INTRUSION DETECTION IN NETWORKED CONTROL SYSTEMS:

FROM SYSTEM KNOWLEDGE TO NETWORK SECURITY

Marco Caselli

Intrusion Detection in Networked Control Systems:
from System Knowledge to Network Security

MARCO CASELLI

Composition of the Graduation Committee:

Prof. dr.	P.M.G. Apers	Universiteit Twente (chairman)
Prof. dr.	F.E. Kargl	Universität Ulm (promotor)
Dr.	E. Zambon	Universiteit Twente SecurityMatters B.V. (assistant-promotor)
Prof. dr.	P.H. Hartel	Universiteit Twente
Prof. dr. ir.	A. Pras	Universiteit Twente
Prof. dr.	R. Baldoni	Sapienza, Università di Roma
Prof. dr.	S. Etalle	Universiteit Twente Techinsche Universiteit Eindhoven
Dr.	M. Dacier	Qatar Computing Research Institute
Dr.	R. Sommer	International Computer Science Institute Lawrence Berkeley National Laboratory

This research has been partially supported by the European Commission through project FP7-SEC-285477-CRISALIS funded by the 7th Framework Program.

UNIVERSITY OF TWENTE.

Services, Cybersecurity and Safety Group
P.O. Box 217, 7500 AE
Enschede, the Netherlands

CTIT

CTIT Ph.D. Thesis Series No. 16-401
Centre for Telematics and Information Technology
P.O. Box 217, 7500 AE
Enschede, the Netherlands

ISBN: 978-90-365-4177-0
ISSN: 1381-3617
DOI: 10.3990/1.9789036541770
<http://dx.doi.org/10.3990/1.9789036541770>

Typeset with L^AT_EX. Printed by: Gildeprint Drukkerijen
Cover design by: Marco Caselli, Marco Locati, Simona di Michele

Copyright © 2016 Marco Caselli, Enschede, the Netherlands
All rights reserved. No part of this book may be reproduced or transmitted, in any form or by any means, electronic or mechanical, including photocopying, microfilming, and recording, or by any information storage or retrieval systems, without the prior written permission of the author.

**INTRUSION DETECTION IN
NETWORKED CONTROL SYSTEMS:
FROM SYSTEM KNOWLEDGE TO NETWORK SECURITY**

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. H. Brinksma,
on account of the decision of the graduation committee,
to be publicly defended
on Friday, 4th of November 2016 at 16:45

by

Marco Caselli

born on 3rd of August 1986
in Palermo, Italy

This dissertation has been approved by:

Prof. dr. F.E. Kargl (promotor)

“The limits of my language mean the limits of my world.”

Ludwig J.J. Wittgenstein

Acknowledgments

I'm on the bike.

University is dark at night.

I'm wondering if somebody is still here. I wouldn't, if it wasn't for writing the abstract in Dutch. I guess it's my fault. What a country, the Netherlands. And the people. I have been around for years and I could stay without knowing the language. Everybody was always so nice speaking in English. I should definitely start thanking the Netherlands.

Crossroad. It's better to wait.

So much time. And I didn't even think I would have done it. I remember Igor coming over and saying "Why don't you start a PhD, I know they've got an open position in Twente". I guess I should thank him. No matter what's next, it was the best experience of my life. I remember arriving here. Meeting the bosses. Pieter and his tricky questions. Sandro and his easygoing attitude. Damiano and his subtle talking. Frank of course. Our "red telephone" connecting Enschede and Ulm was always-on. Despite the distance, it really helped a lot having somebody looking after me. Then, there was Emmanuele. You can't go home at five-thirty when Emma is around (it's not half a day's work). It was always endless talking. . . It was always good talking. Good bosses should be thanked.

Hengelsestraat runs for kilometers.

I can recall one colleague for each pedaling.

Fellow PhDs first. Christoph. Arjan. Elmer. Eleftheria. Michael. Dan. Jan-Willem. Chris. Ines. Susanne. . . Dina takes a couple of pedalings more. She certainly walked me through my first year. . . Bertine. Suse. Geert Jan. Lorena. Jonathan. Maarten. Alexandr. Andreas (and our chats on research, politics and conference rankings). . . My office is another story. Ali and his "Chinese hand". Riccardo and our love for Inter. Tim with his "super-cool" $\text{\LaTeX}$. Bence and his candid look when he suddenly asked me "What month is it?" in one of our late nights working in the office. It may take a while, but I should thank all my colleagues.

It's still cold at the traffic light.

I was surprised it was so cold in Berkeley. When I asked Robin about the internship I kind of pictured myself surfing at the beach. In the end, I was terrible at surfing but I enjoyed working and living there. I should thank the people in Berkeley.

For sure Johanna and Robin. I learned a lot. I guess even my writing improved a little. I still like streams of consciousness but I hope I won't get any more comments like "An 'introduction' at the fourth level of the section structure suggests that there must be a better way to lay things out".

The city center.

There is much more light here.

I'm still wondering where they hide one hundred fifty thousand people in Enschede. But, indeed, I met countless people (even here at Paddy's). So many different moments and so many more friendships that I would like to bring with me. Virginia and the house where we lived (with and without Giorgio), Riccardo, Katerina, Miriam, Giuseppe, Caterina & Federico, Vera, Giorgio, Nicole, Eamon, Federico, Luigi... It would take forever, but I should thank all the people I met here.

Almost at the square. It's warm now.

Cycling remembers me my motorbike. My motorbike remembers me Palermo. Palermo is closest friends and family. Friends almost become family when you know them long enough... Mario, Immo, Alessandro, Enrico, Benedetta, Eugenio... Mauro & Emilia, they are Dutch like me. Krotone (that was Rome, though). Michele. After sixteen years, I still can't forget his Briton archers and the gaming at night. How couldn't I thank them?... Then, there is family. There is mom, dad and Martina. They can be far, but close at the same time. I should always thank them... Finally, there is one more thing. There is Roberta with all the beautiful things we did together and all the things I hope we'll do in the future. This last should be the biggest thanks.

I'm parking the bike and I guess I've something to write.

Was it seventeen minutes, four years, or a lifetime?

Doesn't matter.

I'm home.

Marco Caselli
Enschede, 17/10/2016

Abstract

“Networked control system” (NCS) is an umbrella term encompassing a broad variety of infrastructures such as industrial control systems (ICSs) and building automation systems (BASs). Nowadays, all these infrastructures play an important role in several aspects of our daily life, from managing essential services such as energy and water (e.g., critical infrastructures) to monitoring the increasingly smart environments that surround us (e.g., the Internet of Things). Over the years, NCS technology has progressively switched to IT digital networks and integrated to the Internet. This fact has changed the way operators manage and control their infrastructures and has introduced several security threats. Skilled crackers (also known as black-hat hackers) can remotely access NCSs and change infrastructure behavior potentially endangering human lives (e.g., causing a malfunction of a nuclear power plant). For this reason, NCS stakeholders have been facing the challenge of protecting their infrastructures against cyber-attacks and, especially, targeted attacks, namely those attacks carried out by resourceful and motivated organizations (e.g., Stuxnet). A common practice for protecting NCSs includes the use of standard IT security solutions and techniques. However, most of the times, these solutions do not fit such different environments. Furthermore, any security solution applied to NCSs should never interfere with infrastructure operations. This is particularly important when it comes to NCSs that monitor critical infrastructures and thus, sensitive physical processes (e.g., energy production). Finally, most of today’s NCS security solutions still fail to convey accurate information to the operators and do not allow them to quickly and undoubtedly identify potentially dangerous situations. In fact, this would require more sophisticated techniques capable of understanding the surrounding environment and conclusively discern between malicious activities and valid operations.

