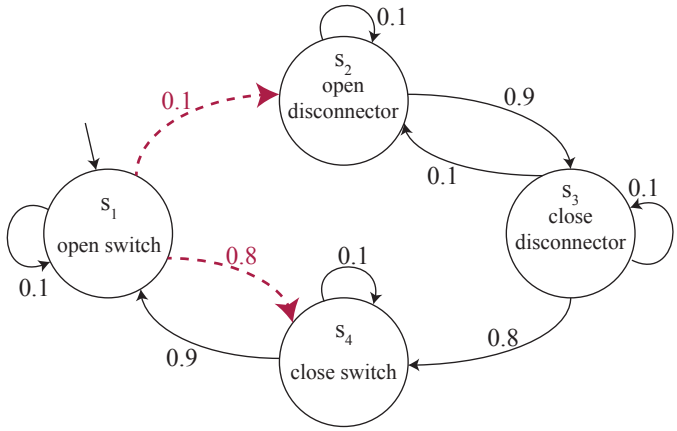


Figure 3.7: DTMC with an anomalous transition frequency violation

One other possibility is a state that does not occur in the testing DTMC $\mathcal{M}'$. This can happen if a specific command is not sent during the currently observed traffic. Although not explicitly analysed, the proposed approach will give some kind of warning in case a state is not present: either a new transition will be

created or the probabilities of the transitions will change significantly, causing an anomalous transition frequency.

3.4 Reduced DTMC Construction

The traffic models that result from applying Algorithm 1 to realistic scenarios can be very large, as shown in [21]. However, many states of the model differ for just a simple state parameter [21]. In the case of IEC-104 traffic we can leverage this by combining states with overlapping IOAs or by completely abstracting from the information contained in IOAs.

Consider a traffic capture where the first message is a measurement message with Type ID 30, that is, *single point infromation*[1], with IO addresses 1-10, in the second message with addresses 6-10, in the third message addresses 1-5, the fourth message sends a single point information (Type ID 45) to address 5, and finally the last message is again a single point information measurement sent from address 125. The original approach explained in [21] would create five different states for these five different requests, as illustrated in Figure 3.8 (A) with 5 different colors. Using the original implementation, we noticed that many new states appear because a request is sent to a different subset of the mentioned addresses, called IOAs.

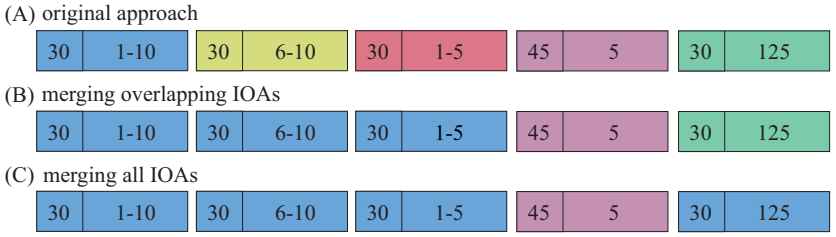


Figure 3.8: Example of influence of the reduction on the number of states

We therefore propose to construct smaller DTMCs in the following way: (i) by considering the states with *overlapping* IOAs as a single state, and (ii) where information contained in IOAs is not taken into account for differentiating states. We compare DTMCs built according to Section 3.3 without reduction to two reduced DTMCs. While the reference case corresponds to the approach in [21], the first reduction relies on the observation that the SCADA server asks for various IOAs, although the function (*Type ID*, see Appendix B.2) remains the same. In the example above, the first reduction corresponds to creating

[1]See Appendix B.2 for details.

three states instead of five: one for Type ID 30 and IOAs 1-10, a second one for Type ID 45 and IOA 5, and a third one for Type ID 30 and IOA 125. They are marked in Figure 3.8 (B) with three different colours.

The second reduction presents a more radical reduction approach, which completely abstracts from the IOAs and only takes the information 'Direction', 'Service' (Type ID) and 'ASDU address' into account from the respective events. For some functions like the *measurement functions*, the process does not suffer if the reading is performed at a different place in the sequence, and merging all IOAs would greatly reduce the size of the DTMCs. In the mentioned example, this would mean that all measurements (Type ID 30) refer to a single state, illustrated in Figure 3.8 (C) with blue color. Commands with Type ID 45 would correspond to a different state, and is represented with purple color.

We have generated traffic models from a SCADA trace obtained at a Dutch gas facility, and in the following we show the resulting DTMCs for the two private IP addresses `172.31.1.100` and `172.31.8.170`. The trace consists of 10 days of traffic captured in 2011. All traffic models presented in this section have been generated using the entire available traffic capture using Algorithm 1, which has been changed slightly to implement also the reductions *overlapping* and *all*. Figure 3.9 roughly depicts the traffic models for the approach that takes into account IOAs in full detail. The model that results from combining states with overlapping IOAs is shown in Figure 3.10, and Figure 3.11 shows the traffic model that does not take into account information about IOAs. Note that in all the figures, the states referring to the same Type ID are marked with the same color.

Figure 3.9 shows a large number of transitions and states representing events in which various subsets of the same IOAs have been requested within the same Type ID. This is indicated in this figure with the same color of the node. Most of the states refer to Type ID 34, which is a normalized sensor measurement value sent to the control room (shown in blue), and to Type ID 1, which is a single point information value sent to the control room (shown in yellow). Despite the difficulty to visualize a DTMC due to its number of states and transitions, as seen in Figure 3.9, the following observations can be made: (i) the majority of events in the DTMC are related to reading measurements, (ii) there are many transitions between the events of the same Type ID, and (iii) no clear sequence of messages is present in this traffic trace.

Figure 3.10 shows the DTMC generated where states with overlapping IOAs are combined. For better readability, this figure has been enlarged[2]. Each state is marked with some color, and contains information on the *Direction*, *Type ID*, *ASDU address*, *IOAs* which it includes and the *count* of how many

[2]See also: `https://github.com/jjchromik/intravis/blob/master/example/over.pdf`.

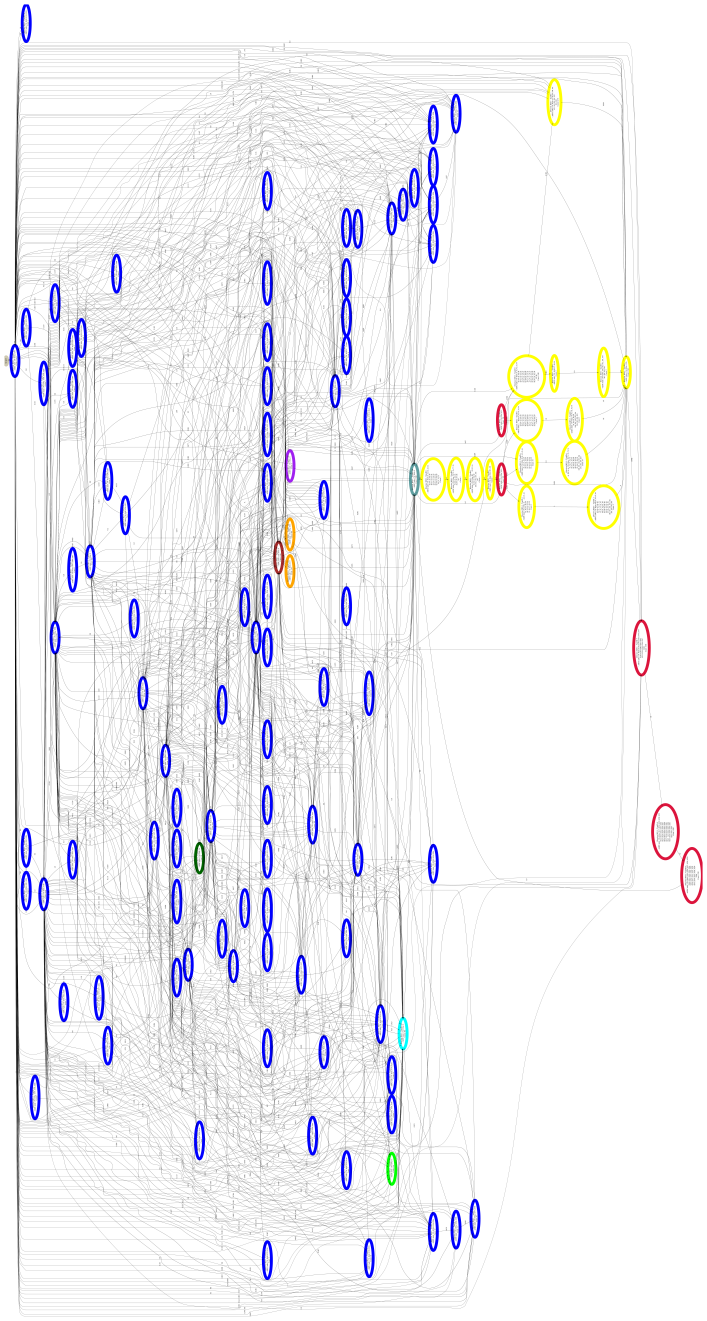


Figure 3.9: Impression of the DTMC generated from the communication between hosts with IP addresses 172.31.1.100 and 172.31.8.170 using the Algorithm 1 as in [21]

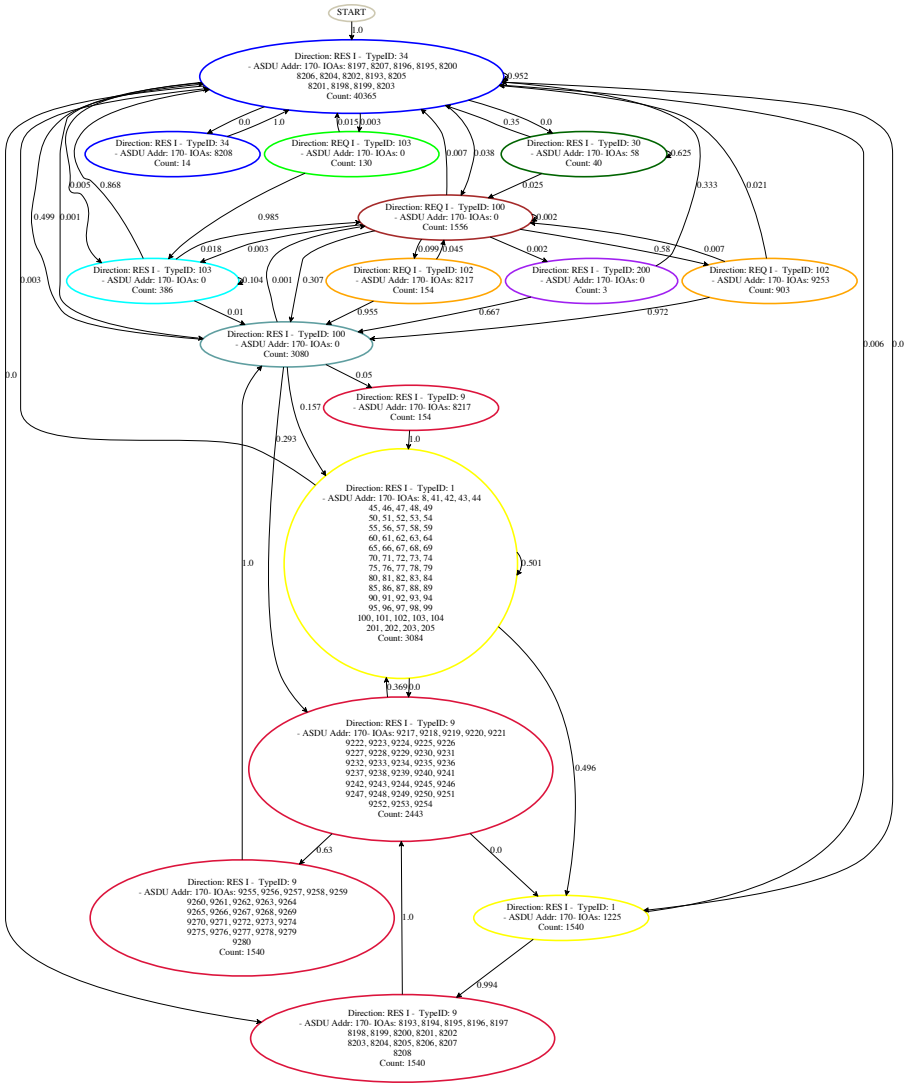


Figure 3.10: DTMCs representing legit activities between devices `172.31.1.100` and `172.31.8.170` over the period of 10 days. States with overlapping IOAs are combined.

packets of this type have been observed. Each transition is labelled with its probability. Almost all of the states marked in blue in Figure 3.9 are reduced to two states only in Figure 3.10, which are also marked in blue. This means that the measurements are sent for various subsets of the 15 IOAs that the Type ID 34 reports.

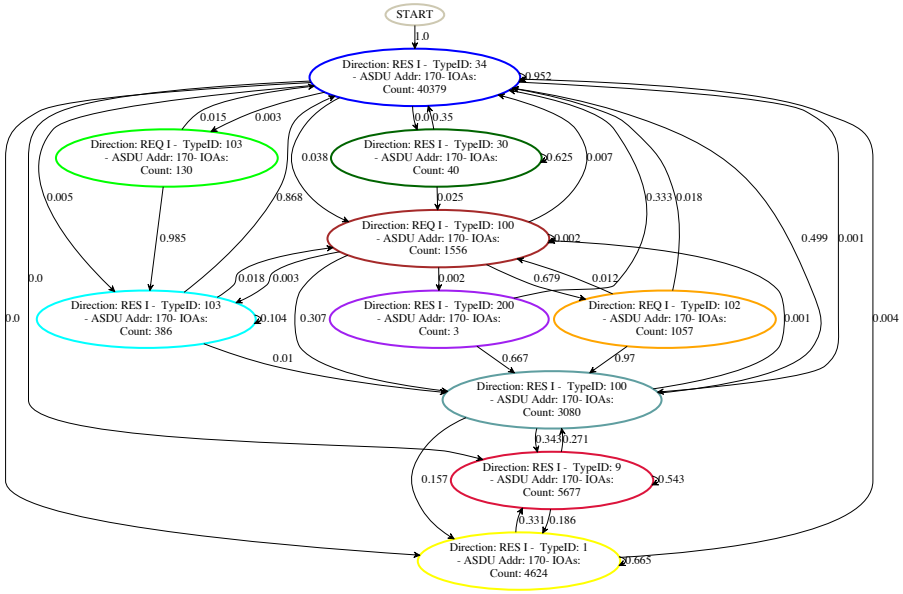


Figure 3.11: Not taking into account IOAs

Reduction *all* then merges all states with the same color as shown in Figure 3.11[3]. Note that some of the probabilities in Figures 3.9-3.11 equal 0. This is due to the fact that these transitions were very rare, for example, occurred only once. Rounding errors when calculating the probability can result in value 0.

Figures 3.9, 3.10, and 3.11, show that the number of states and transitions reduces considerably, when (partly) abstracting from the IOAs. The number of states in the DTMC reduces from 117 to 17 when merging the overlapping IOAs. Further reduction decreases the number of states to 10. We tested all 148 communicating pairs present in the traffic captures and calculated their respective state reduction gain. This is defined as the number of states of the reference DTMC divided by the number of states of the reduced DTMC. The transitions reduction gain is defined analogously.

[3]See also: `https://github.com/jjchromik/intravis/blob/master/example/all.pdf`.

Table 3.1: Reduction gain for three different cases

Communicating pair	Element	Original approach	Overlapping		All	
			#	gain	#	gain
`172.31.8.170 172.31.1.100` (example)	states	117	17	6.88	11	10.64
	transitions	896	51	17.57	39	22.97
`172.31.10.230 172.31.1.100` (best case)	states	189	19	9.95	10	18.9
	transitions	1329	55	24.16	40	33.23
`172.31.3.99 172.31.1.100` (worst case)	states	11	11	1	9	1.22
	transitions	22	22	1	22	1
	average state gain			1.58		2.09
	average transition gain			1.51		2.25

Table 3.1 provides the number of states and transitions for the original and the reduced traffic models for (i) the example shown in Figures 3.9, 3.10 and 3.11, (ii) the DTMC with largest reduction gain (best case), and (iii) the DTMC with smallest reduction gain (worst case). We can see that in the best case, the reduction *overlapping* decreases the number of states almost by a factor 10, and the reduction *all* almost by a factor 19. The worst case shows almost no reduction. We also provide the average state and transition gain that we observed for all 148 communicating pairs.

3.5 Validation

To compare the detection rates of the reduced traffic models to the original one, we introduce anomalies (for example, out-of-order packets) into the traces.[4] Section 3.5.1 explains the type of anomalies introduced in the traffic, before Section 3.5.2 defines what type of sequence violations we aim to detect. Finally, the results are presented and discussed in Section 3.5.3.

3.5.1 Datasets and Anomalies

For validation we use the same traffic capture as mentioned in Section 3.4. This data was split into two parts of 5 days each. One half is used for training a reference DTMC using either no reduction or one of the two approaches listed in Section 3.4. The model obtained from the remaining (testing) trace was then compared with the model obtained from the training traffic trace in order to detect anomalies. Due to the regularity of SCADA traffic the amount of data

[4]See Appendix A for the code used to modify the traces.

(that is, 5 days of training traffic, and 5 days of testing traffic) should be enough to capture all relevant events. The following anomalies have been introduced:

- **copying** a random packet from the used trace and adding it at a random position,
- **removing** a random packet from the trace, and
- **swapping** packets, that is, interchanging the position of two random packets.

We investigate applying the above changes to 0.1%, 1%, and 10% of the packets from the testing trace. Note that while the added anomalies are not necessarily attacks, the results allow valuable insight into the accuracy and usability of the reduced models.

3.5.2 Detection

The detection mechanism compares the DTMCs obtained from the training and the testing traffic captures and checks for the following violations:

- **New transitions violation** - a transition exists in the testing DTMC, but not in the training model.
- **New state violation** - a state created in the testing phase, which does not exist in the training phase.
- **Transition anomaly** - the transition probability in the testing DTMC differs more than a predefined threshold from the corresponding probability in the training phase.
- **State anomaly** - are states affected by transition anomalies. If the total sum of probabilities of incoming transitions for a certain state increases significantly, this means that such packet is requested much more often than normally. If this sum exceeds a certain threshold, such state is marked as anomalous.

For the detection, we build two models according to Algorithm 1: training DTMC $\mathcal{M} = (S, \mathbf{T})$ generated from benign traffic, and testing DTMC $\mathcal{M}' = (S', \mathbf{T}')$ generated from currently observed traffic. Any new state and transition, that was observed in $\mathcal{M}'$, but was not present in $\mathcal{M}$ raises an alert. Moreover, given a known state s_i and known transition t_j in DTMC $\mathcal{M}$, we calculate the distances as explained in [20]:

$$\Delta_{s_i} = \sum_{t_k \in T_i} \frac{|p_{t_k^{\mathcal{M}'}} - p_{t_k^{\mathcal{M}}}|}{2}, \tag{3.2}$$

and

$$\Delta_{t_j} = |p_{t_j^{\mathcal{M}'}} - p_{t_j^{\mathcal{M}}}|, \tag{3.3}$$

where T_i is the set of transitions belonging to a state s_i, $p_x^{t_{\mathcal{M}'}}$ is the probability of transition t_x in the model obtained during the testing, and $p_x^{t_{\mathcal{M}}}$ is the probability of the transition t_x in the model obtained during the training.

Next, we define thresholds for both state and transition violations. We have chosen for threshold $\alpha = 0.1$ for both violations to alert any significant change. The sequence violations are detected by comparing the distances defined in (3.2 - 3.3) between the trained DTMCs and testing DTMCs. In case the difference exceeds the above thresholds, an alert is raised.

3.5.3 Results and Discussion

We modify the second part of the trace 10 times and compute the median and variance of the number of detected anomalies for copying, removing and swapping respectively 0.1%, 1% and 10% of all 6079 IEC-104 packets. We chose median over mean in order to show a result that will not be affected by potential outliers in the results. Furthermore, we compare the results of the original model without reduction to the results of the two proposed reductions. The results of the detection of the sequence violations are presented in Tables 3.2 and 3.3.

Table 3.2 shows the number of state anomalies, including the new states (given in brackets). The row providing the results for the original approach is

Table 3.2: State anomalies (new state violations)

Reduction type	No change		Copy			Remove			Swap		
			0.1%	1%	10%	0.1%	1%	10%	0.1%	1%	10%
none	11 (1)	med	11 (1)	11 (1)	19 (1)	11 (1)	11 (1)	14 (1)	11 (1)	11 (1)	20.5 (1)
		var	0 (0)	0.233 (0)	2.456 (0)	0 (0)	0.1 (0)	0.667 (0)	0.4 (0)	0.178 (0)	0.933 (0)
overlapping	9 (1)	med	9 (1)	9 (1)	15 (1)	9 (1)	9 (1)	12 (1)	9 (1)	10 (1)	15.5 (1)
		var	0 (0)	0.178 (0)	1.956 (0)	0 (0)	0 (0)	0.622 (0)	0.4 (0)	0.267 (0)	0.933 (0)
all	6 (0)	med	6 (0)	6 (0)	8 (0)	6 (0)	6 (0)	7 (0)	6 (0)	6 (0)	10 (0)
		var	0 (0)	0.1 (0)	0.989 (0)	0 (0)	0 (0)	0.622 (0)	0.1 (0)	0.1 (0)	0.767 (0)

marked grey for reference. For the non-modified trace, under the "No change" column, the original model detects 11 state anomalies, out of which 1 state is new. The new state corresponds, for example, to a packet that occurs in the real traffic but was never seen in the 5 days of the training capture, for example, related to a maintenance.

All these anomalous states are considered false positives, as they do not result from a modification of the original trace, but from an irregularity in the trace itself. Comparing the number of the anomalies detected using the original model to the two proposed reduced models, we notice that the number of false positives drops, hence, the detection accuracy of the reduced model improves, due to abstracting from the specific IOAs. In reduction *overlapping*, there are 9 anomalous states, out of which 1 is new. This is the same state as for the reduction *none*. In reduction *all*, 6 anomalous states occur out of which none are new. This means that the packet that was not seen in the training phase uses a Type ID that appears in the training capture, but a different, not earlier seen, IOA.

After introducing anomalies into the traffic sequence, we can see that most of these anomalies do not have an influence on the detection rates. Note that the introduced anomalies will never result in a new state, as we do not generate any *new* packets that would not be seen before. This is confirmed by the results: the number of new states never increases with the anomalies, and the variance of 0 indicates that this was the case for all the tests. Copying and removing 0.1% of the packets did not affect the number of anomalous states in neither of the tests, also confirmed by the variance of 0. When copying or removing 1% of the packets, even though the median of anomalous states equals that number in the unmodified trace, the variance indicates that for some tests this number differed. Swapping 0.1% and 1% of the packets provided some minor changes in the number of detected anomalous states. However, the biggest changes in the number of anomalous states was observed when copying/removing/swapping 10% of the packets. These anomalies in traffic traces increased the number of detections of state anomalies in all of the detection models: the reduced ones and the original one. As the introduced anomalies are performed randomly, we cannot tell whether they could be harmful to the system. Therefore, we cannot judge whether the detected state anomalies should be considered false positives or true positives.

Table 3.3 shows the number of transition anomalies and the number of new transition violations (provided in brackets). Again, the reference case is marked gray (reduction *none*). The original approach detects 24 transition irregularities in the original trace, out of which 10 are new. All those need to be considered as false positives. Either the training traffic capture was too short, as the dataset did not contain messages in this order, or the chosen detection threshold was

set too low. Column "No change", representing the original trace, shows that the reductions decrease the number of false positives w.r.t. the reference case from 10 to 4 in the reduction type *all*.

Modifying the traces introduces additional transition anomalies. Even modifying 0.1% of all packets increases the number of new anomalous transitions considerably. When reducing the traffic models, the number of detected anomalies decreases. For example, when copying 10% of the packets, without reduction, 115 **new** transitions are observed (given in brackets), while *overlapping* results in 81.5 anomalies, and *all* in 41 anomalies. An operator may prefer fewer alerts, as too many notifications may be ignored. However, the question remains, how to distinguish an attack from a false positive alert. A high variance, for example, a variance of 48.22 for anomalous transitions when copying 10% of the packets for the original approach suggests that the number of anomalies deviates in each test. This is to be expected as the introduced anomalies directly affect the order of the packets. Note that the reduced number of detections when applying reductions stems from two sources. First, we lose false positives as in the reference case, which increases the accuracy of detection. Second, not every change is detected in the reduced model, which decreases the sensitivity. The current detection algorithm is not performed after each event, hence, we are unable to distinguish between losing false positive or true positive.

Reconsidering the disconnector and switch attack from Section 3.2 shows that the reduction method should be chosen keeping the application in mind. If a single PLC would control the actuators, the same function (Type ID) referring

Table 3.3: Transition anomalies (new transition violations)

Reduction type	No change		Copy			Remove			Swap		
			0.1%	1%	10%	0.1%	1%	10%	0.1%	1%	10%
none	24	med	31.5 (17.5)	70 (55.5)	134 (115)	28.5 (14.5)	37 (23)	61 (46.5)	38 (24)	89.5 (75.5)	145 (127)
	(10)	var	1.34 (1.34)	15.16 (15.21)	48.22 (36.99)	1.79 (1.79)	1.82 (1.82)	13.66 (11.33)	10.28 (10.28)	38.68 (35.33)	31.83 (32.23)
overlapping	18	med	25 (13)	57 (45.5)	98 (81.5)	22 (10)	29.5 (17.5)	49 (35)	31.5 (19.5)	71 (59.5)	104 (90.5)
	(6)	var	2.28 (2.28)	15.96 (17.11)	34.84 (13.16)	1.43 (1.43)	0.68 (0.68)	8.28 (3.66)	8.54 (8.54)	16.23 (14.62)	12.71 (14.62)
all	14	med	17 (7)	30.5 (20.5)	49.5 (41)	15.5 (5.5)	21 (11)	33 (21)	21.5 (11.5)	37 (27)	57 (44.5)
	(4)	var	0.27 (0.27)	8.71 (7.29)	13.73 (10.22)	0.28 (0.28)	1.33 (1.43)	2.93 (1.96)	7.12 (6.5)	15.21 (10.1)	12.62 (6.99)

to opening or closing respective IOs could appear in a specific order. Therefore, implementing reduction *all* could abstract too much. This could be preserved with the *overlapping* reduction, still reducing the size of the traffic model. In contrast, when dealing with simple measurement commands, such as Type ID 1 or 34, merging all IOAs would not result in a loss of accuracy, still reducing the size of the traffic model.

3.6 Conclusions

Interlocking is an example of a SCADA process where sequence of the executed commands ensures a safe operation. Learning sequences from SCADA traffic and comparing such models to the current traffic pattern is a promising anomaly detection method. However, the developed models can easily become very large and it might not be feasible to maintain large models for each pair of communicating devices.

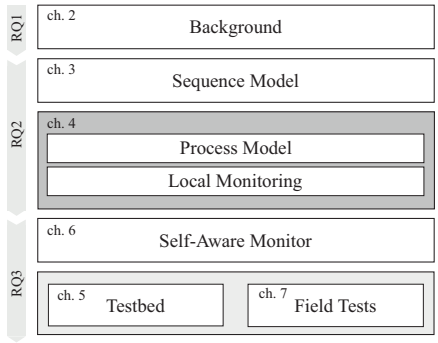
We show that some cases exist where these models can be substantially reduced. For example, the states in the modelled traffic trace using IEC-104 protocol differed mostly with just subsets of requested Information Object Addresss. These states were easily and conveniently combined when generating the proposed *reduced* DTMCs. We observe that abstracting completely from IOAs reduces the model size considerably while losing accuracy. Despite lowering the number of false positives this may cause the IDS to overlook specific attacks, like the disconnector and switch attack. For this reason a more conservative approach combining states with overlapping IOAs has the highest chance to succeed because of the high model accuracy while still reducing model size.

We conclude that when choosing reduction methods the actual purpose of the exchanged functions of the IEC-104 protocol should be taken into account. By understanding the goal of the actual functions (Type IDs), one can use specifically tailored reduction techniques for different functions. However, in most cases, the knowledge needed to develop the reduction function can come only from the operator side.

Sequence attacks are only a small subset of the possible attacks on SCADA systems. As discussed in Section 2.3.1, most of the reported attacks performed some switching commands. In order to address other attacks on the SCADA process, a different method of evaluating the consequences of the currently exchanged packets in the network is needed.

Process Model and Local Monitoring

Sequence attacks detected by the approach shown in the previous chapter comprise only a small, specific subset of possible attacks on the controlled process. To detect other attacks that could disrupt the operation of the electric power distribution system, a model that understands the consequences of the cyber commands on the physical system is needed. Hence, this chapter presents (i) a modelling formalism that can be used to describe the electric power distribution system, and (ii) a monitoring algorithm that is used to detect whether the system is safe. This modelling formalism and the monitoring algorithm are used in the remainder of this thesis for process-aware monitoring of the SCADA traffic.



This chapter is organised as follows:

- *Section 4.1 presents related work on the modelling of the physical process in the power systems domain.*
- *Section 4.2 sketches the proposed method.*
- *Section 4.3 proposes our modelling formalism that can be used to describe the power distribution system.*
- *Section 4.4 outlines an algorithm that is able to detect undesired commands and non-safe or non-consistent system states.*
- *Section 4.5 illustrates the possibilities that arise when using the proposed model and monitoring techniques.*
- *Section 4.6 concludes the chapter.*

This chapter has been based on [27, 28].

4.1 Related Work

This section describes related work on process-aware traffic monitoring. Note that Section 2.4.3 already provided a high-level overview on process-aware IDS. This section, instead, focuses on how physical systems have been modelled in order to be used in the detection mechanism. Section 4.1.1 describes models that can be learned from observing traffic, whereas Section 4.1.2 presents models derived from system specification.

4.1.1 Learning from Traffic

The traffic exchanged between field stations and the control room of a SCADA system contains the measurements from sensors in the field, as well as the commands coming from the control room. The values of certain variables describing the power distribution system can be represented as a set of *invariants*, that is, constant relationships between some of the power values [13]. An invariant dependency is defined as, for example, relation of real power values for buses in the power distribution system: $P1 = k \cdot P2 + C$, where k and C are constants, and $P1$ and $P2$ are real power values on two buses. Any anomalies from such dependency are identified as breaches. The invariants are learned from the observed traffic, and it is not discussed how changes to, for example, system topology influence the invariants. Later work by Jin and others [74] extends the amount of invariants, adding other physical laws, such as Ohm's law or Kirchhoff's law. Another way to model the process values is by performing an *N-gram analysis* [13]. By treating the data within packets as text, and registering the first four characters of the process values, it is possible to build up a database of seen values. The first four characters contain the sign of the reading, the position of the decimal point and the most significant digits. Any new value that significantly differs from the ones learned are reported. It is also possible to model the system values over time. The approach proposed in [57] models the process variables in three categories: (i) variables that change continuously, and gradually over time, (ii) variables that take a value from a discrete set of possible numbers, and (iii) values that remain constant over time. To model the second and third category, a set of expected values has to be derived. To model the first category, autoregression modelling and control limits techniques are used. The former states that the next value of a process variable is a linear function of the previous p quantities plus a prediction error, while the latter defines an upper and lower operation limit of the expected process variable value based on [148]. If a large deviation from the learned trend for variables that change continuously over time is detected, an alert is triggered.

The approaches mentioned above are a sufficient way to describe a stable

system, whose values remain consistent over time, and can only be used to detect suspicious sensor measurements. However, as mentioned in Section 2.1.1, power distribution is facing a rapid transition towards more renewable power sources. Energy-wise, renewables are a source of clean energy, and modelling-wise, renewables are a source of uncertainty. Moreover, it is not clear how the detected suspicious measurements would influence control decisions of the investigated systems.

Approaches considering renewable resources, instead of invariants, for example, use *ranges* to define an expected voltage value [72], or, use weather predictions to create an artificial *neural network model* of the expected power values [83]. Isozaki and others [72] check whether the measurements of the voltage is as expected. If not, they delay the decision to change the transformer setting to avoid putting the system in the unwanted voltage range. Kosek [83] uses the information from the model to detect undesirable commands in systems controlling the photovoltaic panels.

For real applications, models learned from traffic described above have several disadvantages. The invariants can be affected when changing the system topology, leading to false positives in the detection. Also, the N-gram analysis might not adequately detect attacks that change the values of the messages slowly over time. Finally, it is of utmost importance for the anomaly-based approaches to perform the learning phase of a normal behaviour of the system when no intrusions occur, which is difficult to guarantee. On the other hand, supervised approaches such as the one implementing neural network requires to train both, the good behaviour, and all possible malicious activities. Unfortunately, a dataset, containing labelled captures of process-oriented attacks on power distribution, to the best of our knowledge, does not exist.

4.1.2 Specification-based Models

Specification-based approaches require knowledge of the system that can be provided by the system operator [110], for example, in a form of a formal description of the power distribution system.

One way of using such knowledge is to specify *thresholds* of values for each process variable. Once a process variable leaves those predefined bounds, the operator is notified about this [110]. For example, an operator wants to be informed when a value of current exceeds $5A$, that is, $(I > 5)$, raises an alert. This approach can instead use more complex dependency between the process variables, as proposed by [17]. For example, instead of a single condition, it can check that if the mentioned value of current exceeds $5A$ and a switch S is opened, that is, $(I > 5, S == 0)$, then it should raise an alert. The system state is defined by the values of all the system components, for example, for n system

components the system state can be represented by a vector $s \in \mathbb{R}^n$. Some of those vectors are considered *critical*, it is possible to track the evolution of the state towards a critical state [48].

Another way to check whether the information exchanged in the network is true is by checking if the system's process variables adhere with circuit conservation laws [117], that is, *laws of physics*. However, Parvania and others do not implement this idea, nor do they explain the approach to do this. With the description of the entire system, some approaches [7, 37, 94] use a *mathematical representation* of the power system. Lin and others [94] describe the power system by means of matrices G and B that denote the conductance and susceptance of the transmission lines, respectively. The system state is described as the set of voltage magnitudes and angles for each bus i, that is, (V_i, Θ_i). The state can be calculated from two power flow equations. For each bus i, the generated power (P_i^g, Q_i^g), consumed power (P_i^l, Q_i^l), and the power delivered to other buses (indexed by k) are balanced at each timestamp:

$$P_i^g - P_i^l = \sum_k V_i V_k (G_{ik} \cos(\Theta_i - \Theta_k) + B_{ik} \sin(\Theta_i - \Theta_k)), \text{ and}$$

$$Q_i^g - Q_i^l = \sum_k V_i V_k (G_{ik} \sin(\Theta_i - \Theta_k) - B_{ik} \cos(\Theta_i - \Theta_k)).$$

These nonlinear equations can be solved within a predefined error threshold with for example, the Newton-Raphson algorithm.

Having the description of the physical process, a *control-theoretic formulation* of the system state dynamics can be analysed [7, 37, 94, 144] in the following way. A command is sent to the system under analysis and its effect is also computed in the model of that system. Next, the output of the system under analysis is compared with the computed values. If the two differ from each other, such anomaly is reported. Differences in the observed and computed values could possibly mean that either (i) the command was not executed in the real system, (ii) a different command was executed in the real system, (iii) the model of the system is not correct or not accurate, or (iv) one or more sensors are broken.

Methods that require the description of the entire system cannot be used in a local setting, because a monitoring tool deployed at a single substation do not have enough input to solve power flow equations. Approaches that only compare process variables to predefined thresholds lack flexibility in defining rules that depend on the current state of the system. Moreover, even though it is believed that a system can only reach a limited number of unsafe (critical) states [17], the process of listing them manually can be cumbersome and it should be possible to describe these states more generically. Therefore, we propose a different model

to describe a subset of a power distribution system, as explained in the following sections.

4.2 The New Approach

This section sketches the monitoring approach that we aim to implement locally at the field substations. Later, Section 4.4.4 formalizes this concept. Methods described in Section 4.1 work well when the system is stable enough to learn all possible allowed values (for the approaches that learn the models from traffic), or when the monitoring is performed centrally (when modelling the entire distribution system). However, the cyber incidents reported for the power distribution systems [4, 53, 157] show that once the central control room is breached, it is possible to send legitimate commands that disrupt the operation such systems. Also, more and more field stations are connected to a SCADA system and are controlled remotely, while, the protection mechanisms are still maintained centrally in the control rooms. Instead, we propose an approach that would be suitable in the field stations, that bases its decisions on the locally obtained measurements. It connects the previous approaches in the way that

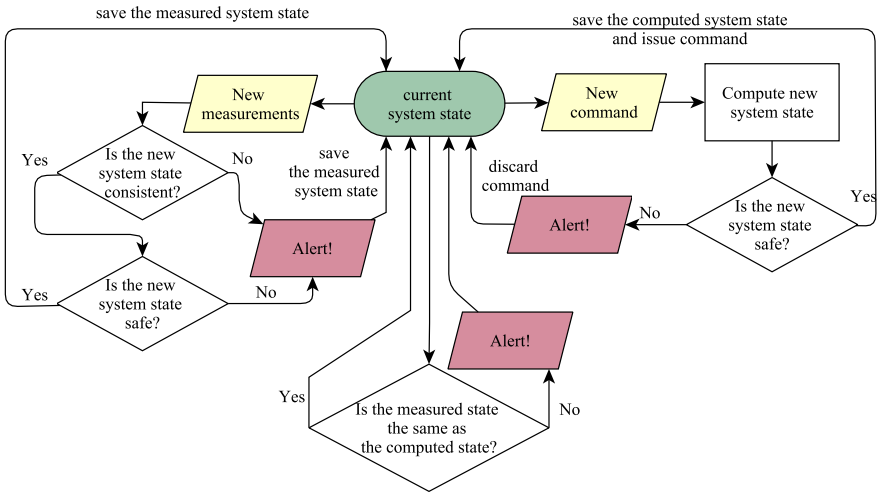


Figure 4.1: Flow chart illustrating the concept of the monitoring approach. Inputs (events) are highlighted in yellow, the initial state is marked as green and the outputs (alerts) are highlighted in red.

it uses the specification of the locally controlled system, and updates the system values based on the information obtained from the traffic. The concept of the proposed approach is illustrated in Figure 4.1. The current system state, represented by the green oval in Figure 4.1, is initialized with the latest sensor measurements.

The left loop of Figure 4.1 is triggered with new measurements obtained from the sensors. If any of the measurements have different values than stored in the current system state, this means that a new system state is observed. Knowing the dependencies between components of a system-at-hand, the monitoring algorithm can verify whether the system obeys them. These dependencies are dictated by the laws of physics and they verify whether the new observed system state is consistent. If the observed state is consistent, the tool then verifies whether the observed values do not violate any safety rules, for example, dictated by predefined thresholds. If that is the case, the newly observed system state is saved into the current system state. Otherwise, if the observed state is not consistent, or not safe, an alert is first generated to the operator, before the current system state is updated.

The right part of Figure 4.1 shows the procedure caused by a new command sent from the SCADA control room to the RTU. New command means that the current system state *will* change. This change is dictated by the physical laws. Therefore, the monitoring tool uses the applicable physical dependencies between the controlled components, and computes the predicted new system state. Next, the safety of the computed system state is checked. If the system state is safe, then this command is forwarded to the RTU and executed in the real system, and the predicted system state is saved as the current system state. Otherwise, the monitoring tool issues a warning to the operator, and discards the command.

Finally, the lower cycle in Figure 4.1, the algorithm also compares the calculated and the observed system state. If they are not equal, the monitoring tool alerts the operator about this.

In order to perform the monitoring outlined above, a modelling formalism that allows to describe the locally controlled power distribution system is needed. This is introduced in the next section. Using this formalism, it is possible to describe the model of a system-at-hand and its state.

4.3 Process Model

This section introduces a modelling formalism that allows to unambiguously describe elements of a power distribution system. The resulting model is used to understand how the physical system evolves over time. These elements are

described with various properties that can be either static or variable over time. The static properties is referred to as the *topology* of the system, while the variable properties is the part of the *system state*. The resulting specification is independent of any programming language, simulation environment or testbed. In Section 4.3.1 the formal description of the system, its elements and properties are defined, Section 4.3.2 describes the static topology of the system, whereas Section 4.3.3 describes the variable system state.

4.3.1 Formal System Description

Formally, (a part of) the power distribution system is described as a tuple $\Omega = (\mathcal{P}, \mathcal{B}, \mathcal{T}, \mathcal{L}, \mathcal{S}, \mathcal{M}, \mathcal{F}, \mathcal{R}, \mathcal{K})$, where:

- $\mathcal{P} = \mathcal{P}^G \cup \mathcal{P}^C$ is a set of power generators ($\mathcal{P}^G$) and consumers ($\mathcal{P}^C$),
- $\mathcal{B}$ is a set of buses,
- $\mathcal{T}$ is a set of transformers,
- $\mathcal{L}$ is a set of power lines,
- $\mathcal{S}$ is a set of switches,
- $\mathcal{M}$ is a set of meters (sensors),
- $\mathcal{F}$ is a set of fuses,
- $\mathcal{R}$ is a set of protective relays, and
- $\mathcal{K}$ is a set of interlocks.

Even though this modelling formalism is general enough to capture a large part of the power grid, smaller models that only represent individual substations controlled by a single RTU can be used. Depending on the scenario, not all elements included in Ω will be part of the local system, since, for example, not every substation contains a transformer.

The system elements are described below. Table 4.1 summarizes all relevant notation, where a set is represented with calligraphic uppercase letters, an element of a set is represented with a normal uppercase letter with a subscripted index, and a vector is represented in bold.

Power Generators and Consumers

Power generators (sources), such as power plants or photovoltaic panels are denoted P_i^G for $i \in \{1, .., |\mathcal{P}^G|\}$. The amount of power produced at source P_i^G is labelled $P_i^G.pv$ and is equal or larger than zero. Power consumers, such as households are expressed as P_i^C for $i \in \{1, .., |\mathcal{P}^C|\}$. Note, that in smart grids there may be customers (houses) that produce electricity using, for example, solar panels and are able to become a source of electricity. In that case they can be treated as a source, instead of a load. The amount of power consumed

at load P_i^C, denoted $P_i^C.pv$, is equal or smaller than zero. The set of all power generators and consumers is defined as $\mathcal{P} = \mathcal{P}^G \cup \mathcal{P}^C$. Vector $\mathbf{P}$ is of size $|\mathcal{P}|$ and collects the current values for all generators and consumers. Power sources and consumers are connected to the distribution system with a single power line. This connection is described as their *position*: $P_i^x.pos = L_j$ for $x \in \{G, C\}$. Symbols of a power generator and a power consumer are shown in Figure 4.2.