For all these reasons, this thesis tackles the challenge of developing more incisive and effective security solutions for NCSs. We focus on intrusion detection to passively monitor and evaluate infrastructure operations without causing any interference and we aim attention at the acquisition of knowledge about the monitored infrastructures to improve the process of detection as well as the feedback to the operators. In what follows, we present a novel approach to NCS security based on the integration between system knowledge acquisition and network intrusion detection. Our work starts by identifying and evaluating valuable sources of information to gain knowledge about the monitored systems. Then, we show how this knowledge contributes to improving intrusion detection systems (IDSs). Finally,

we leverage a specific kind of intrusion detection, namely specification-based intrusion detection, to strengthen the bond between system knowledge and network security. We achieve this by automating the deployment of specification-based IDSs that autonomously use information gathered from NCS network traffic and analyze NCS-related available documentation to describe infrastructure expected behavior. Tests and evaluations performed on real infrastructures support the proposed approach and confirm the advantages of including information about NCS properties and components within the employed security solutions.

Overzicht

“Networked control system” (NCS) is een overkoepelende term voor een breed scala aan infrastructuren, zoals ‘industrial control systems’ (ICSs) en ‘building automation systems’ (BASs). Tegenwoordig spelen al deze infrastructuren een belangrijke rol in verschillende aspecten in ons dagelijkse leven, van het beheren van essentiële diensten, zoals energie en water (zoals vitale infrastructuren), tot het monitoren van steeds slimmer wordende omgevingen die ons heen verschijnen (zoals het ‘Internet of Things’). In de loop der tijd, is de NCS-technologie geleidelijk overgeschakeld naar IT digitale netwerken en geïntegreerd met het internet. Dit gegeven heeft de manier waarop operators hun infrastructuren beheren en controleren veranderd en heeft verschillende beveiligingsdreigingen geïntroduceerd. Kundige ‘crackers’ (ook bekend als ‘black-hat hackers’) kunnen op afstand NCS beheren en mogelijk zelfs mensenlevens in gevaar brengen door de werking van infrastructuren te veranderen (bijvoorbeeld door een ongeluk teweeg te brengen in een kerncentrale). Om deze reden worden NCS-belanghebbende geconfronteerd met het beschermen van hun infrastructuur tegen cyberaanvallen en, in het bijzonder, doelgerichte aanvallen, namelijk aanvallen uitgevoerd door machtige en gemotiveerde partijen (bijvoorbeeld Stuxnet). Het gebruik van standaard IT beveiligingsoplossingen en -technieken is een veelgebruikte manier van om NCS te beveiligen. Echter, vaak zijn deze oplossingen niet voor deze andere omgevingen geschikt. Bovendien zou een beveiligingsoplossing voor NCSs nooit de infrastructuuroperaties mogen hinderen. Dit is in het bijzonder van belang voor NCSs die vitale infrastructuren, en dus gevoelige fysieke processen (bijvoorbeeld het opwekken van energie), monitoren. Tot slot zijn veel van de NCS-oplossingen van tegenwoordig nog steeds niet in staat om nauwkeurige informatie over te brengen aan de operators en stellen hen niet in staat om snel en zonder twijfel potentieel gevaarlijke situaties te identificeren. Eigenlijk zouden geavanceerdere technieken die de omgeving beter begrijpen en ook onderscheid tussen kwaadaardige activiteiten en goedaardige operaties kunnen maken, nodig moeten zijn.

Om al deze redenen, pakt dit proefschrift de uitdaging van het ontwikkelen van doortastende en effectieve beveiligingsoplossingen voor NCSs aan. We richten ons op intrusion detection om passief te monitoren en infrastructuuroperaties te evalueren zonder het teweegbrengen van een verstoring en we richten onze aandacht op het vergaren van kennis over de gemonitorde infrastructuren om zowel het detectieproces als de feedback voor de operators te verbeteren. In wat volgt, presenteren wij een nieuwe aanpak voor NCS-beveiliging gebaseerd op de integra-

tie van het vergaren van kennis over systemen en netwerk intrusion detection. Ons werk begint met het identificeren en het evalueren van waardevolle informatiebronnen om kennis van de gemonitorde systemen te krijgen. Vervolgens laten we zien hoe deze kennis bijdraagt om ‘intrusion detection systems’ (IDSs) te verbeteren. Tot slot maken we gebruik van een specifiek type van intrusion detection, namelijk ‘specification-based intrusion detection’, om de band tussen systeemkennis en netwerkbeveiliging te versterken. We bereiken dit door het uitrollen van specification-based IDSs te automatiseren. Deze specification-based IDSs maken autonoom gebruik van informatie vergaard van NCS netwerkverkeer en analyseren NCS-gerelateerde beschikbare documentatie om het verwachte gedrag van de infrastructuur te beschrijven. Tests en evaluaties uitgevoerd op echte infrastur-
ren staven de voorgestelde aanpak en bevestigen de voordelen van het opnemen van informatie over NCS-eigenschappen en -componenten in de toegepaste beveiligingsoplossingen.

Contents

I	Security in NCS	1
1	Introduction	3
1.1	Motivation	5
1.1.1	Open Problems	6
1.2	Research Questions	8
1.3	Thesis Overview and Contributions	10
2	Networked Control Systems	13
2.1	Industrial Control Systems	13
2.1.1	Architecture Overview	14
2.1.2	Protocols	15
2.2	Building Automation Systems	21
2.2.1	Architecture Overview	21
2.2.2	Protocols	22
3	Intrusion Detection	25
3.1	Anomaly-based Intrusion Detection	26
3.1.1	Classification techniques	26
3.1.2	Feature Selection	28
3.2	Specification-based Intrusion Detection	29
3.3	Summary	30
II	Enhancing Intrusion Detection in NCS	33
4	Requirements & Approach	35
4.1	Research Guidelines	36
4.2	Framework Building Blocks	39
5	Mining System Knowledge	45
5.1	Network Traffic & Fingerprinting	45
5.1.1	Overview	46
5.1.2	State of the Art	47
5.1.3	Reference Model	49
5.1.4	Fingerprinting NCSs	51

5.1.5	Use Case: the Flow Fingerprinter	56
5.1.6	Conclusions	60
5.2	Documentation	60
5.2.1	Overview	60
5.2.2	NCS Documentation	62
5.2.3	Analysis	64
5.2.4	Conclusions	67
5.3	Section Conclusion	67
6	Intrusion Detection in ICS	69
6.1	Sequence Attacks	69
6.2	State of the Art	72
6.3	Security Solution	73
6.3.1	Sequence-based Intrusion Detection	73
6.3.2	Evaluation & Discussion	95
6.4	Section Conclusion	98
7	Intrusion Detection in BAS	101
7.1	Attacks on BACnet	101
7.2	Evaluation Environments and Settings	102
7.3	Security Solution	104
7.3.1	Documentation-based Intrusion Detection	104
7.3.2	Evaluation	116
7.4	Section Conclusion	122
III	Final Remarks	123
8	Conclusion And Outlook	125
8.1	Summary of Contributions	125
8.2	Limitations	128
8.3	Future Research Directions	129

Acronyms

APCI	Application-layer Protocol Control Information (related to IEC104)
API	Application Programming Interface
APDU	Application Protocol Data Unit (related to IEC104)
ASDU	Application Service Data Unit (related to IEC104)
ASN.1	Abstract Syntax Notation One
APT	Advanced Persistent Threat
BACS	Building Automation and Control System
BAS	Building Automation System
BIBB	BACnet Interoperability Building Block
BMS	Building Management System
BSRIA	Building Services Research and Information Association
CAN	Controller Area Network
CERT	Computer Emergency Response Team
CIA	Confidentiality, Integrity, Availability
CSV	Comma-Separated Values
DBN	Dynamic Bayesian Networks
DCS	Distributed Control System
DFA	Deterministic Finite Automaton
D-IDS	<i>Documentation-based Intrusion Detection System</i>
DoS	Denial of Service
DNP3	Distributed Network Protocol v.3
DTD	Document Type Definition
DTMC	Discrete-Time Markov Chain
ECU	Engine Control Unit
EDE	Engineering Data Exchange
EFSA	Extended Finite State Automaton
FF	<i>Flow Fingerprinter</i>

FFT	Fast Fourier Transform
FSM	Finite State Machine
GO	Group Object (related to KNX)
HIDS	Host Intrusion Detection System
HMI	Human Machine Interface
HVAC	Heating, Ventilation and Air Conditioning
ICD	IED Capability Description
ICS	Industrial Control System
IDL	Interface Description Language
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPS	Intrusion Prevention System
ISO	International Organization for Standardization
LBNL	Lawrence Berkeley National Laboratory
MMS	Manufacturing Message Specification
NAT	Network Address Translation
NCS	Networked Control System
NDA	Non-Disclosure Agreement
NIDS	Network Intrusion Detection System
NLP	Natural Language Processing
OLE	Object Linking and Embedding
OPC	OLE Process Control
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
PLC	Programmable Logic Controller
PST	Probabilistic Suffix Trees
RFC	Request For Comment
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SCD	Substation Configuration Description
SCL	Substation Configuration description Language

S-IDS	<i>Sequence-based Intrusion Detection System</i>
SIEM	Security Information and Event Management
SOM	Self-Organizing Map
SPAN	Switched Port Analyzer
TTCN-3	Testing and Test Control Notation v.3
VMD	Virtual Manufacturing Device (related to MMS)
W3C	World Wide Web Consortium

Part I

Security in NCS

CHAPTER 1

Introduction

A *control system* is a device, or a set of devices, aiming to monitor and control the behavior of a process (e.g., a mechanical process, a physical phenomenon, etc.). The first definition of a control system¹, as well as the basic principles of control theory that govern this type of systems, go back to the early 20th century. For decades, control systems have remained standalone infrastructures aimed at monitoring and managing different physical phenomena in various fields of science and engineering. Then, with the advent of communication networks and, especially, the spread of digital networks and the Internet, control systems increasingly developed into large interconnected infrastructures and the traditional control paradigm shifted to a new concept: remote control. This branch of technology comes today under the name of *Networked Control Systems*.

Networked Control Systems (NCSs) are “systems whose constituents such as sensors, actuators, and controllers are distributed over a network, and their corresponding control loops (or control processes) are formed through a network layer” [14] (Figure 1.1). This definition encompasses a large set of systems that, over the last twenty years, have become more and more involved in our daily life. Examples of NCSs include *Industrial Control Systems* (ICSs) [15] and *Building Automation Systems* (BASs) [16].

All NCSs share four key components that define a basic core architecture. These components are:

- **Sensors:** whose task is to monitor physical phenomena
- **Actuators:** whose task is to intervene in the physical phenomena by modifying their properties and characteristics
- **Controllers:** whose purpose is to use sensors’ information and to accordingly operate actuators providing decision-making and issuing command based on these decisions
- **Communication networks:** whose role is to link the previous components together and allow data exchange

¹Referred to early safety systems employed in electric railways [13]

Figure 1.1: NCS logical diagram

This architecture develops and specializes into an infinite number of combinations according to the nature of sensors and actuators, the purposes of control algorithms and the type of communication protocols.

Nowadays, comprehensive statistics on the employment and use of NCSs are difficult to come by. However, a few examples are enough to describe the spread of these systems around the globe. First of all, most of the world population depends on critical infrastructures such as electricity generation, water supply, and food production. All these sectors rely on automation and specifically on ICSs. The importance and the key role of ICSs is confirmed by the effort spent by several nations for guaranteeing the correct functioning of critical infrastructures and for a harmonized interoperation among the different sectors [17, 18]. Furthermore, NCSs have recently gained a less critical but equally significant role in our houses, means of transportations and public facilities. The concept of “Internet of Things” (IoT) has drawn attention on integrating every aspect of our lives to the cyber-world. NCSs have become the way to monitor the environment around us and automatically respond to our needs with a multitude of different (and sometimes targeted) services. For example, HVAC technologies (heating, ventilating, and air conditioning) coordinate their action thanks to inputs from local sensors and remote weather forecast. Likewise, safety alarms warn people of fires and gas leaks while activating countermeasures and signaling the closest fire departments. In the recent past, the employment of IoT solutions has constantly increased and their market share is expected to grow in the following years [19]. Big corporations such as Google and Apple have been working to introduce new technology feeding user requirements for advanced autonomous systems (e.g., autonomous vehicles) and smart devices. In the near future, these systems will likely become less and less expensive and capable of handling and coordinating more complicated tasks ultimately making NCSs a leading technology in a fully interconnected world.

1.1 Motivation

Over the years, NCSs have become a valuable target of cyber-attacks. The reason behind this risk is twofold. On the one hand, the interconnection with the Internet and the gradual shift toward off-the-shelf IT technology (e.g., communications based on TCP/IP, systems based on common operating systems, etc.) has exposed NCSs to common known vulnerabilities of the digital networks (e.g., malware spreads). On the other hand, the underlying threat directly relates to the plausible “physical effects” that cyber-attacks hitting NCSs may cause. In fact, if attacks against standard IT infrastructures usually remain confined to the cyber-world, similar situations concerning NCSs may have consequences in the real world. As outlined in the introduction, NCSs play an important role in different aspects of our daily life. When it comes to ICSs used in critical infrastructures, the cyber-physical nature of NCSs puts a cracker (also known as black-hat hacker) in the condition of remotely operating dangerous equipments and potentially endangering human lives (e.g., causing a malfunction of a nuclear power plant). Besides this, BASs introduce a further issue concerning the risk of crackers being able to access private premises (e.g., by remotely opening house doors and windows [20]) or take control of household electrical appliances.

This thesis focuses on *NCS security*. Generally speaking, protecting NCSs means guaranteeing the correct and safe execution of the underlying control processes. This principle encompasses several aspects, such as: the accomplishment of all services which an NCS is responsible for, the protection of NCS physical components as well as the underlying physical processes, and ultimately the safeguard of human beings employing or depending on the NCS.

NCS security is not a new topic. Both academic publications and ordinary press report numerous examples of cyber-attacks striking these infrastructures. Within ICS, Stuxnet [21] is probably the most famous and documented case of cyber-attack against NCSs. Discovered in 2010, this malware was a highly-sophisticated software designed to manipulate embedded software of Programmable Logic Controllers (PLCs) of Iranian nuclear enrichment facilities and disrupt nuclear centrifuges. Later on, other malware shared similar targets. Particularly interesting, Duqu [22] and Shamoon [23] struck eastern critical infrastructures of the energy sector in 2011 and 2012 respectively. In contrast with Stuxnet, this malware did not directly operates cyber-physical components but worked on gathering information about breached infrastructures and thus paving the way for further intrusions. Besides these notorious cases, ICSs and their related facilities have been targeted by a number of cyber-attacks coming both from inside and outside their premises [24].

The same situation concerns BASs. News about cyber-attacks targeting these infrastructures is increasingly hitting the headlines, especially when it involves big corporations. Already in 2013, vulnerabilities in the HVAC systems of US Target stores allowed crackers to take control of the interconnected IT network and download approximately 40 million debit and credit card accounts [25]. Since then, US government agencies have issued several alerts on the importance of keeping building management systems more secure. The Building Services Research and Information Association (BSRIA) presented a research showing that, in the US,

over 90% of all larger buildings rely on automated BASs and many of these systems are to some degree at risk [26]. This issue was made extremely clear when, in 2015, the CenturyLink data center in Weehawken was in shut down mode as a result of a critical failure in its HVAC system [27].