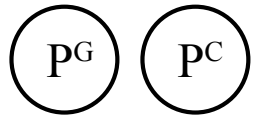


Figure 4.2: Symbols of a power generator and a power consumer

Buses

Buses are labelled B_i for $i \in \{1, ..., |\mathcal{B}|\}$. The power flows from elements with higher energy to the ones with lower energy, for example, from a generator to a load. Knowing the power on each bus, it is also known which lines are *incoming* to and *outgoing* from the bus. Let $\mathcal{B}_j.in$ denote the subset of lines incoming to the jth bus, and $\mathcal{B}_j.out$ the subset of lines outgoing from the jth bus. Note that for each bus, the number of incoming/outgoing power lines can differ. A bus is represented as a bold, vertical line, shown in Figure 4.3.

Figure 4.3: Symbol of a bus

Transformers

Transformers connect parts of the power system that operate at different voltage levels. A transformer T_i for $i \in \{1, ..., |\mathcal{T}|\}$ has the following properties: transformation rate $T_i.r$, which defines the voltage ratio (for example, the ratio 1000:1 transforms voltage from 400 kV to 400 V), and the transformer tap position $T_i.p$. Set $T_i.t$ is a discrete set that collects all the tuples $(T_i.p, T_i.r)$ that are possible for transformer T_i. The position of the tap switch of a medium to

low voltage transformer has to be chosen such that the secondary voltage, that is delivered to the customers, equals 230 V. A transformer T_i has exactly one incoming ($T_i.in$) and one outgoing ($T_i.out$) power line. The measurements of voltage and current are not taken directly on the windings of the transformer, but on these incoming and outgoing lines, which results in an accurate approximation. All properties of the transformers are listed in Table 4.1 and the symbol of a transformer is shown in Figure 4.4.

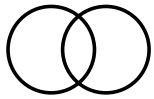


Figure 4.4: Symbol of a transformer

Power Lines

Power lines (or *branches*) labelled L_i for $i \in \{1, ..., |\mathcal{L}|\}$ connect power generators and consumers with each other, or with buses and transformers. They are defined as follows: $\mathcal{L} \subseteq ((\mathcal{P}\times\mathcal{B})\cup(\mathcal{T}\times\mathcal{B})\cup(\mathcal{B}\times\mathcal{B})\cup(\mathcal{B}\times\mathcal{T})\cup(\mathcal{B}\times\mathcal{P}))$. The physical characteristics of a power line impose a maximum current on that branch, that is, $L_i.I_{max}$. Exceeding this maximum value may damage the power line, for example, by wearing it off much faster. The maximum current capacity is provided as a vector over all power lines: $\mathbf{L}.I_{max} = [L_1.I_{max}, L_2.I_{max}, ..., L_{|\mathcal{L}|}.I_{max}]$. The power lines are also described by the reference voltage $L_i.V_{ref}$. This voltage corresponds so the nominal voltage that holds in the voltage level that the power line belongs to. Note that there are special types of power lines, so-called feeders. They are usually of much higher current capacity and they connect main distribution stations. Our modelling formalism treats them as power lines. Also, a single line may consist of multiple connected lines. This may happen when part of a power line is damaged or worn out, or when a branch is extended. In such a case, the minimal I_{max} value among the parts of the line is considered. Power lines can be equipped with a switch, a fuse, a meter and/or a protective relay. These elements are physically located in a substation and can be present at either or both ends of the power line. The summary of characteristics of power lines can be found in Table 4.1. Symbol of a power line is a thin, vertical or horizontal line, shown in Figure 4.5. Power generators and loads, buses, and transformers can be connected using power lines like shown in Figure 4.6.

Figure 4.5: Symbol of a power line

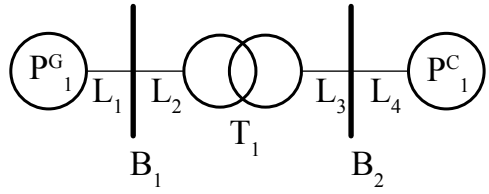


Figure 4.6: A power generator, a power consumer, a transformer and two buses connected with power lines

Switches

Each power line can be connected to or disconnected from the bus by a switch. For each switch S_i, where $i \in \{1, ..., |\mathcal{S}|\}$, the state of the switch is denoted as $S_i.st \in \{0, 1\}$, representing an open (disconnected) and a closed (connected) switch, respectively. The vector $\mathbf{S}$ collects the states of all switches and has size $|\mathcal{S}|$. The position of a switch $S_i.pos = L_i.B_n$ describes its location: on power line L_i next to bus B_n. The properties of a switch are summarized in Table 4.1. A symbol of a switch is shown in Figure 4.7, and Figure 4.8 shows two switches: S_1 and S_2, that are located at power lines L_2 and L_3, respectively.

Figure 4.7: Symbol of a switch

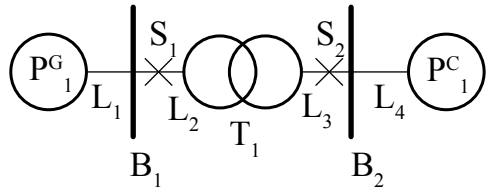


Figure 4.8: Switches S_1 and S_2 located at power lines L_2 and L_3, respectively

Meters

Next to the switches each power line has meters M (also called *sensors*) within the substation where the bus is located. The meter M_i measures usually at least the current on the line $M_i.I$, and the voltage between the line and the ground $M_i.V$. The readings from a sensor are written as a pair of current and voltage: $(M_i.I, M_i.V)$. The vector $\mathbf{M}$ collects all sensors' readings and is of size $|\mathcal{M}|$. The location of the meter is described in a similar way as for a switch: $M_i.pos = L_i.B_n$ means that the meter M_i is located at the power line L_i at the

side of the bus B_n. Set points are values specified on local RTUs that tightly correspond to the values measured by the meters. If the currently measured value exceeds the predefined set point, the RTU issues a SCADA alert. The set points are considered as properties of the corresponding meter, $M_i.I_{sp}$ and $M_i.V_{sp}$ The properties of the meters can be found in Table 4.1. A symbol of a meter is shown in Figure 4.9, and Figure 4.10 shows meters M_1 - M_4 that measure the current and voltage on power lines L_1-L_4 and are connected to a single RTU.

Figure 4.9: Symbol of a switch

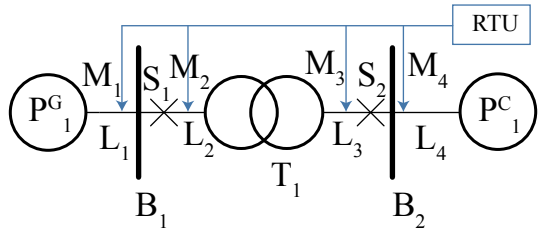


Figure 4.10: Meters M_1 - M_4 measuring the current and voltage on power lines L_1-L_4 are connected to a single RTU.

Fuses

A simpler version of a switch is a fuse, which melts when an overcurrent occurs. It is not possible to turn the fuse back on, it can only be replaced. The fuse is denoted as F_i, where $i \in \{1, ..., |\mathcal{F}|\}$ and the state of the fuse is either one (connected) or zero (disconnected), that is, $F_i.st \in \{0, 1\}$. Vector $\mathbf{F}$ collects the states of all the fuses and is of size $|\mathcal{F}|$. The position of a fuse is described in the same way like for meters and switches. Again, the properties of the fuses are summarized Table 4.1, and a symbol of a fuse is shown in Figure 4.11. Figure 4.12 shows two fuses F_1 and F_2, that protect power lines L_1 and L_4, respectively.

Protective Relays

Protective relays (also called circuit breakers) are mechanical or digital controllers, which control a connected switch. In case the current measured on the power line exceeds some predefined threshold I_{max}, the switch is opened, disconnecting the line with overcurrent. Protective relays are denoted as R_i for $i \in \{1, ..., |\mathcal{R}|\}$, and are assigned to a switch, that is, for relay i, which is

Figure 4.11: Symbol of a fuse.

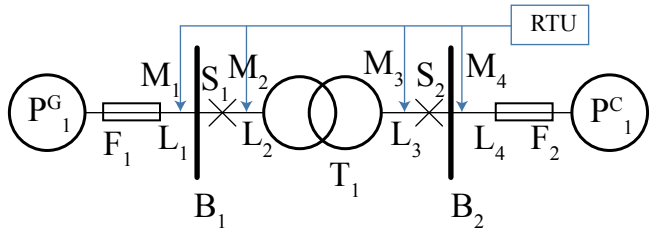


Figure 4.12: Power lines L_1 and L_4 are equipped with fuses F_1 and F_2, respectively.

positioned at switch j, $R_i.S = S_j$. The properties of protective relays are provided in Table 4.1. A symbol of a protective relay is shown in Figure 4.13. In Figure 4.14, switch S_1 has a protective relay R_1 assigned to it.

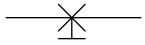


Figure 4.13: Symbol of a protective relay.

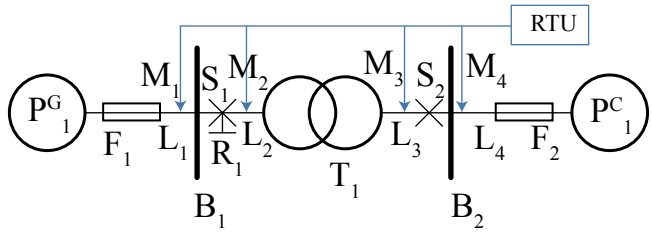


Figure 4.14: Switches S_1 and S_2 located at power lines L_2 and L_3, respectively.

Interlocks

Switches can be interlocked in order to provide a certain current capacity. They can be static or dynamic. An interlock $K_i \in \mathcal{K}$ has a set of interlocked switches $K_i.S$. A *static interlock* guarantees a minimum number of closed (connected) switches $K_i.CS_{min}$, and a *dynamic interlock* defines a minimum guaranteed current capacity $K_i.I_{min}$. For example, if there are three power lines which have a maximum current threshold of $L_1.I_{max} = 100A$, $L_2.I_{max} = 200A$ and $L_3.I_{max} = 300A$ and a dynamic interlock K_1 over those power lines has a minimum guaranteed supply of $K_1.I_{min} = 250A$, then either at least power line L_3 must be closed or both power lines L_1 and L_2. In this example, a static interlock could guarantee that at least 2 switches are closed if $K_2.CS_{min} = 2$. $\mathcal{K}_{static}$ is the set of all static interlocks, $\mathcal{K}_{dynamic}$ contains all dynamic interlocks.

Interlocks do not have a graphical representation, but they are configured within the RTU that controls the local system.

4.3.2 Topology

The topology is a description of the static properties of the power distribution system, for example, how the elements of the power distribution system are connected, and their physical properties. The DSOs maintain the current version of the topology of their system and use it, for example, in the EMS for dimensioning the power distribution system. Describing all elements with the parameters listed in Table 4.1 provides the topology of the example system depicted in Figure 4.14. For example, $L_1 = (P_1^G, B_1)$ means that power line 1 is connecting generator P_1^G with bus B_1, while $S_2 = (L_3.B_2)$ denotes that switch 2 is a switch on line L_3, on the side of bus B_2.

4.3.3 System State

The state of the power distribution system refers to all current values which can change in the system over time. The system state can be described by five vectors indicating: (i) the states of the switches, (ii) the state of the fuses, (iii) the sensor readings, (iv) the power consumption and production, and (v) the position of the transformer taps.

- Vector $\mathbf{S} = [S_1.st, S_2.st, ..., S_{|\mathcal{S}|}.st]$ of size $|\mathcal{S}|$ denotes the state of all switches in the system.
- Vector $\mathbf{F} = [F_1.st, F_2.st, ..., F_{|\mathcal{F}|}.st]$ is of size $|\mathcal{F}|$ and summarizes the states of all fuses present in the system.
- The readings from one sensor can be written as a pair of the measured current and voltage: $(L_i.M.I, L_i.M.V)$. Vector $\mathbf{M}$ collects those pairs for all sensors: $\mathbf{M} = [(L_1.M.I, L_1.M.V), ..., (L_{|M|}.M.I, L_{|M|}.M.V)]$, and is of size $|\mathcal{M}|$.
- Vector $\mathbf{P} = [P_1^G.pv, ...P_{|\mathcal{P}^G|}^G.pv, P_1^C.pv, ..., P_{|\mathcal{P}^C|}^C.pv]$ for $|\mathcal{P}^G|$ sources and $|\mathcal{P}^C|$ consumers, denotes the loads and sources of power.
- Finally, the set of positions of the transformer tap is denoted as vector $\mathbf{T} = [T_1.p, T_2.p, ..., T_{|\mathcal{T}|}.p]$ of size $|\mathcal{T}|$.

Now, the system state Y can be written as a tuple that consists of the above five vectors: $Y = (\mathbf{S}, \mathbf{F}, \mathbf{M}, \mathbf{P}, \mathbf{T})$ and is used in the following to determine whether the system state is consistent and safe, as explained in Section 4.4.

Table 4.1: Properties of the elements of the proposed modelling formalism.

Element	Property	Symbol
Power network	Model	Ω
	State	Y
Power generators and consumers	Combined set	$\mathcal{P} = \mathcal{P}^G \cup \mathcal{P}^C$
	Set of power generators	$\mathcal{P}^G = \{P_1^G, ..., P_{\lvert\mathcal{P}^G\rvert}^G\}$
	Set of power consumers	$\mathcal{P}^C = \{P_1^C, ..., P_{\lvert\mathcal{P}^C\rvert}^C\}$
	Position	$P_i^x.pos = L_j$ for $x \in \{G, C\}$
	Power value	$P_i^x.pv$ for $x \in \{G, C\}$
	Vector of all power values	$\mathbf{P}$
Buses	Set of buses	$\mathcal{B} = \{B_1, ..., B_{\lvert\mathcal{B}\rvert}\}$
	Vector of incoming lines	$\mathbf{B}_i.in = [L_j, ..., L_n]$
	Vector of outgoing lines	$\mathbf{B}_i.out = [L_c, ..., L_h]$
Transformers	Set of transformers	$\mathcal{T} = \{T_1, ..., T_{\lvert\mathcal{T}\rvert}\}$
	Current transformer rate	$T_i.r$
	Current tap switch position	$T_i.p$
	All possible tap switch positions and ratios	$T_i.t$
	Incoming power line	$T_i.in = L_p$
	Outgoing power line	$T_i.out = L_q$
	Vector of current positions of all tap switches	$\mathbf{T}$
Power lines	Set of power lines	$\mathcal{L} = \{L_1, ..., L_{\lvert\mathcal{L}\rvert}\}$
	Position	$L_i.pos = (B_k, B_n)$
	Maximum current	$L_i.I_{max}$
	Reference voltage	$L_i.V_{ref}$
	Meter (side of B_k)	$L_i.B_k.M = M_d$
	Vector of meters on line L_i	$\mathbf{L}_i.M = [M_d, ..., M_h]$
	Switch (side of B_k)	$L_i.B_k.S = S_e$
	Vector of switches on line L_i	$\mathbf{L}_i.S = [S_e, ..., S_o]$
	Fuse (side of B_k)	$L_i.B_k.F = F_u$
	Vector of fuses on line L_i	$\mathbf{L}_i.F = [F_u, ..., F_y]$
Switches	Set of switches	$\mathcal{S} = \{S_1, ..., S_{\lvert\mathcal{S}\rvert}\}$
	Position	$S_i.pos = L_i.B_n$
	State of the switch	$S_i.st$
	Vector of states of all the switches	$\mathbf{S}$

(Continued on next page)

Element	Property	Symbol
Meters	Set of meters	$\mathcal{M} = \{M_1, ..., M_{\lvert\mathcal{M}\rvert}\}$
	Position	$M_i.pos = L_i.B_n$
	Measured current	$M_i.I$
	Measured voltage	$M_i.V$
	Current set point	$M_i.I_{sp}$
	Voltage set point	$M_i.V_{sp}$
	Vector of states of all readings	$\mathbf{M}$
Fuses	Set of fuses	$\mathcal{F} = \{F_1, ..., F_{\lvert\mathcal{F}\rvert}\}$
	Position	$F_i.pos = L_i.B_n$
	State of the fuse	$F_i.st$
	Cutting current	$F_i.I_{max}$
	Vector of states of all fuses	$\mathbf{F}$
Protective relays (circuit breakers)	Set of protective relays	$\mathcal{R} = \{R_1, ..., R_{\lvert\mathcal{R}\rvert}\}$
	Position (on a switch)	$R_i.S = S_j$
	Cutting current	$R_i.I_{max}$
Interlocks	Set of interlocks	$\mathcal{K} = \mathcal{K}_{dynamic} \cup \mathcal{K}_{static} = \{K_1, ..., K_{\lvert\mathcal{K}\rvert}\}$
	Interlocked switches	$K_i.S = \{S_j, ..., S_n\}$
	Minimum number of closed switches	$K_i.CS_{min}$
	Minimum current capacity	$K_j.I_{min}$

4.4 State Evolution and Local Monitoring

In the previous section, the system structure and state was formally defined. This section formalizes the way the system state changes over time based on events. First, the influence of he so-called *events* on the system state is explained in Section 4.4.1. Sections 4.4.2 and 4.4.3 provide sets of rules that are evaluated, before Section 4.4.4 shows how such monitoring can be done.

4.4.1 Events

The system state can change upon receiving any new information, for example, sensor measurements with different voltage and/or current readings result in a new state. Different power values of the sources or loads also update the system state. Moreover, any command to open or close any of the switches, or changing the tap switch position also brings the system to another state. Two types of

events are distinguished:

(i) *readings* update the state Y to a new state $Y' = (\mathbf{S}', \mathbf{F}', \mathbf{M}', \mathbf{P}', \mathbf{T}')$, and
(ii) *commands* provide a new state Y' with an updated vector $\mathbf{S}'$, collecting the states of the switches, or/and new vector of transformer states $\mathbf{T}'$

Every new state Y' has to then be evaluated or precalculated using physical constraints and/or safety requirements, as explained next.

4.4.2 Physical Constraints

Physical constraints are based on physical laws, which must always hold. If these constraints are violated, this means that the monitoring tool is seeing a non-consistent image of the power system, and, therefore, it can infer that the observed state differs from the real state. For example, for every bus, the sum of the incoming currents has to be equal to the sum of the outgoing currents on that bus. If that is not the case, this means that one (or more) of the measurements must be false. Below we explain 6 physical constraints that are analysed in the scope of this thesis.

The first two rules refer to properties of each bus. The first physical constraint refers to the example provided above. **Kirchhoff's current law**, given in the physical constraint P1, states that the incoming current at a bus must equal the outgoing current. In case there is more than a single voltage meter at a single bus, P2 checks that all reported voltages at the bus are equal. Even if a line is disconnected, the meter still reports the voltage at the bus as it is located nearer to the bus than switches, fuses and protective relays are. Note that the notation $L_j.B_i.M$ refers to the meter located at the power line L_j on the side of the bus B_i (see Table 4.1). Physical constraints P1 and P2 are defined as:

$$\forall B_i \in \mathcal{B} : \sum_{L_j \in \mathbf{B}_i.in} L_j.B_i.M.I = \sum_{L_k \in \mathbf{B}_i.out} L_k.B_i.M.I, \tag{P1}$$

and:

$$\forall B_i \in \mathcal{B}, \forall L_j, L_k \in \mathbf{B}_i.in \cup \mathbf{B}_i.out : L_j.B_i.M.V = L_k.B_i.M.V. \tag{P2}$$

The next two rules are evaluated for each power line. Physical constraint P3, states that if a switch is open, then there may not be current on the power line:

$$\forall L_i \in \mathcal{L} : \exists S_j \in \mathbf{L}_i.S : S_j = 0 \Rightarrow \forall M_k \in \mathbf{L}_i.M : M_k.I = 0. \tag{P3}$$

Physical constraint P4, ensures that both the current and the voltage are equal at the beginning and the end of each power line[1]:

$$\forall L_i \in \mathcal{L},\ \forall M_j, M_k \in \mathbf{L}_i.M : M_j.I = M_k.I \wedge M_j.V = M_k.V. \tag{P4}$$

Physical constraints P5a and P5b refer to generators and consumers and verify that the power formula $P = I \cdot V$ holds:

$$\forall P_i^G \in \mathcal{P}^G : P_i^G = P_i^G.pos.M.I \cdot P_i^G.pos.M.V, \tag{P5a}$$

$$\forall P_i^C \in \mathcal{P}^C : P_i^C = (-1) \cdot P_i^C.pos.M.I \cdot P_i^C.pos.M.V. \tag{P5b}$$

Finally, physical constraints P6a and P6b require that the transformation ratio at transformers is consistent with measurements:

$$\forall T_i \in \mathcal{T} : T_i.out.M.V = \frac{T_i.in.M.V}{T_i.r(T_i.p)}, \tag{P6a}$$

$$\forall T_i \in \mathcal{T} : T_i.out.M.I = T_i.r(T_i.p) \cdot T_i.in.M.I. \tag{P6b}$$

4.4.3 Safety Requirements

Safety requirements ensure that the electrical grid is operated in a safe way, protection components are working properly, and all customers are supplied with energy. While it is not physically possible to disobey the physical constraints, it is feasible to violate the safety requirements. For example, every power line has a current threshold I_{max} which may not be exceeded. It is though possible to transport a current higher than I_{max}, however, it is deemed undesirable to do this. Below we present 9 safety requirements that are considered in the scope of this thesis.

The first two safety requirements evaluate the properties of power lines. Requirement (R1) comes from physical properties of a power lines and ensures that the current meets the defined safety threshold. The second safety requirement (R2) validates that the voltage level is within its allowed bounds. The bound $\alpha = \pm 10\%$ for voltage is derived from the *Harmonization Document* from the European Committee for Electrotechnical Standardisation [23]. All low voltage areas ($230V/400V$) must comply with this rule. Higher level voltage grids may have slightly different safety thresholds, but in this work we assume a constant safety boundary factor for all voltage levels [72]. Safety requirements R1 and R2 are defined as:

[1] We neglect the resistance on the power line.

$$\forall L_i \in \mathcal{L}, \forall M_j \in \mathbf{L}_i.M : M_j.I \leq L_i.I_{max}, \tag{R1}$$

$$\forall L_i \in \mathcal{L}, \forall M_j \in \mathbf{L}_i.M : M_j.V \in [(1-\alpha) \cdot L_i.V_{ref}; (1+\alpha) \cdot L_i.V_{ref}]. \tag{R2}$$

The following two safety requirements evaluate the state of fuses and protective relays. Requirement R3 ensures that all fuses and protective relays are functional and requirement R4 checks whether the cutting current I_{max} is exceeded, as follows:

$$\forall F_i \in \mathcal{F} : F_i.st = 1 \wedge \forall R_j \in \mathcal{R} : R_j.S.st = 1, \tag{R3}$$

$$\begin{aligned} &\forall F_i \in \mathcal{F} : (F_i.pos.M.I < F_i.I_{max}) \\ &\wedge \forall R_j \in \mathcal{R} : (R_j.pos.M.I < R_j.I_{max}). \end{aligned} \tag{R4}$$

The tap position of transformers is evaluated with respect to the allowed target voltage range. Requirement R5a calculates the expected effect of the transformer rate on the reference input voltage and checks whether the output is within the allowed voltage bounds. Again, the allowed bounds are defined with respect to the reference voltage: $V_{ref} \pm \alpha \cdot V_{ref}$, where according to [23] $\alpha = 10\%$. Safety requirement R5b calculates the impact of the transformer on the measured input voltage and ensures that the output is within allowed bounds:

$$\begin{aligned} &\forall T_i \in \mathcal{T} : \\ &T_i.in.V_{ref} \cdot T_i.r(T_i.p) \in [(1-\alpha) \cdot T_i.out.V_{ref}; (1+\alpha) \cdot T_i.out.V_{ref}], \end{aligned} \tag{R5a}$$

$$\begin{aligned} &\forall T_i \in \mathcal{T} : T_i.in.M.V \neq 0 \Rightarrow \\ &T_i.in.M.V \cdot T_i.r(T_i.p) \in [(1-\alpha) \cdot T_i.out.V_{ref}; (1-\alpha) \cdot T_i.out.V_{ref}]. \end{aligned} \tag{R5b}$$

The next requirement is in the best interest of the distribution system operators, as they have to compensate their customers (power suppliers) financially when power outages occur. Therefore, reliable supply to all customers is of utmost importance in power distribution. Safety requirement R6 ensures that all consumers receive positive voltage and that switch states are set correctly:

$$\forall P_i^C \in \mathcal{P}^C : (P_i^C.pos.M.V > 0 \wedge \forall S_j \in P_i^C.pos : S_j.st = 1). \tag{R6}$$

The power generated in total should equal the power consumed in the electrical grid. This is validated through R7:

$$\sum_{P_i^G \in \mathcal{P}^G} P_i^G.pv = - \sum_{P_j^C \in \mathcal{P}^C} P_j^C.pv. \tag{R7}$$

The monitoring algorithm should also detect whether set points are in an appropriate proximity to the physical threshold values (R8a, R8b). If they are set too high, the SCADA system might not alert about unsafe incidents, and if they are too low, there might be too many false alerts. The value of the set point, in reference to the physical threshold, should be chosen by the operator. Here, a safety boundary of β is shown. The rules are defined as follows:

$$\forall M_i \in \mathcal{M} M_i.V_{sp} \in [(1-\beta) \cdot L_j.V_{ref}; (1+\beta) \cdot L_j.V_{ref}], \text{ if } M_i \in \mathbf{L}_j.M, \quad \text{(R8a)}$$

$$\forall M_i \in \mathcal{M} M_i.I_{sp} \in [(1-\beta) \cdot L_j.I_{max}; (1+\beta) \cdot L_j.I_{max}], \text{ if } M_i \in \mathbf{L}_j.M. \quad \text{(R8b)}$$

Finally, potential violations of interlocks must be recognized. Safety requirement R9a refers to static interlocks and it checks if a number of switches that belong to an interlock K is connected, while R9b refers to dynamic interlocks and it verifies that the maximum capacity of the connected lines ensures a minimum required current $K.I_{min}$:

$$\forall K \in \mathcal{K}_{static} : \sum_{S_i \in K.S} (S_i.st) \geq K.CS_{min}, \quad \text{(R9a)}$$

$$\forall K \in \mathcal{K}_{dynamic} : \sum_{S_i \in K.S} (S_i.st \cdot S_i.L_j.I_{max}) \geq K.I_{min}, \text{ if } S_i \in \mathbf{L}_j.S. \quad \text{(R9b)}$$

4.4.4 Outline of the Algorithm

Monitoring the system state locally at each substation allows to (i) validate whether the system evolves consistently, and (ii) evaluate whether the execution of a command coming from the control room will lead to a safe system state in advance. This is done by checking the physical constraints $P1$–$P6$ and the safety requirements $R1$–$R9$. The execution of a command is considered to be unsafe if it leads to a system state that violates one of the requirements $R1$–$R9$. On the other hand, if one of the physical constraints $P1$–$P6$ is violated, this indicates that the information that is available on the system state must be incorrect.

In Section 4.3.3, the system state was defined as a tuple $Y = (\mathbf{S}, \mathbf{F}, \mathbf{M}, \mathbf{P}, \mathbf{T})$. Let us distinguish two system state views maintained by the local IDS: (i) Y_O, which is the perceived system state, that the Operator sees by only analysing the sensor readings; and (ii) Y_C, which is the Calculated system state obtained using the physical laws of the system. Note that in an ideal world the two system states, Y_C and Y_O, are the same (within some error margin). Upon

an event, as defined in Section 4.4.1, the expected system state is evaluated with respect to its consistency and safety, that is, the monitoring tool checks whether all the physical restrictions and safety requirements mentioned in Sections 4.4.2 and 4.4.3 are satisfied.

A flowchart of the algorithm used to perform the monitoring is showed in Figure 4.15 and explained below.

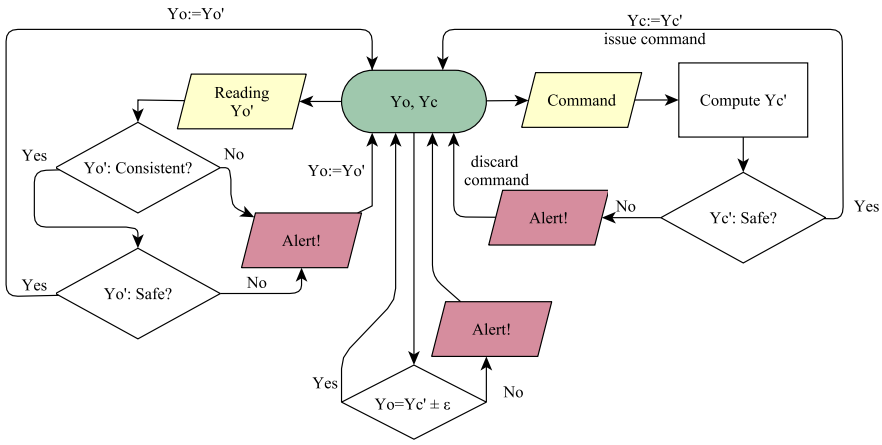


Figure 4.15: Flow chart illustrating the algorithm detecting undesired commands and not consistent states. The current state is marked as a green oval, inputs (events) are highlighted in yellow, and output alerts are marked in red.

The current system state is (Y_O, Y_C), denoted as a green oval. The left part of Figure 4.15 shows the procedure taken when receiving new sensor readings. As mentioned in Section 4.4.1, new readings mean that we have reached an entirely new system state $Y'_O = (\mathbf{S}', \mathbf{F}', \mathbf{M}', \mathbf{P}', \mathbf{T}')$, which could be unsafe and/or inconsistent. Therefore, two checks need to be performed:

(i) the consistency check, which verifies the physical constraints: $P1$ - $P6$, and

(ii) the safety check, which is done by verifying the applicable safety requirements: $R1$ - $R9$.

If the system is both consistent and safe, the old state Y_O is simply replaced by the new reading Y'_O. Otherwise an alert is generated with the reasons listed in Table 4.2.

Table 4.2: Alerts after a sensor reading

safe?	consistent?	alert
-	no	the information from sensor(s) is incorrect: no knowledge about the system safety
no	yes	system is in danger: immediate reaction of system operator required

The right part of Figure 4.15 shows the actions caused by a new command, issued by, for example, the operator. When receiving a new command, that is, a new vector $\mathbf{S}'$ or $\mathbf{T}'$, this command is first "executed" in the model, based on knowledge of the current state Y_O. If the predicted new state Y'_C is safe, the command is executed on the actual system. Otherwise, if the predicted state is unsafe, the command is not executed and a proper alert is sent to the operator, possibly via a different communication channel.

In the lower cycle in Figure 4.15, the algorithm also compares the current state of the system Y_O, as viewed by the local IDS, to the previously calculated system state Y_C. If these two states are not the same (with an error margin of ϵ), this has to be reported to the operator (at the central location), since it indicates a potentially dangerous situation.

4.5 Using the Modelling Formalism

In this section, the proposed modelling formalism and algorithm are used to validate sensor readings and predict the outcome of issued commands. Several examples are analysed: Section 4.5.1 describes a scenario where the safety requirements and physical constraints are evaluated for the normal operation of a system; Section 4.5.2 shows that based on a new reading, a faulty sensor or measurement can be discovered; and Section 4.5.3 shows two scenarios, where based on a new command, the monitoring tool is able to determine that the new state is unsafe, and therefore decide that the command should be discarded.

4.5.1 Normal Operation

Let us first consider an example of a system as described in Figure 4.16. For presenting its normal operation, the entire system is described. The maximum currents are given in column $L_X.I_{max}$ of Table 4.3, for $X = \{1, ..., 9\}$. Moreover, let us assume that the initial state of the system depicted is $Y_O = Y_C = (\mathbf{S}, \mathbf{F}, \mathbf{M}, \mathbf{P}, \mathbf{T})$, where:

$$\mathbf{S} = [S_1.st, ..., S_{12}.st] = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],$$
$$\mathbf{F} = [],$$
$$\mathbf{P} = [P_1^G, P_2^G, P_1^C, P_2^C, P_3^C, P_4^C] = [98.9, 52.9, -20.7, -80.5, -29.9, -20.7],$$
$$\mathbf{T} = [].$$

Initial states of all switches are set to "1" which indicates that switches are closed (and power lines are connected). As neither fuses nor transformers are present in the Figure 4.16, vectors **F** and **T** are empty. Vector **P** collects the (positive) values of power produced by generators and (negative) values of power consumed by loads. The initial sensor readings, and therefore the **M** vector can be found in Table 4.3, under the column "Reading 1". It presents measurements of current (I) and voltage (V) taken at power line X on the side of bus B_1 and B_2 in subcolumns $L_X.B_1.M$ and $L_X.B_2.M$, respectively. Column "X" in Table 4.3 provides numbers 1-9 of power lines, column $L_X.I_{max}$ shows the maximum current of these power lines.
The physical constraints and the safety requirements can be evaluated for this initial state. First, the physical constraints are investigated. If they are not

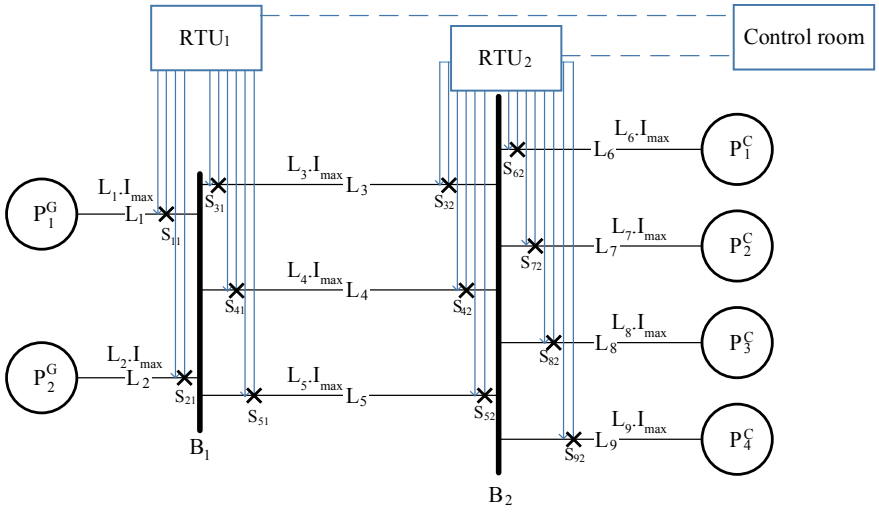


Figure 4.16: The basic power distribution scenario: black lines represent the physical system, blue lines represent the communication network.

Table 4.3: The maximum current on the power lines and the sensor readings

X	$L_X.I_{max}$	reading 1				reading 2			
		$L_X.B_1.M$		$L_X.B_2.M$		$L_X.B_1.M$		$L_X.B_2.M$	
		I	V	I	V	I	V	I	V
1	0.8	0.43	230	-	-	0.43	230	-	-
2	0.5	0.23	230	-	-	0.23	230	-	-
3	0.3	0.22	230	0.22	230	0.22	230	0.22	230
4	0.4	0.22	230	0.22	230	**0.35**	230	0.22	230
5	0.5	0.22	230	0.22	230	0.22	230	0.22	230
6	0.3	-	-	0.09	230	-	-	0.09	230
7	0.5	-	-	0.35	230	-	-	0.35	230
8	0.3	-	-	0.13	230	-	-	0.13	230
9	0.3	-	-	0.09	230	-	-	0.09	230

met, the monitoring tool cannot proceed with performing the other checks, as it operates on inconsistent data. The evaluation is presented in Table 4.4.
Table 4.4 shows the evaluation of physical constraints in the presented system. Columns "rule" and "name" specify the number and the name of the physical constraint. Column "values" provides the values or calculations that satisfy or violate this constraint, and the last column provides additional comment. As shown in Table 4.4, all the physical constraints that can possibly be evaluated in the presented topology are met. Some of the constraints are not applicable in the analysed system. By not applicable, we mean that such constraint is not possible to evaluate due to lack of certain precondition, such as presence of a certain element or its status. For example, all the switches in the presented system are connected, therefore, constraint $P3$ is not evaluated, as it requires that any switch is open. Moreover, because the topology does not contain a transformer, physical constraints $P6a$ and $P6b$ are not evaluated. However, the remaining physical constraints hold: $P1$, that is, Kirchhoff's law, holds for both buses B_1 and B_2. As shown in Table 4.3, the voltage is within the proper bounds for all the sensor measurements, which satisfies constraint $P2$. For three power lines, that is, L_3, L_4 and L_5, constraint P4 can be evaluated. As the values of the current and voltage are equal at both sensors on each line, the constraint is met. Finally, physical constraints $P5a$ and $P5b$ are also satisfied, as the power of both, the sources and the loads equals the product of voltage and current on the connected power lines.

[2]We neglect the power line resistance.

Table 4.4: Evaluation of the safety requirements for the system in Fig. 4.16 and parameters and sensor readings in column "Reading 1" of Tab. 4.3

rule	name	values	comment
P1	Kirchhoff's law	$B_1 : 0.43+0.23[A] = 0.22+0.22+0.22[A]$ $B_2 : 0.22+0.22+0.22[A] = 0.09+0.35+0.13+0.09[A]$	Holds for all the buses.
P2	Voltage on a single bus	$B_1 : 230V$, $B_2 : 230V$	For all power lines on the same bus the measured voltage is equal
P3	No current on disconnected line	n/a	All the switches are closed.
P4	Current and voltage equal on the same line[2]	$L_3.B_1.M.I = L_3.B_2.M.I = 0.22A$, $L_4.B_1.M.I = L_4.B_2.M.I = 0.22A$, $L_5.B_1.M.I = L_5.B_2.M.I = 0.22A$, and $L_3.B_1.M.V = L_3.B_2.M.V = L_4.B_1.M.V = L_4.B_2.M.V = L_5.B_1.M.V = L_5.B_2.M.V = 230V$	For the lines L_3, L_4 and L_5 this holds.
P5a	Power - generators	$P_1^G = 98.9W = 0.43A \cdot 230V$, $P_2^G = 52.9W = 0.23A \cdot 230V$	For all the power sources it holds that $P = V \cdot I$
P5b	Power - consumers	$P_1^C = -20.7W = -0.09A \cdot 230V$, $P_2^C = -80.5W = -0.35A \cdot 230V$, $P_3^C = -29.9W = -0.13A \cdot 230V$, and $P_4^C = -20.7W = -0.09A \cdot 230V$	For all the consumers it holds that $P = V \cdot I$
P6a	Transformation ratio (voltage)	n/a	No transformers in the system.
P6b	Transformation ratio (current)	n/a	No transformers in the system.

Since all physical constraints that could be evaluated are satisfied, also the safety requirements can be tested. Table 4.5 shows the summary of the evaluation of the safety requirements. Similar to the previous table, column "values" lists the measurements that satisfy or violate the analysed requirement, and "comment" provides additional explanation.

Note that, as before, part of the safety rules cannot be evaluated. Since no protective relays, fuses, or transformers are present in the system, safety requirements $R3$ and $R4$ cannot be evaluated. Also, the configuration of the RTU where the set points are specified is not discussed, therefore, requirements $R8a$ and $R8b$ are not assessed. Moreover, no static or dynamic interlocks are defined in this system, therefore, requirements R9a and R9b do not need to be analysed. However, as shown in Table 4.5, the applicable requirements are satisfied. Requirement $R1$ holds as the current values on each of the power lines does not exceed the maximum allowed current for that power line. The

Table 4.5: Evaluation of the safety requirements of the system in Fig. 4.16 and parameters and sensor readings in column "Reading 1" of Tab. 4.3

rule	name	values	comment
R1	Maximum current of power lines	see Table 4.3 under "Reading 1"	For each power line, the measured current value is below the respective maximum current value
R2	Voltage boundary	$230V \in [207; 253]V$ (see also Table 4.3 under "Reading 1")	For all power lines, the measured voltage values lie within the allowed range
R3	State of fuses and protective relays	n/a	No fuses or protective relays in the system.
R4	Maximum current of fuses and protective relays	n/a	No fuses or protective relays in the system.
R5a	Tap position influence on the reference voltage	n/a	No transformers in the system.
R5b	Tap position influence on the measured voltage	n/a	No transformers in the system.
R6	Customer connection	All the switches are connected.	The loads are connected to sources of power.
R7	The sum of the produced and consumed power	$98.9 + 52.9 - 20.7 - 80.5 - 29.9 - 20.7 = 0$	The sum of powers is equal to zero.
R8a	The voltage set points	n/a	Configuration of the RTU is not discussed.
R8b	The current set points	n/a	Configuration of the RTU is not discussed.
R9a	Static interlocks	n/a	No interlocks defined for the system.
R9b	Dynamic interlocks	n/a	No interlocks defined for the system.

measured voltage values also lie within the allowed voltage value range, which satisfies requirement $R2$. $R6$ is fulfilled as all the switches are connected in the analysed scenario. Finally, requirement $R7$ holds, because the sum of the generated power equals the sum of the consumed one.

In the next sections, only the *applicable* physical constraints and safety requirements, that is, the ones that refer to the existing elements of the presented system, are analysed.

4.5.2 Faulty Sensor

In this scenario the system state from Section 4.5.1 is used. Assume that a new set of readings Y'_O was received. The values of Y'_O are presented in column "Reading 2" of Table 4.3. Note that, all the readings are the same, except for $L_4.B_1.M.I$, which in the Table 4.3 is marked with bold font. The new reading is analysed to determine if it meets the physical constraints P1-P6 and the safety requirements R1-R9. The evaluation of the physical constraints is presented in Table 4.6.

Table 4.6: Evaluation of the physical constraints for the system in Fig. 4.16 and parameters and sensor readings in column "Reading 2" of Tab. 4.3

rule	name	values	comment
P1	Kirchhoff's law	B_1 : $0.43 + 0.23[A] \neq 0.22+0.35+0.22[A]$ B_2 : $0.22 + 0.22 + 0.22[A] = 0.09 + 0.35 + 0.13 + 0.09[A]$	Does not hold for bus B_1.
P2	Voltage on a single bus	$B_1 : 230V$, $B_2 : 230V$	For all power lines on the same bus the measured voltage is equal
P4	Current and voltage equal on the same line[3]	$L_4.B_1.M.I$ =0.35, $L_4.B_2.M.I = 0.22A$, $L_4.B_1.M.I \neq L_4.B_2.M.I$	For line L_4, this does not hold.
P5a	Power - generators	$P_1^G = 98.9W = 0.43A \cdot 230V$, $P_2^G = 52.9W = 0.23A \cdot 230V$	For all the power sources, it holds that $P = V \cdot I$
P5b	Power - consumers	$P_1^C = -20.7W = -0.09A \cdot 230V$, $P_2^C = -80.5W = -0.35A \cdot 230V$, $P_3^C = -29.9W = -0.13A \cdot 230V$, and $P_4^C = -20.7W = -0.09A \cdot 230V$	For all the consumers, it holds that $P = V \cdot I$

The following observations were made:

P1 The sum of currents in bus 1 does not sum to zero: $0.43 + 0.23 = 0.66 \neq 0.79 = 0.22 + \mathbf{0.35} + 0.22$. This suggests that at least one of the sensors gave a wrong value: $L_1.B_1.M.I$, $L_2.B_1.M.I$, $L_3.B_1.M.I$, $L_4.B_1.M.I$, or $L_5.B_1.M.I$.