Finally, IoT has recently shown its vulnerability against cyber-attacks by becoming a valuable target of bot herders (criminals running and mastering botnets). In fact, in 2014, researchers from Proofpoint uncovered a large scale cyber-attack against hundreds of thousands of refrigerators, TVs and other smart household appliances with the purpose of taking their control and using them to send malicious spam emails [28].

Most of the examples discussed above belong to a family of cyber-attacks that goes under the name of advanced persistent threats (APTs). APTs represent a class of sophisticated cyber-attacks carried by highly-skilled, well-resourced and motivated organizations [29]. These organizations resort to a wide-range of advanced techniques to strike against their targets and cause severe damages. Moreover, cyber-attacks of this kind are persistent as they generally last for a long time span putting the targeted infrastructures under a lot of stress and likely delaying their recovery to normal operations. APTs usually start with a comprehensive campaign of information gathering (e.g., social engineering). This information later allows the attackers to gain access to targeted infrastructures and eventually fulfill their malicious goals. Moreover, most APTs attempts to maintain control of the breached infrastructures to ease future attacks.

Defending against APTs is a broad field of research and this thesis addresses some of the challenges related to this kind of threats.

1.1.1 Open Problems

The presence of numerous different cyber-incidents shows that NCS security still faces several open problems. Literature studies such as [30, 31] confirm this assumption and draw attention to different areas of research. This thesis particularly tackles three main issues.

First, the presence of **increasingly sophisticated cyber-attacks** and the consequent need for advanced security solutions. Generally speaking, the use of standard IT defensive mechanisms in NCSs is not unusual [30]. However, despite their effectiveness against specific threats, these solutions appear insufficient against sophisticated attacks. In fact, APTs may aim to disrupt NCS control and physical processes. Usually, these attacks take action at a logical level and leverage detailed knowledge of the target system. This means that, more than exploiting vulnerabilities in the employed technology, attackers rather take advantage of the available mechanisms of the infrastructure and maliciously use them to maximize damages (e.g., attackers leveraging weaknesses in the control processes to bring entire infrastructures into unwanted critical states). On this regard, Fovino et al. provide in [32] an example of a destructive attack carried by accurately using legitimate commands. Standard IT security solutions are usually ineffective against attacks of this kind due to a lack of contextual information that allows to recognize the

problem and the related security risk. In fact, experts suggest that a viable way to improve protection against APTs is enhancing security solutions with context information of the monitored infrastructures [33]. These considerations lead the way to custom-made defensive mechanisms that operate by leveraging an in-depth knowledge of the NCS they are deployed in.

Second, the employment of **heterogeneous infrastructures and technologies** within NCSs requires a certain degree of automation in the deployment of the related defensive mechanisms. In the literature, lots of studies about NCSs do not adequately address this issue and overlook the effort security operators must spend tuning their security solutions. Every NCS has its own specific architecture, devices, and communication protocols. Moreover, the control processes embedded in every NCS change depending on the goals and purposes of the system. Encoding all this information in a defensive mechanism is not feasible for every deployment, especially when it comes to large-scale infrastructures such as regional electrical grids. Increasing the capability of a defensive mechanism to autonomously investigate and understand the environment in which it is deployed allows to gather information on the monitored system with no extra effort from the human side [34].

Third, the need for **transparent security solutions** as these solutions must not interfere with any of the on-going processes within NCSs. Most NCSs rely their functioning on carefully timed activities and some of these systems even make use of real-time components to fulfill their purposes [31]. Furthermore, employed devices do not always have the resources nor the computational power to respond to anything that is not their core task [30]. This leads to defensive mechanisms that have minimum impact on the monitored NCSs. In the best case, a security solution is deployed in a standalone machine whose only purpose is monitoring the system. Also, this security solution should not directly interact with any NCS device avoiding every possible interference with the proper execution of its tasks.

The request for *sophisticated, automated and transparent security solutions* coming from the three issues discussed above fits the development of novel *intrusion detection systems* (IDSs). Intrusion detection includes several different techniques for automated analyses of events occurring in the monitored systems with the aim of recognizing and highlighting attempts to compromise those systems. An IDS can work in parallel with other security solutions and does not require any change to the infrastructure where it is deployed in. Particularly interesting, *anomaly-based intrusion detection* techniques usually include mechanisms that aim to understand system behavior, properties and characteristics before starting the actual monitoring. This allows to address the need for an autonomous analysis of infrastructure components. Finally, anomaly-based IDSs are able to face unforeseen attacks. Compared to misuse-based detection (e.g., signature-based intrusion detection), anomaly-based solutions aim to create a reference of correct behavior with which to compare potential anomalous system activities. This feature allows to effectively respond against the sophisticated attacks discussed at the beginning of this section.

In order to fulfill the requirement for sophisticated, autonomous and transparent security solutions we address the research questions presented in the following section.

1.2 Research Questions

The goal of this work is enhancing NCS protection against sophisticated attacks such as APTs. To achieve this goal we design anomaly-based intrusion detection solutions and we combine custom techniques to autonomously gather heterogeneous information about the monitored infrastructures. In what follows, we refer to this heterogeneous information as “system knowledge” meaning knowledge about the network messages (e.g., semantic of the data carried in a network packet), the involved devices (e.g., model and properties), and even the control process (e.g., overall purpose of the infrastructure).

The aforementioned concepts lead this thesis to the following main research question:

Main RQ: *Can we defend NCSs from APTs by developing more effective anomaly-based IDSs that leverage system knowledge?*

To address this point, we present a general framework for integrating system knowledge acquisition and anomaly-based intrusion detection and we show how security solutions based on this approach may exceed results of state-of-the-art solutions.

In what follows, we break the main research question down to four inter-related sub-questions and we treat them one by one (Figure 1.2).

The first two sub-questions address the acquisition of system knowledge. We identify two possible sources of information, namely network traffic and available documentation. In the first case, we investigate the problem of gathering knowledge from NCS network communications. Therefore:

RQ1: *To what extent can we infer system knowledge from NCS network traffic?*

Acquiring knowledge from NCS network traffic relies on analyzing messages flowing in NCS networks and identifying the main communication actors. Device recognition or “fingerprinting” is an activity that uses the information available on the network to recognize and describe devices exchanging messages and to outline their main characteristics. Fingerprinting has been widely studied for standard IT but little research in literature investigates its feasibility for NCSs. Moreover, no available approach exploits NCS-specific properties to improve its results. To answer RQ1, we test IT state-of-the-art fingerprinting solutions, we study NCS fingerprinting challenges and opportunities and we develop novel techniques specifically crafted for NCSs, such as the *Flow-fingerprinter*. Finally, we investigate the semantic meaning of the network messages based on the identification of sender and receiver.

Figure 1.2: Research questions overview

For what concerns the second sub-question, we observe that most NCSs often come with configuration and description documents used by operators to manage their infrastructures. For this reason, we investigate to what extent we can automatically extract the information stored in this documentation and integrate it to what we gather from the network traffic. Therefore:

RQ2: *Can we automate the process of gaining system knowledge from NCS documentation?*

To achieve this we analyze available configuration files, reference books and user manuals either offline (within repositories owned by NCS operators) or online (over the Internet). Moreover, we investigate on the use of results coming from NCS network traffic analysis to conduct an extensive and automated search over these documents.

The last two sub-questions put our approach into practice by designing and developing two novel anomaly-based IDSs. The two solutions are tailored to NCSs and aim to detect APTs. Moreover, these IDSs take advantage of the acquired system knowledge to improve their effectiveness and refine their results. In the first case, we focus on a specific kind of APTs not yet tackled by state-of-the-art security solutions. In fact, we define “sequence attacks” meaning cyber-attacks that rely on the possibility of crafting a sequence of individually harmless network messages that is capable of disrupting the correct functioning of an NCS. Therefore, we investigate the following research question:

RQ3 *How do we detect APTs such as sequence attacks against NCSs?*

In this regard, we develop a purely anomaly-based intrusion detection approach, namely the *sequence-based intrusion detection system* or *S-IDS*. The S-IDS aims to analyze sequences of network messages with the goal of detecting changes

in the control processes. Moreover, we use system knowledge gained from our work on RQ1 to improve intrusion detection results and decrease the number of false positives. The S-IDS uses fingerprinting techniques to focus the detection on programmable logic controllers (PLCs) and employs knowledge about the semantic value of network messages to refine its results.

For what concerns the second IDS, we try to tackle a broader variety of cyber-attacks by exploiting a different paradigm of intrusion detection, called specification-based intrusion detection. A specification-based IDS leverages functional specifications of a system to model its properties and consequently creating a reference of correct behavior. Differently from standard anomaly-based IDSs, this modeled behavior does not derive from a learning phase (always lacking the complete set of infrastructure properties and operations) but is directly extracted from documentations. The following question evaluates the feasibility of specification-based intrusion detection for NCSs:

RQ4 *Can we develop specification-based intrusion detection that relies on automatically acquired system knowledge?*

This analysis further pushes the integration between intrusion detection and system knowledge and eventually develops into an IDS that automatically generates its specification rules from the retrieved technical documentation. Our *documentation-based intrusion detection system (D-IDS)* relies on system knowledge and, specifically, on the automated acquisition of the related information and uses this knowledge to detect any action of an APT that causes the monitored NCS to deviate from its optimal behavior.

1.3 Thesis Overview and Contributions

Figure 1.3 depicts an outline of this thesis and shows the main research areas.

We begin with a description of NCS architectures, components and protocols with a special focus on ICS and BAS technologies. These concepts, discussed in §2, are necessary to understand the remainder of this thesis. Furthermore, we present a brief analysis of intrusion detection state of the art. In §3, we discuss anomaly-based and specification-based detection techniques and we describe the most important results in these two fields of research. Additionally, we investigate potential weaknesses of current approaches and we outline future research directions.

In §4 we resume the discussion started in this introduction and we describe NCS security requirements in more details. Consequently, we define our approach for integrating system knowledge acquisition and anomaly-based intrusion detection. This basic framework applies to general NCSs and gives a reference on how to link the solutions proposed and discussed in the following chapters of the thesis. Part of this chapter has appeared in a refereed conference publication [4].

We introduce the concept of system knowledge mining in §5. We define two types of information sources coming from inside and outside monitored infrastructures, namely internal and external information. Internal information concerns all

Figure 1.3: Thesis outline

knowledge we can derive by observing the network traffic. We primarily focus on passive fingerprinting and, particularly, on the development of fingerprinting techniques tailored to the characteristics of NCSs. The *Flow-Fingerprinter* is an example of these techniques as it relies on NCS stable communication patterns. Recognizing devices communicating on the network allows a network intrusion detection system (NIDS) to tune its detection techniques on the basis of the characteristics of the monitored target or directly focus on a specific subset of targets (e.g., monitoring a particularly important subsystem rather than the whole network). Instead, external information consists of configuration files, reference books and user manuals. This information helps to refine the detection mechanisms of the employed NIDS and improve their effectiveness. Part of this chapter has appeared in a refereed conference publication [7].

In §6 we focus on ICSs, we introduce “sequence-based” cyber-attacks and we present a novel NIDS solution that analyzes network message sequences. The *sequence-based intrusion detection system (S-IDS)* uses discrete-time Markov chains to spot anomalies over stable communication patterns and takes advantage of fingerprinting as well as a technique to assess the importance and semantic value of network messages to refine the models and, consequently, to improve the overall detection. This approach has been tested against three real critical infrastructure facilities over two weeks of normal operations. Part of this chapter has appeared in a refereed conference publication [3] and in a refereed workshop publication [2].

In §7 we design and prototype a method to develop an entire NIDS based on automatically acquired system knowledge. The *documentation-based intrusion detection system (D-IDS)* represents a novel approach to intrusion detection that uses fingerprinting techniques to identify devices operating a BAS and search for their information over the Internet (e.g., vendors’ websites) as well as private document repositories. Once the information is acquired, the D-IDS is able to automatically craft effective intrusion detection rules and spot potentially malicious activities over the network. This approach has been tested against two real building automation infrastructures over more than two months of analysis. Part of this chapter has appeared in a refereed conference publication [1].

Finally, we conclude this thesis in §8 by summarizing the main contributions and discussing plausible future research directions.

CHAPTER 2

Networked Control Systems

As discussed in the introduction, *networked control system* (NCS) is an umbrella term that covers a broad set of infrastructures. These infrastructures share the property of managing and monitoring one or more physical processes through communications exchanged over a digital network. NCSs include three fundamental components, namely *sensors*, *actuators*, and *controllers*, physically linked together through a fourth component, the *communication network*. The control-loop, or control process, logically connects the three fundamental components and models the flow of information traveling from the sensors to the actuators through the controllers and their decisioning process. In fact, the operations carried out by the actuators are previously determined by the controllers on the basis of the readings of the sensors (Figure 2.1).

Figure 2.1: NCS control loop

This basic diagram develops and specializes into complex different systems. In this thesis, we focus on two important categories of NCSs, namely *industrial control system* and *building automation system*.

2.1 Industrial Control Systems

ICS is a term generally used to indicate several types of control systems employed in industry (e.g., power plants, electrical grids, water treatment facilities, etc.) to

monitor and control physical processes. The most known and well studied examples of ICSs are “Distributed Control Systems” (DCSs) and “Supervisory Control And Data Acquisition” (SCADA) [30]. DCSs are usually smaller-scale systems whose purpose is to gather sensor data and present it to control engineers. Instead, SCADA systems more commonly extend over large geographical areas and collect data from different locations implementing complex mechanisms of process control. Over the years, these differences have progressively faded away due to new advanced and low-cost technologies, making now the two infrastructures more similar with each other (i.e., SCADA hardware and software are cheaper and commonly used in smaller-scale systems as well).

2.1.1 Architecture Overview

It is possible to logically divide ICS communication networks in two main sub-networks: the *field network* and the *process network*. While the former hosts devices close to the monitored physical processes, the latter conveys collected information to the operators and supervises the correct functioning of the overall control process. According to [35], the field network may include:

- **Sensors** that are components monitoring physical processes with the aim to report any change in their status.
- **Actuators** that are components intervening within the physical processes with the aim of modifying some of their characteristics.
- **Programmable Logic Controllers (PLCs)** that are key components governing the industrial control process. A PLC is in charge of executing a control program and handling digital and analog signals from sensors and towards actuators. Its main characteristic is its robustness as it is usually placed in noisy environments (e.g., electrical interferences, hot/cold temperatures, etc.). PLCs are also meant to work twenty-four hours per day on systems that never stop their operations.
- **Remote Terminal Units (RTUs)**, also known as Remote Telemetry Units, that are electronic control devices operating as interfaces to the physical processes. An RTU is built on top of a micro-processor and implements simple control rules. Compared to a PLC, there is usually no actual logic running within the device. However, due to increasingly cheaper hardware, this situation has changed over time and RTUs have progressively acquired many PLC-like features.

The process network includes all devices used to design and manage industrial processes. According to [36], what follows is a list of common components of a process network:

- **SCADA servers** manage and coordinate actions of PLCs and RTUs and operate as data collectors.

- **Distributed control servers**, also known as DCS servers, share most of their features with SCADA servers. Originally, they were employed to coordinate field network devices only.
- **Human Machine Interfaces (HMIs)** provide user-friendly interfaces to operators and engineers. An HMI allows to easily manage and control simple aspects of the monitored infrastructure. Without accessing SCADA or DCS servers, a HMI allows on-site users to directly access inner information of the monitored physical processes.
- **Historian servers** are database software applications. Historian servers log and arrange process information in a chronological order. Their employment fulfills the requirement of keeping a record for all operations performed in the infrastructure as well as computing statistics.
- **OLE process control (OPC) servers** provide application programming interfaces (APIs) and protocol conversions for different devices of the process network. An OPC server allows field devices, such as PLCs and RTUs, and process devices such as HMIs and historian servers that are based on different technologies to transparently work together through the Object Linking and Embedding (OLE) standard.
- **Engineer's workstations** are those servers or personal computers used to program PLCs. Usually, a workstation directly connects to field devices and allows an operator to update the implemented control algorithms and, consequently, the control process.