P4 For line L_4 it is observed that: $L_4.B_1.M.I = \mathbf{0.35} \neq 0.22 = L_4.B_2.M.I$. This suggests that at least one of the sensors gave a wrong value: $L_4.B_1.M.I$ or $L_4.B_2.M.I$.

[3]We neglect the power line resistance.

Even without knowing which sensor is broken, from these two observations it can be concluded that the value of $L_4.B_1.M.I$ is incorrect. Note that the above algorithm is able to detect that the system state is not consistent. However, it cannot draw the conclusion as presented above. Knowing that the system state violates the physical constraints, the algorithm will not perform a safety requirements analysis. The SCADA monitoring device can then issue a warning to the operator about a faulty reading; in case the warnings repeat, a conclusion can be drawn that the sensor is broken and needs to be replaced. The state Y_O is updated with Y'_O, and therefore, it is not equal to the previously calculated Y_C. The bottom loop in Figure 4.15 checking whether $Y_O =^{\epsilon} Y_C$ yields an alert notifying the operator about this inconsistency.

4.5.3 Undesirable Command

For the analysis of an undesirable command, two scenarios are investigated which are based on two smaller parts of a real power distribution system. In both scenarios, commands are sent by a hacker to local RTUs, aiming at disrupting the safe operation of the power distribution system. The first scenario shows an attack involving switching, and the second scenario describes an attack involving a tap changer on a transformer.

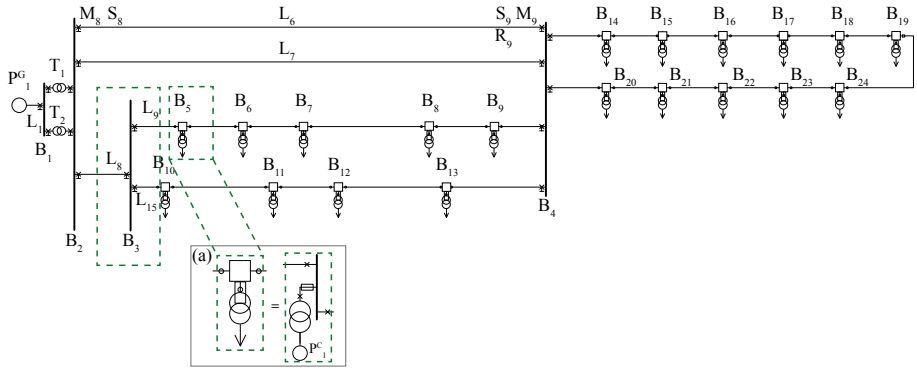


Figure 4.17: A part of the distribution grid

Figure 4.17 depicts a part of a distribution network of a small town in the Netherlands, which is controlled by a SCADA system. Most of the symbols in that figure are introduced in Section 4.3.1, while some shorthand notation of buses are translated into known symbols in box *a*. Two buses, B_3 and B_5 are additionally marked with a green dashed line: these are two case studies

that are analysed later in this section. The power network analysis tool Vision[4] is used to solve the power flow equations, which determine the distribution of current in the system of a defined topology. With each new command, the change has to be applied in the system, and the power flow equations need to be recalculated in order to provide the current sensor readings. This way the input for evaluating physical constraints and safety requirements in each state of the system in Figure 4.17 is obtained.

Switching Scenario

The first scenario focuses on bus B_3. The RTU located in the substation with bus B_3 is equipped with a device, which runs the local monitoring algorithm explained in Section 4.4.4. This RTU sends the sensor measurements taken on the power lines connected to bus B_3 and controls the switches on these power lines. Figure 4.18 shows the part of the system which is supervised by the local monitoring tool.

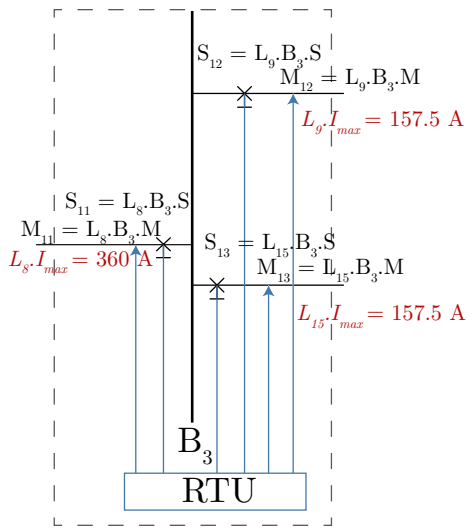


Figure 4.18: Part of the system controlled by the RTU located on bus B_3

To illustrate how the proposed approach complements the existing central approaches, such as the EMS, first it is shown how the system evolves locally and report on both the local and the central view, before the checks performed by the local monitoring tool on bus B_3 are explained in detail. The system state

[4] Accessed via `https://phasetophase.nl/vision-network-analysis.html`.

as displayed on the HMI of the EMS in the central location (control room) is denoted Y_{ems}. The state of the system that is observed locally by the monitoring tool at the RTU is called Y_O, and Y_C indicates the state calculated by the algorithm. Assume that two RTUs, one at bus B_3, and one on bus B_2, are equipped with a local monitoring tool, later referred to as an IDS. Table 4.7, explains steps of this attack scenario in column "action", and summarises the resulting view on the system as seen by the EMS, the IDS on B_2, and the IDS on B_3 in columns EMS, B_2 and B_3, respectively. Assume that a hacker gains control over communication channel between the SCADA server and the RTUs.

Table 4.7: Switching attack scenario

action	EMS	B_2	B_3
The hacker sends false information to EMS: replaying the data from the past	EMS relies on false information	n/a	n/a
The hacker sends a command to RTU on bus B_2 to open line L_6	command bypassed EMS; EMS relies on false information	local IDS computes $Y_{C_{B_2}}$ and does not see any violation. Command is executed by RTU on bus B_2	the value of the current is increasing on all the connected lines
The hacker sends a command to RTU on bus B_2 to open line L_7	command bypassed EMS; EMS relies on false information	local IDS computes $Y_{C_{B_2}}$ and does not see any violation. Command is executed by RTU on bus B_2	the value of the current is increasing on all the connected lines
System operator (or hacker) sends a command to RTU on bus B_3 to open switch S_{13} for some maintenance	command is allowed by EMS, because based on old information it cannot detect a violation	n/a	local IDS computes the resulting $Y_{C_{B_3}}$ and detects violation. Command is **discarded** and an **alert** is sent to the operator (or hacker...).

In the presented scenario, a hacker sends false information to the central EMS, which is a replayed historical information showing a safe state of the power distribution system. The EMS is therefore corrupted and shows inaccurate data on the screen of the control engineer. In the next step, the hacker opens power line L_6. RTU on bus B_2, even when equipped with a local monitoring tool allows such action, as the maximum current on all its connected lines lies within the allowed bounds.The same happens when the hacker requests to open line L_7.

Table 4.8: Sensor readings in the switching scenario

measurements	prediction	calculation
$S_{11}.st = 1$	$S_{11}.st = 1$	$S_{11}.st = 1$
$S_{12}.st = 1$	$S_{12}.st = 1$	$S_{12}.st = 1$
$S_{13}.st = 1$	$S_{13}.st = 0$	$S_{13}.st = 0$
$M_{11}.I = 203$	$M_{11}.I = 203$	$M_{11}.I = 201$
$M_{11}.V = 10.61$	$M_{11}.V = 10.61$	$M_{11}.V = 10.61$
$M_{12}.I = 60$	$M_{12}.I = 204$	$M_{12}.I = 202$
$M_{12}.V = 10.61$	$M_{12}.V = 10.61$	$M_{12}.V = 10.61$
$M_{13}.I = 144$	$M_{13}.I = 0$	$M_{13}.I = 0$
$M_{13}.V = 10.61$	$M_{13}.V = 10.61$	$M_{13}.V = 10.61$

At the same time, the local monitoring tool at bus B_3 observes an increasing value of current on all the connected power lines. In this situation, all the power needed at bus B_4 has to be transported through bus B_3. However, this information is not shown on the EMS. The operator decides to open switch S_{13}. Knowing the capacity of power lines (as indicated in Figure 4.18), and the transported power (listed in Table 4.8), this RTU can decide whether this action is safe or not. The column "measurements" in Table 4.8 explains the current system state: $S_{11}.st = 1$ means that switch S_{11} (located at power line L_8, see Figure 4.18) is connected, while current (I) and voltage (V) values of the meters M_{11} - M_{13} are given next. The values were obtained by modelling the system from Figure 4.17 in Vision and performing power flow equations. The column "prediction" shows the values for the same elements calculated by the monitoring tool, and column "calculation" shows the results obtained with Vision. It can be observed that the command is going to cause overcurrent in line L_9 (marked as red in Table 4.8), since $L_9.I_{max} = 157.5 < 204 = M_{12}.I$, and may wear a power line off faster. This risky command should be reported to the SCADA control room, and can, for example, be rejected. Note that for this simple analysis the IDS at bus B_3 did not need information about the lines that are not connected to it. Also, note that there are some differences between the predicted resulting system state, and the one obtained from the Vision computation. The difference is most likely a rounding error.

Table 4.7 shows that the first two commands sent by the hacker have been executed, because the IDS on bus B_2 could not detect an unsafe state.

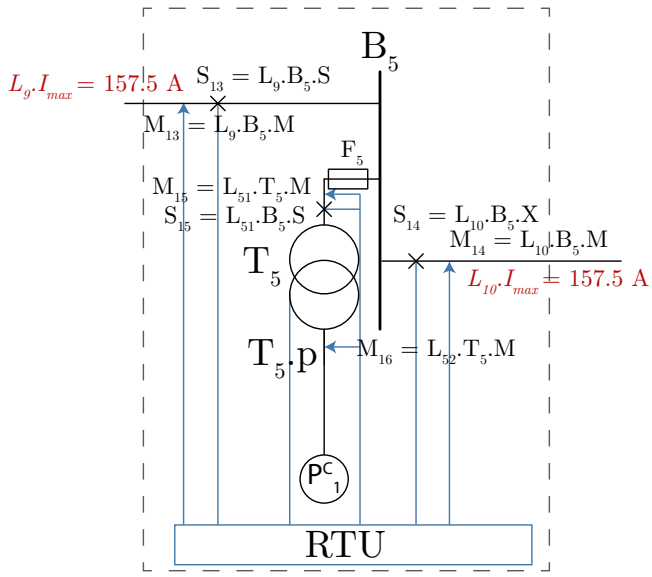


Figure 4.19: Part of the system controlled by RTU on bus B_5

Transformer Tap Switch Scenario

In the next scenario, presented in Figure 4.19, the local IDS is monitoring the messages sent to and from the RTU on bus B_5 and it involves a tap switch of the transformer T_5. Tap switches in low voltage used to be manually set to predefined position, because of the predictable voltage drops. Dynamic control used to not be necessary. However, with renewable energy sources, such as photovoltaic panels, the voltage drop is no longer so predictable. Therefore, more transformers are equipped with online tap switches, which are adjusting the settings automatically and locally [72]. Often it is also possible to overwrite this setting by a command sent from the central control room.

We assume that the hacker takes control over the EMS, like in the previous scenario. The initial values of current and voltage measured for power lines in Figure 4.19 are given in column "measurements" in Table 4.9. The hacker then sends a command to switch the tap setting of the transformer $T_5.p$ from 0 to -2. The values of transformer position $T_5.p$ and the corresponding transformation ratio $T_5.r$ are given under the horizontal line in Table 4.9. The executed command results in an increase of the secondary voltage from 244 V to 256 V (marked in red in column "prediction" for $M_{16}.V$ in Table 4.9). That value is outside of the allowed range, therefore, posing a risk to all connected devices in

Table 4.9: Sensor readings in the transformer scenario

measurements	prediction	calculation
$M_{13}.I = 32$	$M_{13}.I = 32$	$M_{13}.I = 32$
$M_{13}.V = 11.032$	$M_{13}.V = 11.032$	$M_{13}.V = 11.032$
$M_{14}.I = 19$	$M_{14}.I = 19$	$M_{14}.I = 19$
$M_{14}.V = 11.032$	$M_{14}.V = 11.032$	$M_{14}.V = 11.032$
$M_{15}.I = 13$	$M_{15}.I = 13$	$M_{15}.I = 13$
$M_{15}.V = 11.032$	$M_{15}.V = 11.032$	$M_{15}.V = 11.032$
$M_{16}.V = 0.244$	$M_{16}.V = 0.256$	$M_{16}.V = 0.256$
$T_5.p = 0$	$T_5.p = -2$	$T_5.p = -2$
$T_5.r = 26.14$	$T_5.r = 24.9$	$T_5.r = 24.9$

that neighbourhood. The local IDS therefore rejects such a command coming from the central control room and raises an alert at the control room.

4.6 Summary

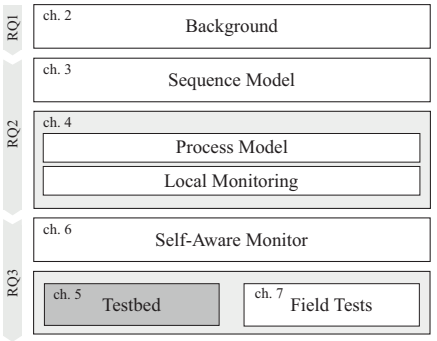
This chapter introduced a power distribution system model that can be used in a local monitoring approach. The models in the related work often (i) require knowledge of the entire system, (ii) are located centrally, (iii) are not able to prevent unsafe commands from being executed, or (iv) require a learning phase that is error prone and often it is not clear how long it should last. Instead, this chapter introduces a model, that uses local information about the physical process, that is controlled by a single RTU, for example, at a substation. Moreover, a new monitoring algorithm was presented. By following the system state evolution, it allows to predict the outcome of the commands that are sent to the RTU. Examples showing how this monitoring approach can be used to detect faulty sensor readings, as well as commands, are provided. In examples involving commands from a hacker, the situation that would lead the system to unsafe state was not visible to the centrally based monitoring system.

Having a concept of the local monitoring algorithm, the next step is to evaluate it. However, this is not a trivial task, as it is not possible to do this in a real system. Therefore, in the next section, a testbed for evaluating a local monitoring algorithm is proposed.

CHAPTER 5

Testbed

This chapter describes a testbed for evaluating the local monitoring approach introduced in the previous chapter. In general, it is difficult to test or validate solutions for critical infrastructures in real-life. Especially, in the proposed method, we want to test whether such monitoring method is able to detect and notify about commands that bring the system into an unsafe state. In real-life, such commands cannot be tested, as we cannot risk that system ends up in an unsafe state. Hence, most of the available security mechanisms are tested in simulation environments where mainly network traffic properties of the control network are being analysed. Physical testbeds with expensive hardware in place are often difficult to access. Therefore, to overcome some of these shortcomings, we propose a testbed for validating the proposed local approach, based on co-simulating the SCADA control network and the power system and monitor both, the SCADA network and the physical process.



This chapter is organised as follows:

- *Section 5.1 provides an overview of related work on co-simulation testbeds.*
- *Section 5.2 describes the proposed testbed and its components.*
- *Section 5.3 presents how the monitoring tool is implemented.*
- *Section 5.4 demonstrates the use of the testbed based on two example scenarios.*
- *Section 5.5 concludes this chapter.*

This chapter has been based on [30, 31, 32].

5.1 Related Work

Research on critical infrastructures requires either a dedicated physical testbed or a simulation testbed. The former is often expensive, not very flexible or hard to access, while the latter often uses software that requires a paid license, or is not suitable for analysis that examines consequences of commands on the controlled physical process. The monitoring approach that we proposed in Chapter 4 aims to perform the detection analysis locally at field stations, hence, there is no need to simulate the entire control network. At the same time, we require simulating the consequences of the cyber commands on the physical system.

In contrast, current co-simulation environments focus on simulating the *entire* network using Omnet++ [6, 87], ns2 [91], RINSE [36] or OPNET [126], and evaluate non-semantic attacks, for example, denial of service attacks on the control network *only*. These fully simulated approaches are highly flexible, while more advanced testbeds [54, 76, 85, 126], may require a connection to emulate real hardware or the use of proprietary software. Non-virtualized testbeds at Distribution System Operators (DSOs) are less flexible and often difficult to access. All simulation-based approaches require a power simulator, like Power World [36, 54], OpenDSS [6, 87], PSFL [91], or MATPOWER-based *Matlab/Simulink* [85, 126] or *Mosaik* with PyPower simulator [130]. The latter easily integrates existing simulators in the smart grid co-simulation framework. Moreover, if needed, new simulators can be attached to the *Mosaik* co-simulation framework by using the provided API.

Table 5.1 summarizes the characteristics of the investigated co-simulation environments. For each framework the availability is specified: either it is available under an Open Source license (OS), or tools are openly available, but the source code is not (OS*). The table indicates if a paid license is required for an element used in the co-simulation environment (LIC), or if it is a physical TestBed (TB) or uses some other Hardware In the Loop (HIL). Next, the integration of the simulator is discussed: if a programming language (such as Python, Java or C++) is specified in the table, the approach has a dedicated interface written in that language which enables the integration of simulators. Two of the approaches use other communication protocols, such as HTTP requests [87] or VPN connections [36]. One approach uses a physical testbed, which requires physical connections between hardware components [76]. The table shows which simulator is used for the SCADA network and for the power grid system. The extensibility of the co-simulation testbed is specified, and all available communication protocols are listed.

Table 5.1: Table comparing related work on testbed environments

source	availability	integration	network simulator	power simulator	extensibility	protocols
Davis et al. [36] (2008)	LIC	Server requests over VPN	RINSE, TCP/IP	PowerWorld	HIL	Modbus, TCP/IP
Lin et al. [91] (2011)	OS*	C++/Java	ns2	PSFL	Not discussed	Not discussed
Levesque et al. [87] (2012)	OS*	HTTP requests	Omnet++	OpenDSS	Not discussed	Not discussed
Sadi et al. [126] (2015)	LIC	C	OPNET	Matlab/ Simulink	Not discussed	Not discussed
Kang et al. [76] (2015)	TB	Physical links	IEC61850 device	PV simulator	Not discussed	Modbus, IEC61850
Koutsandria et al. [85] (2015)	HIL/ TB	C, C++, Python	Modbus	Matlab/ Simulink	Other protocols	Modbus, TCP/IP
Awad et al. [6] (2016)	OS	C++	Omnet++	OpenDSS	Not discussed	TCP/IP, 802.11, Ethernet
Gunthilaka et al. [54] (2016)	LIC	Java	IEC61850 simulator	PowerWorld	Other protocols	IEC-104, IEC61850
Mosaik [131] (2012)	OS	Python	Not built in	Not built in; demo shows use of PyPower	Other simulators can be integrated	Not built in

OS - Open Source, OS* - open source elements, however the source code is not made available, LIC - License required , TB - TestBed, PSLF - Positive Sequence Load Flow, HIL - Hardware In Loop

While older approaches mostly do not investigate particular SCADA protocols, newer approaches are tailored towards Modbus and or substation automation protocols.

5.2 Implementation of the Testbed in Mosaik

From the available simulation testbeds, described in detail in Section 5.1, the co-simulation framework *Mosaik* appeared to be the most flexible. Through including several specifically developed simulators, *Mosaik* was extended with communication network capabilities, to be presented in the following section. Section 5.2.1 describes the *Mosaik* framework, before Section 5.2.2 explains how the physical system elements are simulated. Then, Section 5.2.3 presents the implementation of the SCADA system in the proposed testbed, and Section 5.2.4 provides an analysis of the interaction between co-simulated components.

5.2.1 Mosaik Co-simulation Framework

Mosaik is an open source co-simulation framework written in Python (under GNU LGPL) [113], using a discrete-event simulation library based on SimPy. With the provided API, different custom-made or existing simulators can be connected, while *Mosaik* interfaces their data transfer and tracks the execution order.

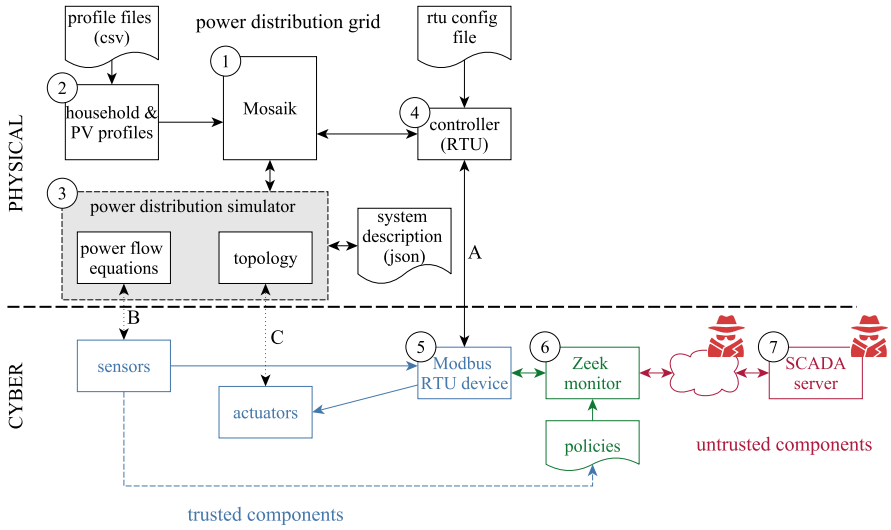


Figure 5.1: Overview of the testbed: the simulated power distribution grid model appears above the dashed line, and trusted (green and blue) and untrusted (red) components appear below the horizontal dashed line.

An overview of the proposed testbed is illustrated in Figure 5.1, with *Mosaik* presented as a box marked with Number 1. Black elements above the horizontal dashed line indicate the *physical* elements of the testbed. They are simulated here, but they refer to the physical parts of the power distribution system. Values provided by this part are considered the "ground truth", that is, if a sensor value on the cyber side deviates from the one on the physical side, then the one on the physical side is considered true. The most significant parts co-simulated in *Mosaik* are: a household and a PV panel profile simulator (Number 2), which are available in the *Mosaik* example scenario[1]; a power distribution simulator (Number 3), and an RTU simulator (Number 4), enabling communication with

[1] `http://mosaik.readthedocs.io/en/latest/installation.html`

the (cyber) Modbus RTU device (Number 5).

The power distribution simulator solves power flow equations using the PyPower package [120] implementing the Newton-Raphson AC power flow method. This simulator has been adapted to allow for topology changes. The proposed extensions and adjustments are described in detail in the following sections.

Below the horizontal dashed line in Figure 5.1, the *cyber* elements of the testbed are presented: the control network, which consists mainly of a Modbus/TCP [105] RTU server (Number 5), a monitoring device (Number 6), and a SCADA server (Number 7).

The integration of the RTU device into the (simulated) physical system is enabled by making the following connections, as indicated in Figure 5.1 by black vertical lines: the controller (RTU) API invokes a thread which creates a simulation of the Modbus RTU device (Connection A). This connection is the *actual* link between the *cyber* and *physical* part of the testbed, therefore in Figure 5.1 it is indicated with a solid line. It allows for the following relations: based on the values obtained from the power flow equation solver via the *Mosaik* interface, the Modbus RTU device determines the sensor measurements (Correspondence B, marked with a dashed line) and forwards them to the control network; upon a command received from a SCADA server in the Modbus RTU device, this device applies the changes on the actuators in the testbed by changing the topology in the power distribution simulator (Correspondence C, marked with a dashed line).

With the physical and cyber system co-simulated within the *Mosaik* framework, it is possible to include all elements necessary to describe the system Ω as explained in Chapter 4. The power buses, branches, transformers are described within the power distribution system simulator, meters, switches, set points and interlocks are described within the controller simulator, and power sources and loads are taken from the household and PV panel simulators, or represented as the reference bus.

5.2.2 Power Distribution System in Mosaik

The power distribution system description consists of houses, PV panels and a distribution network built from buses, branches and transformers. The simulator for houses and PV panels (see Number 2 in Figure 5.1) uses historic consumption profiles, with samples collected every 15 minutes and stored in the form of CSV files. The power distribution system simulator (see Number 3 in Figure 5.1) solves the power flow equations using the Newton-Raphson power solving method and processes the topology changes. It uses a system description stored in a human-readable JSON file. The description formalism includes buses (a reference bus, PQ buses, and isolated buses), branches (or: power

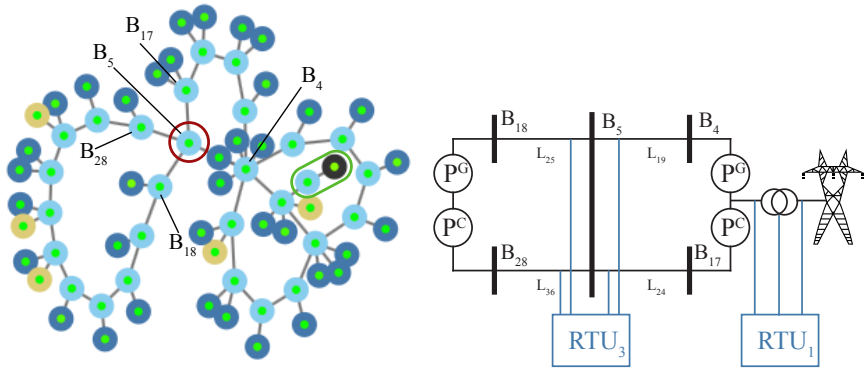
lines) and transformers, which here are a special kind of branch connecting the medium and low voltage buses. In order to solve the power flow equations, one bus marked as the reference bus balances the active and reactive power in the system. Additionally, a PQ bus is is a bus containing a load. In order to analyse a situation where a part of the power grid is disconnected from the reference bus, we have introduced so-called *isolated* buses to PyPower. Isolated buses are buses that are not connected to the remaining grid, if, for example, all the power lines connected to that bus are disconnected. An example of a branch description is shown in Table 5.2. As can be seen, a power line is defined by its ID (name), the IDs of the buses it connects (*from bus* and *to bus*) and its physical properties such as its length, resistance, reactance, capacitance and maximum allowed current. These physical properties, except for the maximum allowed current, are not used in the local monitoring algorithm, however, they are necessary for solving power flow equations. The description of power lines is expanded to include their state: *online* (all switches on the power line are closed) or *offline* (at least one of the switches on the branch is opened).

Table 5.2: Example of a power line description

name	from	to	length [km]	R' $[\frac{\Omega}{km}]$	X' $[\frac{\Omega}{km}]$	C' $[\frac{nF}{km}]$	I_{max} [A]	*online*
L_{13}	B_9	B_4	0.35	0.2542	0.080425	0.0	240.0	true

The power distribution system simulator was extended to take into account changes in the topology, as follows. The initial PyPower simulator is enhanced with topology functions, which identify isolated buses based on information about the state of switches on the branches. This information is obtained from the controller and is then adjusted in the power distribution (topology) model, which in turn is stored in the JSON file. This new model is then used by the power flow equation solver. Detailed information about the power distribution system simulator and its extensions is provided in Appendix C.

The power system used in the following to validate the monitoring approach is based on the topology of a small Dutch town (see Figure 4.17 in Section 4.5.3) and is shown in Figure 5.2. Figure 5.2(a) shows the power system model in *Mosaik*, with the bus B_5 marked with a red circle, and the nodes corresponding to the primary an secondary parts of the transformer are marked with a green oval. These nodes are highlighted, as they will be further used for the analyses. Figure 5.2(b) shows bus B_5 controlled by RTU_3 and the transformer controlled by RTU_1 in more detail, where the rest of the grid is abstracted to load(s) and generator(s).



(a) Mosaik representation of the topology (b) Position of the RTU_1 and RTU_3

Figure 5.2: System under analysis

5.2.3 SCADA System

In order to test the proposed monitoring tool, we require a simulation engine for the controller (an RTU or a PLC) which receives information from the SCADA server and sends commands to actuators influencing the physical process. Hence, the controller is the main interface between the physical process and the control network including the control room.

In the presented testbed, the SCADA system consists of an RTU located in a field station and one SCADA server located in the control room (see Numbers 5 and 7 in Figure 5.1). In the investigated topology, presented in Figure 5.2, the SCADA server communicates with two RTUs: RTU_1 and RTU_3, in two different scenarios. These RTUs communicate with the control server over an untrusted network, using Modbus/TCP protocol. Note that the central SCADA server is assumed to be an untrusted component as well, because of the possibility of the occurrence of insider attacks. The RTUs read the measurements from the sensors on power lines directly connected within the substations they are located at, and they control a set of actuators (switches) connecting power lines attached to bus B_5 or the transformer, see Figure 5.2(b). In the proposed testbed, the *Mosaik* controller (RTU) simulator creates a Modbus RTU device for an RTU, which is a Modbus server listening on TCP port `10502` on the host machine. It uses the PyModbus library[2] to implement the Modbus/TCP protocol [105]. The SCADA server is a Modbus/TCP client created in a Virtual Machine.

[2] `http://pymodbus.readthedocs.io/en/latest/index.html`

RTU_3 controlling bus B_5 stores values of the state of the switches as coils (see Appendix B.1) and the other values (voltage, current) as holding registers. RTU_1 controlling the transformer stores all values (of the state of the tap switch and the voltage and current values) as holding registers. Once a command to change a switch state arrives from the SCADA server, this change is saved on the proper coil within the simulated RTU. The *Mosaik* controller (RTU), upon every simulator step, checks whether the coil value of the RTU device has changed as compared to the stored value. If this is the case, this triggers the RTU to send information about the commands to the power distribution simulator. Details on the Modbus/TCP simulator are given in Appendix C.

5.2.4 Data Exchange Between Simulators

The *Mosaik* framework allows for information exchange between the connected simulators. The order of information exchange between them, their synchronisation and handling data-flows is managed by *Mosaik*'s scheduler. At the beginning of the simulation, all simulators are set to time 0. When *Mosaik* interacts with a simulator, it passes the current simulation time to that simulator. After executing the API `step()` call (see Appendix C), the simulator returns the time at which it wants to perform its next step. The step size does not need to be constant. This is illustrated in Figure 5.3: the simulation actions performed by simulator A at time 0 are valid for the step-size 1, while the simulation actions performed at time 1 are valid for 3 units of simulation time.

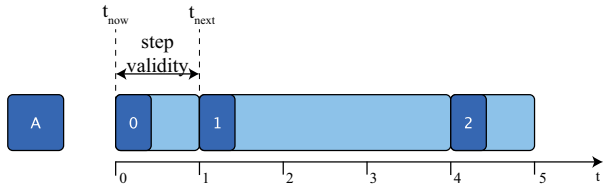


Figure 5.3: Step size and validity of a simulator A [115].

When connecting two simulators, for example, simulator A and B, and adding a relation between them, for example, simulator B provides input for simulator A, the scheduler does the following. For equal step sizes, as shown in Figure 5.4, the scheduler searches for inputs for simulator A, collects them from B, and passes them to A. In our simulation, the step-size for all the simulators has been set to 60 seconds, except for the household and PV panels profile simulators, which have a step-size of 900 seconds. This is because, as mentioned previously, consumption and production profiles consist of samples

collected every 15 minutes, and in the proposed testbed the controller updates information every 60 seconds.

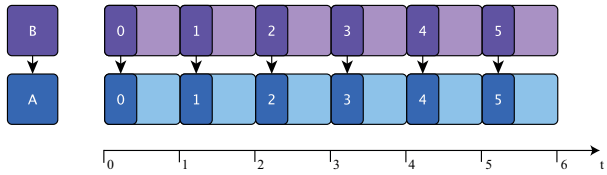
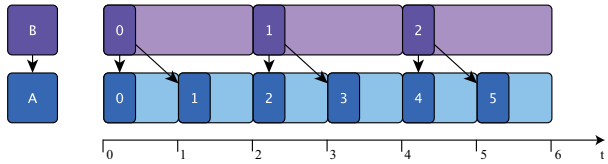
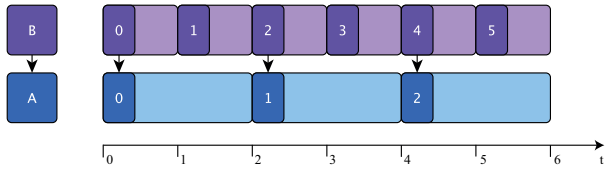


Figure 5.4: Steps and data exchange with input from simulator B to simulator A [115].

In general, choosing different step-sizes for the controller and power distribution simulator is not needed. It would possibly result in one simulator performing calculations more than once on the same data (like relation between simulators B and A in Figure 5.5(a)), or performing calculations, whose result would not be directly used. For example, simulator B in Figure 5.5(b) performs additional simulation step that is not passed to A. Of course, during the additional simulation step, some internal values of simulator B can be, for example, incremented; however, extra changes can be taken into account during the regular simulation step.



(a) Step size of A equals 1, while step size of B equals 2 [115].



(b) Step size of A equals 2, while step size of B equals 1 [115].

Figure 5.5: Various step sizes for simulators in *Mosaik* [115].

One-way data exchange in *Mosaik* framework is sufficient for co-simulation where the simulators do not influence one another. However, simulating an entity providing information to a controller, while the controller provides command input parameters for the simulated entity might be problematic. Consider the RTU simulator and the power distribution simulator present in our testbed. The simulated RTU not only executes commands issued by the power distribution simulator, but also sends sensor readings to the power distribution simulator, which can then be requested by the control room. This cyclic dependency is handled in *Mosaik* by asynchronous requests, illustrated in Figure 5.6.

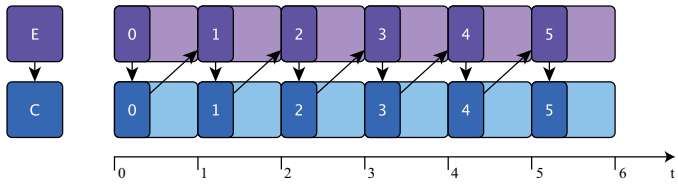


Figure 5.6: Data exchange between a controlled simulator E and controller C. E provides information to C, and C sends commands to E, which are used in E in the next step [115].

The data from controlled entity E is passed to the controller upon the beginning of the step. The control functions from controller C to entity E are actively set in the `set_data()` call in the `step()` function of the controller, and are applied by the entity E in the next simulation step. In the proposed testbed a cyclic dependency exists between the RTU controller and the topology of the power distribution simulator, illustrated in Figure 5.7 as C and D, respectively.

Due to the interaction of these simulators, commands that are issued within the network simulation part of *Mosaik* first need to be handled by the simulated controller, before they are propagated to the power distribution system simulator. This corresponds to a delay of one step-size in the simulation framework, which does not occur in real systems, as commands that have been processed by the controller directly impact the distribution system.

To see how this delay works out in practice, let us now consider executing a command in the proposed testbed for bus B_5, as presented in Figure 5.2(b). The command is sent from the SCADA server to RTU_3 to open the switch located at power line L_{25}. A detailed analysis is shown in Figure 5.8. The upper graph shows simulator events occurring in the RTU simulator and the power distribution simulator. The lower graph shows the influence of the command on the electric current readings reported by the RTU_3 simulator. For clarity, constant values of house consumption and PV panel production are used.

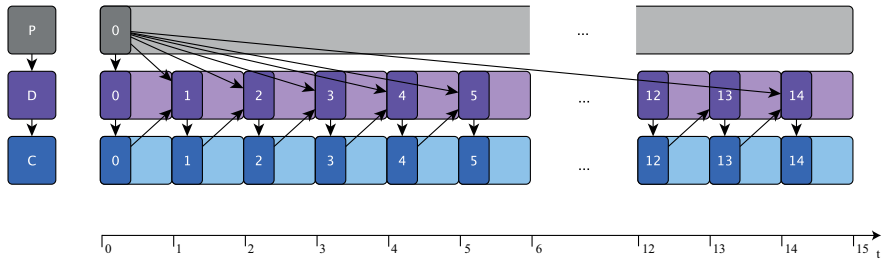


Figure 5.7: Relations between three simulators in the proposed testbed. The simulator of households and PV panels **P**rofiles is illustrated in gray color, the simulator of power **D**istribution in purple, and the simulator of the RTU **C**ontroller is shown in blue.

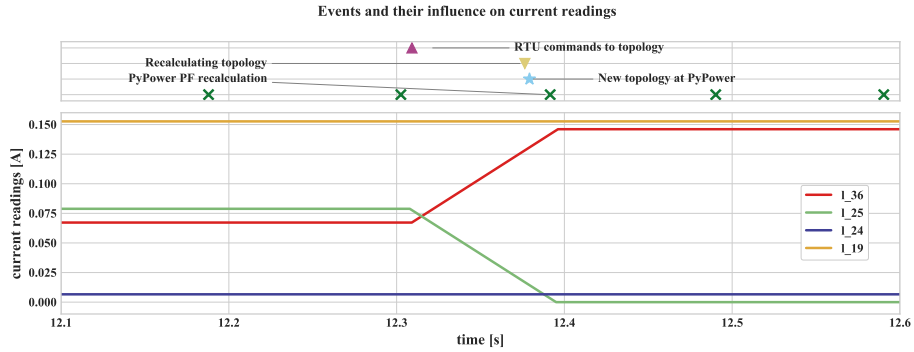


Figure 5.8: Execution and effects of one event in the testbed

The time given on the x-axis refers to the simulation time, which is running with a real-time factor of 120 (that is, 120 times faster than real time). At the beginning, the current reading of power line L_{19} (orange line) equals 0.153 A, the current of power line L_{25} (green) equals 0.078 A, the current of power line L_{36} (red) equals 0.067 A, and the current of power line L_{24} (dark blue) equals 0.007 A. The simulator events (upper) graph shows recurring simulator event of recalculating power flow equations (green crosses X). At a time point just after 12.3 seconds, the power flow equations are recalculated. Soon after this, the controller simulator receives a command (purple triangle) which has to be passed to the power distribution simulator, because the values of the switch state(s) changed. This information is sent to the power distribution simulator

and at the next step of that simulator, the topology is recalculated (yellow triangle) and the power flow equations are recalculated using PyPower again. This last event has direct influence on the readings of the current seen in the graph below. Since power line L_{25} is now opened, the current value on that line decreases to zero. To compensate for that, the current on power line L_{36} increases to 0.145 A.

Note that the delay between receiving a command to change the tap switch position and its influence on the voltage value is similarly influenced by the inter-dependencies of the various simulators, as shown here for the currents in the switching scenario.

5.3 Traffic Monitor

Previous section described how (using *Mosaik* co-simulation framework) the interaction between the commands sent over a SCADA communication link and the simulation of the distribution grid can be modelled. This section explains the implementation of the monitoring algorithm introduced in Section 4.4.4 of Chapter 4. Section 5.3.1 introduces Zeek network security monitor[3] before Section 5.3.2 outlines the method for creating Zeek policies.

5.3.1 Zeek Network Monitor

Among the available open-source network monitoring tools which can be used for SCADA protocols, the most popular are Snort [124] and Zeek [94, 118, 143]. While Snort allows for rule-based pattern matching within packets to determine their legitimacy, Zeek provides various frameworks, which allow rule-based evaluation of packet content.

Zeek, for the incoming packets, creates high-level events that can apply custom-made scripts that determine the necessary procedures. For example, it is possible to configure real-time alerts about new hosts in the network, execute external programs upon an event, and log data. Moreover, Zeek includes a Modbus/TCP parser, that generates events upon parsing packets of this protocol. The parser, for example, generates a `modbus_write_single_coil_request` event when parsing a Modbus/TCP packet containing a "write single coil request", that is, a Modbus packet with a function number `05`. By creating a custom event handler, it is possible to process the content of such packet in a desired way. For example, one can extract the number of the coil that this function addresses and the desired state ("on" or "off") of that coil. This information can be further processed, either in a Zeek script itself, or, it can be passed to

[3]Recall that the Zeek network monitor is the new name for the Bro network monitor.

some external program. However, in order to assess the information contained in a certain coil, the script needs the knowledge what that coil represents. This will be explained in the next section.

For the purpose of testing the monitoring in the testbed proposed in this chapter, we will investigate the option of processing the content of packets within a Zeek script. In the proposed testbed, the monitoring device is placed between the Modbus RTU device and the rest of the network; in Figure 5.1, Zeek is indicated with Number 6.

5.3.2 Creating Zeek Policies

To enable process-aware scripts (so-called *policies*) in Zeek, among others, the physical restrictions and safety requirements from Sections 4.4.2 and 4.4.3 are used in combination with local measurements. Moreover, the system state of the part of the power distribution system (shown in Figure 5.2(b)) has to be described. The system state is defined by the tuple Y explained in Section 4.3.3. However, Zeek needs to also understand the relation between the parsed coil address and the physical entity that this address refers to. This can be provided by, for example, parsing the configuration file of an RTU. Figure 5.9 illustrates the approach to create such Zeek policies. The left-side of Figure 5.9 shows the

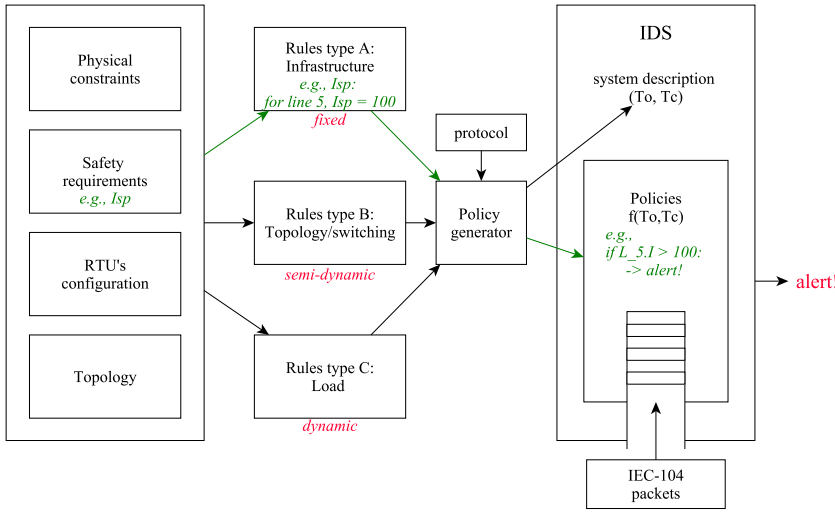


Figure 5.9: Example of generating Zeek policy for maximal current from the system description components.

input needed to create the process-aware policies: the physical constraints, the safety requirements, the RTU's configuration file and the topology of the investigated system. The first two were already discussed in Sections 4.4.2 and 4.4.3. The **RTU configuration** contains the mapping of process variables to the addresses, where the values are stored. Modbus/TCP stores either 1 bit values (so-called coils) or 1 byte values (so-called registers). Both coils and registers can be either read-only values (discrete inputs and input registers, respectively) or read/write values (coils or holding registers, respectively). In order to allow for, for example, floating point variables, some vendors allow for combining registers to hold 32-bit and 64-bit values [57]. An example part of the configuration of the Modbus/TCP server in xml format is shown below:

```
...
<ip>192.168.33.1</ip>
<port>10502</port>
<reg type="co" index="0" label="switch_19-branch_19" dt="bool">True</reg>
<reg type="hr" index="16" label="sensor_19-branch_19" dt="64bit_float">0</reg>
<reg type="hr" index="32" label="max-branch_19" dt="64bit_float">0.08</reg>
...
```

The first two lines describe the parameters needed to connect to the RTU server: its IP address and the port number. The next line describes a coil ("`co`") containing a boolean value `True`. This value describes the state of switch S_{19} on power line 19, and the address (index) of that coil is 0. The next two lines describe the content of two holding registers ("`hr`"). The one *starting* at address 16 contains a 64 bit floating point value describing the sensor measurement of the current on power line 19. As a single register has only 1 byte of information, and the floating point number needs 64 bits, the value has to be stored in registers 16-19. The initial value of the sensor reading is 0, as it will directly be updated with the current sensor measurements once the server is running. Similarly, the maximum current on the power line 19 is described with the holding register that starts at index 32. However, here the initial value is set to the actual maximum allowed current and will only be changed from the control room, not by the sensor measurements.