Figure 2.2 illustrates an overview of a common ICS.

2.1.2 Protocols

Each ICS uses a well-defined set of open standard or proprietary communication protocols to exchange information and manage devices. All these protocols may rely on different technologies and multiple technologies may be used together within the same infrastructure. It is worth noting that ICSs have relied on serial communications for decades. However, over the years, TCP/IP- and Ethernet-based networking technologies have been increasingly integrated in these infrastructures. For this reason, several industrial protocols have been ported onto the TCP/IP protocol stack.

2.1.2.1 Modbus

Originally developed as a serial protocol by Modicon, Modbus [37, 38] has become a TCP/IP application layer protocol in 2002 as well as a de facto standard for industrial systems [39, 40]. Thanks to its royalty-free nature and the simplicity of its functioning and implementation, Modbus is one of the most widely-used and known ICS protocol. From a technical point of view, Modbus relies on a client/server scheme: on the one side, the client (usually a workstation) asks for a

Figure 2.2: ICS network layout

service and waits for an answer; on the other side, the server (usually a PLC or RTU) executes the requested service and responds to the client (Figure 2.3). Figure 2.4 shows a standard Modbus/TCP frame format. The transaction identifier uniquely identifies a Modbus request and the related response. The protocol ID announces the employed protocol (0 for Modbus TCP) while the length field indicates the amount of bytes left to read in the frame. The unit identifier is used to convey the request to a specific device as multiple devices can communicate behind the same IP address. Finally, the function code defines the requested service as well as its response and allows to interpret any further data in the frame. Modbus services can be of two types, namely data access and diagnostics. The former type includes the most common services such as writes and reads. The latter type includes services that allow clients to check on the status of a server, identify its type, and probe for event logs. Also, Modbus standard allows to define further function codes referring to new services and operations available on a device.

Modbus services work on variables of different kind. Modbus defines two types of variables. Variables consisting of single bits are called containers or coils and they represent inputs or outputs respectively. Instead, variables consisting of 2-bytes words are called registers. Whenever a device cannot fulfill an operation conveyed by a service on a specific variable, exception codes are used. Exception codes are special function codes used by a Modbus server to report an error related

Figure 2.3: Modbus transaction example: the SCADA server requests to read a single coil and gets its value back from the controller

Figure 2.4: Modbus frame

to any operation (these exception codes consist of the same function code used in the related service request plus 128).

2.1.2.2 MMS

The Manufacturing Message Specification [41, 42] is a comprehensive standard defining protocols for all seven layers of the ISO/OSI stack. Nevertheless, its top layers (application, presentation and session) are designed to work on TCP/IP (Figure 2.5). The application protocol defines a set of standard objects, services, and access methods. Objects represent either entire devices or their single elements and characteristics (e.g., a variable, a file, etc.). Every device implementing MMS has a top level object called Virtual Manufacturing Device or VMD that exposes a network-visible address with which other MMS devices can exchange messages. The role of the VMD is to map every element and characteristic of the related device onto a virtual object that is made available on the network. Objects exposed by the VMD can be of different types depending on the characteristics they

Application	Manufacturing Message Specification (MMS) – ISO/IEC 9506 Association Control Service Element (ACSE) – ISO 8649/8650
Presentation	Connection Oriented Presentation – ISO 8822/8823 Abstract Syntax Notation (ASN) – ISO 8824/8825
Session	Connection Oriented Session – ISO 8326/8327
Transport	Transmission Control Protocol (TCP)
Network	Internet Protocol (IP)
Datalink	MAC – Ethernet – ISO 8802-3
Physical	Ethernet

Figure 2.5: MMS over TCP/IP stack

represent. Common object types are: variables, programs, domains, semaphores, files, transactions.

MMS is a client/server protocol with synchronous or asynchronous communication schemes. MMS services allow the client to read or write objects exposed by server’s VMD (Figure 2.6). Other services allow to gather information on

Figure 2.6: MMS transaction example: the SCADA server requests to write “0” to variable “mu” and gets back a confirmation of the operation from the controller

a VMD (e.g., **Information Report**) or modify its content (e.g., **Define Named Variable**). Client can access objects within server VMDs in several ways such as querying “list of variables” (by using variable addresses) or using “accesses by description” (by using variable addresses and descriptions). All services and access methods encode information into byte strings according to the rules defined in the standard.

2.1.2.3 IEC 60870-5-104

IEC 60870 Part 5 describes a set of standards for telecontrol and, specifically, for supervisory control and data acquisition. IEC 60870-5-104¹ [43, 44] is the standard describing protocol access to digital network and TCP/IP support. IEC 60870-5-104 provides balanced/unbalanced communications. In balanced mode both master and slave devices can start a connection while unbalanced mode only allows master devices to initiate the communication. Figure 2.7 shows the template of a IEC 60870-5-104 frame also called IEC 60870-5-104 Application Protocol Data Unit (APDU). The frame includes a header (Application-Layer Protocol Control Infor-

Figure 2.7: IEC 60870-5-104 frame

mation or APCI) and a payload (Application Service Data Unit or ASDU). The first byte is always equal to 0x68 and is used to identify IEC 60870-5-104 frames while the length field indicates the amount of octets left to read in the frame. The next four bytes identifies the control fields whose information is used either to check communication among devices or to interpret data carried in the ASDU. IEC 60870-5-104 defines three types of control field formats (Figure 2.8). The I

Figure 2.8: IEC 60870-5-104 control fields

¹In the remainder of this thesis, we will refer to IEC 60870-5-104 as “IEC104”

format implements “numbered information transfers” and always implies further data in the ASDU (e.g., encoded variables, values, etc.). The S format implements “numbered supervisory functions” and checks the amount of data exchange over a communication. Finally, the U format implements “unnumbered control functions” and allows to test a connection or to start and stop data transfer. In the U format, **act** and **conf** fields are set to 1 when a message is a request (“activation”) or a confirmation respectively.

IEC 60870-5-101 defines available ASDUs used in IEC 60870-5-104. ASDUs encode a number of services employed for different purposes such as information exchange about systems, control processes, parameters, files, etc. (Figure 2.9).

Figure 2.9: IEC 60870-5-104 transaction example: the controller pushes a notification of an updated value of variable 9217 (Information Object Address) to the SCADA server

2.1.2.4 Other Protocols

DNP 3 The Distributed Network Protocol [45] represents a set of communication protocols generally used in process automation but mostly employed by electric companies for communication between electrical substations. DNP3 covers three out of the seven ISO/OSI layers: physical, datalink and application. Its significant employment in the field of electricity is due to its high efficiency (e.g., low bandwidth usage) and reliability. Moreover, it ensures strong data integrity. Application data is organized into data types which in turn belong to different object groups (e.g., **Binary Inputs**, **Analog Outputs**, etc.). Individual data points, or objects within each group, are further defined using the so-called Object Variations. Message exchange relies either on a client/server scheme (even with multiple masters) or peer-to-peer mechanisms.

Profinet Profinet [46] is a standard for industrial ethernet. It derives from Profibus, defining mechanisms for field bus communications, and describes its implementation over TCP/IP. As for DNP3, Profinet defines application layers of the ISO/OSI model as well as physical and datalink layers to be used for real time applications. Devices running Profinet can communicate with each other in an

efficient manner and also operate self-diagnosis and connection diagnosis. Furthermore, Profinet defines the controllers data exchange as well as parameter setting.

2.2 Building Automation Systems

Building automation systems (BASs), also known as building management systems (BMSs), are networked infrastructures controlling different operations and services within a building (or a group of buildings). Among other uses, BASs can monitor and control HVAC (heating, ventilation, and air conditioning), water and energy consumption, lighting, and physical security and safety [16].

2.2.1 Architecture Overview

Compared to ICS networks, BAS networks are often arranged into a hierarchical layout [47] (Figure 2.10). However, the two networks share almost similar components (e.g., DCS servers, engineering workstations, etc.).

Figure 2.10: BAS network layout

At the bottom, sensors and actuators directly connect to the monitored physical processes and send information back and forth to BAS controllers. A **BAS controller**, also known as Building Automation and Control System (BACS), is the analogous of ICS network PLCs and RTUs. In fact, a BACS is in charge of

making decisions on the basis of the information collected by sensors and changing the status of the actuators. Also, BACSs can be either general purpose or designed for specific control applications and their architecture may vary depending on their characteristics. BACSs can directly exchange information with each other but often coordinate their action by communicating with SCADA and DCS servers. These servers provide high-level control procedures and policies. Finally, operators can access and manage BACSs connecting through their workstations and human-machine interfaces (HMIs).

2.2.2 Protocols

As for ICSs, BASs use different communication technologies and protocols. Also in this case, many of these protocols have been ported on TCP/IP to ease operator management and access to building automation infrastructures.

2.2.2.1 BACnet

The “Building Automation and Control Network” (BACnet) protocol [48] allows the communication of BASs for a wide array of different devices and different settings. While exact statistics of the proliferation of BACnet are difficult to come by, already back in 2003 there were more than 28,000 BACnet installations in 82 countries [47]. BACnet can be implemented without having to pay any licensing fees; the detailed standard is available for a nominal fee.

BACnet has a layered protocol architecture, similar to the ISO/OSI model (Figure 2.11). The BACnet protocol has an application layer, containing the actual

BACnet Application Layer						
BACnet Network Layer						
ISO 8802-2		MS/TP	PTP	LonTalk	BVLL	BZLL
ISO 8802-3 (Ethernet)	ARCNET	EIA-486	EIA-232		UDP/IP	ZigBee

Figure 2.11: BACnet stack

application data payload as well as a network layer that abstracts the differences of the network architectures supported by BACnet and implements its own routing protocol (using variable length BACnet addresses). Underneath the network layer, BACnet also specifies how it can be used with different types of data links. The BACnet standard specifies how to use BACnet on top of Ethernet, ARCNET, Master-Slave/Token-Passing (MS/TP), Point-to-Point (PTP) as well as LonTalk. In addition, BACnet also specifies how it can be used with higher-level network protocols like IP and ZigBee.

The BACnet application layer rests on two important core concepts: objects and services (Figure 2.12). A BACS² includes one or more BACnet objects that

²In the remainder of this thesis, we will refer to to any controller implementing BACnet as “BACS”

Figure 2.12: BACnet transaction example: a BACS requests to read the values of two properties of object “analog-value, 7” and gets their values back from the other BACS

are used to represent its functions. Objects are of a specific type like **Analog Input** or **Analog Output**. BACnet supports a wide range of high-level object types like **Calendars**, **Date Value**, or **Credential Data Input** objects. BACnet users and vendors can define “proprietary” objects as well to serve specific functionalities. Object types have different attributes that are called properties (which are extensible for specific purposes). The second core concept of the BACnet application layer are services. While objects describe the different functions that are implemented by BACS, services define how to communicate with the BACS and offer functionality to, for example, read object information from a device. Services have names like **ReadProperty** (to read a property of an object).

2.2.2.2 Other Protocols

KNX KNX [49] is an open standard supported by more than 300 vendors worldwide [50]. The standard defines an architecture based on the seven layers of the ISO/OSI stack and includes a flexible IP tunneling protocol to port the technology on TCP/IP networks. All devices employing KNX share the so-called KNX common kernel that implements networking and object addressing functions. The application layer interface abstracts the description of a device and its characteristics through a predefined set of group objects (GOs) and I/Os. Each GO includes one or more DataPoints defining single variables handled by a specific device.

LonWorks Lonworks [51] is a communication technology aiming to provide high-performance and reliable data exchange for distributed automation systems. This technology and specifically the protocol defined in the standard, called LonTalk, has been introduced and used worldwide to support lighting and HVAC systems. Recently, LonWorks has moved to more general and comprehensive BAS solutions. Some of its technical characteristics differentiates LonTalk from technologies such as BACnet and KNX. First of all, LonTalk is a native peer-to-peer technology where each node communicates independently eliminating any single point of failure. Furthermore, the protocol works through the concept of “event”. An event represents a change in the status or the value of a specific variable. When any of these changes happens, the related information is communicated even without an explicit request (e.g., variables are not polled but their values are automatically shared to devices running LonWorks when a modification occurs). Finally, the standard focuses on making LonWorks totally independent from the ISO/OSI physical layer implementing mechanisms for several different kinds of transmission media.

CHAPTER 3

Intrusion Detection

Starting from Anderson’s seminal work in 1980 [52] and Denning’s publication in 1986 [53], *intrusion detection* has evolved into a number of different approaches. Over the years, these approaches have been classified according to different taxonomies (e.g., [54, 55]). Garcia-Teodoro et al. divide intrusion detection approaches into two main categories: *misuse-based* and *anomaly-based*. Misuse-based IDSs (e.g., signature-based IDSs) identify intrusions thanks to precise descriptions of malicious elements and behaviors (e.g., the execution steps of a malware, the content of a potentially dangerous payload in a network message, etc.). Every misuse-based IDS uses these descriptions as a reference point. Specifically, misuse-based intrusion detection employs a set of heuristics to compare known malicious elements and behaviors with unknown ones, searching for similarities. If similarities are found, the misuse-based IDS reacts by notifying the users of a potentially dangerous situation and conveys the information by sending one or more explanatory messages called alerts. Instead, anomaly-based IDSs exploit a complementary approach. With respect to misuse-based IDSs, anomaly-based IDSs rely on a definition of normal behavior and alert everything that does not match with it. Modeling normal behavior rather than malicious one is usually harder [56] and requires a so-called “learning phase”. In this phase, an anomaly-based IDS observes the monitored infrastructure with the assumption that no malicious activities are performed and extracts key features and characteristics defining infrastructure’s normal operations. A different way to perform this task implies learning about the same normal operations from infrastructure’s specifications (e.g., technical documentation) rather than direct observations. This approach is sometimes referred as an entirely different category of intrusion detection, namely *specification-based* [57].

Both misuse-based and anomaly-based approaches have advantages and disadvantages. From the definitions of “accuracy” and “completeness” proposed by Porras et al. [58], it is generally agreed that misuse-based intrusion detection has high accuracy (few false positives) but low completeness (many false negatives). This is generally due to the fact that known attacks are always detected while unknown attacks are never flagged (unless they share some characteristics in common with known attacks). On the other hand, anomaly-based intrusion detection

has higher completeness as it is designed to foresee novel attacks but this increases the probability to have false positives.

As discussed in the introduction, this thesis leaves misuse-based intrusion detection out of its scope. This is primarily due to the need for security systems that are able to face unforeseen attacks. In what follows we discuss the most important approaches and examples of anomaly-based and specification-based intrusion detection.