The **topology** is a description of the system at hand, shown in Figure 5.2(b). We use the format introduced in Chapter 4 to describe the local topology for the RTU that is located at bus B_5.

$\mathcal{B} = \{B_5\}$	$L_x.B_5.S = S_x$ for $x \in [19, 24, 25, 36]$
$\mathbf{B}_5.in = [L_{19}, L_{24}]$	$S_x.st = True$ for $x \in [19, 24, 25, 36]$
$\mathbf{B}_5.out = [L_{25}, L_{36}]$	$\mathcal{K} = \{K_1, K_2\}$
$\mathcal{L} = \{L_{19}, L_{24}, L_{25}, L_{36}\}$	$K_1.S = \{S_{19}, S_{24}\}$
$L_x.I_{max} = 0.08$ for $x \in [19, 24, 25, 36]$	$K_1.CS_{min} = 1$
$L_x.V_{ref} = 10$ for $x \in [19, 24, 25, 36]$	$K_2.S = \{S_{25}, S_{36}\}$
$\mathcal{M} = \{M_{19}, M_{24}, M_{25}, M_{36}\}$	$K_2.CS_{min} = 1$
$L_x.B_5.M = M_x$ for $x \in [19, 24, 25, 36]$	$\mathcal{P} = \emptyset$
$M_x.I_{sp} = 0.08$ for $x \in [19, 24, 25, 36]$	$\mathcal{T} = \emptyset$
$M_x.V_{ref} = 10$ for $x \in [19, 24, 25, 36]$	$\mathcal{F} = \emptyset$
$\mathcal{S} = \{S_{19}, S_{24}, S_{25}, S_{36}\}$	$\mathcal{R} = \emptyset$

In a real-life scenario, the topology can be provided by the distribution system operator.

Based on the interactions between the power distribution system elements, it is possible to develop three sets of rules: (i) *infrastructure*-based, (ii) *topology*-based, and (iii) *load*-based, that help to keep the system in a safe and secure state. The middle of Figure 5.9 illustrates the three types of rule, which are explained below:

A Infrastructure: These rules depend on the present infrastructure and change only when the physical system is changed, for example, if a cable is replaced by one with a different maximum allowed current. Infrastructure changes are relatively rare, hence, we decided to not adjust these rules automatically within the Zeek policies. Hence, a change in the infrastructure requires an explicit update by the operator, which we believe helps to keep the system safe and secure. An example of this type of rule is checking whether the current sensor measurement stays below the threshold for the maximum current on a power line:

$$M_{19}.I < 0.08.$$

B Topology/Switching: These rules are to some extent dynamic and can be updated based on the system state. They change when the topology of the system is changed, for example, due to a switching command. An example of such a rule is interlocking, that is, a mutual dependency between the state of some switches. Upon receiving a *command* about changing a state of a switch, first, the model of the switch states in Zeek is updated, before the policy evaluates the new, resulting model. For example, in the presented system, a static interlock K_1 defines the minimum number of connected power lines among lines L_{19} and L_{24}:

$$K_1.CS_{min} = 1.$$

C Load: These rules depend on the load and are highly dynamic. They are updated automatically in the system, for example, a decision to open a power line depends on the load of all connected power lines. The state of the system depends on the load that is newly reported with every reading of sensor *measurements*, and is updated in Zeek accordingly. The policies referring to the load depend on that reference state. An example of a load-based rule is checking whether executing a command to open a switch does not cause an overcurrent on any of the connected lines. As in this scenario, all the thresholds are equal, we write this as:

$$\forall L_k \in B_5.in \cup B_5.out : L_k.B_5.M.I < 0.08.$$

The system state changes over time, and as the rules depend on the state of the system, they have to be implemented in adaptive Zeek policies in order to monitor and alert about potentially malicious traffic. The local monitoring algorithm as explained in Section 4.4.4 is implemented for both readings and commands:

(i) Upon a *new reading*, the Zeek policy tests whether the physical consistency rules hold and whether safety requirements are maintained, as indicated in Sections 4.4.2 and 4.4.3. In case no violations are detected, the observed values are stored in the local model of the physical system. If violations are detected, an alert is additionally sent to the operator.

(ii) Upon receiving a *new command*, the Zeek policy precomputes the outcome of executing such a command based on the constraints in Section 4.4.2, and performs safety checks according to Section 4.4.3.

The following section will show, step by step, how the system state and relevant policies are defined for the example shown in Figure 5.2(b).

5.4 Examples of Traffic Monitoring

This section describes how monitoring the state of the physical system can improve field station security. Section 5.4.1 discusses the general threat model and attack scenarios. Section 5.4.2 applies monitoring to identify attacks on the system's interlocks, and Section 5.4.3 applies them to a transformer tap switch.

5.4.1 Threat Model and Attack Scenario

In the following, an attacker can either perform a *man-in-the-middle attack* and inject false messages between the Modbus RTU device and the SCADA server,

or can directly take control over the SCADA server, as illustrated in Figure 5.1. Both attacks result in a corrupted communication channel to the field station. Hence, both the network and the SCADA server cannot be trusted. Assume that an adversary sends well-formatted packets from the control room to the remote stations and has all necessary privileges to perform the requested commands. This means that other security mechanisms, such as standard Network IDS would not recognize such packets as potentially malicious.

In the initial attack scenario an attacker attempts to disconnect power lines controlled by the RTU_3 (see Figure 5.2(b)), one by one. That RTU initially does not perform any of the safety checks, that is, it directly executes the received commands. Then the attack scenario is changed, such that the attacker attempts to change the tap switch controlled by RTU_1 (see Figure 5.2(b)) to an unsafe position.

5.4.2 Interlocks

Interlocks are used to manage mutually dependent elements. This logic is supposed to work locally and independently from the central control room. However, distribution operators are concerned [128] that for some solutions checks are not performed locally, but only in the central control room. This means, that it is possible to bypass any security checks by injecting a command via an outside communication channel, which is not analysed by the central EMS. Consider the interlocks that are required for the system from Figure 5.2(b), where bus B_5 is a node operating at medium voltage. When disconnecting either the two power lines L_{19} and L_{24}, or the two power lines L_{25} and L_{36}, the neighbourhood behind bus B_5 is left without electricity. Hence, there are two groups of interlocks, where at least one switch has to be connected (closed).

Implementation of the Interlocks using Zeek

The interlocks are configured in a Zeek script as follows. First, the state of the switches is stored in a global object. This is, in fact, the vector **S** mentioned in Section 4.3.3 and it is part of state Y. These values will be updated each time a read command is parsed by Zeek. Moreover, the mapping function m between coil address and switch instance has to be stored.

```
S[switch_name] = { state }
m(coil_address) = switch_name
```

Secondly, information about static interlocks K_s is added to the script. The sets of interlocked switches from the topology description are added to the Zeek policy that will be configured in RTU on bus B_5.

K_s = { K_1, K_2}
$K_1.S$ = { "S_19.st", "S_24.st"}
$K_1.CS$ = 1
$K_2.S$ = { "S_25.st", "S_36.st"}
$K_2.CS$ = 1

Thirdly, updating switch states upon receiving a new read or write command has to be implemented. Modbus/TCP is a request/response type of protocol. This means that the SCADA server, in order to obtain the information, acts like a client and requests data from the RTU server. The Modbus request contains the start address and the number of requested coils, while the response contains their value(s). A response is matched with the request with the connection and the Modbus Header ID. This is also how we implement this in Zeek. Since the switch states are stored on the RTU as Modbus coil values, the event handlers for the read coil request and response events are created, as shown below.

modbus_message ∈ { **read_coil_req**, **read_coil_res** }
event **read_coil_req** → store **temp**(**id**) = { coil_address, quantity }
event **read_coil_res** → verify **id** is in **temp**
→ retrieve coil **value**
→ update S(m(**temp**(**id**)(coil_address))) = **value**
where **id** = { connection id, Modbus header id }

Upon a "read coil request", the address and number of requested coils are stored in a temporary variable **temp**, identified by the connection identifier and transaction identifier from the Modbus header. Upon a "read coil response", the script checks if the response **id** is stored in the temporary variable. If such a reference is present, the value of the coil is retrieved from the response and the state of switch stored in the vector **S** is updated.

Finally, the safety requirement checked upon receiving a new command is implemented. Upon a *write* coil request, the function testing interlocks is triggered.

modbus_message ∈ { **write_coil_req**}
event **write_coil_req** → check_interlocks(coil_address, value)

The function testing interlocks is shown in Algorithm 2.
The function shown in Algorithm 2 tests whether the outcome of the command does still satisfy the interlock constraints. Line 4 checks whether the switch that is supposed to be opened is part of any of the interlock sets. If so, the number of closed switches in that set is counted (lines 5-8) and the command is applied on that interlock (lines 9-13). Line 14 checks if the switch can be opened comparing the number of the closed switches in the resulting system state with the minimum number of closed switches $k.CS$ that this interlock requires.

```
   Output: bool
 1 check_interlocks(address: count, value: bool)
 2 local amt:count = 0;
 3 for k in K_s do
 4 |  if m(coil_address) in k.S then
 5 |  |  for s in k.S do
 6 |  |  |  if S[s] == True then            // Count closed switches
 7 |  |  |  |  ++amt;
 8 |  |  end
 9 |  |  if value== True then          // Apply the current command
10 |  |  |  ++amt;
11 |  |  else
12 |  |  |  - -amt;
13 |  |  end
14 |  if amt < k.CS then                      // Interlock violation
15 |  |  return False;
16 end
17 return True;                              // No violations occurred
```

Algorithm 2: Function testing the interlocks.

Attack Without Local Monitoring

In the example shown in Figure 5.2(b), a successful attack on bus B_5 is performed by disconnecting a pair of lines: either L_{19} and L_{24}, or L_{25} and L_{36}. An example of the effect of such a successful attack on RTU_3 is shown in Figure 5.10. In this attack, the SCADA server sends three commands to open switches on power lines L_{25}, L_{19} and L_{24}, respectively. Similar to Figure 5.8, the upper graph shows events in the co-simulation framework, and the lower graph shows the effect of those events on the current readings in the power lines that are directly connected to bus B_5. Again, the profiles of power demand in houses and production of PV panels are set as constant for the sake of better visibility, and the time on the x-axis refers to the simulation time.

In Figure 5.10 the current reading of the electric current in power line L_{19} is shown in orange, L_{24} in dark blue, L_{25} in green and L_{36} in red. Initially, the electric current readings on the lines have a constant value. After the first event, that is, opening power line L_{25} (at approximately 12.5 seconds simulation time), the current which was carried by line L_{25} is taken by power line L_{36}. After opening the switch on power line L_{19} (at around 15.2 seconds simulation time), the bus B_{28} and the rest of the neighbourhood is now only connected via lines L_{36} and L_{24}. The current on lines L_{36} and L_{24} is therefore equal:

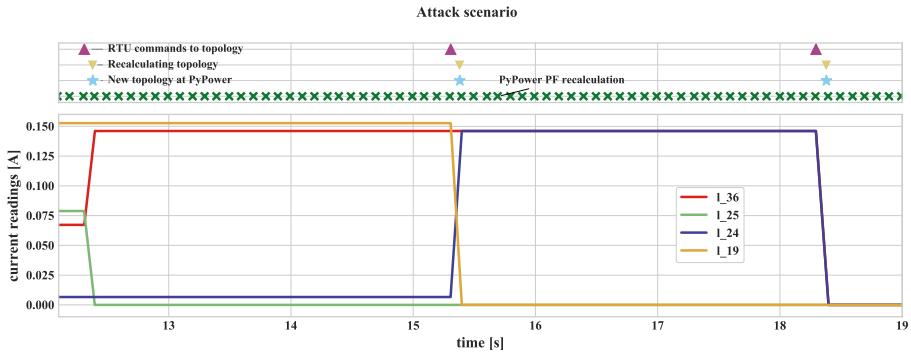


Figure 5.10: An example of attack scenario on RTU_3

in Figure 5.10, the dark blue line (for L_{24}) overwrites the red line (for L_{36}). Finally, at the time point around 18.5 seconds, power line L_{24} is opened. This causes isolation of part of the neighbourhood and all the power lines around RTU_3 have zero current (orange overwrites green).

Although disconnecting only power lines L_{25} and L_{19} influences the power flow in the distribution system, it does not disrupt the operation of the distribution system, as all the houses can still be connected to a source of power.

Results

In the following, the influence of the proposed local monitoring approach on the security of the field stations for *all* possible initial settings is investigated. The left side of Table 5.3 shows all possible initial (safe) values of vector **S** describing the state of the switches in the subsystem controlled by RTU_3. In this context, *safe* means that all houses are still connected to the source of electricity.

The right side of Table 5.3, under column "command", shows all possible commands that can be sent to RTU_3. These commands could be sent from the control room either by the operator or by an attacker. The outcome of each of the 4 commands for each of the nine safe initial states is tested and the output of the detection mechanism is presented. Mark '–' means that the system does not execute a requested command, as the current state of the switches already matches the requested one. Mark 'safe' indicates that the command is safe to perform and allowed. Mark 'alert!' means that the command is not safe to perform, an alert is raised and the command is discarded.

Out of a total of 36 cases, 12 cases are marked with "–", as the execution of the command would not change the state of the system. An operator should

Table 5.3: Output of the monitoring script on commands to change state of switches for all possible safe values of vector **S**

safe S				**command**			
L_{19}	L_{24}	L_{25}	L_{36}	$S_{19}.st = 0$	$S_{24}.st = 0$	$S_{25}.st = 0$	$S_{36}.st = 0$
1	1	1	1	safe	safe	safe	safe
0	1	1	1	–	alert!	safe	safe
1	0	1	1	alert!	–	safe	safe
1	1	0	1	safe	safe	–	alert!
1	1	1	0	safe	safe	alert!	–
0	1	0	1	–	alert!	–	alert!
0	1	1	0	–	alert!	alert!	–
1	0	0	1	alert!	–	–	alert!
1	0	1	0	alert!	–	alert!	–

still be notified about such a situation, since the command could have been sent by an attacker who is unaware of the current state of the system, trying to perform an attack in a opportunistic or random way. Another 12 cases are marked as safe. This means, that after performing the attack, the resulting vector **S** indicating the switch states is also one of the 9 listed safe vectors. This possible type of attack (if sent by an attacker) goes unnoticed, but also does not harm the system. The remaining 12 cases were marked as attack. Here it is clear that the resulting vector of switch states **S** is not safe for the system. All these alerts are cases which would otherwise go unnoticed, thus stressing the extra security and safety precautions provided by the local monitoring approach.

5.4.3 Transformer Tap Switch

The previous scenario analysed an RTU controlling a bus operating at medium voltage level. The scenario addressed next monitors an RTU that controls different voltage levels, namely high and medium voltage. A transformer changes the voltage values on the outgoing power lines with some predefined ratio. Changing a ratio setting of a transformer influences the value of the secondary voltage, while the voltage on the primary side remains the same. By changing the position of so-called tap switches, it is possible to adjust the exact value of that ratio. The transformer T_1 marked in Figure 5.2(a) as yellow oval is presented in Figure 5.11. It connects the high and medium voltage levels and contains a controllable tap switch. The operator can send commands from the control room in order to change the value of the voltage on the secondary side of the transformer.

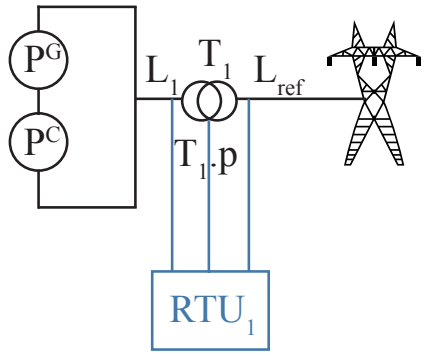


Figure 5.11: An example of attack scenario on RTU_3

The main safety requirement that is checked when changing the tap switch position is the voltage value on the secondary windings of the transformer (compare safety requirement R5b in Section 4.4.3). That requirement defines that the measured secondary voltage value has to be equal to the reference value ±10%. This is defined for the low voltage area [23], however, in the proposed approach it is also possible to perform the same check for medium voltage, like proposed in [72]. The implementation of the monitoring tool on RTU_1 that controls the transformer, can be done similarly as in Section 5.4.2 for interlocks (and is not shown here in detail). In the following, only the outcome of the performed tests are shown.

Attack Scenario

A successful attack is performed by changing the tap switch to such a position that the value of the secondary voltage exceeds the allowed bounds. Since the nominal value of the secondary voltage is 10 kV, this means the voltage must stay within 9 kV and 11 kV. The initial ratio of the transformer, that is, the ratio of the primary to secondary voltage is 11 in the following scenario. The transformer has 3 tap switch positions, resulting in ratios 11 (position 1), 10.5 (position 2) and 10 (position 3), respectively. The ratios are provided in Table 5.4. If the primary voltage equals the nominal value of 110 kV, then setting the transformer's tap switch to position 3 results in violating the bound of the secondary voltage.
The attacker opportunistically changes the tap switch position to different values, aiming to disturb the physical process. Figure 5.12 shows the result of commands issued by the attacker. The lower part of Figure 5.12 shows the

Table 5.4: The tap switch settings $T_1.t$ of the transformer T_1

$T_1.p$	$T_1.r$
1	11
2	10.5
3	10

voltage value on the secondary side of the transformer. It can be seen that at 16s (simulation time; x-axis), the attacker changed the position from 2 to 1, resulting in a voltage of 10 kV. This is a failed attack attempt, as the resulting voltage is well within bounds. Next, at around 32s another change is made: the tap position is changed back to 2, as the attacker does not know the initial value of the tap switch. Finally, at around 48s, the attacker changes the tap switch to position 3 which results in the undesired voltage value of 11 kV. If the attacker continues to perform changes, the monitoring approach will continue to filter actions that lead to unsafe states. However, our approach does not warn about attacks that lead to safe states.

Results

While the previous scenario covered all initially safe configurations, this section focuses on the analysis of the interaction in the testbed between receiving commands and issuing alerts, as presented in Figure 5.12. The upper part of Figure 5.12 indicates the time when commands are sent by the attacker and the reaction of the monitoring tool to these commands.

The events marked with a green pentagon represent alerts issued by Zeek upon receiving the command to change the tap switch to a position that would result in a too high secondary voltage (see safety rule R5b). This is a result of implementing the voltage safety requirement (see Section 4.4.3) upon receiving a new command (see the right-side loop of Figure 4.15). Note that Figure 4.15 indicates that the command that may bring the system to an unsafe state should be discarded. Here, only an alert was given in order to analyse the further behaviour of the system.

The blue diamonds represent the warnings issued by Zeek due to violations of the voltage safety requirement R2 upon receiving a new reading (see Figure 4.15, the left-side loop).

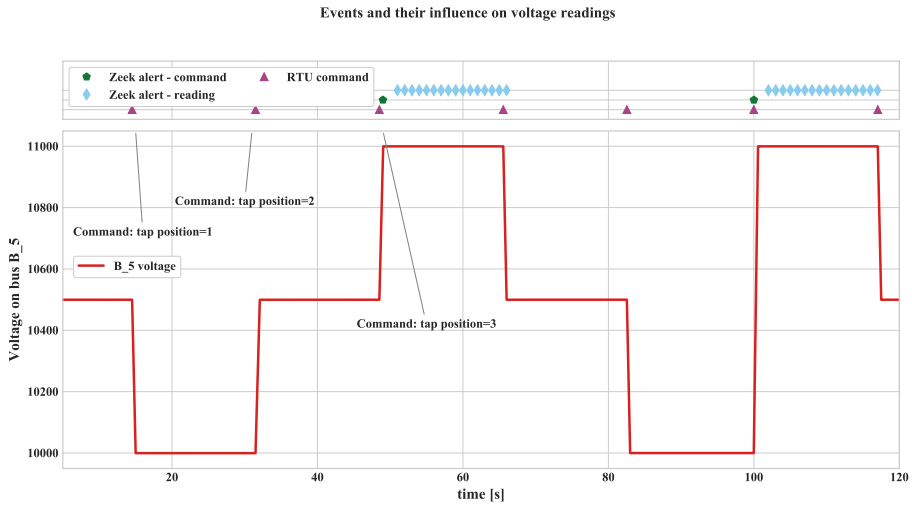


Figure 5.12: An example of attack scenario on RTU_1

5.5 Summary

In this chapter, we introduced a testbed that can be used for evaluating the feasibility of a process-aware monitoring approach. Although we use it for the monitoring method explained in Chapter 4, this testbed can be also used to evaluate other techniques that require information from both the cyber and the physical elements of the power distribution system [29].

Sections 5.4.2 and 5.4.3 presented how the proposed testbed can be used to investigate the effect of the proposed process-based monitoring on the safety and security in field stations. In both cases, the experiments have shown that the monitoring tool responds accurately to the processed command, for example, generates alerts for commands that would bring the system to an unsafe state. For so-called interlocks, that is, mutually dependent states of system elements, the proposed monitoring prevents the execution of 33.3% of all possible commands. Without the proposed approach in place, those commands would result in an unsafe state of the power distribution. The remaining two-thirds of the commands yield a safe state of the power distribution, that is, all the neighbourhoods remain connected to the power grid. Hence, the approach allows the RTU to execute them, even though they might come from an untrusted source. In a second scenario, local monitoring is used to identify commands to change the tap switch position of a transformer, which lead the system into an unsafe

state. The monitoring approach correctly detects all the commands that bring the system to an unsafe state. Moreover, it warns when the voltage values stay in the unsafe range.

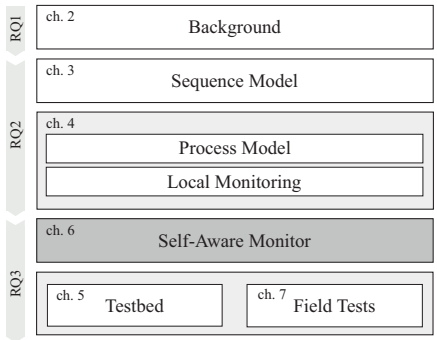
Using a simulation testbed allows to investigate the consequences of executing a malicious command versus discarding it or simply issuing an alert. This would not be possible in real infrastructures and would be very difficult in a physical testbed. Moreover, the proposed co-simulation testbed lends itself to stress tests, for example, regarding the frequency of reading commands and how this influences the number of alerts and the accuracy of the monitoring tool.

Also the real-time capabilities of the proposed approach can be evaluated for the presented test cases. The first investigated scenario, that is, monitoring the interlocks, focused on 4 elements in the switch vector describing part of the system state, and on the sensor measurements on the 4 connected power lines. The second scenario investigated the transformer tap switch position vector with a single element and the sensor readings of two power lines on the primary and secondary side of the transformer. In these scenarios, calculating the resulting system state and the policy checking within Zeek caused message delays of only 0.002 ms on average. Clearly, a more thorough investigation of the real-time performance is needed for different sizes of field stations, before drawing more general conclusions. However, as the approach is meant to work locally at field stations, the models probably will not become much larger than for the scenarios analysed here. Hence, scalability should not be a problem in the proposed approach.

The next chapter proposes a tool that can generate the policies automatically given the input mentioned in Section 5.3.2: physical constraints, safety requirements, RTU configuration and the topology of the investigated system.

Self-Aware Monitor

Previous chapters introduced a modelling formalism used to describe a model of a locally controlled part of the power distribution system, a new algorithm that can be used to perform local monitoring, and a testbed, where this algorithm can be evaluated. However, manually generating the rules for the monitoring tool can be cumbersome. Moreover, once a set of rules is generated, it is then hard to maintain the rules upon changes in the system topology. Therefore, in this chapter, a tool that, given the system topology, automatically verifies relevant rules is introduced. Based on inputs, such as the configuration of a local RTU and the system topology, the proposed Self-Aware Monitor (SAM) implements the monitoring algorithm from Chapter 4. Upon each incoming packet, the SAM tool creates events that trigger an update of the state of the locally stored model of the system. That state is then evaluated w.r.t its consistency and safety.



This chapter is organised as follows:

- *Section 6.1 states the design objectives of the prototype and outlines the architecture of the SAM tool.*
- *Section 6.2 describes four sets of tests performed with the SAM tool: (i) normal operation, (ii) sensor manipulation, (iii) set point changes, and (iv) malicious commands in the system.*
- *Section 6.3 presents the results of the above mentioned tests.*
- *Section 6.4 discusses how the knowledge scope of the SAM tool and the topology influence the output of the tool.*
- *Section 6.5 concludes the chapter.*

Section 6.1 is based on [45], whose work the author has been co-supervising.

6.1 The SAM Architecture

This section outlines the architecture of the proposed Self-Aware Monitor. Section 6.1.1 lists design objectives before Section 6.1.2 provides the tool's architecture. Section 6.1.3 describes the first component of the SAM tool: the power distribution grid model and its evaluation logic, whereas Section 6.1.4 addresses the second component of the tool: the IDS and the custom made events. Finally, Section 6.1.5 discusses the advantages and disadvantages of the proposed architecture.

6.1.1 SAM Design Objectives

The Self-Aware Monitor, marked in Figure 6.1, is located between the SCADA server and the Modbus RTU. It monitors the network traffic exchanged between these two elements of the testbed. To understand the choices taken in the development of the prototype tool, first, we establish some design objectives of the implementation, listed below.

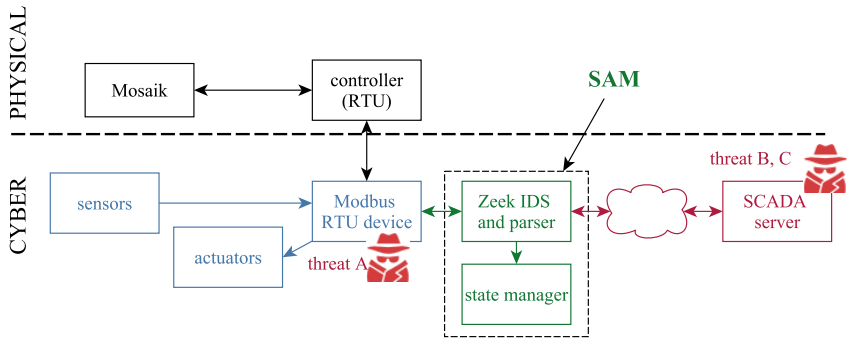


Figure 6.1: Location of the tool with reference to the testbed.

The SAM tool has to support all elements introduced in Section 4.3, as well as all physical constraints and safety requirements introduced in Sections 4.4.2 and 4.4.3, respectively. Moreover, if needed, new components and new rules must be easy to add. The tool should be using open source components, so it is freely available for use. It should be possible to use various SCADA protocols, such as Modbus/TCP, IEC-60870-5-104 or DNP3. The model evaluation should use the contents of packets, independently from the protocol used. The SAM tool should be able to work with real-time data, for example, obtained from Zeek, but it should also be able to analyse historical information,

for example, stored in a network traffic trace file. When parsing malformed packets, the tool should not deny further service, but handle and inform about the errors. Finally, the prototype should be well-documented and well-tested and provide extensive debug capabilities.

6.1.2 Architecture Overview

To achieve the desired protocol and data source independence and to ensure extensibility, the SAM tool is split into two main components as depicted in Figure 6.2. Both components are completely independent and run as separate processes. The source of network traffic which Zeek captures and parses is depicted as a white rectangle on the right-hand side of Figure 6.2.

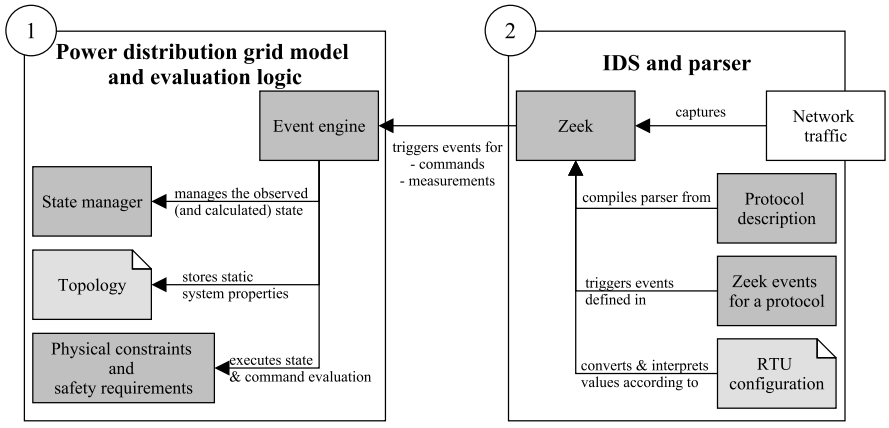


Figure 6.2: Overview of the SAM tool's architecture. Network traffic is the evaluated input, illustrated as white box. Topology and RTU configuration are directly based on inputs from the power operator, and are used to generate rules. They are marked as bright gray boxes.

The left-hand side of Figure 6.2, marked with number 1, shows the component maintaining the power distribution grid model and the policy evaluation. This component is split into three major subcomponents: (i) the state manager, (ii) the data structures describing the topology, and (iii) the rule evaluation logic for physical constraints and safety requirements. Furthermore, an event engine, which offers an interface to receive process measurements and commands, connects this component to an input source, for example, to Zeek, and triggers the evaluation of rules on different occasions. Note that this component requires

the input source to provide a custom tag name and value to update the system state, therefore, it works protocol-independent. This component of SAM is further described in detail in Section 6.1.3.

The second component of the tool, marked with number 2 in the right-hand side of Figure 6.2, is the extension of the IDS Zeek and the protocol parser. It encompasses (i) a protocol description, (ii) protocol-specific events, and (iii) a segment for mapping the packets' raw contents into the physical process values, based on the RTU configuration. Zeek supports parsing and creating events for some of the SCADA protocols such as Modbus/TCP and DNP3. For unsupported protocols, such as IEC-104, it is possible to write and compile other protocols in the open source Spicy framework [71, 136]. This makes Zeek the perfect choice for parsing network traffic also for protocols that are not yet supported. The IDS and parser component are described in detail in Section 6.1.4.

6.1.3 The System Model and the State Evaluation Logic

The Python application maintaining the power distribution model separates the *topology*, which contains the static elements from the power distribution system and their connections, from the *system state* that changes over time. While the topology is created upon the initialization of the monitoring tool, the state is continuously updated every time new measurements and/or commands are detected in the traffic. In the following, we explain the implementation of these two components, as well as the implementation of the events engine that determines when the process evaluation is triggered.

Topology Representation

Every element of the power distribution system is represented as a class that has the topology properties defined in Section 4.3.1. For example, a property of a power line is its maximum allowed current I_{max}. A single component is an instance of the corresponding class with its values set to the ones defined in the topology. State-dependent information like sensor measurements are not saved in the objects themselves but are referenced by so-called tags, which serve as keys to retrieve values from state objects (see Section 4.3.3). Those tags are also saved in attributes of the objects. Figure 6.3 shows a UML class diagram for the data structures that compose a topology.

Classes define a function for every physical constraint and safety requirement that is applicable to that component type. For example, the transformer class, marked in Figure 6.3 with number 1, contains the physical constraints `P6a` and `P6b`, for checking whether the measurements of primary and secondary

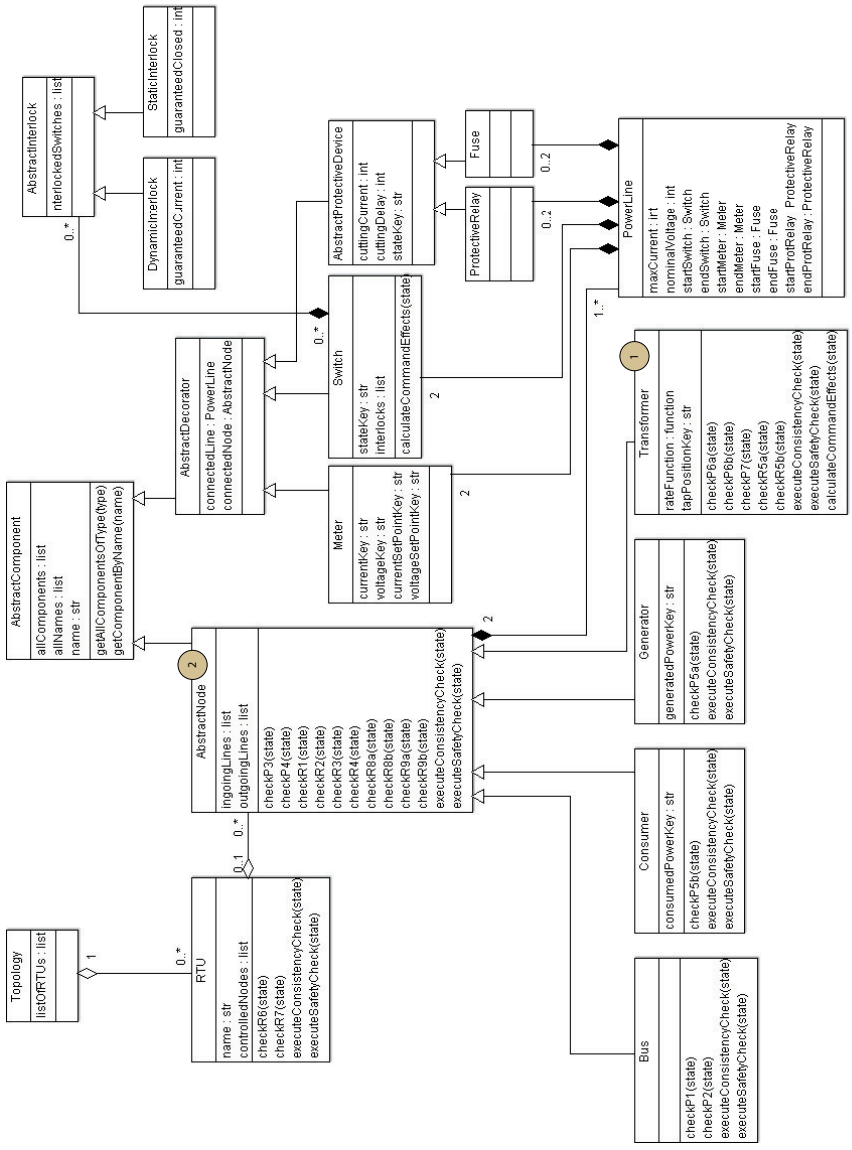


Figure 6.3: UML class diagram of the topology implementation.

voltage and current are consistent with the transformer rate. Rule evaluation functions may also be part of the abstract parent classes if they apply to multiple component types (for example, rules that affect all node components, that is, power generators and consumers, transformers and buses, are defined in the `AbstractNode` in Figure 6.3, marked with 2). Functions evaluating rules always return *False* if there is a violation and *True* otherwise. The evaluation of a component can be either called separately (for example, by `checkR1(state)`) or by functions which bundle all physical consistency and safety rules for that component (for example, by `executeSafetyCheck(state)`). A global rule evaluation of an observed or calculated state of the topology is done recursively. For each tested RTU in a topology those bundled functions are called on each connected node component.[1] In this way, the evaluation process takes locality into consideration and does not only return the consistency and safety of the whole grid state, but also the physical consistency and safety of every single component, node and substation (represented as an RTU).

All rules are implemented in a fail-safe way, which means that all exceptions and errors that occur during the evaluation process are caught and do not propagate. All comparisons of floating-point numbers are done with a special function which takes an allowed relative or absolute error margin ϵ as a parameter for its precision.

Managing the System State

If a command (for example, a switch opening command) is captured, the SAM tool should be able to evaluate, whether this command could lead the system into an unsafe state. To achieve this, SAM calculates new possible state ST_c based on the previously observed state ST_o and evaluates the consistency and safety rules over this state given the static topology.

In practice, the state of the electrical grid is saved in a dictionary. Every time a new measured or reported value is seen on the traffic, the previous value is updated in the observed state dictionary. Every state-dependent property is referenced in the dictionary by a globally unique key $u \in U$. As described above, this key is part of the fixed topology. For example, the key which references the measured voltage is saved in the attribute `voltageKey` of the class `Meter` as depicted in Figure 6.3.

A state property $(v, t, i) \in V$ contains the observed value v, the time of observation t, which is important to judge its freshness, and its validity i. Mathematically speaking, a state ST is defined as a function. Let U be the set of

[1] A local substation contains usually only one RTU. However, deploying this tool at an MTU would allow to obtain the information from different substations. In such case, the evaluation process would be done for each RTU for which state information is known.

all keys defined in the topology, $V \in \mathbb{Q} \times \mathbb{Q}_{\geq 0} \times \{0, 1\}$ and $V^\epsilon = V \cup \{(\epsilon, \epsilon, 0)\}$: $ST : U \to V^\epsilon$

$$u \mapsto \begin{cases} (v, t, 1) & \text{if value } v \text{ of property } u \text{ was seen most recently at time } t \text{ and still holds.} \\ (v, t, 0) & \text{if value } v \text{ of property } u \text{ was seen most recently at time } t \text{ and is invalidated.} \\ (\epsilon, \epsilon, 0) & \text{otherwise.} \end{cases}$$

For example, if the key $u \in U$ references the measured voltage of a specific meter, then $ST(u).v$ would return the last measured voltage value, which has been seen in the traffic at time $ST(u).t$. The second element contains the elapsed time in seconds. Implementation-wise it uses UNIX epoch time. Milliseconds are represented by the decimals of this number. $ST(u).i$ indicates if this value is still considered valid. A more recent change in switching or transformer configuration could have invalidated the value. If there has been no corresponding measurement ever before, then its state is unknown and $ST(u) = (\epsilon, \epsilon, 0)$. The rule evaluation considers $(\epsilon, \epsilon, 0)$ as unavailable. Hence, the rule evaluation process takes the unavailability and freshness of data into account. This ensures a separation between the process-independent state manager and the process-aware rule evaluation model. One more task of the state manager is the management of the history of all measured values for offline analysis.

Event Engine and Rule Evaluation

The event engine offers the interface for process data input. It acts in an event-based manner. If connected to Zeek, it uses the Python bindings from the *Bro Client Communications Library*, called *broccoli* [142], which allows to receive and send events and data between a Python application and a running Zeek instance in a client-server fashion. Every time a process variable has been parsed from the traffic, Zeek triggers an event in the state manager (written in Python) containing the measured value, its physical context and the time. The same applies to commands sent over the network. This is illustrated in Figure 6.4: the left-hand side illustrates the actions happening in Zeek, and the right-hand side of the figure shows the operation of the Python server application. Upon receiving the first packet, Zeek extracts two process variables. For each of them, it triggers an event to call to the Python application. The Python application, upon receiving the process variable, updates the stored value characterizing an element of the physical grid model, using the physical context as reference. If the application does not receive any new input, or receives a command to change some element in the physical system, it performs calculations on the stored model, which is explained next.

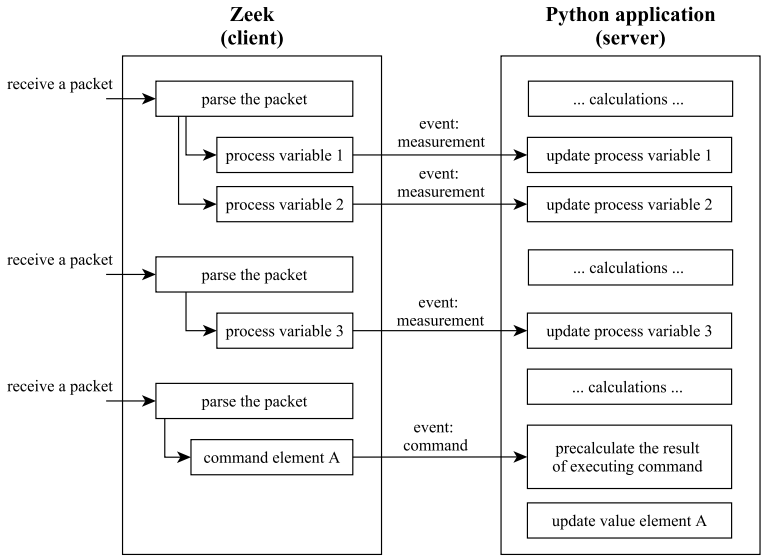


Figure 6.4: Example information flow from Zeek to the Python application.

- **Measurements: periodic assessment of the observed state.** Ideally, each time the state gets updated (that is, upon a new measurement), all rules are immediately checked (see Figure 4.15). However, in a real deployment, new measurements can arrive in different network packets, or be delayed. Updating the state for only a subset of the values can lead to an inconsistent system state. Furthermore, assessing the consistency and safety of the state upon each packet may cause performance issues, due to constant calculations. In order to mitigate those issues, the evaluation process is triggered periodically every x seconds if there has not been a state update in the last y seconds. If there was an update within this time, the evaluation is postponed until the next point in time for which there was no update in the last y seconds. To prevent denial of service attacks, the evaluation is forced after z seconds of delay. Figure 6.5 illustrates the different timers. These custom values can be chosen depending on the system at hand. In the following, $x = 5$, $y = 1$, $z = 10$ are used.

- **Commands changing the topology:** When commands like circuit breaks are parsed by the monitoring tool, it calculates their effect on the observed state. An accurate, or worst-case system state is estimated as the

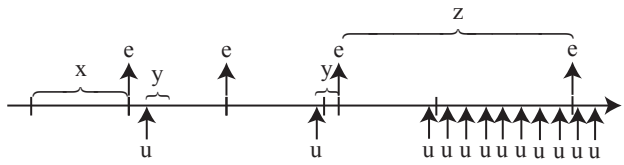
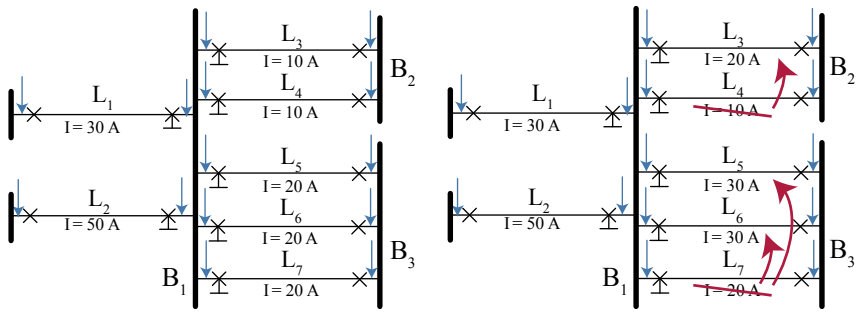


Figure 6.5: Evaluation of the system state (e) happens every x seconds. If an update (u) of measurements has recently happened, the tool waits y seconds from the last update. To avoid denial of service on the evaluation, a forced evaluation happens always after z seconds from the last evaluation.

calculated state, depending on the available information (or lack thereof). The safety of this state is subsequently analysed to assess the validity of the command.
Switching commands are evaluated in the following way. If a switch is opened, the current that the corresponding power line is conducting must be distributed on other power lines. The tool always considers the worst-case, where the entire amount of current could be transported by a single other power-line, which connects the same endpoint. If there are multiple power-lines connected directly to the same endpoint as the power line that should be disconnected, then the current can be evenly distributed over those power lines. This is illustrated in Figure 6.6. Assume lines L_1 and L_2 are incoming power lines, and the remaining ones are outgoing power lines. When disconnecting power line L_4, the current transported by this power line has to be transported by line L_3. Therefore, the calculated state will show that the worst-case prediction for line L_3 is $20A$. When disconnecting power line L_7, the current transported by this power line will be evenly divided between lines L_5 and L_6, yielding a total of $30A$ on each of these power lines. The calculated currents for all local power lines are evaluated with regard to $R1$, $R4$, $R9a$ and $R9b$, defined in Chapter 4. A single violation is sufficient for an alert to prevent false negatives. A command that closes a switch is always considered safe.

The local effect of a transformer tap position change can be calculated very accurately with the known rate function. The resulting state is tested regarding $R1$, $R2$, $R4$, $R5a$ and $R5b$.
The current implementation of command evaluation is limited to a local and immediate effect scope.

- **Commands changing the RTU configuration:** Set points are values specified by the system operator, to control the process, used by the SCADA systems to issue alarms. Configuring, for example, the set point



(a) Before switching. (b) After switching.

Figure 6.6: Topology illustrating the approach for calculated state of the system after disconnecting a power line.

for the electric current too high would prevent necessary alarms in the SCADA system. If setting such a value too low, the SCADA operator may get flooded with unnecessary notifications. Every command that changes the set point configuration is assessed. The calculated state, which contains the updated set points, is tested against the rules *R*8*a* and *R*8*b*, which are set point related safety rules defined in Chapter 4.

- **Manual assessment of the observed state:** This option allows to test and debug the intrusion detection process.

6.1.4 Connection to the Zeek Network Monitor

Until now, all described components have been considered to work independently from the source of process variables, hence, the network protocol used. This section presents the usage of Zeek as source of process information, which is transferred over the network within SCADA protocol packets. In the following,we describe the use of an event-based scripting engine that Zeek offers, and outline how to process the raw values and their protocol address to a meaningful process variable with its context, based on the RTU configuration.

Protocol Parser and Events

Zeek supports the parsing of several SCADA protocols, such as Modbus/TCP and DNP3. For non-supported SCADA protocols, Zeek provides a seamless integration of custom protocol parsers written in the Spicy language [71, 136].

Zeek offers an event-based scripting engine, which calls user-defined scripts written in the Zeek language upon different events. With Spicy's integration into Zeek, it is possible to define new Zeek events for parsing events of unsupported protocols. For supported protocols, it is possible to create custom handlers for the existing events of that protocol. The user-defined event handler has knowledge about the parsed raw value and its protocol address, which are both passed as parameters to user-defined Zeek code.

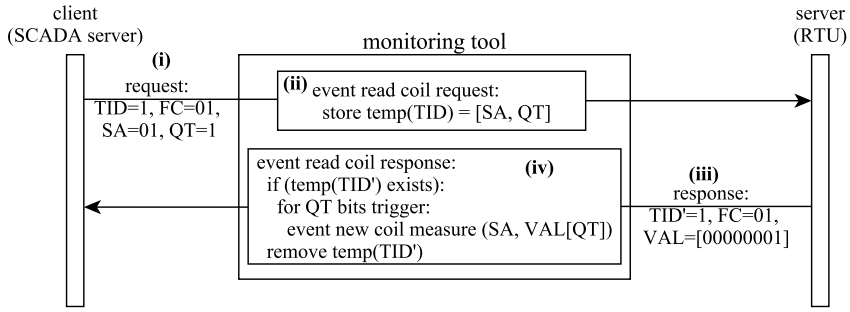
For the Modbus protocol, Zeek creates a different event for every Modbus function code[2], one for the request and another one for the response. For example, one type of event is created when processing a request to read a coil[3], and another type of event is created for the response to such a request. However, in order to be able to match a value (contained in a read response) to a physical variable (identified with the address contained in the read request), it is necessary to store one more mapping in Zeek. This is illustrated in Figure 6.7(a). Upon the arrival of a read coil request, marked with (i) in the figure, Zeek triggers an event (ii) that creates a temporary entry in a table, identified with the transaction identifier[4] of the Modbus communication (denoted as `TID` in Figure 6.7(a)). That entry contains the starting address (`SA`) of the requested coil or register, and the quantity (`QT`) of the requested entities. Upon a response from the server (iii), Zeek matches the stored transaction identifier with the currently processed one (`TID'`) (iv), in order to match which starting address does the processed value belong to. Once a match is found, Zeek creates an event to the Python application and deletes the temporary element in the table. If the number of the requested coils is larger than 1, it will trigger an event for every coil, incrementing the passed address, and referring to the proper value in the response byte (`VAL`). This extra mapping is not necessary for other protocols - as long as the measurement values are sent in the same packet as their identifier. An example of such a case is shown in Figure 6.7(b).

The IEC-104 protocol usually operates in asynchronous mode, so no request is issued. Packets sent in the monitoring direction contain the necessary identifiers, as well as the measured values.

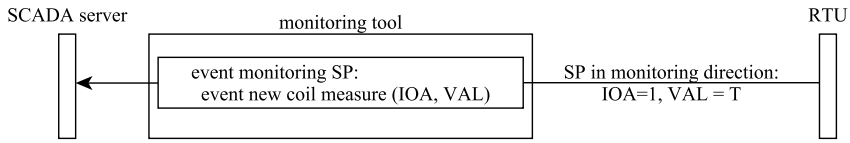
Interpretation of the Physical Context

In order to convert the parsed raw values into the actual physical values and their context, custom event handlers have to be developed for all relevant function types of measured and commanded variables. Moreover, those values have to be unambiguously referenced to variables stored on the RTU. For example, for a

[2]Function codes are listed in Table B.2 of Appendix B.1.
[3]Modbus coil is a 1 bit data type, see Appendix B.1.
[4]Used for pairing the transaction request with response, see Appendix B.1



(a) Modbus



(b) IEC-104

Figure 6.7: Zeek triggers the events differently for protocols that operate in request/response method (like Modbus) (part (a)), and for protocols operating in an asynchronous mode (like IEC-104) (part (b)).

single RTU implementing the Modbus/TCP protocol, an unambiguous reference is the start address and the data type (that is, coil, discrete input, holding or input register)[5]. If more than one device use the same IP address, the Modbus Unit ID (or: Slave ID) is used next to the data type and starting address in order to identify a variable. All necessary information for this preprocessing can be found in an RTU configuration, which often is a table containing information about what is stored in each memory address on the RTU. Table 6.1 shows an example of a configuration within a Modbus/TCP device introduced in the testbed. Column `‘‘Device ID’’` refers to a single Modbus device. When using Modbus/TCP, this value is most often not used, as each device uses its own IP address. Column `‘‘Data Type’’` refers to the Modbus data type, which can be a single bit coil (`CO` and `DI`), or a 2 byte register (`HR` or `IR`). `‘‘Start address’’` refers to the address of the first coil/register belonging to this variable. Then, a description of the physical process context is given in column `‘‘Description’’`

[5]For the description of the data types and start address see Appendix B.1.

Table 6.1: Example of Modbus RTU configuration.

Device ID	Data Type	Start Address	Description	TagName	Real Data Type
101	CO	0001	Switch State	R1_B1_SW11_ST	bool
101	DI	20005	Fuse State	R1_B1_F15_ST	bool
101	HR	30012	Voltage Set Point	R1_B1_M11_V_SP	float64
101	IR	40012	Measured Voltage	R1_B1_M11_V	float64

and an explicit tag reference is given in column ‘‘`TagName`’’. This tag name should be unique within the description of the entire power distribution system, as it is used by the first component of the tool: the grid model and the state manager do not work with registers or information object addresses as they are protocol-specific. Instead, they use this tag in the topology as a key to reference the corresponding value in the state. Finally, column ‘‘`Real Data Type`’’ provides the actual data type of the variable. The real data type indicates how many registers belong to that value. For example, if the real data type is marked as `float32`, 2 registers are needed ($2 \cdot 16$ bits); if the real data type is marked as `float64`, 4 registers are needed ($4 \cdot 16$ bits).

The Zeek module for Modbus has to map raw variables contained in an RTU into physical context variables that can be interpreted by the state manager described in Section 6.1.3. This mapping is illustrated in Figure 6.8. The state

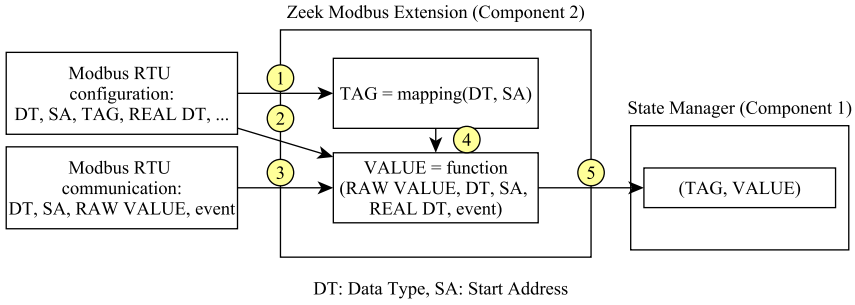


Figure 6.8: Based on the RTU configuration and Modbus communication, the Zeek Modbus extension can unambiguously map raw value and addresses to process variables interpreted by the state manager component.

manager, shown on the right-hand side of Figure 6.8 expects a tuple (*tag*, *value*) as its input format for process variables. Hence, Zeek must do a conversion of the protocol-specific unique address tuple to its protocol-independent *tag*. First, a script generates a mapping function between the Modbus data type and start address (`DT`, `SA`) and a unique tag (illustrated in Figure 6.8 as 1) for a specific Modbus RTU configuration. Based on that configuration, it also has information about the real data type of the processed variable (2). When processing Modbus communication (3), Zeek creates events for each request/response (see previous section). In order to retrieve the actual value of the process variable, Zeek interprets the current event, the raw value, data type and starting address (that are translated to the tag, shown as 4), and the expected real data type. It then creates an event, where the (*tag*, *value*) tuple is passed to the state manager component (5). Using broccoli [142], the state manager acts as a server, whereas Zeek acts as a client (see previous section). Hence, Zeek can trigger events remotely in the state manager. Each parsed Modbus packet triggers a corresponding event in the state manager and passes the (*tag*, *value*) tuple with the current time as an argument (see Figure 6.4).

6.1.5 Discussion of the Architecture

The proposed architecture satisfies the design objectives stated in Section 6.1.1. It is important, however, that the SAM tool operates on the up-to-date topology and RTU configuration. Upon any changes in either of those inputs, an update of the configuration in the SAM tool is required as well. Outdated information could cause false alarms. Moreover, performing a global state evaluation of the entire topology can introduce some delay. In the tests performed later in this chapter, the maximum time of this evaluation took only 2 *ms*, while evaluating a command took at most 1 *ms* on the test machine[6]. In the proposed architecture, the one-way information flow from Zeek to the Python application shown in Section 6.4 only allows to analyse the content of the packets and to warn about a possible unsafe state or malicious command. However, it cannot react by blocking the malicious command. This would require, for example, a feedback from the Python application back to Zeek, that could then discard the packet and block the user who sent it.

The proposed architecture has several advantages. First of all, the state manager works independently from its input source, and therefore, from the used protocol. This allows for easy integration to other protocols. In such case, one needs to provide the mapping of the protocol's registers and values to the corresponding (*tag*, *value*) tuple of the physical process. Moreover, it is possible to simply connect a grid simulator to the state manager. To do so, a

[6]Intel Core i5 6500, 4x 3.20GHz, 128kB/102kB/6144kB L1/L2/L3 Cache, 8GB RAM.

function which passes $(tag, value)$ tuples from the simulation framework to the state manager is needed. Secondly, as the *broccoli* connection uses sockets, the state manager does not need to run on the same machine as Zeek. It is possible to run the intrusion detection logic on a different device, which does not have any information about the RTU configuration, thereby improving the security of this approach.[7] Finally, this architecture allows the connection and data exchange of multiple RTUs. A state manager can connect to multiple instances of Zeek simultaneously. As some rules need semi-local or even global information, process information of different RTUs can be exchanged over a network connection secured by the SSL protocol [142]. Zeek can also be connected to different clients and trigger the corresponding events for observed values and commands at every client.

6.2 Evaluation Scenarios

In this section, the SAM tool is evaluated using traffic obtained in the testbed described in Chapter 5. Scenarios for evaluation of the tool have to cover testing all attack types listed in the description of the threat model in Section 2.3.3. Moreover, all of the physical constraints and safety requirements introduced in Sections 4.4.2 and 4.4.3 need to be violated at least once. Section 6.2.1 describes a topology of a part of a power distribution system which is used to evaluate the proposed tool, Section 6.2.2 then lists four types of scenarios that are investigated. Finally, Section 6.2.3 explains the implementation of the scenario cases in the proposed testbed.

6.2.1 Topology Description

In order to test all physical constraints and safety requirements, a topology that contains all elements introduced in Chapter 4 is proposed. Figure 6.9 presents a diagram of part of a power distribution system, whereas Table 6.2 provides an overview of the most relevant physical attributes of that system. The left-hand side of Figure 6.9 shows a single generator P_1^G, while the right-hand side contains two consumers: P_1^C and P_2^C. The generator and consumers are connected through a network of buses, power lines, and transformers. Buses are thick vertical lines numbered from B_1 to B_3. Transformer T_1 is located between bus B_3 and customer P_1^C, and transformer T_2 lies between bus B_3 and customer P_2^C. T_1 transforms the medium voltage value of $10kV$ to a low voltage

[7] The RTU configuration is considered confidential as an eavesdropping attacker can not interpret the traffic without it. The state manger does not need this configuration, because it receives already interpreted values.

Table 6.2: Physical properties of the investigated topology.

component	value of property		
	I_{max}	V_{ref}	t_{cut}
L_3	$300A$	$10kV$	-
L_4	$200A$	$10kV$	-
L_5	$200A$	$10kV$	-
L_6	$300A$	$10kV$	-
L_7	$300A$	$10kV$	-
L_{10}	$500A$	$230V$	-
L_{11}	$450A$	$6kV$	-
F_{104}	$500A$	-	$5s$
R_{114}	$450A$	-	$5s$

value of $230V$, while T_2 transforms the $10kV$ to $6kV$, which is typically used in energy-intensive factories. Power lines, numbered from L_2 to L_{11}, are illustrated as horizontal thin lines. Their maximum current I_{max} and reference voltage V_{ref} are listed in Table 6.2 in columns which are named accordingly. Each power line is equipped on each side with a switch and a meter, illustrated in Figure 6.9 as a black cross and a blue arrow, respectively. Meters M_{xy} and switches S_{xy} are labelled with numbers xy, where x stands for the power line number, and y stands for the bus number. This labelling is illustrated in Figure 6.9 on example of switch S_{72} and meter M_{72}. For nodes other than buses the following numbers are used: for the power source $y = 0$, for transformers $y = 4$, and for consumers $y = 5$. The initial state of all switches is `True`, meaning that

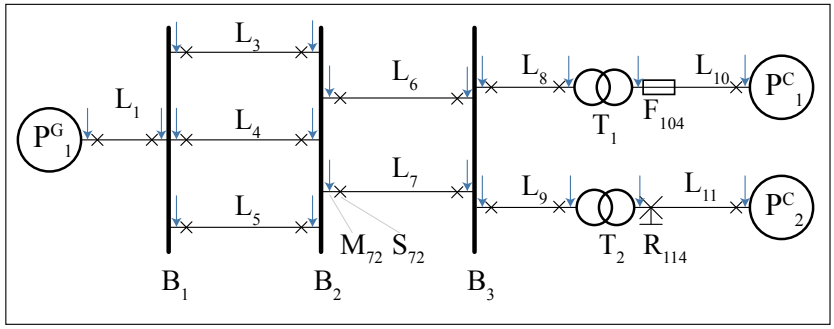


Figure 6.9: Topology of a part of a power distribution grid used in the testbed for tool evaluation.

Table 6.3: Position to ratio mapping of transformers.

component	$t(p, r)$
T_1	(0, 39.13), (1, 41.3), (2, 43.48), (3, 45.65), (4, 47.82)
T_2	(0, 1.5), (1, 1.59), (2, 1.67), (3, 1.75), (4, 1.84)

they are all connected. On power line L_{10} one switch is replaced by a fuse F_{104} next to transformer T_1, and on power line L_{11} one switch is replaced by a protective relay R_{114} next to transformer T_2. The transformer position and the accompanying rate is given in Table 6.3, and the cutting current of the fuse and protective relay as well as the cutting delay are given in Table 6.2 in columns I_{max} and t_{cut}, respectively. Interlocks of this system are defined in Table 6.7, in Section 6.3.4. For the testing of the SAM tool, we assume that the entire system shown in Figure 6.9 is controlled by a single RTU, that is, that RTU has a global view of the entire system.

The system shown in Figure 6.9 was implemented in *Mosaik*. Both, the fuse and the protective relay are modelled in *Mosaik* as switches, in order to be able to change their state.

6.2.2 Scenario Types

The investigated scenarios are divided into four groups: normal operation (N), threat type A, threat type B, and threat type C, referring to the threats defined in Section 2.3.3, and recalled here in Figure 6.10. In each of the groups, several cases testing each of the physical constraints and safety requirements are defined.

1. **Normal operation**. This scenario only evaluates the content of measurements sent to the control room. Three scenario cases, marked with letter

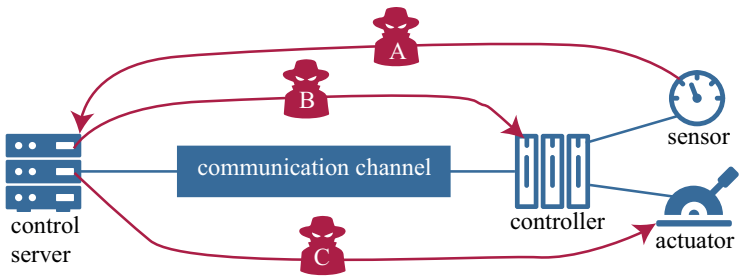


Figure 6.10: Threat types used to define the investigates scenarios, repeated from Section 2.3.3.

N, are analysed: a case for normal and safe operation of the power distribution system, and two cases where the power system experiences some problems. One of the investigated problems is that the power consumed by the customers is very high, and the other addressed problem is a broken fuse. The main objective of these scenarios is to test the integration between the tool components, and to perform a model validation for a consistent and safe, as well as for an unsafe system state. The monitoring tool is expected to generate no alerts for the safe operation of the grid, however, it should generate safety alerts for the other two cases.

2. **Threat type A**. The values reported by the RTU are not consistent. Eight scenario cases, marked with letter A, are investigated in Section 6.3.2. The not consistent measurements could be a result of a broken sensor or a deliberate change of the readings by an adversary. The objective in this scenario is to validate the consistency rules. The monitoring tool is expected to generate at least one alert on each investigated case.
3. **Threat type B**. Commands from the control room request an undesired value for set points. Two scenario cases, marked with letter B, are investigated in Section 6.3.3. The objective of this scenario is to evaluate the calculated new state after changing the set point. The monitoring tool is expected to generate at least one alert about the command to change the set point, and then generate alerts about the set point value stored in the RTU.
4. **Threat type C**. Commands from the control room request changes to the system that can bring it into an unsafe state. Six scenario cases, marked with letter C, are presented in Section 6.3.4. The objective of this scenario is to validate the safety rules. The monitoring tool is expected to generate a safety violation alert upon each case that brings the system to an unsafe state.

The exact description of each case in each scenario is provided in Section 6.3, where we discuss the results of the evaluations.

6.2.3 Implementation of Scenario Cases

Every scenario case begins with the power grid being in normal operation, as explained in Section 6.2.2. Then, depending on the threat type, either the RTU or the SCADA server will act in an undesired way. For the threat type A, the RTU changes values of process variables, and for the threat type B and C, the SCADA server issues unwanted commands (see the threat types on Figure 6.10). For every scenario case, the traffic is stored as a `*.pcapng` file using `tcpdump`.

In normal operation, the RTU is simply acting as a server listening on port `10502`, while the SCADA server asks for regular updates of all values stored

on the RTU. Scenarios for threat type A are generated either by creating a special thread within the RTU that alters a chosen value, or by adding an extra register or coil and creating a fake mapping for a tag name. The former approach is taken for changing value of sensors, while the latter method is used for altering the position of a transformer tap or the state of a switch. Traffic observed for each case is stored as ‘‘ModbusScenario_CaseAX.pcapng’’, where X stands for the number of the case. The console output is provided in ‘‘ModbusScenario_CaseAX.txt’’ file. Scenarios of threat type B or C are generated by issuing the SCADA server with a case description, which is stored in a `.txt` file. For implementing scenarios of type B or C, the SCADA server is executed with a scenario case description ‘‘*_CaseBX.txt’’, or ‘‘*_CaseCX.txt’’, respectively; X stands for the number of the case. The case description contains one of the actions given in Listing 6.1 and discussed below.

```
1 GI
2 WAIT <seconds>
3 CO <start address>, <unit id>, <value>
4 REG <start address>, <number of registers>, <unit id>, <value>
```

Listing 6.1: Possible case actions for scenario type B and C.

‘‘GI’’ indicates a “general interrogation” command which requests all values located on the RTU. The ‘‘WAIT’’ command tells the SCADA server to not perform any actions for a number of seconds specified by the parameter. ‘‘CO 8 1 False’’ is a request to change coil number 8 located on the device with unit ID 1 to False. Finally, ‘‘REG 40 4 1 12.5’’ is a command to change the value of 4 consecutive holding registers starting at number 40 to 12.5.

All scenario cases are captured using `tcpdump` and are available on our github repository[8]. Therefore, all analyses can also be performed offline. The next section provides a detailed description of the investigated scenario cases, together with a discussion of the results provided by the SAM tool.

6.3 Discussion of the Results

This section provides a detailed description of the analysed scenario cases and the explanation of the output of the SAM tool. Section 6.3.1 provides discussion of the tool evaluation of the normal operation of the power distribution grid. Section 6.3.2 evaluates 8 different cases of not consistent sensor readings, and Section 6.3.3 discusses 2 cases of unsafe set points. Finally, Section 6.3.4 analyses executing 3 cases of safe and 3 cases of unsafe commands.

[8]See Appendix A for details.

Table 6.4: Descriptions of scenarios of normal operation of the power grid, and the violated rules.

#	description	violated rules
N1	basic scenario: just reading the measurements, for a safe operation of the grid	None
N2	run the basic scenario but for higher values of power consumed by the loads	R1
N3	run the basic scenario but with the state of fuse F_{104} (this can be done by disconnecting the switch imitating the fuse)	R3, R6

6.3.1 Normal, Safe and Unsafe Operation

During the normal operation of the power distribution grid, only the content of the measurements is tested with respect to the system state's consistency and safety. We run three scenarios: a basic scenario, where no problems occur, a scenario with high load, where the values of the current on the power lines can get very high, and a scenario with a broken fuse. These cases are labelled with letter "N" and are summarised in Table 6.4.

```
1  Checking CONSISTENCY BUS node_b3
2    [x] P1: Kirchhoff's current law holds.
3    [x] P2: All voltage values equal.
4    [x] P3: Switch/Fuse/Protective Relay open -> no current.
5    [x] P4: Current and voltage equal at beginning and end of line.
6  [x] CONSISTENCY BUS node_b3
7
8  Checking SAFETY BUS node_b3
9    [x] R1: Current does not exceed maximum of power line.
10   [x] R2: Nominal voltage boundary of power line is obeyed.
11   [x] R3: No fuse is molten / protective relay is open.
12   [x] R4: No fuse / protective relay has current above cutting current.
13   [x] R8a: Voltage set points are safe.
14   [x] R8b: Current set points are safe.
15   [x] R9a: Static interlocks are ensured.
16   [x] R9b: Dynamic interlocks are ensured.
17 [x] SAFETY BUS node_b3
```

Listing 6.2: Console output for state evaluation of bus B_3 for scenario case N1.

Listing 6.2 shows an excerpt of the console output of the evaluating rules for bus B_3 in scenario case N1. For each physical constraint, and for each safety requirement relevant for a specific element, a check is performed. If a rule is

satisfied, the corresponding box is marked with a cross (for example, "`[x] P1: Kirchhoff's current law holds.`"). If a rule is violated, the box is left empty (for example, "`[ ] P1: Kirchhoff's current law holds.`").

After all rules of a component have been processed, the result is marked with a cross if there has been no violation in any of the recursive evaluations (for example, "`[x] CONSISTENCY BUS node_b3`" in line 6 of the Listing 6.2).

Below, each of the three scenario cases are discussed in detail. We only discuss evaluating the state, as no commands are present in this scenario.

- **N1:** The normal operation scenario describes the operation of the system without performing any changes. After requesting values of all process variables located on the RTU, the tool performs automatic evaluation of the observed state. All consistency and safety checks hold true, as presented for bus B_3 in Listing 6.2.
- **N2:** When running the simulation for larger loads, a higher value of current is measured on all power lines. For the areas with high voltage this is still not a problem, as the values of the current do not exceed a few amperes, while the maximum allowed current equals, for example, $200A$. However, when transforming the voltage to $230V$, this increases the value of the current significantly, eventually resulting in current exceeding the allowed value and, therefore, violating rule $R1$. The tool warns about this with the following message: `Line 110. A=536.603990 (<= maxI = 500.000000). (False).`
- **N3:** In this scenario case, the state of one of the fuses is marked as `False`. The tool directly notifies about a molten fuse, marking rule $R3$ as failed. Moreover, as not all of the customers are connected to a source of power, the tool also warns about violating rule $R6$.

The above tests show that Zeek parses and interprets the process variables without any errors. The state evaluation component performs the assessment of consistency and safety rules without failures. For scenario cases N2 and N3, the tool detects violated rules, which correctly reflect the safety of the system under test.

6.3.2 Non-consistent Sensor Readings

As described in Section 5.4.2, non-consistent sensor readings are created by making changes to the RTU execution logic. A total of 8 cases have been investigated, as listed in Table 6.5. They correspond to the threat type A, therefore, they are labelled with that letter in the table. Column "description"

Table 6.5: Descriptions of scenarios of type A, and the violated rules.

#	description	violated rules
A1.1	changing current value ($+5A$) of sensor M_{32}, located at bus B_2 and power line L_3	P1, P4
A1.2	changing current value ($+7A$) of sensor M_{94}, located at transformer T_2 and power line L_9	P4, P6b
A2.1	changing voltage value ($+1kV$) of sensor M_{84}, located at transformer T_1 and power line L_8	P4, P6a
A2.2	changing voltage value ($+1kV$) of sensor M_{63}, located at bus B_3 and power line L_6	P2, P4
A3.1	changing the value of power produced by generator P_1^G to $5000W$	P5a, R7
A3.2	changing the value of power consumed by load P_1^C to $5000W$	P5b, R7
A4	changing the reported position of the tap switch of transformer T_2 to 0 (real value: 2)	P6a, P6b, R5a, R5b
A5	changing the reported state of switch S_{42}, located at bus B_2 and power line L_4, to `False` (real value: `True`)	P3

explains the change performed in the system, and column "violated rules" lists the rules that did not hold, as reported by the tool.

Below we discuss how the tool assesses the scenarios described in Table 6.5. Again, since no command is present in the traffic, only state evaluation is discussed.

- **A1.1:** Assume that sensor $M_{32}.I$ is either broken or manipulated. Because of that, it reports continuously $5A$ more than it should, violating rule $P1$, that stands for the Kirchhoff's current law, and rule $P4$, which indicates that the current at the beginning of the line is not equal to the value at end of the line. This is reported by the tool, as shown in Listing 6.3.

- **A1.2:** Sensor $M_{94}.I$, located at the primary windings of transformer T_2, is either broken or manipulated, measuring $7A$ more than it should. This causes again violation of rule $P4$. Moreover, as the measurements of current on the primary and secondary side of the transformer are not consistent with the transformer ratio, rule $P6b$ is violated.

- **A2.1:** Sensor $M_{84}.V$ reports $1kV$ more than it should. As voltage at the beginning of the power line differs from the one at the end of the power line, rule $P4$ is violated. Also, the measurements of voltage on the primary and secondary side of the transformer T_1 are not consistent with

the transformer ratio, which defies rule $P6a$, as sensor $M_{84}.V$ is located at the primary windings of a transformer.

- **A2.2:** In this case voltage sensor $M_{63}.V$, located at bus B_3, reports a value which is $1kV$ larger than it should. This violates again rule $P4$, but also, as voltage measured by various sensors at the bus B_3 differs, $P2$ does not hold.

- **A3.1:** The value of power produced by generator P_1^G is set to a constant value of $5000W$. This change causes a violation of the physical constraint $P5a$, that indicates that the power of the generator is not equal to voltage times current measured on the sensor located at the generator. Also, safety requirement $R7$, stating that value of consumed and produced power should be equal at all times, does not hold.

- **A3.2:** The value of power consumed by the load P_1^C is set to a constant value of $5000W$. This change causes a violation of the physical constraint $P5b$, which checks whether the power of the consumer is equal to voltage times current measured on the sensor located at that consumer. Moreover, the safety requirement $R7$ is violated again.

- **A4:** The RTU reports a different value of the position of the tap switch. The tool uses the transformer ratio, and the values of voltage and current on the primary side of the transformer, in order to predict the resulting secondary values of voltage and current. As the ratio is artificially changed, the predicted values differ from the actual measurements of voltage and current on the secondary side, violating physical constraints $P6a$ and $P6b$. Moreover, using the (fake) tap switch position, the tool predicts that the secondary voltage will exceed the allowed voltage bounds, violating safety requirements $R5a$ and $R5b$. Here, the *measured* value is within bounds. However, given the (fake) transformer rate, and the voltage on the primary side of the transformer, the *predicted* value is unsafe.

- **A5:** Finally, the RTU reports a different value of the state of switch S_{42}. The tool reports a violation of rule $P3$, since the line separated by a disconnected switch should not carry any current.

Listings 6.3 and 6.4 show two excerpts of the console output for scenario cases A1.1 and A4. In Listing 6.3, the tool evaluates the relevant rules, one by one. Above rules $P1$ and $P4$, in lines 2 and 6, respectively, an explanation is provided which justifies why that rule is not satisfied.

```
Checking CONSISTENCY BUS bus2
    Ingoing current: 6.748472 (==) Outgoing current: 1.748472. (False)
  [ ] P1: Kirchhoff's current law holds. (local)
  [x] P2: All voltage values equal. (local)
  [x] P3: Switch/Fuse/Protective Relay open -> no current. (local, semi-global)
    Line l3. Local: V=9999.966485,A=5.582824 (==) Remote: V=9999.971270,A
        ↪ =0.582824. (False)
  [ ] P4: Current and voltage equal at beginning and end of line. (semi-global)
[ ] CONSISTENCY BUS bus2
```

Listing 6.3: Console output for state evaluation of bus B_2 for scenario A1.1.

Listing 6.4 evaluates all rules relevant for transformer T_2. For rules $P6a$ and $P6b$ it provides the expected value of the secondary voltage and current, based on the known voltage ratio.

```
Checking CONSISTENCY TRANSFORMER tap-transformer_2
  [x] P3: Switch/Fuse/Protective Relay open -> no current. (local, semi-global)
  [x] P4: Current and voltage equal at beginning and end of line. (semi-global)
    Transformer tap-transformer_2. Measured input voltage: 9999.835298V.
        ↪ Measured output voltage: 5999.849926V. Expected output voltage:
        ↪ 6666.556865V. (False)
  [ ] P6a: Transformer transformation rate is consistent in voltage. (local)
    Transformer tap-transformer_2. Measured input current: 3.798832A. Measured
        ↪  output current: 6.331391A. Expected output current: 5.698249A. (False
        ↪ )
  [ ] P6b: Transformer transformation rate is consistent in current. (local)
  [x] P7: Transformer has transformation rate defined. (local)
[ ] CONSISTENCY TRANSFORMER tap-transformer_2
```

Listing 6.4: State evaluation of transformer T_2 for scenario A4.

In the above tests, several rules are violated, depending on the analysed case (see Table 6.5). All of them correctly reflect the change applied in each scenario case. The investigated scenarios together caused violation of all the physical constraints $P1$–$P6$, and additionally, of safety requirements $R5a$, $R5b$ and $R7$. For the traffic captures taken during each case, and the console output generated from evaluating that capture, see Appendix A.

6.3.3 Unsafe Set Points

In these scenario cases, the commands requesting to change a set point are evaluated. Two cases are defined, which are described in Table 6.6. Each of the cases consists of several commands of changing a set point.

Table 6.6: Descriptions of scenarios of type B, and the violated rules.

#	description	violated rules
B1	changing the current set point value of sensor M_{51} to $230A$, $205A$ and $150A$	R8b
B2	changing the voltage set point of sensor M_{104}, located at transformer T_2 and power line L_{10} to $260V$, $225V$ and $180V$	R8a

In both scenarios, a command to change a set point is sent from the control room to the RTU. To assess safety of such command, the tool has to first precalculate the result of the command on the system, based on its current state. An example of a command evaluation is shown in Listing 6.5. The tool detects a set point command (line 1), then it evaluates the currently observed state (lines 3–4), and the state of the system that is calculated after executing the command (lines 5–7). As shown, rule *R8a* is not satisfied for the calculated system state. Therefore, the tool concludes that the new set point value is not safe (line 10).

```
1  Command detected: Set REF−NODE_B4 to 260.0
2  (Voltage set point change)
3  Observed state evaluation:
4          [x] R8a: Voltage set points are safe (local)
5  Calculated state evaluation:
6              Line l10. Voltage set point = 260.000000V (in [207.00,253.00]V). (False)
7          [ ] R8a: Voltage set points are safe (local)
8  Safety before command (R8a): True
9  Safety after command (R8a): False
10 New set point is NOT safe.
```

Listing 6.5: Console output for evaluation of the calculated state when changing set point in scenario B2.

- **B1:** The safe range of set point for electric current for meter M_{51} is defined as the maximum current of line $L_5.I_{max} \pm \beta \cdot L_5.I_{max}$, where $\beta = 10\%$. As $L_5.I_{max} = 200A$, this range equals $[180.00, 220.00]A$. The first ($230A$), and the third ($150A$) change of the current set point of meter M_{51} is evaluated to be unsafe, as they are outside of the desired range. The second change ($205A$) is considered safe, so rule *R8b* is not violated.
- **B2:** The safe range of voltage set point for meter M_{104} is defined as the reference voltage of line $L_{10}.V_{ref} \pm \alpha \cdot L_{10}.V_{ref}$, where $\alpha = 10\%$. As

$L_{10}.V_{ref} = 230V$, this range equals $[207.00, 253.00]V$. Again, the first ($260V$), and the third ($180V$) change of the voltage set point are evaluated as unsafe, as they are outside of the desired range. The second change ($225V$) is considered safe, so rule $R8a$ is not violated.

For both cases, the state of the system is considered unsafe when set points are configured to a value outside of their predefined bounds[9]. This violates the safety rules $R8a$ (set point of voltage) as well as $R8b$ (set point of current). In each case, the effect of changing the value of a set point is correctly predicted and classified as either safe (for values that belong to the predefined value range) or unsafe (otherwise).

6.3.4 Unsafe Commands

For this scenario type, 6 cases have been evaluated, summarised in Table 6.8. The commands refer either to changing a switch state or to changing a tap switch position of a transformer. The analysed power distribution system has two interlocks defined in Table 6.7. A static interlock K_1 contains switches S_{62} and S_{72} and requires that at least one of these switches is connected. A dynamic interlock K_2 contains switches S_{31}, S_{41}, and S_{51} and requires that the minimum guaranteed current capacity of the connected power lines equals $290A$.

Table 6.7: List of interlocks in the investigated topology.

interlock	type	property	value
K_1	static	$K_1.S$	S_{62}, S_{72}
		$K_1.ST_{min}$	1
K_2	dynamic	$K_2.S$	S_{31}, S_{41}, S_{51}
		$K_2.I_{min}$	$290A$

Using the command and the most recent system state, the tool predicts the outcome of that command on the controlled physical system. This computed state is then evaluated w.r.t. the safety rules. An example of such an evaluation is shown in Listing 6.6. Lines 1–2 show that the tool received the command of changing the state of the switch. Lines 3–5 evaluate the currently observed state of the system as safe. Lines 6–17 evaluate the safety of the calculated state. Only the violated safety rules are presented in Listing 6.6, replacing the rest with ellipses.

[9]The bounds are defined with parameters $\alpha = 10\%$, and $\beta = 10\%$, see Section 4.4.3.

```
Command detected: Set TAP−TRANSFORMER_2_TAP to 0.0
(Transformer tap position change)
Observed state evaluation:
    Checking SAFETY TRANSFORMER tap−transformer_2
    [x] SAFETY TRANSFORMER tap−transformer_2
Calculated state evaluation:
    Checking SAFETY TRANSFORMER tap−transformer_2
      (...)
        Line l11. Local: V=6666.637208 (in [5400.00;6600.00]). (False)
      [ ] R2: Nominal voltage boundary of power line is obeyed. (local)
      (...)
        Transformer tap−transformer_2. Nominal input voltage: 10000.000000V.
            ↪ Nominal output voltage: 6000.000000V. Transformed nominal
            ↪ output voltage: 6666.666667V (should be in [5400.00;6600.00]). (
            ↪ False)
      [ ] R5a: Transformer transformation rate is safe on nominal voltage. (local)
        Transformer tap−transformer_2. Actual input voltage: 9999.955813V.
            ↪ Nominal output voltage: 6000.000000V. Transformed actual output
            ↪ voltage: 6666.637208V (should be in [5400.00;6600.00]). (False)
      [ ] R5b: Transformer transformation rate is safe on actual voltage. (local)
      (...)
    [ ] SAFETY TRANSFORMER tap−transformer_2
Safety before command (R1,R2,R4,R5a,R5b): True
Safety after command (R1,R2,R4,R5a,R5b): False
New transformer tap position is NOT safe.
```

Listing 6.6: Evaluation of the calculated state for command $T_2.p = 0$.

Table 6.8: Descriptions of scenarios of type C, and the violated rules.

#	description	violated rules
C1.1	changing the state of switch S_{62}, located at bus B_2 and power line L_6 to `False` and `True`	None
C1.2	changing the state of switch S_{62} and S_{72} to `False`	R9a, R6, P4
C2.1	changing the state of switch S_{51}, located at bus B_1 and power line L_5 to to `False` and `True`, then doing the same for switch S_{31}	None
C2.2	changing the state of switch S_{41}, S_{31} and S_{51} to `False`	R9b
C3.1	changing the tap switch position of transformer T_2 from position 2 to 1, 1 to 2 and from position 2 to 3	None
C3.2	changing the tap switch position of transformer T_2 from position 2 to 0 and from position 0 to 4	R2, R5a, R5b

For scenarios C1.1, C2.1 and C3.1 the state is always assessed as safe. For the remaining scenarios, the state is considered unsafe with the reason provided in the "violated rules" column of Table 6.8. The evaluation is performed after every command: not only for unsafe commands but also for allowed actions.

- **C1.1 and C1.2:** Switches S_{62} and S_{72} belong to the static interlock K_1 listed in Table 6.7. Disconnecting just one of the two (as in scenario C1.1) does not result in a violation of the interlock, and neither does reconnecting it. However, disconnecting both switches (as in scenario C1.2) results in a violation of rule testing the static interlocks, $R9a$. The evaluation tool displays the following message when evaluating the calculated future state: `Open switch on line l7. Switch is interlocked. Interlock switch states: [False, False] (>= 1). (False)`. Later readings show also that some customers are not connected to any source of power, which violates rule $R6$. This additionally triggers a violation of the consistency check $P4$, as the voltage at the beginning of power lines L_6 and L_7 is different than on the end of them.

- **C2.1 and C2.2:** Switches S_{31}, S_{41} and S_{51} belong to the dynamic interlock K_2 listed in Table 6.7, and are located on power lines L_3, L_4 and L_5, respectively. The maximal current values of these power lines are given in Table 6.2, and are equal $300A$, $200A$, $200A$, respectively. The required minimal current for the dynamic interlock K_2 is $290A$ (see Table 6.7). In scenario C2.1 switches S_{51} and S_{31} are disconnected, then reconnected one at the time. When switch S_{51} is disconnected, the capacity of connected power lines equals $500A$; when switch S_{31} is disconnected, that capacity equals $400A$. This does not violate the dynamic interlock K_2, therefore, the commands are evaluated as safe. In scenario C2.2, the first command to disconnect switch S_{41} is still safe, as the dynamic interlock is still obeyed (the resulting maximal current equals $L_3.I_{max} + L_5.I_{max} = 500A > K_2.I_{min}$). However, a command to disconnect also power line L_3 causes a violation of rule $R9b$ with the following message: `Open switch on line l3. Switch is interlocked. Interlock switch states + current capacities: [(False, 300), (False, 200), (True, 200)] (>= 290A). (False)`. Disconnecting switch S_{51} results in the same violation.

- **C3.1 and C3.2:** Transformer T_2 is initially set to position $T_2.p = 2$, providing a rate equal to $T_2.r = 1.67$ (see Table 6.2), which results in secondary voltage value of $5987.997V$. In scenario C3.1, this position is changed, resulting in rates 1.59 for position 1 and 1.75 for position 3. These changes do not cause any violations, as the resulting secondary

voltage is still within the allowed bounds $[5400; 6600]V$, equal $6289.28V$ and $5714.26V$, respectively. In scenario C3.2, changing the position of the transformer to 0, results in violation of rules $R2$, $R5a$ and $R5b$ for the precalculated state. Safety rule $R2$ validates that the measured voltage value lies within the allowed bounds, and for the sensor located at power line L_{11} this is not the case (see lines 9–10 of Listing 6.6). Rules $R5a$ and $R5b$ calculate the secondary voltage given the nominal, and the measured value of the primary voltage, respectively (see lines 12–15 of Listing 6.6). For tap switch position 0, the value of the secondary voltage given the nominal voltage on the primary side, equals $6666.667V$. The secondary voltage value given the measured primary voltage, equals $6666.637V$. Since, in both cases, the calculated secondary voltage lies outside of the safe bounds, these rules are also violated. Changing the tap position to 4 brings the secondary voltage value (equal $5434.759V$) back to the safe voltage range, hence, not resulting in a violation.

Scenarios C1.2, C2.2 and C3.2 were constructed to violate rules referring to a static interlock, a dynamic interlock and voltage safety bounds. All the violations were properly assessed by the tool in the precomputed, future state of the system at hand. The above evaluations show that the tool is correctly classifying inconsistencies in the measured data. Moreover, it is able to assess the safety of the future state of the system, which is computed based on the currently observed state and the parsed command.

6.4 From Local to (Semi-) Global Monitoring

Section 6.3 has shown that the SAM tool properly evaluates both the consistency and safety of the system state upon sensor readings, as well as the safety of the system state after executing a command. The goal of this section is to investigate how the following factors affect the outcome of the evaluated physical constraints and safety requirements.

- **Controlled elements.** Not all rules are relevant for all field stations. For example, not all field stations contain a transformer, therefore, rules related to transformers do not apply to these stations. Moreover, some of the rules need information from the entire system in order to be tested. This analysis is provided in Section 6.4.1.
- **Knowledge scope.** We want to evaluate how the amount of information available at the RTU affects the outcome of evaluated rules. To do this, first, we use the system as analysed in Section 6.2, and change the RTU configuration so that it only receives and updates information about elements located at bus B_2. We execute several commands for this scenario

and discuss the outcome of the SAM tool. Then, we expand the so-called *knowledge scope* of the RTU by adding information from the neighbouring nodes. This analysis is provided in Section 6.4.2.

- **System topology.** Here, we compare the system presented in Figure 6.11 with a system where an additional power line L_2 is added between buses B_1 and B_3. Again, we perform the same commands and increase the knowledge scope of the RTU and discuss the output of the SAM tool. This analysis is provided in Section 6.4.3.

6.4.1 Controlled elements

In this section we list which rules can be evaluated based on elements located in a substation. We assume that every substation contains at least one of the node types: a bus, a power generator, a power consumer and/or a transformer. Moreover, these nodes have to be connected to other elements of the power distribution system with at least one power line. Each power line is equipped with a meter. Optionally, a switch, a fuse, and/or a protective relay is present on a power line. An RTU controlling a single substation configures set points of current and voltage values, and, if applicable, also interlock relations. Table 6.9 shows an overview of the analysed rules, depending on the controlled components within the field station. Every station will have at least one of the first three elements: bus, transformer and/or a power source and/or consumer, and optionally one or more of the last two component categories: switches, fuses and/or protective relays. Letter "L" (local) denotes a rule that applies to the component named in the first column, and which only requires local information

Table 6.9: An overview of rules that are evaluated for each topology component

component	P1	P2	P3	P4	P5 a/b	P6 a/b	R1	R2	R3	R4	R5 a/b	R6 a/b	R7	R8 a/b	R9 a/b
bus	L	L		G			L+	L+						L*	
transformer				G		L	L+	L+			L			L*	
power source/ consumer				G	L		L+	L+					G	L*	
switch			L+									L+			L*
fuse/PR			L						L	L					

"L" denotes that only local information is needed for the rule to be evaluated, "G" requires knowledge of more than one RTU. Symbol "+" means that the rule can be evaluated more thoroughly if information from more nodes is available and "*" indicates that the rule is applicable if the RTU has that aspect configured.

to be evaluated. The symbol "+" indicates that the scope of the rule can be extended if information from other RTUs is available. As the set points and interlocks in the system can be, but do not have to be configured, we mark this with a "*", which stands for "applicable if configured". Finally, if information from more than one node is absolutely needed in order to evaluate this rule, then it is marked with "G" (global). For example, if a substation contains a bus connected to some power lines, the SAM tool can test whether the Kirchhoff's law for those power lines is met ($P1$) based only on local information from sensors monitoring the directly connected power lines. This is indicated with letter "L". However, checking if the current and voltage are equal at the beginning and end of the power line ($P4$) is only possible when the monitoring tool has the information coming from two neighbouring nodes. This is shown in Table 6.9 under column "$P4$": for every node (bus, transformer, or power source and/or consumer), this rule needs global information. Therefore, this rule is marked with letter "G". Safety rule $R1$ checks whether measurements of current reported by all sensors on a single power line are the same. It can be tested locally (for the local sensor only), but it can also verify the remote measurement, if such information is available. That is why this is marked with letter "L+". Interlocks (rule $R9$) can be checked locally only for substations that contain switches and only if they are configured. This is indicated with "L*".

6.4.2 Knowledge scope

In this section we evaluate how the amount of information available to an RTU affects the outcome of evaluated rules.

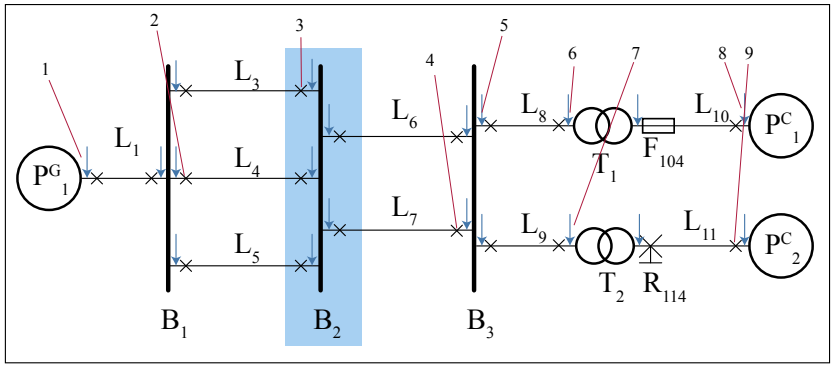


Figure 6.11: Topology used to test the rule evaluation depending on the amount of information available. Numbers 1-9 indicate targets of issued commands.

We use the system as analysed before, depicted again in Figure 6.11. *Knowledge scope* defines the amount of information available to a certain RTU. In Figure 6.11, the knowledge scope includes bus B_2, which is marked with a blue rectangle. This means that the RTU receives information about the states of the switches and sensor measurements on all power lines connected to bus B_2. If the knowledge scope includes buses B_2 and B_3, then, the RTU additionally monitors components connected to bus B_3. The knowledge scope of an RTU also applies to the SAM tool, if placed locally. The SAM tool only evaluates an incoming command, if it affects a component that belongs to the knowledge scope of the respective RTU.

Initially, the RTU is configured so that it only receives and updates information about elements located at bus B_2, marked with a blue rectangle in Figure 6.11. Then, we expand the knowledge scope of the RTU by adding information from the neighbouring nodes. As reference, we also monitor with the tool that has knowledge about the entire system. We perform the commands listed below. For better illustration, the elements that are referred to are also marked in Figure 6.11 with numbers 1–9.

1. Change voltage set point $M_{10}.V_{sp}$ to $11.5kV$.
2. Disconnect switch S_{41} (line L_4 with switch at bus B_1).
3. Disconnect switch S_{32} (line L_3 with switch at bus B_2).
4. Disconnect switch S_{73} (line L_7 with switch at bus B_3).
5. Change current set point $M_{83}.I_{sp}$ to $115A$.
6. Change transformation rate of transformer T_1 to 0.
7. Change transformation rate of transformer T_2 to 4.
8. Change current set point $M_{105}.I_{sp}$ to $560A$.
9. Disconnect switch S_{115} (line L_{11} with switch at customer P_2^C).

The traffic exchanged between the SCADA server and the RTU is captured and analysed with the SAM tool for different knowledge scopes. Table 6.10 shows which rules were violated on which elements, after sending each of the commands. The column "knowledge scope" contains the name of the node to which the command refers. In the next two columns, if applicable, the violated rule(s) and the component where they are violated, are listed. The remaining columns show the outcome of the evaluation of the tool, depending on the knowledge scope: global, B_2 only, B_1 only, $B_2\&B_1$, $B_2\&B_3$, and $B_1\&B_2\&B_3$. A - (dash) denotes that a command refers to an element outside of the knowledge scope of the RTU. In such case, this command was not evaluated by the SAM tool. A * (star) marks that a command is safe, refers to an object within the knowledge scope and the tool properly evaluated it as benign. A ✓(check mark) indicates a malicious command that the tool also correctly identifies as malicious. Finally, an X denotes a case where the tool did not detect a malicious command.

Table 6.10: Violated rules and components they refer to, for each command.

#	knowledge scope	violated rule	component	global	B_2	B_1	B_2 & B_1	B_2 & B_3	B_1 & B_2 & B_3
1	P_1^G	$R8a$	L_1	✓	-	-	-	-	-
2	B_1	none	none	*	-	*	*	-	*
3	B_2	none	none	*	*	-	*	*	*
4	B_3	none	none	*	-	-	-	*	*
5	B_3	$R8b$	L_8	✓	-	-	-	✓	✓
6	T_1	none	none	*	-	-	-	-	-
7	T_2	$R5b$	T_2	*/✓	-	-	-	-	-
8	P_1^C	$R8b$	L_{10}	✓	-	-	-	-	-
9	P_2^C	$R6$	P_2^C	✓	-	-	-	-	-

✓violation detected; X violation not detected; - not within the knowledge scope; * no violation

As shown in Table 6.10, not all commands result in an unsafe state of the system. For the commands that do generate a violation, the tool with the global knowledge correctly identifies that. Command 5 is detected as malicious also by the tool without the global knowledge, as long as the tool contains B_3 within its knowledge scope. Command 7, that requests a change of the tap switch position of transformer T_2 from 0 to 4, is correctly assessed as safe. However, after some time, the value of voltage decreases until it reaches $5399.997352V$, which is just out of the safe voltage bounds of $[5400.00; 6600.00]V$. Therefore, safety requirement $R5b$ is violated, which is reported by the tool upon receiving an update of the measurements.

Note that commands 1, 8 and 9 are detected as malicious by the tool with global knowledge. They can also be detected by the tool with only local knowledge, as long as the knowledge scope of the tool contains the knowledge scope that the command refers to. For example, the first command to change the set point on line L_1 can be detected by an RTU with knowledge of P_1^G.

6.4.3 Topology

Finally, we check how the investigated topology affects the local evaluation of the rules. To do that, we investigate the system shown in Figure 6.12, which is created by adding an additional power line L_2 between the buses B_1 and B_3 in the system from Figure 6.11. The maximum capacity of current on this power line is only $20A$.

We perform the same commands as listed in the previous section. The results

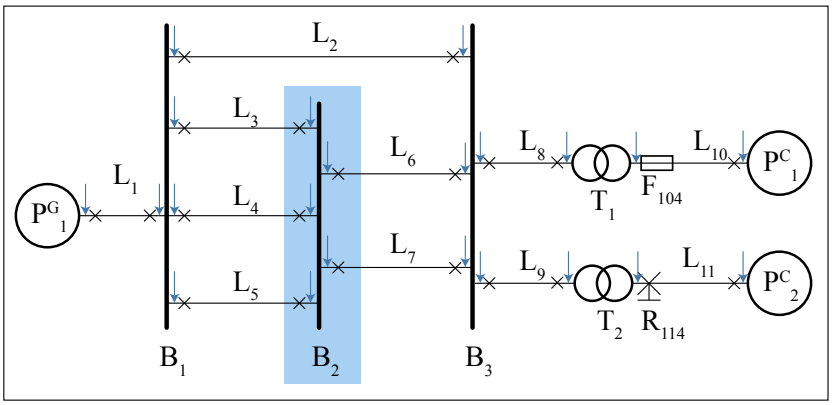


Figure 6.12: Altered topology used to test the rule evaluation: additional power line L_2 is added between buses B_1 and B_3.

Table 6.11: Violated rules and components they refer to, for each command.

#	knowledge scope	violated rule	component	global	B_2	B_1	B_2 & B_1	B_2 & B_3	B_1 & B_2 & B_3
1	P_1^G	*R8a*	L_1	✓	-	-	-	-	-
2	B_1	*R1*	L_2	✓	-	✓	✓	-	✓
3	B_2	*R1*	L_2	✓	X	-	✓	X	✓
4	B_3	*R1*	L_2	✓	-	-	-	✓	✓
5	B_3	*R8b*	L_8	✓	-	-	-	✓	✓
6	T_1	none	none	*	-	-	-	-	-
7	T_2	*R5b*	T_2	*/✓	-	-	-	-	-
8	P_1^C	*R8b*	L_{10}	✓	-	-	-	-	-
9	P_2^C	*R6*	P_2^C	✓	-	-	-	-	-

✓violation detected; X violation not detected; - not within the knowledge scope; * no violation

are shown in Table 6.11. Because the system contains a very weak power line L_2, more violations occur. When evaluating commands 2, 3 and 4, which were benign in the previous topology, the tool with a global knowledge correctly detects an unsafe state of the system. Command 2, that requests disconnecting switch S_{41}, results in an overcurrent on power line L_2. This is detected by the monitoring tool in every case where B_1 is within the knowledge scope of the tool. A correct detection of malicious command is indicated with a ✓symbol in the Table 6.11. Command 3, to disconnect switch S_{32} makes the situation even

worse. However, when the tool does not contain information about bus B_1, it misclassifies command 3 as safe. The reason for this is because this command is safe for the buses within the knowledge scope. These two cases, where the tool incorrectly classified the command as safe, are indicated in Table 6.11 with symbol X. Because the system is already in an unsafe sate, command 4 to disconnect switch S_{73} also results in such state.

6.5 Summary

This chapter described the architecture and an implementation of the Self-Aware Monitor - a tool that can be used to monitor the safety and security of an electrical substation by analysing the contents of the network traffic. The tool uses the modelling formalism introduced in Chapter 4 to describe the physical power distribution system. Upon sensor measurements, the model of the physical system stored in the tool is updated. Upon receiving a command, the tool precomputes the outcome of executing such a command on the observed system state to assess whether the outcome is safe in the context of the controlled system.

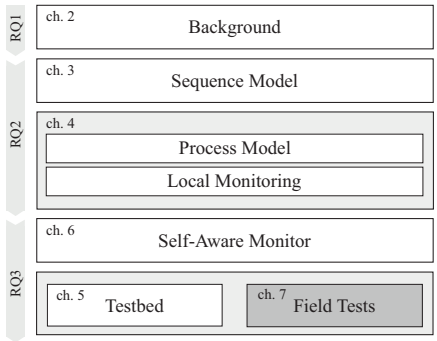
The evaluation of physical constraints and safety requirements, presented in Section 6.3, shows that the SAM tool correctly identified all inconsistencies and warned about commands that brought the system to an unsafe state.

The analysis presented in Section 6.4 shows that monitoring the system in a local fashion can be challenged if, for example, the topology of the monitored system contains loops. Changing the state of a switch connected to one bus can seem safe from the perspective of that bus, however, the effect of that change can influence power lines that are not connected to that bus. Old power distribution systems often contain loops, however, nowadays, a radial structure is advised for low voltage areas [146]. This can be achieved in the current infrastructure by disconnecting some of the switches permanently.

When implementing the monitoring tool in practice, it is therefore important to be aware of the topology of the system. If loops exist, it is necessary to broaden the knowledge scope of the tool.

Field Tests

This chapter presents a real-life case study of the use of the SAM tool described in the previous chapter. We investigate safety and security of a power distribution substation that communicates with the control room using the IEC-60870-5-104 protocol. In order to perform this case study, the proposed tool has been adapted so that it supports the IEC-104 protocol. A parser for that protocol has been developed using the Spicy framework, and appropriate event handlers are constructed and integrated with the prototype tool. The operation of the prototype tool is tested by connecting it to a network that controls a real power distribution substation, and by allowing it to analyse measurements and commands exchanged with the control room.



This chapter is organised as follows:

- *Section 7.1 introduces a parser for the IEC-104 protocol that allows for deep packet inspection of network traffic exchanged between a field station and the control room of the SCADA system.*
- *Section 7.2 explains the adaptations made to the SAM tool that allow for integrating the IEC-104 parser.*
- *Section 7.3 describes a case study used for testing the SAM tool.*
- *Section 7.4 concludes the chapter.*

Parts of this chapter have been based on [33]. Section 7.3 has been based on [45], whose work the author has been co-supervising.

7.1 IEC-104 Protocol Parser

This section presents a dedicated IEC-104 parser, implemented using the Spicy deep packet inspection framework [136], and its connection to the Zeek network monitoring tool [118]. We extend the parser proposed by Udd et al. [143] by implementing 34 IEC-104 functions (referenced by Type IDs), and extracting relevant process variables from the IEC-104 Data Units. Section 7.1.1 describes related work on parsers for real-time deep packet inspection. Section 7.1.2 explains our implementation of the IEC-104 parser, before Section 7.1.3 outlines the basic integration of that parser with Zeek. Finally, Section 7.1.4 evaluates the performance of the parser used in Zeek on traces of various lengths.

7.1.1 Related Work

As shown in Chapter 6, the Zeek network monitor can be used to implement process-aware detection of malicious commands. Zeek supports several SCADA protocols, such as Modbus/TCP or DNP3, however, the parsing of the widely used protocol IEC-104 is currently not supported. To the best of our knowledge, only one parser has been proposed for real-time monitoring for IEC-104, using a compiler-assisted tool called BinPac++ [143]. However, BinPac++ is no longer in use (it was replaced by Spicy [136]), which limits its current usability; moreover, [143] implements only 5 different Type IDs and parses solely control information w.r.t. the connection from the IEC-104 packets, and not the contents of the packets. In contrast, the solution presented here allows for parsing the content of the most relevant Application Service Data Units (ASDU) for monitoring individual substations.

7.1.2 IEC-104 Protocol Analyzer

This section presents the main components of the IEC-104 parser and illustrates its use within an Intrusion Detection System implemented in Zeek [118]. Every IEC-104 packet, a so-called Application Protocol Data Unit (APDU), contains a header called Application Protocol Control Information (APCI). S-frames (for numbered supervisory functions) and U-frames (for unnumbered control functions) are built from only the APCI. I-frames (used for information transfer), consist additionally of Application Service Data Units (ASDUs). ASDUs determine what kind of function and data type (the so-called Type ID) they carry, and they can contain up to 127 Information Objects (IOs), referring to different addresses on the RTU that is being controlled.[1]

[1] See Appendix B.2 for details on the IEC-104 datagram structure.

Table 7.1: IEC-104 protocol Type IDs supported by the proposed parser

nr	Type ID	nr	Type ID
1	M_SP_NA_1	49	C_SE_NB_1
2	M_SP_TA_1	50	C_SE_NC_1
3	M_DP_NA_1	51	C_BO_NA_1
5	M_ST_NA_1	58	C_SC_TA_1
7	M_BO_NA_1	59	C_DC_TA_1
9	M_ME_NA_1	60	C_RC_TA_1
11	M_ME_NB_1	61	C_SE_TA_1
13	M_ME_NC_1	62	C_SE_TB_1
21	M_ME_ND_1	63	C_SE_TC_1
30	M_SP_TB_1	64	C_BO_TA_1
31	M_DP_TB_1	70	M_EI_NA_1
32	M_ST_TB_1	100	C_IC_NA_1
33	M_BO_TB_1	101	C_CI_NA_1
34	M_ME_TD_1	102	C_RD_NA_1
35	M_ME_TE_1	103	C_CS_NA_1
36	M_ME_TF_1	107	C_TS_TA_1
45	C_SC_NA_1	142	proprietary
46	C_DC_NA_1	143	proprietary
47	C_RC_NA_1	200	proprietary
48	C_SE_NA_1		

Zeek uses a set of protocol parsers to process network data, and the information generated is analyzed, for example, to detect intrusions. For currently unsupported protocols, a parser can be built using the deep packet inspection framework Spicy [136]. It provides a format specification language and a compiler toolchain that compiles the protocol specification into a robust and efficient native code protocol parser.

We extend previous work of Udd et al. [143] to define the syntax of the IEC-104 protocol and implement 34 protocol Type IDs, as listed in Table 7.1. The five Type IDs highlighted in orange were already implemented in [143]. To summarize the applicability of the proposed solution, we characterize the implemented Type IDs below.

The measurement information from field stations to the control room is transported using data types with Type IDs beginning with 'M_' (monitoring direction). The controlling functions sent from the control room to field stations are sent using Type IDs with names beginning with 'C_' (control direction). Process values are sent using Type IDs and a data format defined by the operator. For example, a measurement of the current could be sent either using a normalized value with Type ID 9 or 34, a scaled value with Type ID 11 or 35, or as a floating point using Type ID 13 or 36.

Type IDs 1, 3, 5, 7, 9, 11 and 13 transport various data types without time

tags. They transport single point information[2], double point information[3], step positions, bit strings, normalized or scaled measured values and floating points, respectively. Type IDs 30–36 transport the same data types but with time tag of format CP56Time2a as defined in the IEC-60870-5-101 standard [68]. Type IDs 45–51 refer to commands of setting values for the same data types as listed before, without time tag, while Type IDs 58–64 do the same and include the time tag of format CP56Time2a. Type ID 2 is not supported by IEC-104, as it contains a single point information with an outdated time tag CP24Time2a, defined in IEC-101 [68].

Type IDs 100 and 101 are interrogation and counter-interrogation commands, respectively. They are always sent after establishing a connection between the control room and the field station, and when the control room requests the most up-to-date view of the measurements in the field station. With the Type IDs described above, it is possible to parse traffic containing the most commonly used functions. The parser is available on Github (see Appendix A) and can be tested using exemplary IEC-104 traffic [149].

7.1.3 Basic Connection of the Parser and Zeek

Having specified the grammar of the IEC-104 protocol and compiled a parser, it can be connected to Zeek to parse network traffic. When processing predefined objects within the Spicy code, events are generated and evaluated within Zeek. For example, processing a control function-related U-frame triggers an event as follows:

```
on T104::Apci if (self.ctrl.mode == 3) -> event t104::u($conn);
```

Here, if parser `T104` encounters an `Apci` object with property `ctrl.mode` equal 3, a `t104::u` event is created with argument `$conn`, which refers to the connection information.

Depending on the desired level of detail, various events can be created. For example, in order to analyse the values from a single Information Object within an ASDU carrying multiple objects, it may be necessary to create events for every created object. Depending on the Type ID, the parser needs to export a proper component, and create an event. For example, for Type ID 34 - M_ME_TD_1, which contains the monitoring information for a measured value in normalized format with time tag CP56Time2a, the following event is created:

[2]Single point infromation is a single bit of data. It allows to represent two states of an element: `on` or `off`.

[3]Double point information datatype are two bits of data. It allows for representing 4 states of an element: `on`, `off`, and two intermediate states.

```
on T104::Measured_Val_Normalized_Val_TTCP56 ->
             event t104::m_me_td_1s($conn, T104::bro_m_me_td_1(self));
```

The event generated above has two arguments: `$conn` contains the connection information, while `T104::bro_m_me_td_1()` is a function call that returns `self` (ASDU information). This function is located in the *.spicy file and defines the tuple which is passed to Zeek as an argument. For the above Type ID with measured values in normalized format and time tag, we obtain the Object's address and the normalized value measured for that Object.

```
tuple <uint64, double>
bro_m_me_td_1 (asdu : Measured_Val_Normalized_Val_TTCP56) {
        return (asdu.info_obj_addr, asdu.normalized_value); }
```

This function allows us to access the normalized value of the Information Object in Zeek, based on which, detection policies can be implemented, as explained in the following section.

7.1.4 Evaluating the Parser

To illustrate the feasibility of the proposed parser, we evaluated its throughput for varying flow lengths. We obtained an IEC-104 traffic trace on March 2, 2018 at a Dutch power distribution station, operated by our industrial partner Coteq. SCADA networks in general have a relatively low throughput; in this capture the IEC-104 throughput was around 0.1 packets per second (pps). However, in the future this throughput might increase due to high communication demands related to load balancing and self-healing. Therefore, we have decided to stress-test the parser.

In order to stress-test the monitoring tool linked to the parser, we used the Scapy Python library to create four groups of IEC-104 traffic traces, each with a different number of packets per TCP flow, that is, 1048, 10130, 50043 and 100233 packets per flow. From these, we created different traffic traces, with total lengths ranging between 20000 and 500000 packets. These traces have each been processed ten times by the parser connected to Zeek. We ran the tests on an Oracle VM VirtualBox with two logical processors with Intel Core i5, 2.9 GHz per core and 4 GB RAM. The resulting processing time (in seconds) and throughput (in pps) are shown in Figures 7.1 and 7.2, respectively. To indicate the variability in the measurements, 95% confidence intervals are depicted.

Figure 7.1 shows that the processing time for each TCP flow length increases linearly with the total trace length. However, the longer a single TCP flow is, the faster the processing time increases. The throughput as presented in Figure 7.2, however, shows that for different TCP flow lengths, the throughput

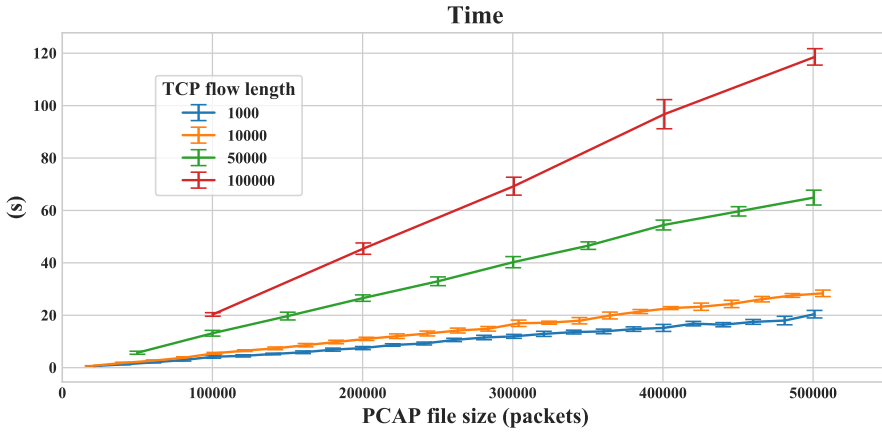


Figure 7.1: Processing time as function of the file size for multiple TCP flow lengths.

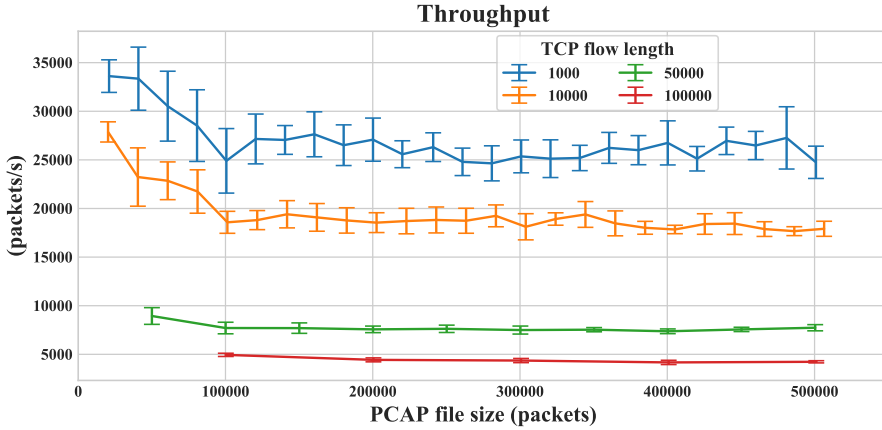


Figure 7.2: Throughput as function of file size for multiple TCP flow lengths.

of the monitoring tool reaches a constant (albeit different) value. Moreover, for shorter TCP flows (1000 and 10000 packets), the throughput is initially higher and then decreases until that constant value is reached. The processing times have been measured within Zeek and the measurements were only started after the initialization phase of Zeek was finished. The measurements demonstrate that the performance of Zeek is strongly influenced by the total flow length in the trace. As indicated on the Zeek website[4], the performance of Zeek highly depends on various parameters and currently no performance benchmarks are available for comparison.

As SCADA traffic usually consists of long TCP flows [11], it is important to be aware of the throughput drop for such long TCP flows. The obtained results are based on a single trace only, hence, may not be generic enough to fully evaluate the performance of our parser. However, we do believe that the proposed parser in combination with Zeek is fast enough to be used for real-time monitoring of SCADA traffic on single RTUs (as also proposed in related work [29, 92]), since such networks have a relatively low throughput [9].

7.2 The SAM Tool with IEC-104 Parser

This section describes how the SAM tool (as introduced in Chapter 6) is adapted to be used with the IEC-104 protocol. Note that only the subcomponents highlighted in yellow in Figure 7.3 are altered. They all belong to the "IDS and parser" component of the tool, marked with number 2 in Figure 7.3. The component for the power distribution grid and the evaluation logic (numbered 1) remains unchanged: it just requires the updates of $(tag, value)$ for the tags it has obtained from the topology.

In order to facilitate the operation of the monitoring tool for the IEC-104 protocol, the following elements have been adjusted:

(i) the subcomponent "protocol description" uses the parser defined in Section 7.1,
(ii) new event handlers are defined for that protocol, and
(iii) a new method for interpreting the values according to the RTU configuration is provided.

In the following, changes in handling the RTU configuration and in processing the events are explained.

[4] `https://www.zeek.org/development/projects/benchmark.html`

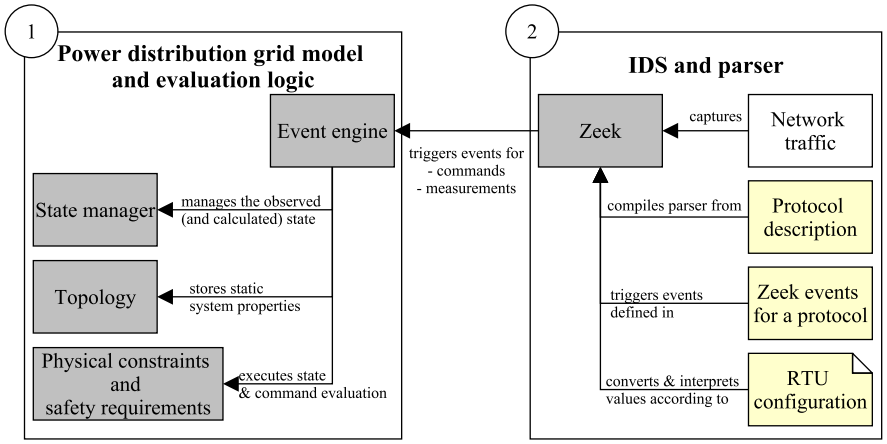


Figure 7.3: The elements of the prototype tool highlighted in yellow require modification in order to be compatible with the IEC-104 protocol.

RTU Configuration

The configuration of an IEC-104 RTU differs from the configuration of a Modbus RTU. Table 7.2 shows three rows of the configuration of the RTU used in the case study. Only the most relevant columns from the configuration of that RTU are presented. For anonymity, the values are changed, however, the structure of the configuration remains the same. The column `CA` provides the common address of the local device. The information object addresses (`IOA_M`, `IOA_C`) refer to the memory location at the local device. Note that they both refer to the same underlying memory location, because IEC-104 uses different virtual address spaces for measurement and command functions. These addresses are followed by a description of the physical context value of the variable, whereas the name provided in the column "tag name" serves as a unique reference to that variable. The last two columns provide the lower and upper bounds of the variable, which are used to write the variable in the normalized or scaled format, as explained in the next section. Some entries do not require any bound values, such as the last entry in Table 7.2, referring to a switch state.

The state manager requires an unambiguous tag name, that identifies a physical system component. The "tag name" provided in the configuration of the RTU file is unique within a system, and can be used for this purpose. This tag name has to be mapped to the process variable stored on the RTU. The process variable in the IEC-104 protocol is identified with the common address,

Table 7.2: Excerpt from the RTU configuration of the RTU from the case study.

CA	IOA_M	IOA_C	description	tag name	lower bound	upper bound
1001	5001	10001	Measured Current	R1_B1_M11_I	-2000	2000
1001	5002	10002	Measured Voltage	R1_B1_M11_V	-15000	15000
1001	5301	10301	Switch State	R1_B1_SW11_ST		

which refers to a unique SCADA device like the local RTU, and the information object address, which refers to a memory register on that device. The tuple (`CA`,`IOA`) is then translated into the physical process context (that is, the tag name).

To enable Zeek to map the protocol-specific and RTU-depending (`CA`,`IOA`, $rawValue$) tuple to an independent ($tag, value$) tuple, a script was developed which reads an RTU configuration and automatically generates Zeek code for the conversion process.

The translation of raw values into the actual process values is explained in the next section.

IEC-104 Events

As compared to the Modbus/TCP protocol, the IEC-104 protocol provides support for a more practical data representation. Dedicated functions are used to transfer analog values as floating point values, as scaled or as normalized values [69].

- **Normalized values** are numbers in the range $\langle -1; 1 \rangle$, where the precision depends on the number of bits available. If the resolution of the measured value is less precise than the unit of the least significant bit, then the least significant bit is set to 0 [35, 68].
- **Scaled values** transmit values with a defined fixed decimal point. Values span the interval $\langle -32768; 32767 \rangle$, and the range and position of the decimal point are stored in the configuration of the RTU and central server [35, 68].

For example, when using normalized values, both the main server in the control room and the RTU in the field stations store reference values of all process variables, for example, PV_{ref}. The transmitted value PV_{trans} of a real value PV_{real} is then defined as $PV_{trans} = \frac{PV_{real}}{PV_{ref}}$. For example, if the reference value

for the current on a line is equal to $1000A$ and the measured value is $150A$, then 0.15 is sent as normalized value to the control room. The reference values stored in the RTU are present also in the configuration file, as shown before.

For the tool, this means that all the functions (that is, all of the Type IDs used in the analysed SCADA communication) have to be supported and interpreted. Also, the raw value transported by each function has to be interpreted to its real value. For example, if the variable R1_B1_M11_I (as depicted in the Table 7.2) stores measured current as $1000A$ and the normalization interval of that variable is $[-2000, 2000]$, the transmitted normalized value is $+0.5$. Hence, a parsed raw value of, for example, $+0.75$ should be interpreted as $1500A$.

7.3 Case Study

This section describes the case study performed at a Dutch power distribution substation. Section 7.3.1 explains the goal of the case study. Section 7.3.2 describes the topology of the investigated system, Section 7.3.3 lists the test setup and describes the collected data. Finally, Section 7.3.4 provides the results and discussion.

7.3.1 The Aim of the Case Study

Our goal is to test the capabilities of the SAM tool to warn about commands that bring the power system into an unsafe state. At the same time, we do not want to risk putting the real power system in that unsafe state. As the tool only *detects* the commands and *does not prevent* them, unsafe commands cannot really be performed in the real-life system. Moreover, we would like to test as many physical constraints and safety rules as possible. Not all are relevant for the substation under test: for example, no transformer is present in this station, therefore, rules concerning transformation rate are not evaluated. Therefore, together with the power distribution operator, we have discussed a way to design three attack scenarios, as follows. In some of the tests the monitoring tool is using slightly different information about the topology, imitating a "weaker" system than the real one. By weaker, we mean a system that has smaller capacity and more constraints, than the real one. The exact changes to the topology information are presented in Section 7.3.4.

7.3.2 Description of the Physical System

The tests were conducted on August 15th, 2018 at a Dutch power distribution substation implementing the IEC-104 protocol. Figure 7.4 shows an excerpt of the topology of the power distribution system at the field station. The part

of the system directly monitored and controlled by an RTU present in this substation, is marked with a blue rectangle.

The description of the topology including values of all elements' properties, such as maximum allowed currents on all power lines, was extracted from a `*.vnf` file[5] provided by the operator. The four power lines on the left-hand side of Figure 7.4, that is, L_2, L_5, L_{10} and L_{13}, are feeders. They are considered as incoming power lines. All power lines on the right-hand side are considered as outgoing. The allowed maximal current values of all power lines, as indicated in the topology file, are provided in Figure 7.4. All power lines have a meter, and a switch with a protective relay, denoted as M_X, S_X and R_X, respectively. X is the number of the corresponding power line, as illustrated in the bottom left corner of Figure 7.4 for power line L_{13}. We only consider the meters, switches and protective relays directly controlled by the RTU. Furthermore, feeders L_5, L_{10} and L_{13} are connected to the same source bus. Initially, all of the switches controlled by the local RTU are closed. The remote switch of power line L_9 is disconnected.

7.3.3 Description of the Test Setup

In order to perform the tests, a hub was connected between the RTU and the gateway to the control room. A hub copies packets to all interfaces, therefore, it copied the incoming packets not only to the local RTU but also to the connected monitoring device. In this way, it was possible to perform the packet analysis without interrupting the regular operation of the RTU on the controlled system. A scheme of the test setup is shown in Figure 7.5(a), and an actual photo of the test setup is shown in Figure 7.5(b). The monitoring was performed on a laptop running a Docker image[6] that contains all necessary packages and configurations installed. Commands were sent by the operator using HMI, shown in the right-hand side of Figure 7.5(a).

The tests were performed using the approach illustrated in Figure 7.6. Every test started with a general interrogation command sent to the local RTU, which provided the initial values necessary to evaluate the safety and consistency of the physical system. If necessary, the topology information could still be adapted after this initial check. The initial interrogation command also allowed for verifying whether all the physical constraints and safety requirements are met for the observed system state. Then, for every command sent from the control room, first, a general interrogation command was sent to the RTU, to refresh the

[5]The operator uses a program called `Vision` for power system planning. This program stores the description of the power system in a "Vision Network File", which uses `*.vnf` as extension.

[6]See Appendix A for the source to the SAM tool.

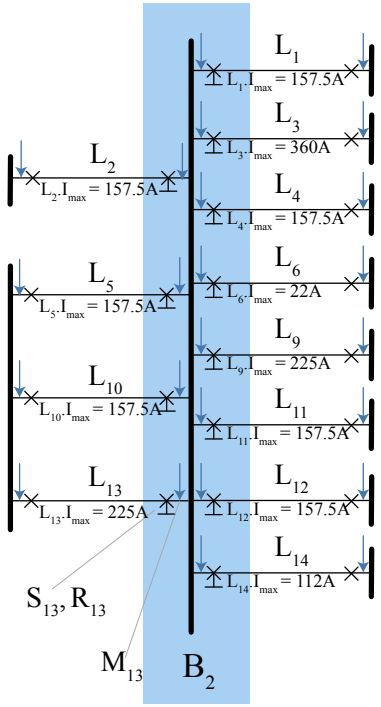


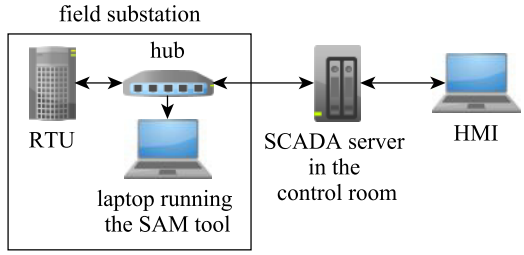
Figure 7.4: Excerpt from the power distribution system topology at the case study substation.

observed state of the physical system. This was followed by the actual command, and another general interrogation command that allows for assessing the safety of the resulting system state. If necessary, an additional general interrogation was done at the end of the tests. Beside being analysed by the monitoring tool, all the network traffic was additionally captured using `tcpdump`.

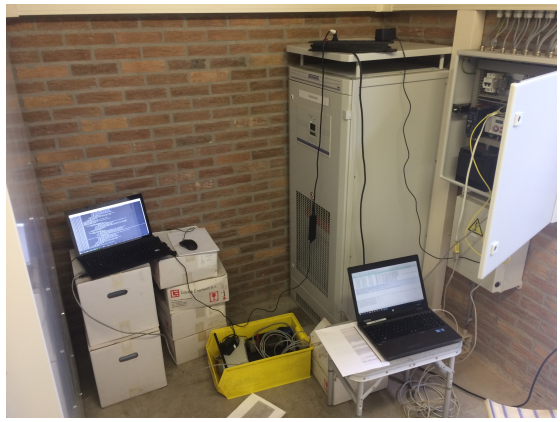
A total of four files containing network traffic was captured. Table 7.3 summarizes some properties of the captured traffic: the duration, the packet count[7], and the number of IEC-104 commands and measurements. Each test has a different goal, as summarised in its name:

- The **reference** test is used to obtain a baseline of normal system operation; apart from the general interrogation commands, 2 set point and 4 switching commands are sent during this test.

[7]The total packet count includes, for example, DNS queries and ARP packets.



(a) Scheme of the test setup.



(b) A photo of the actual test setup.

Figure 7.5: The test setup.

- The **set point** test contains commands to change set points to an unsafe value; 4 set point commands are sent in this test.
- The **switching** test uses commands to change states of switches, causing violations of interlocks; 4 switching commands are issued.
- The **protective relay** test also uses commands to change states of switches, resulting in (a fictitious) overcurrent on one of the protective relays; 2 switching commands are sent in this test.

The first test was done to evaluate whether errors occur when parsing the IEC-104 traffic, and whether the tool is able to perform the state evaluation. Also, for several safe commands, we test whether the SAM tool performs the command evaluation without failures. Therefore, the captured traffic is longer (31 minutes and 34 seconds) than the other three tests (all below 9 minutes).

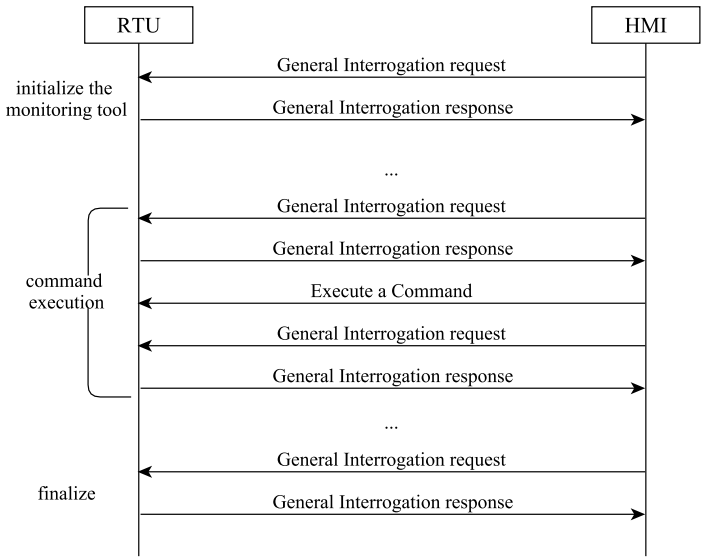


Figure 7.6: Sequence diagram visualizing IEC-104 communication between the SCADA server and the RTU during the tests.

Table 7.3: Summarized information about the captured traffic.

test	duration	packet count		IEC-104 events	
		total	IEC-104	commands	measurements
reference	31:34	2663	300	6	4796
set points	8:02	804	150	4	2789
switching	4:29	479	137	4	2351
protective relay	6:29	706	132	2	2369

7.3.4 State and Commands Evaluation

As mentioned before, in some of the tests, the SAM tool is using slightly different information about the topology. The changes to the topology information are indicated in the column “topology changes” in Table 7.4. This table also summarizes the executed commands and the violated rules.

Table 7.4: Summary of all the tests: the necessary preconditions, topology information changes, performed commands and violated rules.

test	preconditions	topology changes	commands	violated rules
reference	switches S_9 and S_{10} are closed.	-	$M_{10}.I_{sp} = 154A$, $S_9.st = False$, $S_9.st = True$, $S_{10}.st = False$, $S_{10}.st = True$, $M_{10}.I_{sp} = 158A$	$P1$, $P3$, $R1$, $R8a$
set point	current set point of $M_{10}.I_{sp} = L_{10}.I_{max}$	reference voltage $L_6.V_{ref} = 6kV$	$M_{10}.I_{sp} = 154A$, $M_{10}.I_{sp} = 126A$, $M_{10}.I_{sp} = 190A$, $M_{10}.I_{sp} = 157.5A$	$R2$, $R8a$, $R8b$
switching	switches S_9 and S_{10} are closed.	static interlock $K_1.S = \{S_9, S_{10}\}$, $K_1.CS_{min} = 1$, dynamic interlock $K_2.S = \{S_5, S_{10}, S_{13}\}$, $K_2.I_{min} = 320A$, $L_{13}.I_{max} = 120A$	$S_9.st = False$, $S_{10}.st = False$, $S_9.st = True$, $S_{10}.st = True$	$P1$, $P3$, $R1$, $R8b$, $R9a$, $R9b$
protective relay	switch S_{10} is closed.	$R_{13}.t_{cut} = 10s$, $R_{13}.I_{max} = 120A$	$S_{10}.st = False$, $S_{10}.st = True$	$P1$, $P3$, $R1$, $R4$

In the following, we present the four test scenarios. For each of them, first, the description of the output of the initial general interrogation is explained. Next, the outcome of each of the commands is provided. Following that, the violated rules are analysed, and finally, a general discussion of each test is provided.

A. Reference Test

Description The reference test is used to validate the integration and proper operation of the SAM tool. The initial general interrogation command generated some warnings, and consistency and safety violations, which were not expected. We list and explain them below.

- Violation of rule $P1$, which stands for the Kirchhoff's current law. The difference between the sum of current values on the incoming power lines,

and the sum on the outgoing lines was quite significant - up to 6–10%. After some tests, it was concluded that one of the feeders was, in fact, an outgoing line, while another power line, L_1 was an incoming line. After adjusting this information in the topology, the values between incoming and outgoing lines differed with approximately 3.5%, which can be explained by imprecisions of the measurements.

- Scripts generating the mapping of tags for Zeek supposedly used the latest configuration of the local RTU. Still, addresses of 12 unknown information objects were seen in the parsed traffic. This did not affect the operation of the prototype, but generated some warnings. As it turned out, the configuration was slightly out-of-date, and the new 12 objects referred to process variables indicating SCADA alarms on the power lines.
- Some of the set points were configured to much lower values than the values obtained from the topology file, violating rule *R8b*. There were two reasons for that: (i) the set point of power line L_{13} was not updated in the configuration of the RTU after the physical power line was upgraded, and (ii) for L_9, although the maximum current of the cable directly connected to bus B_2 equals $225A$, the threshold of a more distant power line equals only $157.5A$. These two deviating set points were changed manually in the topology file to reflect the currently configured set points.

The above described findings show that the proposed tool can additionally detect inconsistencies between the configurations and the current topology. Before further tests, the information about the feeders, current thresholds, and the imprecision of the measurements were taken into account in the topology description and in comparing the floating point variables. The unknown RTU configuration objects did not affect the evaluation of the rules, so the problem was investigated later. After these changes, the tool was not generating any consistency or safety violations.

Commands After the changes explained above were made, some set point commands and switching commands were executed. Below we discuss the outcome of each command.

- **Change set point of line L_{10} to $154A$.** As shown in 7.4, the maximum current for $L_{10}.I_{max} = 157.5A$, therefore, changing it to $154A$ does not violate the safety bounds of $I_{max} \pm 10\%$.
- **Disconnect switch S_9.** Disconnecting one switch at a time should be a safe operation, especially in case of switch S_9, the switch on the other side of L_9 was opened anyway. However, after executing the command, the measurements showed a mismatch between the sum of currents on incoming and outgoing power lines, violating rule $P1$. As it turned out, the current on the disconnected power line L_9 was not equal to 0, violating

rule $P3$. These inconsistencies are discussed later.

- **Connect switch S_9.** The command is assessed as safe, however, after its execution, another violation of $P1$ is observed, as the current on power line L_9 is still equal to 0 for some time.
- **Disconnect switch S_{10}.** An unexpected issue is reported: the tool predicts an overcurrent on line L_5, which violates rule $R1$. This will also be discussed below.
- **Connect switch S_{10}.** The command is assessed as safe, however, after its execution, $P1$ is violated again.
- **Change set point of line L_{10} to its original value of $158A$.** This does not result in any violations.

Violated rules All violated rules were either a result of outdated configuration information of the RTU, of a precision issue, or of delayed measurement updates.

Two physical constraint violations were observed every time a switch was disconnected: $P1$ (Kirchhoff's current law) and $P3$, which indicates that the current on a disconnected power line should equal 0. When disconnecting power line L_9, the measured current on that power line $M_9.I$ is directly invalidated. However, upon the next general interrogation, 16 seconds later, the sensor still shows a non-zero value of current on that power line, violating rule $P3$. Only after 22 seconds, the reported value changed to $0A$. In the meantime, also Kirchhof's current law ($P1$) is violated. When connecting power lines, rule $P1$ is violated for up to half a minute. This delay shows, that not only the frequency of updating the values on the SCADA server has to be taken into account when evaluating rules, but also the frequency of updating the meter measurements on the local RTU.

Moreover, when disconnecting power line L_{10}, the tool predicts that the current on power line L_5 will exceed its maximum allowed value (rule $R1$). Line L_{10} is one of three power lines connected to the same source bus, therefore, the current should be transported over the other feeders. However, disconnecting switch S_{10} reports an unexpected issue: the violation of rule $R1$. Just before disconnecting, power line L_{10} was carrying $92A$, while the other two feeders connected to the same source bus, L_5 and L_{13}, transported $114A$ and $101A$, respectively. Therefore, the tool calculates the resulting current carried over those lines, after disconnecting switch S_{10}. It results in values $M_5.I = 114A + \frac{92A}{2} = 160A$ and $M_{13}.I = 101A + \frac{92A}{2} = 147A$, while the maximum currents of these power lines equal $157.5A$ and $225A$, respectively. For power line L_5 this results in a violation. However, during the switching, the power flowing through bus B_2 decreases, and the next readings show values $M_5.I = 157A$ and $M_{13}.I = 142A$. This violation shows the importance of updating the measurements frequently, especially for the variables changing with the dynamic load.

Discussion The list of relevant rules for this case study does not cover all the rules from Chapter 4, as the knowledge scope of the monitoring tool is restricted only to local process information, and not all elements of the topology are present in this system.

Unfortunately, the reference test reveals that updates of meter measurements are delayed by up to half a minute. Beside that, in some cases, responses to a single general interrogation command contained inconsistent current measurements at a single point in time. This inconsistency was not expected, and can cause some false positive violations of physical constraints.

The reference test also provided insights into error margins in floating-point comparisons. The precision is influenced, for example, by the precision of the measurement devices, the data representation within the RTU and its representation in the monitoring tool. During the measurements, the maximal detected relative difference, was around 3.5%. Therefore, an error margin of 4% was chosen to cope with imprecision, and at the same time, not to lose the precision in the comparisons.

The response to the general interrogation command reports all values stored on the RTU. The total response took sometimes up to five seconds, with a maximum delay of one second between two consecutive measurement messages. Because of these observations, (i) the maximal allowed age of measurements was set to 7 seconds, so that all the values obtained in the same general interrogation response are still valid, and (ii) the parameter y[8], defining the delay between the last obtained measurement and an automatic evaluation, was set to 2 seconds.

B. Set Point Test

Description In this test, the topology information is first adjusted such that the voltage reference point for power line L_6 is set to $6kV$. Note that this value is not changed on the RTU configuration. We consider the topology information to be the ground truth. As a result, after the initial general interrogation command, the tool reports that the set point of voltage value on power line L_6 is outside of the allowed bounds, violating rule $R8a$. Moreover, the measured value of voltage on this power line equals $M_6.V = 9.139896kV$ during the capture. This value exceeds the (fake) reference voltage set point number significantly, violating $R2$. The following commands alter the set point value for the maximum current on line L_{10}.

Commands The commands listed in Table 7.4 are issued and the following observations are made:

[8]Parameter y has been defined in Section 6.1.3.

- **Change the set point of current value of line L_{10} to $154A$.** This command was also performed in the reference test, and is again classified as safe, as the new value is contained within the desired bounds.
- **Change the set point of current value of line L_{10} to $126A$.** This is classified as an unsafe command, as the new set point value is lower than the allowed minimal value, violating *R8b*.
- **Change the set point of current value of line L_{10} to $190A$.** This, again, is classified as an unsafe command, as the requested value is higher than the maximum allowed value, also violating *R8b*.
- **Change the set point of current value of line L_{10} to the original value of $157.5A$.** This command is classified as safe.

Violated rules The observed rule violations correctly reflect on the safety of the system.

The first two violations are a result of emulating a weaker system, than in reality. Under the assumption that the reference voltage for power line L_6 equals $6kV$, the measurements of voltage for this line exceed the allowed voltage range, and the set point is not configured properly. This was expected, as the tool's configuration was changed on purpose. The resulting violations of rules *R2* and *R8a* occur during each measurement update, as the topology information is not changed during the course of the test.

When changing the set point of the current, the allowed range for the set point for line L_6 is $[141.75; 173.25]A$. The second and the third command request a value outside of this range; therefore, *R8b* is violated. In each case, the response to the next general interrogation request contains these unsafe set points, which raises the same warning.

Discussion This test showed a correct response of the monitoring tool to both, command evaluation, and the system state evaluation. The tool warned when the set point configured on the RTU deviated too much from the value obtained from the topology file.

C. Switching Test

Description The switching test was designed to test three rules: a static interlock rule, a dynamic interlock rule, and the maximum allowed current threshold rule. For this purpose, two interlocks were defined in the topology (see Table 7.4). A static interlock K_1 for switches S_9 and S_{10} requires that at least one of these two switches is connected. A dynamic interlock K_2 for switches S_5, S_{10} and S_{13} requires that power lines, connected through these switches, provide a capacity for a total current of at least $320A$. Moreover, the maximum allowed current on power line L_{13} was changed in the reference topo-

logy to $120A$. Because of that last change, upon every measurement update, the tool reports that the set point value, which is configured to the original value of $225A$, exceeds the allowed range defined by the value altered in the topology information, violating rule $R8a$.

Commands The outcome of each of the commands listed in Table 7.4 is provided below:

- **Disconnect switch S_9.** This command is classified as safe. However, due to the delay in updating sensor measurements on the RTU, Kirchhoff's law is violated, and the non-zero current is reported on the disconnected power line L_9.
- **Disconnect switch S_{10}.** When evaluating this command, the monitoring tool warns about several issues: (i) the static interlock K_1 (see Table 7.4) will not have its minimal number of connected switches, (ii) the dynamic interlock K_2 will have less than its required minimal total current, and (iii) current value on power line L_{13} will exceed the lowered maximal allowed current. This is explained in detail below.
- **Connect switch S_{10}.** This command is evaluated as safe.
- **Connect switch S_9.** This command is also evaluated as safe.

Violated rules Rules $P1$ and $P3$, that represent Kirchhoff's law and the requirement that the current on a disconnected power line should be 0, are violated after every switching command for up to 30 seconds. The reason for these warnings was explained in detail in the reference test. Also, due to the altered value of the $L_{13}.I_{max} = 120A$, the originally configured set point $M_{13}.I_{sp} = 225A$ exceeds the allowed range. Therefore, the evaluated state will violate rule $R8b$ upon each general interrogation response.

Three rules are violated when disconnecting switch S_{10}. When evaluating this command, the calculated value of current for lines L_5 and L_{13} equals $170A$ and $155A$, respectively. Both values exceed the maximum allowed current of the power lines, which equal $157.5A$ and $120A$, respectively. Upon the next measurement, the observed values equal $M_5.I = 167A$ and $M_{13}.I = 150A$, triggering a violation of rule $R1$. For power line L_{13} this value is indeed above the maximum current value, which was altered in the topology. Still, it is below the real maximum current of $225A$. For power line L_5 a violation occurred, which was not expected, nor intended. The measurements showed that the current exceeded its real current threshold by $10A$ for around one minute. Disconnecting switch S_{10} also triggered violation of both static and dynamic interlock rules ($R9a$ and $R9b$). The static interlock K_1 requires that at least one switch from

the set $K_1.S = \{S_9, S_{10}\}$ is closed. As S_9 is already opened, the command to disconnect S_{10} triggers a violation. The dynamic interlock K_2 demands that the minimum value for the sum of the maximum allowed currents of lines connected via a switch from the set $K_2.S = \{S_5, S_{10}, S_{13}\}$ equals $320A$. The maximum current values for power lines L_5, L_{10} and L_{13} equal $157.5A$, $157.5A$, and $120A$, respectively, providing a total of $435A$. When disconnecting line L_{10}, the resulting total equals $277.5A < K_2.I_{min}$.

Discussion The switching test showed a correct prediction of the safety of the system state after executing each command. Especially, when disconnecting switch S_{10}, the monitoring tool displays multiple warnings about safety consequences of this action. The consistency rules $P1$ and $P3$ are violated again due to the delays in updating the measurements on the local RTU. Exceeding the maximum allowed current on line L_5 was not indented, and as later explained by the power grid operator, the configured maximum allowed value is chosen conservatively. Therefore, the *actual* maximum allowed currents of power lines are higher than the ones given in Figure 7.4, and the real system was in fact not in an unsafe state.

D. Protective Relay Test

Description For this test, two alterations of the topology were performed: the protective relay located at switch S_{13} is configured with a maximum current of $120A$, and a cutting timer of $10s$. This means that if the value of current exceeds $120A$ for more than $10s$, the protective relay is considered broken. To test this, power line L_{10} was disconnected in order to increase the current value on power line L_{13}. Only after around a minute, power line L_{10} was connected again.

Commands The following commands were issued:

- **Disconnect switch S_{10}.** This command is evaluated as unsafe, as the expected current on power line L_5 will exceed its maximum allowed current.
- **Connect switch S_{10}.** This command is considered safe.

Violated rules Switching causes a violation of Kirchhoff's law, and a violation of the restriction checking if the current on a disconnected power line equals $0A$, that is, $P1$ and $P3$, respectively. This is again caused by the delay of measurement updates on the RTU.

Moreover, when disconnecting power line L_{10}, the tool calculates that the resulting current on line L_5 will equal $165A$, which exceeds the maximum allowed

current on that line $L_5.I_{max} = 157.5A$. The next update shows the current value of $162A$, which indeed is above the maximum allowed value, violating rule $R1$.

Additionally, disconnecting power line L_{10} results in a higher current on power line L_{13}, exceeding the fictitious threshold of protective relay R_{13}. The resulting violation of rule $R4$ reports that the maximum current of protective relay is surpassed. Within the cutting time t_{cut}, it is only a warning of a recent violation. After the $10s$, the warning changes to information about a broken protective relay.

Discussion The command evaluation proved to be very accurate, as it predicted the violation of safety rule $R1$.

7.4 Conclusions

This chapter showed the feasibility of SAM a power distribution substation with the monitoring tool proposed in Chapter 6. Two main contributions have been presented.

First of all, a parser for IEC-104 protocol was presented. This parser builds on the Spicy framework and is able to extract traffic- and control-relevant information by deep packet inspection. Based on a real IEC-104 trace, taken at a Dutch power distribution station, several IEC-104 traces with variable length were created. The realized throughput proved to be sufficient to employ the parser in real-time with Zeek in real SCADA systems. The evaluation also shows that the performance of Zeek highly depends on the offered load.

Secondly, the SAM tool introduced in Chapter 6 was used in a real SCADA system. The performed experiments provided many practical insights about using the proposed monitoring algorithm in real-life, especially w.r.t. availability, freshness and precision of the data, which are discussed next.

- **Availability** In a local implementation, the amount of data is limited to information measured at only a single substation, therefore, it was not possible to evaluate all rules. Moreover, it is necessary to request *all* values stored on the RTU, for example, in periodic updates. During the tests we have achieved this by sending general interrogation commands before and after every command.

- **Freshness** When calculating the future state of the system, the SAM tool applies the parsed control command to the currently observed state. It is therefore important that the observed state is up to date. Moreover, after a command is executed in the system, many of the currently measured values may not hold true anymore. For example, after changing a switch

state, the current on the power line is different than the last measured. The outdated measurements should not be used to evaluate the system consistency and or safety, otherwise, the tool might generate false positives, until the next update of all of the values is performed. Hence, all measurements affected by a change are invalidated after a command has been executed and no rules are evaluated with invalid information. Unfortunately, even when doing so, we noticed that due to delays in updating measurements on the local RTU, the tool *was* generating false positives. This could be prevented, for example, by changing the frequency of these updates, or by performing the consistency checks only after a period of 30 seconds.
Moreover, measurements of dynamically changing variables should only be used if they are recent. The prototype uses a so-called *freshness period* to define the maximum age allowed for a measurement. If a measurement is older than its allowed freshness period, it is not considered reliable anymore and is not used for evaluation.

- **Precision** In a real implementation, the measurements are bounded by the precision of the measurement devices, precision loss coming from the data types used in transit, and the data representation within the monitoring tool. This imprecision influenced the evaluation of the rules, making it necessary to evaluate the rules using not exact values, but intervals around that calculated or measured value. Due to the imprecision of the measurements, an error margin of 4% was chosen when comparing sums of currents on the incoming and outgoing power lines.

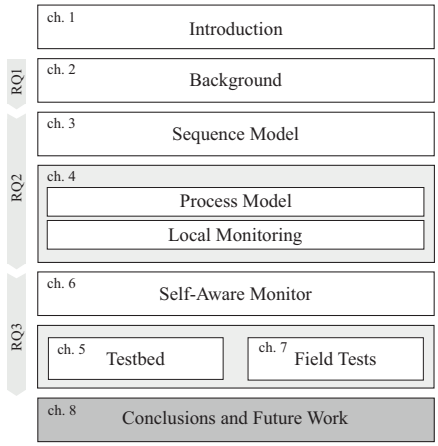
Overall, the results show many differences between the emulated SCADA traffic from Chapter 6 and the real SCADA traffic. The constraints discussed above, however, could be solved by a different configuration of the equipment at the field station.

The SAM tool proved to be robust to unexpected information objects that were not present in the RTU configuration. The unknown objects generated warnings, however, the tool continued the state and command evaluation without problems. Finally, the case study revealed that some of the set points on the local RTU were not chosen properly for the system at hand. The set point configured for power line L_{13} could have caused false positive warnings, as it was configured to a lower value.

CHAPTER 8

Conclusions

The final chapter of this thesis presents the conclusions to the research questions presented in Chapter 1. First, we reflect on the main contributions of the research, before we revisit each of the research questions. Finally, we discuss future research directions that stem from the work presented in this thesis.



8.1 Summary

Since 2010, the name Stuxnet has been giving nightmares to security specialists working with SCADA systems. This was the first known malware that used knowledge about the operation of the controlled process, in order to disrupt it. Stuxnet also served as main motivation when starting our work on process-aware monitoring. For a while this was an isolated, known incident which disrupted a controlled process by means of cyber commands. However, while working on this thesis, in the course of the last 4 years, we have witnessed many new examples of dedicated cyber attacks which could harm the physically controlled process. These new incidents, such as disconnecting 225 000 customers in Ukraine in 2015, often affected the electric power sector.

This thesis, investigates the security of SCADA systems controlling electrical power distribution systems. We have observed that many of the mentioned incidents were successful due to a lack of security mechanisms at remote substations. We have proposed an approach that uses information obtained only within the locally controlled substation in order to asses whether the system is in a safe state and whether the incoming commands sustain this safe state.

Our main contribution as presented in this thesis is the development of a modelling formalism and a local monitoring algorithm that can be used for supervising the traffic exchanged between the SCADA control room and a field station. Moreover, this approach has been thoroughly validated both in a co-simulation testbed and in a real-life case study. Validations have shown that the presented approach allows for detecting attacks aiming at disrupting the physical process in field stations. To the best of our knowledge, no currently implemented security mechanisms are capable of performing such detection.

8.2 Revisiting the Research Questions

In this section, we provide detailed conclusions to each of the research questions, as presented in Chapter 1.

> ***RQ1*** – *Where in the power distribution system is an extra layer of security needed? How can it be designed and implemented?*

We conclude that the **field stations** are the most vulnerable components and require an extra layer of security in the power distribution system. We also consider the **process-aware approach** highly suitable to do this, as information about the local process is the only reference that a controller located in the field

station has direct access to. However, we have not found a suitable process-aware detection mechanism that would allow to notify the operator about an unsafe physical system state, given only the local information. Below we summarize the motivation to these findings.

As shown in Chapter 2, the ongoing **energy transition** potentially has a very high impact on power distribution systems. On the one hand, the traditional one-way flow of electrical power is affected by Distributed Energy Resources such as solar panels, and by using electrical vehicles. This change requires better insight in the current state of the infrastructure. On the other hand, smart grid goals, such as being self-healing, require remote control of currently non-automated field stations.

Recall, that **SCADA architecture** usually consists of a control room, which contains the supervisory software on a SCADA server, and several field stations, equipped with RTUs or PLCs. In power distribution the centrally located SCADA server runs an Energy Management System. Based on information from the field stations, the EMS is able to monitor, and if necessary, control the system. Moreover, an EMS is able to detect whether received sensor measurements are consistent w.r.t. physical laws and the system topology. Instead, RTUs in field stations usually do not have any additional security mechanisms in place, and simply apply the commands sent from the control room. Once the control room is corrupted, these field stations cannot protect themselves.

The recent **incidents** that happened in the electric power sector confirm the issue highlighted above. Many incidents aimed at connecting to and gathering information about the control network, to later send commands from an authorised account to the field stations. The hackers in Ukraine managed to disconnect several field stations in this way.

Finally, when analysing state-of-the-art **intrusion detection methods**, we have noticed a trend of using process-aware methods for SCADA systems. Only these methods are able to detect when the process is in an unsafe state. Unfortunately, the available techniques often require the knowledge of the entire system state, in order to perform similar calculations as done by the EMS. Approaches that are able to work in a local fashion often alert only about some process variables that were out of static bounds (thresholds). Therefore, a new method is required, that uses only local process information and can depend on the current system state.

RQ2 *– Which aspects of the physical system state should be modelled in a local process-aware monitoring system?*

The modelled features depend on the types of attacks that we intend to detect. We have presented two approaches for incorporating the information about the process in the detection method. Both could be applied locally in the field stations.

The first method, presented in Chapter 3, investigated a model of the **traffic sequence**. It is often important to follow certain sequences when performing switching in power distribution systems. Not following these sequences can lead to catastrophic consequences, such as blackouts. However, generated models of traffic sequences can be very large, which can challenge the detection in real-time. Moreover, modelling all packets can be misleading, as the sequence conditions apply only to a subset of the types of packets. The majority of the packets within SCADA traffic are related to reading the sensor measurements, where order of packets is not highly important. It is therefore important to tailor such method to the packets whose order is crucial and model only sequences of these packets.

Sequence attacks, however, are only a small subset of the possible attacks on the physical process. Therefore, we have decided to investigate a more general detection approach. The second method uses a **process model**, which we presented in Chapter 4. We proposed a modelling formalism that can describe the locally controlled system, and a **local monitoring algorithm**, that maintains the current state of the physical process. This state is assessed w.r.t. its consistency and safety based on a set of rules, which we have derived from physical laws and safety standards relevant to power distribution systems. Moreover, by following the evolution of the system state, our algorithm allows to predict the outcome of the commands that are sent to the RTU. We provided several examples showing how this monitoring approach can be used to detect faulty sensor readings, as well as commands. The presented examples involved commands that would lead the system to an unsafe state. This was not visible to the centrally-based monitoring system, however, it was recognised by the proposed approach.

RQ3 *– Is a local process-aware monitoring solution feasible to protect power distribution systems?*

The proposed approach was implemented and validated both in a co-simulation testbed and in a real-life case study.

Chapter 5 presented a **co-simulation testbed** which we used to investigate the effect of the proposed process-based monitoring on the safety and security

of field stations. Using this testbed allowed us to investigate the physical consequences of executing malicious commands. An advantage of using simulation was that the proposed algorithm could be evaluated for various scenarios.

As writing specific policy rules for substation elements was cumbersome, Chapter 6 introduced a **Self-Aware Monitor** - a tool that, given the topology of the station, automatically derives the physical constraints and safety requirements relevant to the tested system. The SAM tool uses the modelling formalism introduced in Chapter 4 to describe the physical power distribution system. Upon sensor measurements, the model of the physical system stored in the tool is updated. Upon receiving a command, the tool precomputes the result of executing this command on the observed system state to assess whether the outcome is safe in the context of the controlled system.

We noticed, however, that monitoring the system in a local fashion can be challenged if, for example, the topology of the monitored system contains loops. A change done at one bus can be considered safe for that bus, however, it can affect remote power lines, that are not connected directly to this bus. This can be especially challenging for old power distribution systems, because these often contain loops. Nowadays, a radial structure is advised for low voltage areas. Therefore, when implementing the SAM tool in practice, it is important to carefully analyse the topology of the system.

Finally, Chapter 7 implemented the Self-Aware Monitor tool in a **real SCADA system**. The results show many differences between the SCADA traffic emulated in Chapter 6 and the real SCADA traffic. The latter brought insights w.r.t. precision, delays, freshness, and availability of the data, which are discussed next. The experiments showed that the **imprecision** error of the measurements significantly affected the tested physical constraints. We chose an error margin of 4% when comparing sums of currents on the incoming and outgoing power lines in order to avoid false alarms resulting from the imprecision error. Furthermore, due to **delays** in updating measurements on the local RTU, it might be necessary to either change the frequency of these updates, or only perform the consistency checks after a period of 30 seconds. This delay also affects data **freshness**, as the SAM tool cannot use old data when evaluating the rules. Using old data, the tool might generate false positives, until the next update of all of the values is performed. Especially measurements of dynamically changing variables should only be used if they are recent. Finally, the SAM tool assumes knowledge of all values within a substation that the RTU is controlling. To ensure data **availability**, it is necessary to request *all* values stored on the RTU, for example, in periodic updates. In the real-life case study we requested these updates manually, because not all values were updated periodically to the control room.

With the conclusions above we have achieved our objective that we have defined as:

> *Design a process-aware monitoring system for power distribution, that detects when the physical process is in an unsafe state.*

Of course, further work is needed before implementing the proposed SAM tool. Possible directions for the further development of this tool are listed in the next section.

8.3 Future Work

Future research directions can be divided into three parts: improving the algorithm, changing the locality of the tool, and the practical implementation.

First of all, the proposed algorithm could be further improved by including anomaly detection next to the proposed specification-based detection. As indicated in Chapter 2, specification-based methods focus only on not bringing the system into an unsafe state. Hence, they alert when the model assumptions are violated. Such approaches do not detect changes in the normal behaviour of the power distribution system, for example, will not detect that the power consumption does not exhibit a diurnal pattern. Therefore, specification-based approaches will not detect unknown attacks, that fall within the model constraints. This limitation could be improved by, for example, using anomaly-based detection, possibly implementing machine learning methods.

Second of all, we have discussed that the local approach can be challenged for some topologies of the system. This could be circumvented by identifying the most crucial substations which would need to exchange the information to validate the system state correctly. Another approach would be to divide the distribution grid into sub-topologies, which could each employ the local monitoring approach.

Finally, in order to implement the local monitoring approach, or any other local or global approach, a better ICT infrastructure is required, which can provide synchronized and real-time measurements to the monitoring tool. We have shown that the freshness of data, the unavailability of some of the values and the imprecision of measurements potentially affect the performance of the local monitoring algorithm. The currently observed precision, could lead to semantic attacks being unnoticed. With a better measurement infrastructure, for example, implementing Phasor Measurement Units, the power operators could facilitate many new monitoring approaches.

APPENDIX A

Open Data Management

In this appendix, we provide links to access the source code used in this thesis. Table A.1 mentions the chapter in which the code was referred to, and provides a brief description of the code together with corresponding URL.

Table A.1: Description and URLs of source code used in each chapter.

chapter	description	URL
Chapter 3	The code used to modify traces: it randomly copies a packet, removes a packet, or swaps two packets from a trace.	`https://github.com/jjchromik/manipulateTraces.git`
Chapter 5	Co-simulation testbed based on Mosaik.	`https://github.com/jjchromik/mosaik-cosim`
Chapter 6	The Self-Aware Monitor, and `*.pcap` files used to evaluate the tool.	`https://github.com/jjchromik/RuleGeneratorSCADA/`
Chapter 7	IEC-60870-5-104 parser for using with Zeek network monitor.	`https://github.com/jjchromik/hilti-104`

SCADA protocols

In this appendix we present SCADA protocols investigated in this thesis.

B.1 Modbus/TCP

Modbus is originally a serial communication protocol developed in 1979 by Modicon (now Schneider Electric) for use with Programmable Logic Controllers [105]. Later it became an open standard communication protocol and since 2004 it is managed by Modbus Organization.

Modbus/TCP is one of the widely-used protocols to connect the remote RTUs with a central supervisory computer [80]. Although Modbus is a generally accepted industrial process standard, especially popular in the oil and gas sector, it also plays an important role in power distribution [16, 77]. It is a master/slave type of protocol, where only one of the communicating devices, called master (or "client"), can initiate the communication. The slave (or "server") continuously listens for incoming connections on TCP port 502[1] (see Figure B.2). Modbus stores either 1 bit values (so-called coils) or 2 byte values (so-called registers). Both coils and registers can be either read-only values (discrete inputs and input registers, respectively) or read/write values (coils or holding registers, respectively). In order to allow for, for example, floating point variables,

Table B.1: Modbus data types

name (shortcut)	access	length
coil (CO)	read/write	1 bit
discrete input (DI)	read-only	1 bit
holding register (HR)	read/write	2 bytes
input register (IR)	read-only	2 bytes

[1] Port 802 for Modbus Security version.

some vendors allow for combining registers to hold 32-bit and 64-bit values [57]. Table B.1 summarises the data types supported by Modbus.

Modbus protocol data unit consists of the elements presented in Figure B.1 and explained next.

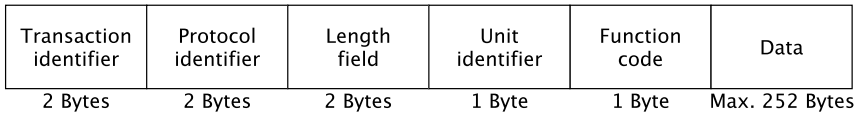


Figure B.1: Modbus/TCP protocol data unit.

- **transaction identifier** - is used for pairing the transaction request with response.
- **protocol identifier** - is used for intra-system multiplexing. For Modbus protocol it is always set to 0.
- **length field** - is the byte count of all the following fields.
- **unit identifier** - is used for intra-system routing purpose, for example, when communicating to Modbus serial devices over a Modbus/TCP gateway. If not used, this field is set to 0.
- **function code** - is used to identify Modbus operations, such as read or write, on coils and registers. A list of the most commonly used function codes is provided in Table B.2.
- **data** - contains the content of the message. Exact content depends on whether the packet is a request, response, or an error, and on the function code.

As explained above, the content of the data field can vary. Figure B.1 shows an example communication for reading values stored on a Modbus/TCP server. In Figure B.1, Modbus server is shown on the left side, and Modbus client on the right side. The exchanged information, limited to the content of function code and data fields, is shown on the horizontal lines. In a request, Modbus Client specifies (i) a function code, (ii) the starting address, and (iii) number of the requested entities. For example, function code can be set to 01 for reading coils, and 04 for reading holding registers. Quantity of the requested entities (that is, coils or registers) can be equal from 1 to 2000 for coils and from 1 to 125 for registers [105]. The starting address is the address of the first entity. The server can send two types of messages back: a response or an error. A response contains (i) the same function code, (ii) the byte count and (iii) values of the entities. The byte count is the number of the bytes that follow. For coils and

Table B.2: Selected function codes of Modbus

number	action	explanation
01	read	read coils
02	read	read discrete inputs
03	read	read holding registers
04	read	read input registers
05	write	write a single coil
16	write	write multiple holding registers

discrete inputs, it is equal $n = \lceil x/8 \rceil$, where x is the number of the requested coils, while for holding and input registers it is equal to $n = 2 \cdot y$, where y is the number of the requested registers. If some error occurs, the Modbus server responds with the function code followed by an exception code.

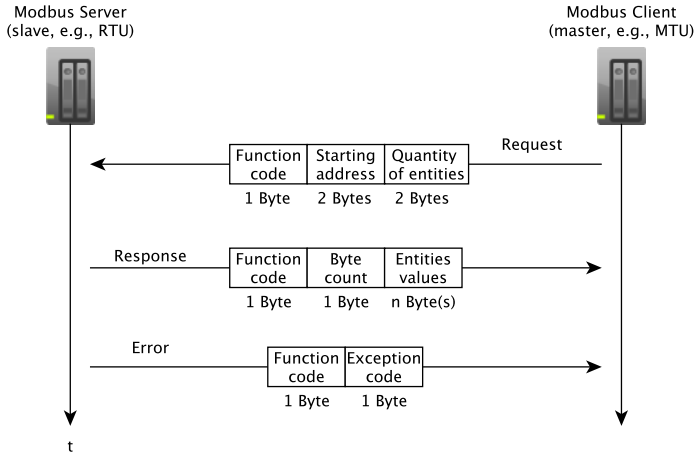


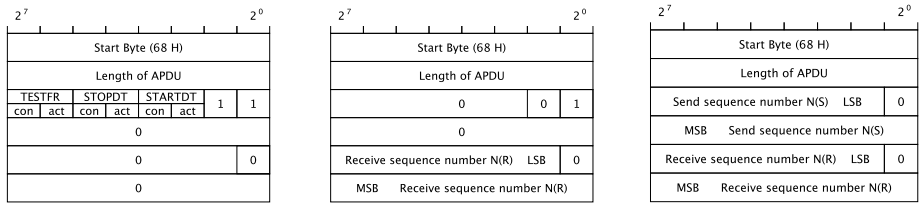
Figure B.2: Modbus/TCP data exchange.

Security extensions for Modbus/TCP protocol have been proposed, for example, [41, 46, 134], which, however, do require changes on the protocol level of operating devices. This is expected to be difficult as companies are reluctant to such changes and global standardization. Without a uniform standard, the proposed approaches may be incompatible with existing systems. For long time, no dedicated Modbus security standards existed, however, one could argue that

IEC62351 [67] also encompasses Modbus as it is nowadays usually runs over TCP/IP. Recently, a Modbus Security standard was proposed [106], which introduces Transport Layer Security (TLS) to the traditional Modbus protocol. TLS encapsulates Modbus packets to provide authentication and message integrity. Even with the extra security standard, attacks focusing on disrupting the process are still possible.

B.2 IEC-61850-5-104

IEC-60870-5-104 (IEC-104) is a part of IEC standard series 60870 used for telecontrol in power systems and electrical engineering applications. It is an extension of IEC-60870-5-101 protocol that allows communication over a TCP/IP connection. At the application layer, IEC-104 uses the basic telecontrol tasks defined already in IEC-101, with few differences. For example, IEC-104 provides timestamps for commands, while IEC-101 does not; Type IDs using time stamp of format CP24Time2a[2] are not used in IEC-104 anymore; some fields of variable length in IEC-101, such as ASDU address and Information Object Address, have a fixed length in IEC-104. The IEC-104 protocol is used for communication between field stations and control room [1].



(a) U-Format (b) S-Format (c) I-Format

Figure B.3: The APCI formats.

An IEC-104 packet, shown in Figure B.4, consists of the Application Protocol Control Information (APCI) and the Application Service Data Unit (ASDU) which together are called the Application Protocol Data Unit (APDU). APCI determines whether the packet has U-, S- or I-format. Figure B.3 shows the three formats, which are distinguished based on the so-called `control octet 1`, which is the first and second bits of the third byte of the APCI.

[2]Time formats CP24Time2a and CP56Time2a are defined in IEC-101.

- **U-format** initiates and terminates sessions between two devices, or checks the responsiveness of a device by setting flags *STARTDT*. *STOPDT*, and *TESTFR*, respectively. U frames are identified by setting the `control octet 1` to "11".
- **S-format** acknowledges the data received in I-frames. S frames are identified by setting the `control octet 1` to "10".
- **I-format**, which contains relevant information in the ASDU fields. The first bit of the `control octet 1` is set to "0" for this type of frame. I frames are always followed by an ASDU containing information.

An ASDU contains the actual data transferred by that data frame. Figure B.4 shows a datagram of an I frame. The ASDU part contains the following fields:

- **Type identification (Type ID)** is the type of the sent object. It defines the *direction* of the communication, the *type* of the sent information object, indication if it contains a *timestamp*, and the *format* of the sent value. A single ASDU contains up to 127 objects of the same Type ID. It is possible to define custom Type IDs.
- **Variable structure qualifier** consists of S/Q bit that specifies the method of addressing the information objects, and the number of objects contained in the ASDU.
- **Cause of transmission** specify the reason of the transmission, for example, 1 stands for periodic/cyclic message, 2 is background interrogation, and 13 is data transmission. Additional two bits, P/N and T, indicate positive or negative confirmation of a function, or label ASDUs created during testing, respectively. Each bit, if not relevant, is set to 0.
- **Originator address** indicates whether the message comes directly from the source (set to 0) or whether it was relayed (set to the address of the original sender).
- **ASDU address fields** contain the address of the sender. This address is associated with all of the objects contained in that ASDU. If set to 0, then the address is not used; 65 535 is the broadcast address.
- **Information object address (IOA)** refers to the address of a specific information object, can be equal up to 65 535.

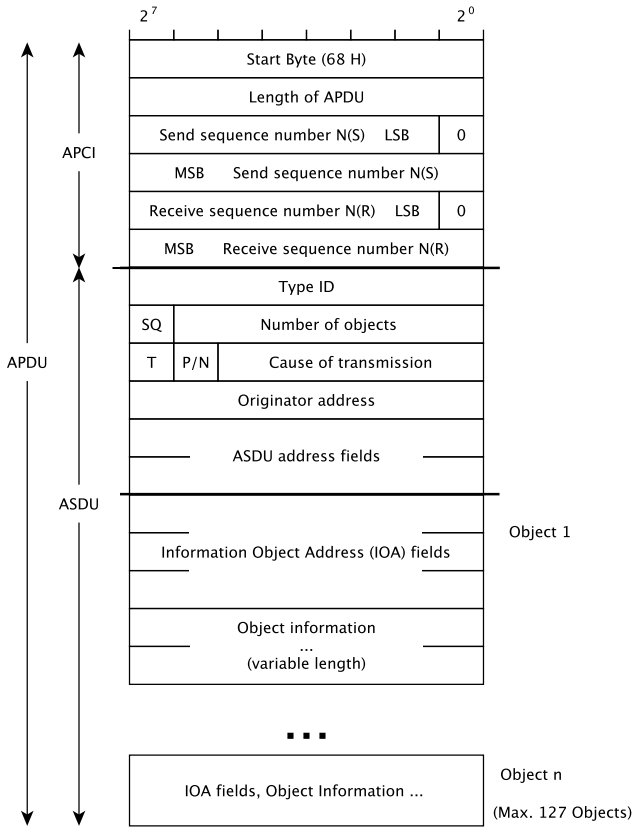


Figure B.4: I frame format: APCI and ASDU.

- **Information elements** is the actual data. This object can contain, for example, a measurement of voltage or current, a state of an element (on/off), or a threshold setting. The data can be sent in different formats, such as floating point value, normalized value, or scaled value. A packet may have up to 127 information objects.

As mentioned above, the Type ID included in the ASDU contains such information as *direction* of the communication. In general, the name of a Type ID provides information about the contained object, as shown in Figure B.5. The first letter indicates the mentioned direction. For example, letter "M" indicates the monitoring direction, while "C" stands for the controlling direction. Next two letters describe the information object sent. Different abbreviations exist for either controlling or monitoring direction. The following letter indicates whether the information object is sent including a time tag or not. The next letter defines the format of the sent object, for example, "A" means that the value is sent in normalized format, while "B" stands for scaled value. The last symbol is currently always set to 1. For example, Type ID "M_SP_NA_1" is

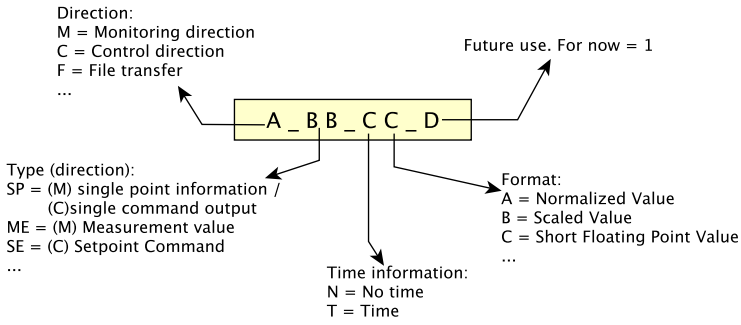


Figure B.5: Naming convention of Type IDs.

sent in the monitoring direction (that is, from the RTU to the control room), and it contains a single point information without a time tag. It is not necessary to normalize a single point information, as it contains a single bit of information. Type ID "C_SE_NB_1" is sent in the controlling direction (that is, from the control room to the substation), and it transports a set point command in a scaled value format, without a time tag.

IEC-104 itself has no built-in security mechanisms. IEC 62351 [67] defines the security extensions for the IEC 60870-5 standard; these extensions, such as using the Transport Layer Security (TLS) encapsulation, are applied in the IEC TS 60870-5-7 document. Even though security extensions are proposed, they are not widely adopted [100].

APPENDIX C

Mosaik Simulators

The function of *Mosaik*, co-simulation framework used in the testbed described in Chapter 5, is to exchange information between various simulators. Every simulator, in order to interact with *Mosaik*, consists of two parts: (i) the actual *simulator* containing the model of the simulated entities, and (ii) *Mosaik handler*, that uses *Mosaik* API calls to initiate the simulator, create instances of the simulated entities, and request simulation steps.

In this appendix, we describe the actions performed in `step()` call for the two simulators: power distribution simulator and Modbus/TCP simulator, for better understanding of the Sections BLABLA in Chapter 5.

C.1 Connecting Mosaik and a Simulator

Communication between the *Mosaik* core component (later referred to as *Mosaik*) and a simulator is done in 4 stages, illustrated in Figure C.1. In stage 1, *Mosaik* calls `init()` function of the simulator handler. In this stage, *Mosaik* can pass some global parameters to the simulator, and the simulator returns meta data describing itself. In stage 2, *Mosaik* creates instances of the models implemented by the simulator using `create()` function. This function returns information about created entities. Once all the entities are created, and all the relations between them are defined, a call to `setup_done()` is done. Stage 3 indicates that the simulation is started and *Mosaik* will repeatedly call `step()` function of the simulator handler. At every call of the `step()` function, the simulator steps forward in time (with predefined value), and will return the time at which it wants to perform its next step. Stage 4 is the end of simulation and *Mosaik* sends a `stop()` call to all the simulators to request a shut-down.

Within a single `step()` call of a simulator, the simulator can perform some asynchronous requests back to Mosaik. These are: `get_progress()` for getting current simulation progress information, `get_related_entities()` for collecting information about other relevant simulated components, `get_data()` for querying other entities and `set_data()` for sending data for other entities. Most importantly, within a `step()` function, all the simulation done by the simulator

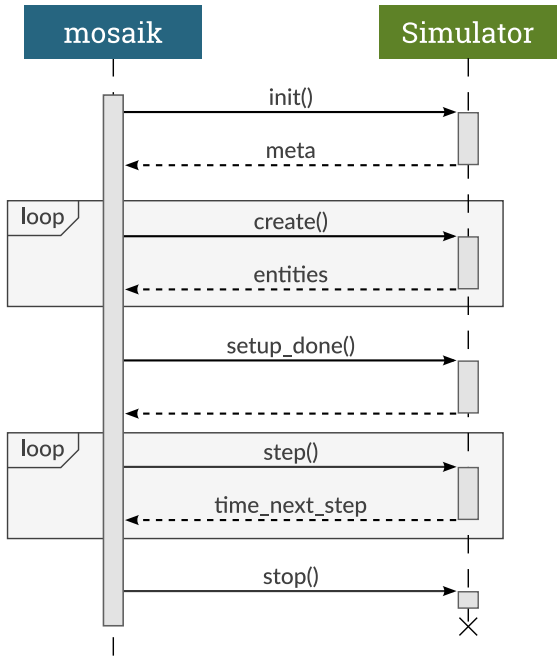


Figure C.1: Communication between *Mosaik* and a simulator [114]

have to be executed. For example, the power distribution system simulator has to (i) check if any topology changes were done and if so, refresh the topology, (ii) read the new power values and the topology and perform power flow equations.

C.2 Power Distribution Simulator

The power distribution simulator consists of two parts: (i) the power flow equations solver, and (ii) the topology manager. In order to calculate the new sensor readings (that is, voltage and current measurements), the power flow equation solver needs the latest house and PV panels consumption values, and the newest topology information. House and PV panel profiles are stored in a CSV file and provide a new value every 900 simulation time units. Topology changes, if they occur, are provided as input from the Modbus/TCP simulator.

A single step of the power distribution simulator is illustrated in Figure C.2. The step of power distribution simulator begins with checking whether inputs contain a command from the RTU simulator. If so, this command is applied

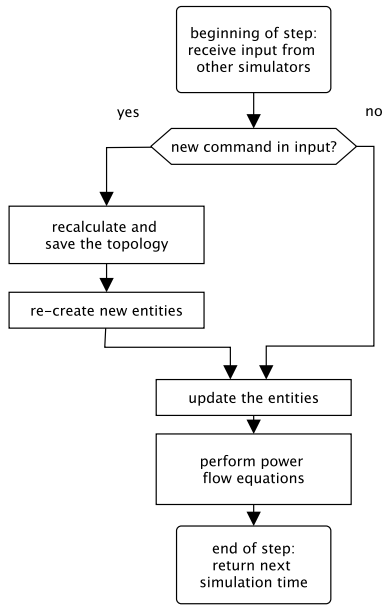


Figure C.2: Illustration of a single step of simulation of the power distribution system simulator.

in the model of the topology, by changing the requested value of, for example, switch, and verifying if all the buses are still connected to the power grid. The latter is done in the topology simulator by representing the topology as a graph, with buses seen as nodes, and power lines as edges and testing whether all buses are connected to the reference bus. If a bus (of more) is no longer connected to the reference bus, it is marked as 'isolated'. Such buses are de-energized and do not supply the connected houses with power. The new topology is saved as the new reference topology, and the entities generated in `create()` call of the Mosaik handler are built again.

Next, the entities are updated with the new values of the production/consumption. This action is performed despite of receiving a new command from the RTU simulator. Finally, the PyPower package is used to recalculate the power flow equations for the updated values of the power consumption/production and new topology.

Finally, the simulator returns the next time that it would like to perform a simulation step.

C.3 Modbus/TCP Simulator

Modbus/TCP server simulator is a middle point between the simulated power distribution system and the RTU server that is creates. It has to update the server instance with the values it receives from the power distribution simulator, and it has to update the power distribution simulator about commands it detects on the RTU server. This simulator consists from a *Mosaik* handler for the API calls, and the actual RTU simulator.

The `create()` call of the *Mosaik* API handler performs the following actions. First, it reads the configuration of the RTU it is supposed to create. Next, it creates a memory of that device, called a data block, and a local cache of the values. Moreover, it starts an RTU server running Modbus/TCP protocol, and assigns it the created memory with all the coils and registers. The created server runs on the local machine using the IP address and port number specified in the configuration of the RTU. Finally, it creates entities of sensors and switches that will be receiving the data from the power distribution simulator.

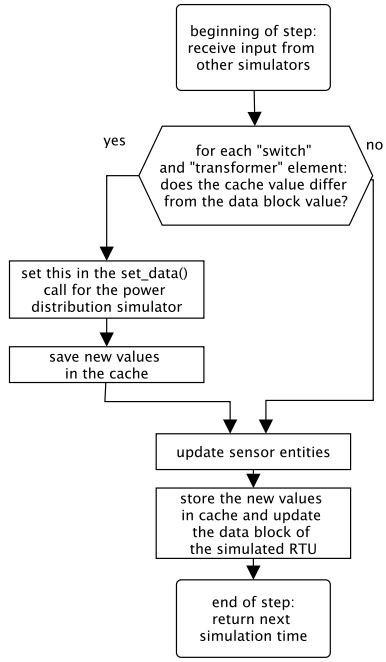


Figure C.3: Illustration of a single step of simulation of the RTU simulator.

A single step of the RTU simulator is illustrated in Figure C.3.

The `step()` call of the *Mosaik* API handler performs the following actions. First, for the state of switches and the transformer tap switch position, it checks whether the current content of the RTU's data block differs from the cached values. If so, this means that there was a recent change requested on the RTU server, and this change has to be applied in the power distribution simulator. The changes are sent to the power distribution simulator using `set_data()` call. If no change of switches or tap position is seen, it continues. Next, all the sensor entities are updated with the new values received from power distribution simulator. These values are stored in cache and updated on the RTU datablock.

Bibliography

Note: all URLs were accessed on the 18^{th} of April 2019.

[1] C. Alcaraz, J. Lopez, J. Zhou, and R. Roman. Secure SCADA framework for the protection of energy control systems. *Concurrency and Computation: Practice and Experience*, 23(12):1431–1442, 2011.

[2] A. Anwar and A. N. Mahmood. Cyber Security of Smart Grid Infrastructure. In *The State of the Art in Intrusion Prevention and Detection*, pages 449–472. CRC Press, Taylor & Francis Group, USA, 2014.

[3] W. Aoudi, M. Iturbe, and M. Almgren. Truth Will Out: Departure-Based Process-Level Detection of Stealthy Attacks on Control Systems. In *Proceedings of the ACM Conference on Computer and Communications Security, Toronto, Canada*, pages 817–831. ACM, 2018.

[4] M. Assante. Confirmation of a Coordinated Attack on the Ukrainian Power Grid. `https://ics.sans.org/blog/2016/01/09/confirmation-of-a-coordinated-attack-on-the-ukrainian-power-grid`, 9 Jan 2016.

[5] Associated Press. Flights cancelled at Schiphol airport as power outage hits Amsterdam. `https://www.theguardian.com/world/2015/mar/27/flights-cancelled-schiphol-airport-power-outage-amsterdam`, 27 March 2015.

[6] A. Awad, P. Bazan, and R. German. SGsim: Co-simulation framework for ICT-enabled power distribution grids. In *Proceedings of the International GI/ITG Conference on Measurement, Modelling, and Evaluation of Dependable Computer and Communication Systems, Münster, Germany*, pages 5–8. Springer, 2016.

[7] H. Bao, R. Lu, B. Li, and R. Deng. BLITHE: Behavior rule-based insider threat detection for smart grid. *IEEE Internet of Things Journal*, 3(2):190–205, 2016.

[8] R. R. R. Barbosa and A. Pras. Intrusion detection in SCADA networks. In *Proceedings of the Mechanisms for Autonomous Management of Networks and Services*, pages 163–166, Zurich, Switzerland, 2010. Springer.

[9] R. R. R. Barbosa, R. Sadre, and A. Pras. A first look into SCADA network traffic. In *Network Operations and Management Symposium, 2012 IEEE*, pages 518–521. IEEE, 2012.

[10] R. R. R. Barbosa, R. Sadre, and A. Pras. Towards periodicity based anomaly detection in SCADA networks. In *17th Conference on Emerging Technologies & Factory Automation*, pages 1–4. IEEE, 2012.

[11] R. R. R. Barbosa, R. Sadre, and A. Pras. Flow whitelisting in SCADA networks. *International Journal of Critical Infrastructure Protection*, 6(3):150–158, 2013.

[12] M. Bell, F. Berkel, and S. Liu. Real-Time Distributed Control of Low Voltage Grids with Dynamic Optimal Power Dispatch of Renewable Energy Sources. *IEEE Transactions on Sustainable Energy*, 10:417–425, 2019.

[13] J. Bigham, D. Gamez, and N. Lu. Safeguarding SCADA systems with anomaly detection. In *Computer Network Security*, pages 171–182, Russia, 2003. Springer.

[14] M. H. Bollen and F. Hassan. *Integration of distributed generation in the power system*, volume 80. John Wiley & Sons, USA, 2011. ISBN 9780470643372.

[15] R. E. Brown. *Electric power distribution reliability*. CRC press, USA, 2 edition, 2008. ISBN 9780849375675.

[16] S. Bush. *Smart Grid: Communication-Enabled Intelligence for the Electric Power Grid*. John Wiley & Sons, UK, 2014.

[17] A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. N. Fovino, and A. Trombetta. A multidimensional critical state analysis for detecting intrusions in SCADA systems. *IEEE Transactions on Industrial Informatics*, 7(2):179–186, 2011.

[18] A. A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry. Challenges for securing cyber physical systems. In *Proceedings of the Workshop on future directions in cyber-physical systems security*, pages 1–5, Newark, New Jersey, USA, 2009.

[19] A. A. Cardenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM symposium on information, computer and communications security*, pages 355–366, Hong Kong, China, 2011. ACM.

[20] M. Caselli, E. Zambon, and F. Kargl. Sequence-aware intrusion detection in industrial control systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, pages 13–24, Singapore, Republic of Singapore, 2015. ACM.

[21] M. Caselli, E. Zambon, J. Petit, and F. Kargl. Modeling Message Sequences For Intrusion Detection in Industrial Control Systems. In *Critical Infrastructure Protection IX*, volume 466, pages 49 – 71, London, UK, 2015. Springer.

[22] CBS. Centraal Bureau voor de Statistiek: Hernieuwbare elektriciteit; productie en vermogen. `http://statline.cbs.nl/Statweb/publication/?DM=SLNL&PA=82610ned&D1=a&D2=5&D3=a&HDR=T&STB=G1,G2&VW=T`, 30 Oct 2018.

[23] CENELEC. Harmonisation Document: Nominal voltage for low voltage public electricity supply systems, HD 472 S1, 1988. European Committee for Electrotechnical Standardization.

[24] CENELEC. EN 50160:2010, Voltage characteristics of electricity supplied by public distribution networks, 2010. European Committee for Electrotechnical Standardization.

[25] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes. Using Model-based Intrusion Detection for SCADA Networks. In *Proceedings of the SCADA Security Scientific Symposium*, pages 1–12, Miami Beach, FL, USA, 2007.

[26] S. Choi, Y. Chang, J.-H. Yun, and W. Kim. Traffic-locality-based creation of flow whitelists for SCADA networks. In *International Conference on Critical Infrastructure Protection*, pages 87–102, Arlington, VA, USA, 2015. Springer.

[27] J. J. Chromik, A. Remke, and B. R. Haverkort. Improving SCADA security of a local process with a power grid model. In *Proceedings of the 4th International Symposium for ICS&SCADA Cyber Security Research*, pages 114–123, Belfast, UK, 2016. BCS Learning & Development Ltd.

[28] J. J. Chromik, A. Remke, and B. R. Haverkort. What's under the hood? Improving SCADA security with process awareness. In *Proceedings of the Joint Workshop on Cyber-Physical Security and Resilience in Smart Grids*, pages 1–6, Vienna, Austria, 2016. IEEE.

[29] J. J. Chromik, C. Pilch, P. Brackmann, C. Duhme, F. Everinghoff, A. Giberlein, T. Teodorowicz, J. Wieland, B. Haverkort, and A. Remke. Context-aware local Intrusion Detection in SCADA systems: a testbed and two showcases. In *Proceedings of the International Conference on Smart Grid Communications (SmartGridComm)*, Dresden, Germany, 2017. IEEE.

[30] J. J. Chromik, A. Remke, and B. R. Haverkort. A Testbed for locally Monitoring SCADA Networks in Smart Grids. In *International workshop Energy-Open*, Enschede, the Netherlands, 2017. IEEE.

[31] J. J. Chromik, A. Remke, and B. R. Haverkort. Bro in SCADA: dynamic intrusion detection policies based on a system model. In *Proceedings of the 5th International Symposium for ICS & SCADA Cyber Security Research 2018*, pages 112–121, Hamburg, Germany, 2018. BCS Learning & Development Ltd.

[32] J. J. Chromik, A. Remke, and B. R. Haverkort. An integrated testbed for locally monitoring SCADA systems in Smart Grids. *Energy Informatics*, pages 1–29, 2018.

[33] J. J. Chromik, A. Remke, B. R. Haverkort, and G. Geist. A Parser for Deep Packet Inspection of IEC-104: A Practical Solution for Industrial Applications. In *The 49th IEEE/IFIP International Conference on Dependable Systems and Networks*, Portland, Oregon, USA, 2019. IEEE/IFIP. To appear.

[34] A. Ciocia, G. Chicco, P. Di Leo, M. Gai, A. Mazza, F. Spertino, and N. Hadj-Said. Voltage control in low voltage grids: A comparison between the use of distributed photovoltaic converters or centralized devices. In *International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe*, pages 1–6, Milan, Italy, 2017. IEEE.

[35] G. R. Clarke, D. Reynders, and E. Wright. *Practical modern SCADA protocols: DNP3, 60870.5 and related systems.* Elsevier, Oxford, UK, 2004.

[36] C. Davis, J. Tate, H. Okhravi, C. Grier, T. Overbye, and D. Nicol. SCADA cyber security testbed development. In *Proceedings of the 38th North American Power Symposium (NAPS)*, pages 483–488, Carbondale, IL, USA, 2006. IEEE.

[37] M. Di Santo, A. Vaccaro, D. Villacci, and E. Zimeo. A distributed architecture for online power systems security analysis. *IEEE Transactions on Industrial Electronics*, 51(6):1238–1248, 2004.

[38] DRAGOS. CRASHOVERRIDE, Analysis of the Threat to Electric Grid Operations. Technical report, DRAGOS, 2017.

[39] ENISA. Communication network dependencies for ICS/SCADA Systems, 2017. European Union Agency For Network And Information Security, ISBN: 978-92-9204-192-2, doi: 10.2824/397676.

[40] European Commission. Energy 2020. a strategy for competitive, sustainable and secure energy. Technical report, European Commission, 2011.

[41] Á. Éva, J. Gábor, and S. P. Tamás. Proposal of a Secure Modbus RTU Communication with Adi Shamir's Secret Sharing Method. *International Journal of Electronics and Telecommunications*, 64(2):107–114, 2018.

[42] N. Falliere, L. O. Murchu, and E. Chien. W32. Stuxnet Dossier. Technical report, White paper, Symantec Corp., Security Response, 2011.

[43] D. Fauri, B. de Wijs, J. den Hartog, E. Costante, E. Zambon, and S. Etalle. Encryption in ICS networks : a blessing or a curse? In *IEEE SmartGridCom : Proceedings of the 2017 IEEE International Conference on Smart Grid Communications, 23-26 October 2017*, pages 289–294, Dresden, Germany, 2017. IEEE Computer Society.

[44] B. Ferling, J. Chromik, M. Caselli, and A. Remke. Intrusion detection for sequence-based attacks with reduced traffic models. In *Proceedings of the 19th International GI/ITG Conference, Measurement, Modelling and Evaluation of*

Computing Systems, volume 10740, pages 53–67, Erlangen, Germany, 2018. Springer.

[45] R. Flosbach. A prototype implementation for process-aware intrusion detection in electrical grids. Master's thesis, Westfälische Wilhelms Universität, Münster, 2018.

[46] I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta. Design and implementation of a Secure Modbus Protocol. In *International conference on critical infrastructure protection, IFIP Advances in Information and Communication Technology*, volume 311, pages 83–96, Berlin, Heidelberg, Germany, 2009. Springer.

[47] I. N. Fovino, A. Carcano, T. D. L. Murel, A. Trombetta, and M. Masera. Modbus/DNP3 state-based intrusion detection system. In *24th IEEE Int. Conf. on Advanced Information Networking and Applications*, pages 729–736. IEEE, 2010.

[48] I. N. Fovino, A. Coletta, A. Carcano, and M. Masera. Critical state-based filtering system for securing SCADA network protocols. *IEEE Transactions on Industrial Electronics*, 59(10):3943–3950, 2012.

[49] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18–28, 2009.

[50] A. Ginter. The top 20 cyberattacks on Industrial Control Systems. Technical report, Waterfall, 2017.

[51] A. Ginter. Defining Control Security. `https://ics-cert.us-cert.gov/sites/default/files/ICSJWG-Archive/QNL_JUN_18/defining_control_security_S508C.pdf`, 2018. ICS-CERT.

[52] N. Goldenberg and A. Wool. Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International Journal of Critical Infrastructure Protection*, 6(2):63–75, 2013.

[53] A. Greenberg. 'Crash Override': The Malware That Took Down a Power Grid. `https://www.wired.com/story/crash-override-malware/`, 12 Jun 2017.

[54] P. Gunathilaka, D. Mashima, and B. Chen. SoftGrid: A Software-based Smart Grid Testbed for Evaluating Substation Cybersecurity Solutions. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, pages 113–124, Vienna, Austria, 2016. ACM.

[55] D. Hadžiosmanović. *The process matters: cyber security in industrial control systems*. Centre for Telematics and Information Technology (CTIT), Enschede, The Netherlands, 2014. ISBN 978-90-365-3604-2.

[56] D. Hadžiosmanović, D. Bolzoni, and P. H. Hartel. A log mining approach for process monitoring in SCADA. *International Journal of Information Security*, 11(4):231–251, 2012.

[57] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel. 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, New Orleans, Louisiana, USA, 2014. ACM.

[58] M. T. Hagh, S. M. Mahaei, and K. Zare. Improving bad data detection in state estimation of power systems. *International Journal of Electrical and Computer Engineering (IJECE)*, 1(2):85–92, 2011.

[59] B. R. Haverkort. *Performance of Computer Communication Systems: A Model-Based Approach*. John Wiley & Sons, Inc., New York, NY, USA, 1998. ISBN 0471972282.

[60] J. Hong, C.-C. Liu, and M. Govindarasu. Integrated anomaly detection for cyber security of the substations. *IEEE Transactions on Smart Grid*, 5(4):1643–1653, 2014.

[61] ICS-CERT. Alert (TA17-163A) CrashOverride Malware. `https://www.us-cert.gov/ncas/alerts/TA17-163A`, 12 Jun 2017.

[62] ICS-CERT. ICS-CERT. Year in Review, Industrial Control Systems Cyber Emergency Response Team. Technical report, NCCIC, ICS-CERT, 2016.

[63] ICS-CERT. Alert (IR-ALERT-H-16-056-01) Cyber-Attack Against Ukrainian Critical Infrastructure. `https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01`, 25 Feb 2016.

[64] ICS-CERT. Advisory (ICSA-10-272-01), Primary Stuxnet Advisory. `https://ics-cert.us-cert.gov/advisories/ICSA-10-272-01`, 29 Sep 2010.

[65] IEA. Technology Roadmap: Smart Grids. International Energy Agency. `https://www.iea.org/publications/freepublications/publication/smartgrids_roadmap.pdf`, 2011.

[66] IEC. IEC 62443 (old ISA99) Security of Industrial Automation and Control Systems , 2015.

[67] IEC. Power systems management and associated information exchange - data and communications security. `https://webstore.iec.ch/publication/6912`, 2018.

[68] IEC-101. IEC TS 60870-5-101:2003. Technical specification, TC 57 - Power systems management and associated information exchange, Geneva, 2003.

[69] IEC-104. IEC TS 60870-5-7:2013. Technical specification, TC 57 - Power systems management and associated information exchange, Geneva, 2013.

[70] V. M. Igure, S. A. Laughter, and R. D. Williams. Security issues in SCADA networks. *Computers & Security*, 25(7):498–506, 2006.

[71] International Computer Science Institute and Networking and Security Group. About Spicy. `http://www.icir.org/hilti/`, 2018.

[72] Y. Isozaki, S. Yoshizawa, Y. Fujimoto, H. Ishii, I. Ono, T. Onoda, and Y. Hayashi. On detection of cyber attacks against voltage control in distribution power grids. In *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 842–847. IEEE, 2014.

[73] Y. Jang, I. Shin, B.-g. Min, J. Seo, and M. Yoon. Whitelisting for critical IT-based infrastructure. *IEICE transactions on communications*, 96(4):1070–1074, 2013.

[74] X. Jin, J. Bigham, J. Rodaway, D. Gamez, and C. Phillips. Anomaly detection in electricity cyber infrastructures. In *Proceedings of the International Workshop on Complex Networks and Infrastructure Protection*, Rome, Italy, 2006.

[75] G. Kabasele Ndonda and R. Sadre. A Two-level Intrusion Detection System for Industrial Control System Networks using P4. In *Proceedings of the 5th International Symposium for ICS & SCADA Cyber Security Research 2018*, pages 31–40, Hamburg, Germany, 2018. BCS Learning & Development Ltd.

[76] B. Kang, P. Maynard, K. McLaughlin, S. Sezer, F. Andrén, C. Seitl, F. Kupzog, and T. Strasser. Investigating cyber-physical attacks against IEC 61850 photovoltaic inverter installations. In *Proceedings of the 20th Conference on Emerging Technologies & Factory Automation*, pages 1–8, Luxembourg, Luxembourg, 2015. IEEE.

[77] S. Kenner, R. Thaler, M. Kucera, K. Volbert, and T. Waas. Comparison of smart grid architectures for monitoring and analyzing power grid data via Modbus and REST. *EURASIP Journal on Embedded Systems*, 2017(1):1–13, 2016.

[78] M. Kerkers, J. Chromik, A. Remke, and B. R. Haverkort. A Tool for Generating Automata of IEC60870-5-104 Implementations. In *Proceedings of the 19th International GI/ITG Conference, Measurement, Modelling and Evaluation of Computing Systems*, volume 10740, pages 307–311, Erlangen, Germany, 2018. Springer.

[79] R. Khan, P. Maynard, K. McLaughlin, D. Laverty, and S. Sezer. Threat Analysis of BlackEnergy Malware for Synchrophasor based Real-time Control and Monitoring in Smart Grid. In *Proceedings of the 4th International Symposium for ICS&SCADA Cyber Security Research*, pages 53–63, Belfast, UK, 2016.

[80] S. Khan and J. Mauri. *Green Networking and Communications: ICT for Sustainability*. CRC Press, Boca Raton, FL, 2013. ISBN 9781466568754.

[81] A. Kleinmann and A. Wool. A statechart-based anomaly detection model for multi-threaded SCADA systems. In *International Conference on Critical Information Infrastructures Security*, pages 132–144. Springer, 2015.

[82] A. Kleinmann, O. Amichay, A. Wool, D. Tenenbaum, O. Bar, and L. Lev. Stealthy Deception Attacks Against SCADA Systems. In *International Workshop on the Security of Industrial Control Systems and Cyber-Physical Systems*, volume 10683, 2017.

[83] A. M. Kosek. Contextual anomaly detection for cyber-physicalsecurity in Smart Grids based on an artificial neuralnetwork model. In *Joint Workshop on Cyber-Physical Security and Resilience in SmartGrids (CPSR-SG2016)*, Vienna, Austria, 2016. IEEE.

[84] G. Koutsandria, V. Muthukumar, M. Parvania, S. Peisert, C. McParland, and A. Scaglione. A hybrid network IDS for protective digital relays in the power transmission grid. In *Proceedings of the International Conference on Smart Grid Communications (SmartGridComm)*, pages 908–913, Venice, Italy, 2014. IEEE.

[85] G. Koutsandria, R. Gentz, M. Jamei, A. Scaglione, S. Peisert, and C. McParland. A real-time testbed environment for cyber-physical security on the power grid. In *Proceedings of the 1st ACM Workshop on Cyber-Physical Systems-Security*, pages 67–78, Denver, CO, USA, 2015. ACM.

[86] A. Lemay, J. Rochon, and J. M. Fernandez. A Practical flow white list approach for SCADA systems. In *Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research*, pages 1–4, Belfast, UK, 2016. BCS Learning & Development Ltd.

[87] M. Lévesque, D. Q. Xu, G. Joós, and M. Maier. Communications and power distribution network co-simulation for multidisciplinary smart grid experimentations. In *45th Annual Simulation Symposium*. Society for Computer Simulation International, 2012.

[88] C. Li, T. Dragicevic, N. L. D. Aldana, A. C. L. Hernández, Y. Guan, T. B. Rasmussen, and S. Beheshtaein. Grid architecture for future distribution system—a cyber-physical system perspective. In *Industrial Electronics Society, IECON 2017-43rd Annual Conference of the IEEE*, pages 5235–5239. IEEE, 2017.

[89] C.-Y. Lin and S. Nadjm-Tehrani. Understanding IEC-60870-5-104 Traffic Patterns in SCADA Networks. In *Proceedings of the 4th ACM Cyber-Physical System Security Workshop (CPSS) at AsiaCCS, Incheon, Republic of Korea*, pages 51–60, 2018.

[90] C.-Y. Lin, S. Nadjm-Tehrani, and M. Asplund. Timing-based anomaly detection in SCADA networks. In *International Conference on Critical Information Infrastructures Security*, pages 48–59. Springer, 2017.

[91] H. Lin, S. Sambamoorthy, S. Shukla, J. Thorp, and L. Mili. Power system and communication network co-simulation for smart grid applications. In *Proceedings of the Innovative Smart Grid Technologies (ISGT), Anaheim, CA, USA*, pages 1–6. IEEE, 2011.

[92] H. Lin, A. Slagell, C. Di Martino, Z. Kalbarczyk, and R. K. Iyer. Adapting Bro into SCADA: Building a Specification-based Intrusion Detection System for the DNP3 Protocol. In *Proceedings of the 8th Annual Cyber Security and Information Intelligence Research Workshop*, pages 5:1–5:4, Oak Ridge, TN, USA, 2013. ACM.

[93] H. Lin, A. Slagell, Z. Kalbarczyk, P. W. Sauer, and R. K. Iyer. Semantic security analysis of SCADA networks to detect malicious control commands in power grids. In *1st ACM Work. on Smart energy grid security*, pages 29–34. ACM, 2013.

[94] H. Lin, A. Slagell, Z. Kalbarczyk, P. Sauer, and R. Iyer. Runtime semantic security analysis to detect and mitigate control-related attacks in power grids. *IEEE Transactions on Smart Grid*, PP(99):1–16, 2016.

[95] O. Linda, T. Vollmer, and M. Manic. Neural network based intrusion detection system for critical infrastructures. In *International Joint Conference on Neural Networks*, pages 1827–1834. IEEE, 2009.

[96] Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. *Transactions on Information and System Security*, 14(1):13:1–13:33, 2011.

[97] Louis Marinos. European Union Agency for Network and Information Security (ENISA): Smart Grid Threat Landscape and Good Practice Guide, 2013.

[98] S. Lu, S. Repo, D. Della Giustina, F. A.-C. Figuerola, A. Löf, and M. Pikkarainen. Real-time low voltage network monitoring—ICT architecture and field test experience. *IEEE Transactions on Smart Grid*, 6(4):2002–2012, 2015.

[99] A. N. Mahmood, C. Leckie, J. Hu, Z. Tari, and M. Atiquzzaman. Network traffic analysis and SCADA security. In *Handbook of Information and Communication Security*, pages 383–405. Springer, 2010.

[100] D. Mashima, P. Gunathilaka, and B. Chen. An active command mediation approach for securing remote control interface of substations. In *Proceedings of the International Conference on Smart Grid Communications (SmartGridComm)*, pages 147–153, Sydney, Australia, 2016. IEEE.

[101] S. Massoud Amin. Smart Grid: Overview, Issues and Opportunities. Advances and Challenges in Sensing, Modeling, Simulation, Optimization and Control. *European Journal of Control*, 17(5-6):547–567, 2011. ISSN 09473580.

[102] P. Maynard, K. McLaughlin, and B. Haberler. Towards Understanding Man-in-the-middle Attacks on IEC 60870-5-104 SCADA Networks. In *Proceedings of the 2nd International Symposium for ICS&SCADA Cyber Security Research, St Pölten, Austria*, pages 30–42, 2014.

[103] S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, and P. McDaniel. Multi-vendor penetration testing in the advanced metering infrastructure. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 107–116. ACM, 2010.

[104] R. Mitchell, I. Chen, et al. Effect of intrusion detection and response on reliability of cyber physical systems. *IEEE Transactions on Reliability*, 62(1):199–210, 2013.

[105] Modbus. The Modbus Organization, Modbus application protocol specification, ver. 1.1b3. `http://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf`, 2012.

[106] Modbus. MODBUS/TCP Security - Protocol Specification. `http://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf`, 2018.

[107] S. Mustard. Security of distributed control systems: The concern increases. *Computing & Control Engineering Journal*, 16(6):19–25, 2005.

[108] A. Nicholson, S. Webber, S. Dyer, T. Patel, and H. Janicke. SCADA security in the light of Cyber-Warfare. *Computers & Security*, 31(4):418–436, 2012.

[109] NIST. Guidelines for Smart Grid Cyber Security. Technical Report 7628, National Institute of Standards and Technology (NIST), Gaithersburg, MD, 2010.

[110] J. Nivethan and M. Papa. A SCADA Intrusion Detection Framework that Incorporates Process Semantics. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, pages 1–5, Oak Ridge, TN, USA, 2016. ACM.

[111] J. Nivethan and M. Papa. On the use of open-source firewalls in ICS/SCADA systems. *Information Security Journal: A Global Perspective*, 25(1-3):83–93, 2016.

[112] J. Nugteren. ACM completes investigation into power outage in Diemen. `https://www.acm.nl/en/publications/publication/16469/ACM-completes-investigation-into-power-outage-in-Diemen/`, 3 Oct 2016.

[113] OFFIS. Mosaik Documentation. `http://mosaik.readthedocs.io/en/latest/overview.html`., 2017.

[114] OFFIS. How mosaik communicates with a simulator. `https://mosaik.readthedocs.io/en/latest/mosaik-api/overview.html`, 2018.

[115] OFFIS. Scheduling and simulation execution. `https://mosaik.readthedocs.io/en/latest/scheduler.html`, 2018.

[116] P. Oman and M. Phillips. Intrusion detection and event monitoring in SCADA networks. In *International Conference on Critical Infrastructure Protection*, pages 161–173. Springer, 2007.

[117] M. Parvania, G. Koutsandria, V. Muthukumary, S. Peisert, C. McParland, and A. Scaglione. Hybrid control network intrusion detection systems for automated power distribution systems. In *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 774–779. IEEE, 2014.

[118] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.

[119] U. K. Premaratne, J. Samarabandu, T. S. Sidhu, R. Beresh, and J.-C. Tan. An intrusion detection system for iec61850 automated substations. *IEEE Transactions on Power Delivery*, 25(4):2376–2383, 2010.

[120] PYPOWER. Pypower. `https://pypi.org/project/PYPOWER/`, 2018.

[121] M. A. Rahman and H. Mohsenian-Rad. False data injection attacks with incomplete information against smart power grids. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 3153–3158, 2012.

[122] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly. Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE control systems magazine*, 21(6):11–25, 2001.

[123] RISI. The Repository of Industrial Security Incidents. `http://www.risidata.com/`, 2015.

[124] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX Conference on System Administration, Seattle, WA, USA*, LISA '99, pages 229–238. USENIX Association, 1999.

[125] G. Sabaliauskaite and A. P. Mathur. Aligning cyber-physical system safety and security. In *Complex Systems Design & Management Asia*, pages 41–53. Springer, 2015.

[126] M. A. H. Sadi, M. H. Ali, D. Dasgupta, R. K. Abercrombie, and S. Kher. Co-Simulation Platform For Characterizing Cyber Attacks in Cyber Physical Systems. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 1244–1251, Cape Town, South Africa, 2015. IEEE.

[127] J. Santanna, J. Chromik, J. Ceron, A. Pras, and B. R. Haverkort. Online Discoverability and Vulnerabilities of ICS/SCADA Devices in the Netherlands, 2019. To be published.

[128] SCADA Security meeting of the Dutch Distribution System Operators Netbeheer Nederland. "Personal communication, 7 June 2017", 2017.

[129] K. Scarfone and P. Mell. Guide to Intrusion Detection and Prevention Systems (IDPS), 2007. NIST Special Publication 800–94.

[130] F. Schloegl, S. Rohjans, S. Lehnhoff, J. Velasquez, C. Steinbrink, and P. Palensky. Towards a classification scheme for co-simulation approaches in energy systems. In *Proceedings of the International Symposium on Smart Electric Distribution Systems and Technologies*, pages 516–521, Vienna, Austria, 2015. IEEE.

[131] S. Schütte, S. Scherfke, and M. Sonnenschein. Mosaik-smart grid simulation API. In *Proceedings of International Conference on Smart Grids and Green IT Systems*, pages 14–24, 2012.

[132] H. Security. Homeland Security. Recommended Practice: Improving Industrial Control Systems Cybersecurity with Defense-In-Depth Strategies, 2009.

[133] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based anomaly detection: a new approach for detecting network intrusions. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 265–274. ACM, 2002.

[134] A. Shahzad, M. Lee, Y.-K. Lee, S. Kim, N. Xiong, J.-Y. Choi, and Y. Cho. Real time MODBUS transmissions and cryptography security designs and enhancements of protocol sensitive information. *Symmetry*, 7(3):1176–1210, 2015.

[135] R. Singh, F. Tuffner, J. Fuller, and K. Schneider. Effects of distributed energy resources on conservation voltage reduction (cvr). In *IEEE Power and Energy Society General Meeting*, pages 1–7, San Diego, CA, USA, 2011. IEEE.

[136] R. Sommer, J. Amann, and S. Hall. Spicy: a unified deep packet inspection framework for safely dissecting all your data. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 558–569. ACM, 2016.

[137] S. Sridhar, A. Hahn, and M. Govindarasu. Cyber-physical system security for the electric power grid. *Proceedings of the IEEE*, 100(1):210–224, 2012.

[138] K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems (ICS) Security, 2011. NIST Special Publication 800–82.

[139] Symantec Security Response Attack Investigation Team. Dragonfly: Western energy sector targeted by sophisticated attack group. `https://www.symantec.com/blogs/threat-intelligence/dragonfly-energy-sector-cyber-attacks`, 20 Oct 2017.

[140] A. Teixeira, S. Amin, H. Sandberg, K. H. Johansson, and S. S. Sastry. Cyber security analysis of state estimators in electric power systems. In *49th IEEE Conference on Decision and Control (CDC). Atlanta, GA. DEC 15-17, 2010*, pages 5991–5998, 2010.

[141] A. Teixeira, G. Dán, H. Sandberg, and K. H. Johansson. A cyber security study of a SCADA energy management system: Stealthy deception attacks on the state estimator. *IFAC Proceedings Volumes*, 44(1):11271–11277, 2011.

[142] The Bro Network Security Monitor. Broccoli: The Bro Client Communications Library. `https://www.bro.org/sphinx/components/broccoli/broccoli-manual.html`, 2018.

[143] R. Udd, M. Asplund, S. Nadjm-Tehrani, M. Kazemtabrizi, and M. Ekstedt. Exploiting Bro for intrusion detection in a SCADA system. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, pages 44–51, Xi'an, China, 2016. ACM.

[144] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria*, pages 1092–1105. ACM, 2016.

[145] A. Valdes and S. Cheung. Communication pattern anomaly detection in process control systems. In *Technologies for Homeland Security, 2009. HST'09. IEEE Conference on*, pages 22–29. IEEE, 2009.

[146] P. van Oirsouw. *Netten voor distributie van elektriciteit.* Atelier Rijksbouwmeester, 2011.

[147] A. Wain, S. Reiff-Marganiec, H. Janicke, and K. Jones. Towards a Distributed Runtime Monitor for ICS/SCADA Systems. In *Proceedings of the 4th International Symposium for ICS&SCADA Cyber Security Research*, pages 132–141, Belfast, UK, 2016. BCS Learning & Development Ltd.

[148] G. B. Wetherill and D. W. Brown. *Statistical process control : theory and practice.* Chapman and Hall, London ; New York, 1991.

[149] Wireshark. Wireshark Sample Captures, viewed 24.04.2018. https://wiki.wireshark.org/SampleCaptures#IEC_60870-5-104.

[150] D. Yang, A. Usynin, and J. W. Hines. Anomaly-based intrusion detection for SCADA systems. In *5th intl. topical meeting on nuclear plant instrumentation, control and human machine interface technologies (npic&hmit 05)*, pages 12–16, 2006.

[151] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, B. Pranggono, and H. Wang. Intrusion detection system for IEC 60870-5-104 based SCADA networks. In *Power and Energy Society General Meeting (PES), 2013 IEEE*, pages 1–5. IEEE, 2013.

[152] Y. Yang, K. McLaughlin, S. Sezer, T. Littler, E. G. Im, B. Pranggono, and H. Wang. Multiattribute SCADA-specific intrusion detection system for power networks. *IEEE Transactions on Power Delivery*, 29(3):1092–1102, 2014.

[153] Y. Yang, K. McLaughlin, S. Sezer, Y. Yuan, and W. Huang. Stateful intrusion detection for IEC 60870-5-104 SCADA security. In *PES General Meeting| Conference & Exposition*, pages 1–5, 2014.

[154] Y. Yang, H.-Q. Xu, L. Gao, Y.-B. Yuan, K. McLaughlin, and S. Sezer. Multidimensional intrusion detection system for IEC 61850-based SCADA networks. *IEEE Transactions on Power Delivery*, 32(2):1068–1078, 2017.

[155] E. Zambon, I. Cairo, E. Costante, M. Guadagnoli, D. Lavernia, G. Leon, J. Marin, R. Barbosa, A. Ribak, A. Ruiz, and L. Trilla. D2.3 Reference Taxonomy on Industrial Control Systems Networks for Utilities. Technical Report. Technical report, PREEMPTIVE, 2015.

[156] M. Zeller. Myth or reality – Does the Aurora vulnerability pose a risk to my generator? In *Protective Relay Engineers, 2011 64th Annual Conference for*, pages 130–136. IEEE, 2011.

[157] K. Zetter. The Ukrainian Power Grid Was Hacked Again. `https://motherboard.vice.com/en_us/article/bmvkn4/ukrainian-power-station-hacking-december-2016-report`, 10 Jan 2017.

[158] B. Zhu and S. Sastry. SCADA-specific intrusion detection/prevention systems: a survey and taxonomy. In *Proc. of the 1st Workshop on Secure Control Systems (SCS)*, 2010.

[159] B. Zhu, A. Joseph, and S. Sastry. A taxonomy of cyber attacks on SCADA systems. In *Internet of things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing*, pages 380–388. IEEE, 2011.

Acronyms

AMI Advanced Metering Infrastructure.
APCI Application Protocol Control Information.
APDU Application Protocol Data Unit.
ASDU Application Service Data Unit.
AVR Automatic Voltage Regulator.

BDD Bad Data Detection.

CA Contingency Analysis.
CIA Confidentiality, Integrity, Availability.
CSV Comma Separated Values.

DER Distributed Energy Resource.
DNP3 Distributed Network Protocol.
DSO Distribution System Operator.
DTMC discrete-time Markov chain.

EMS Energy Management System.

HMI Human Machine Interface.

ICS Industrial Control System.
ICT Informations and Communications Technology.
IDS Intrusion Detection System.
IEC International Electrotechnical Commission.
IEC-101 IEC-60870-5-101.
IEC-104 IEC-60870-5-104.
IED Intelligent Electronic Device.
IO Information Object.
IOA Information Object Address.

IP Internet Protocol.
IPS Intrusion Prevention System.
IT information technology.

JSON JavaScript Object Notation.

MITM man-in-the-middle.
MTU Master Terminal Unit.

OPC Open Platform Communications.

PLC Programmable Logic Controller.
PMU Phasor Measurement Unit.
PV PhotoVoltaic.

RTU Remote Terminal Unit.

SAM Self-Aware Monitor.
SCADA Supervisory Control and Data Acquisition.
SE State Estimation.
SSL Secure Sockets Layer.

TCP Transmission Control Protocol.
TLS Transport Layer Security.
TSO Transmission System Operator.

UML Unified Modeling Language.

VPN Virtual Private Network.

WAN Wide Area Network.

About the author

Justyna Chromik was born in Pszczyna, in Poland on March 30^{th}, 1989. She received a Bachelor of Science degree in Electronics and Telecommunications in 2011 at the AGH University of Applied Science and Technology, in Kraków, Poland. Her work addressed reviewing challenges and providing guidelines for building a secure and functional company network. Her interest in network and information security started already then.

After spending the summer internship at a web hosting company *Keenondots* in Enschede, the Netherlands, she decided to move there and pursue her further education at the University of Twente. In 2012, she started Telematics Master of Science programme at that university. She obtained her M. Sc. degree *cum laude* in 2015, with the work addressing classification of websites offering Distributed Denial of Service attacks as a paid service.

Years 2015-2019 she has spent working on the MOSES project (*More Secure SCADA Through Self-Awareness*). She had the opportunity to learn about the real life security problems concerning electric power distribution, thanks to collaboration with project partners: Coteq Netbeheer, ENCS and TNO. This cooperation allowed for applying the theoretical models into practice, which contributed to the final result of Justyna's thesis.