3.1 Anomaly-based Intrusion Detection

Anomaly-based intrusion detection draws community's attention in the late '90s. Despite its general concepts being introduced before ([52, 53, 59]), studies such as [60] in 1998 have the merit of formally defining the approach, its advantages and limitations. In this work, Lee et al. present the basic guidelines for the development of anomaly-based IDSs. Specifically, the authors have the merit to define an anomaly-based intrusion detection framework that relies on three key components: an inductive learning engine, a classifier, and a detection engine. The authors test their framework against two different use cases by developing a host-based and a network-based IDSs respectively. The former focuses on detecting unusual behaviors of software applications by analyzing the related system calls. The latter focuses on identifying anomalies within network traffic traces. This work represents a mile stone in the development of anomaly-based intrusion detection and contributes to identify two fundamental research questions of the field, namely the search for effective classification techniques with which to spot anomalies and the choice of features leveraged by those techniques. In what follows, we discuss some examples of anomaly-based intrusion detection according to these two research questions.

3.1.1 Classification techniques

Over the years, several techniques have been proposed to support anomaly-based intrusion detection. Garcia-Teodoro et al. outline the most important techniques in [55] and critically discuss their application in different use cases. The authors take the cue from [61] and organize 12 different techniques into three macro categories: statistical-based, knowledge-based, and machine learning-based (Figure 3.1):

Statistical-based techniques model normal behavior of the monitored system looking at its stochastic properties. Techniques used by Ye et al. [62] belong to this group. In this work, the authors present an anomaly-based IDS relying on multivariate statistical analysis. Specifically, the authors use Hotelling's T^2 distributions to detect outliers in series of records coming from generic information systems (e.g., log files generated by a web server).

Figure 3.1: Anomaly detection classification techniques (from [55])

Knowledge-based techniques include all classification techniques based on deductive reasoning as well as specification-based intrusion detection approaches [63–65]. For example, Wespi et al. describe Unix process behaviors by modeling either the system calls or the generated audit events. Instead, Michael et al. exploit a finite state machine (FSM) to monitor deviations on trails created by the Sun Solaris Basic Security Module.

Machine learning-based techniques encompass all techniques that leverage explicit or implicit models to identify anomalous patterns. Most of the studies about anomaly-based intrusion detection fall in this category. Furthermore, everyone of these studies belong to a different sub-category according to the employed modeling technique. These sub-categories are: bayesian networks (e.g., [66]), Markov models (e.g., [67]), neural networks (e.g., [68, 69]), fuzzy logic (e.g., [70]), genetic algorithms (e.g., [71]), and clustering (e.g., [72], [73]).

The choice of a specific technique among the others always relies on the context in which the anomaly-based IDS is deployed. However, the security community often disagree on the advantages and disadvantages that specific techniques bring to intrusion detection. In this regard, two important works discuss the effectiveness of entire classes of classification techniques; the former comes from Sommer et al.,

the latter from Hadžiosmanović et al.

Sommer et al. discuss the role of machine learning in network anomaly-based intrusion detection [33]. The authors begin by emphasizing the existing gap between the extensive academic research on the topic and the lack of actual deployments in real scenarios. In fact, several challenges make the use of machine learning for intrusion detection troublesome. First, machine-learning algorithms are meant to find similarities. Anomaly-based intrusion detection relies on finding outliers and thus misuses machine-learning techniques. Furthermore, most of the research underestimates network traffic heterogeneity and completely offloads the classification to machine-learning algorithms. Heterogeneous data is difficult to categorize and often drags the algorithms to misclassification. This is due to a “semantic gap” that does not allow a machine to understand underlining similarities and differences in the actual meaning of the analyzed network messages. Without understanding the data behind the analysis, machine-learning and, consequently, anomaly-based intrusion detection may provide meaningless or useless results. Finally, the authors discuss on the cost and the inherent risk of a misclassification in anomaly-based intrusion detection compared to other applications based on machine-learning.

Hadžiosmanovic et al. focus on the use of n-gram analysis for intrusion detection [74]. The authors perform a thorough benchmark of anomaly-based IDSs adopting n-gram analysis techniques and compare their effectiveness with respect to the detection rate and the number of false positives. This study targets binary-protocols and leaves text-based protocols out of its scope. This is due to the leading role these protocols have in infrastructures such as NCSs. Experiments on real data clearly show the limitations of these approaches in real environments with little differences among the n-gram analysis techniques employed by the different IDSs. Also in this case, the authors identify the heterogeneity of the traffic as the greatest obstacle against the use of these techniques and conclude that the basic principle of alerting any irregularity of the data detected by n-gram analysis is rarely effective.

3.1.2 Feature Selection

An anomaly-based IDS focus its analysis on a set of specific characteristics or properties of the monitored infrastructure. For example, host-based intrusion detection commonly analyzes system calls [63, 64, 75], software data structures [76, 77], and application execution flows [78]. On the other hand, network-based intrusion detection generally focuses on network communications and, often, on single network messages. In this last case, IDSs may analyze either message headers or payloads.

The approach proposed in [79] is an example of header analysis. In their work, Haris et al. detect TCP SYN flood attacks by parsing IP and TCP headers data. The authors succeed to differentiate between malicious and valid messages and conclude that IP and TCP headers are the most valuable source of information to detect network floods and thus prevent Denial of Service (DoS) attacks.

Instead, works such as [80–82] are examples of payload-based approaches. In the first work, Wang et al. present a payload-based IDS called PAYL in [80]. PAYL

uses the work of Damashek et al. on n-gram analysis ([83]) to model network message information and detect unusual byte frequency distributions. The authors compute Mahalanobis distance between known and unknown payload models to detect potentially malicious messages. Tests performed on the “1999 DARPA off-line IDS evaluation data set” as well as on live traffic from a university network support the approach especially for what concerns HTTP connections. Related to the work of Wang et al., Bolzoni et al. present POSEIDON (Payl Over Som for Intrusion DetectiON) in [81]. POSEIDON is a two-tier network IDS that improve PAYL by introducing a smart profiling system. In fact, the tool takes advantage of a self-organizing map (SOM) used to organize and classify payload data before comparing known and unknown messages. Finally, Scheirer et al. [82] improve state-of-the-art payload-based approaches by extracting binary code and performing semantic analyses. Performed tests show the effectiveness of this approach against polymorphic shellcodes and support the principle that links the best detection results to solutions that attempt to gain knowledge from captured data.

Different approaches to network intrusion detection rely on network flow analysis. Flow-based IDSs look at the network in terms of open connections and traffic properties (e.g., data bursts). Works such as [84] and [85] are examples of this kind. In the former, Kim et al. detect malicious traffic patterns by FFT (Fast Fourier Transform) and wavelet analysis. In the latter, Barbosa et al. exploit the substantial stability of SCADA traffic to white-list open connections and thus detect malicious devices communicating on the network.

3.2 Specification-based Intrusion Detection

As discussed in the introduction of this chapter, specification-based intrusion detection relies on a set of human-crafted rules precisely describing the correct behavior of a system.

Ko et al. introduce specification-based intrusion detection in [86]. The authors describe their approach towards automated detection of Unix privileged program misuses and suggest to “specify programs’ intended behavior” by modeling their normal execution beforehand. The proposed solution works through the definition of a Program Policy Specification Language aiming to formally define programs’ operations by simple predicate logic and regular expressions. Later works such as [87] resume and improve the proposed ideas. Ko et al. improve their introductory research in [88] by defining a formal framework used to define and detail security-relevant behavior of Unix programs. In [89], the framework gets integrated into a comprehensive specification-based IDS, called SHIM. SHIM merges several different detection approaches that apply to both network communications and operating system activities. Its use, together with machine learning techniques has proven to be an effective solution towards the development of automated intrusion response strategies [90].

From the previous works, Sekar et al. continue investigating the research field by proposing complementary approaches in [91] (based on the use of the Auditing Specification Language), [92] and [57]. Particularly, this last work presents a hybrid

approach aiming to increase the information of a specification-based IDS with the use of anomaly-based intrusion detection techniques. The authors use Extended Finite State Automata (EFSA) to model and detail network protocols' behaviors. Then, they refine the set of monitored features via a learning phase exploiting statistical analyses made on the traffic traces.

Over the years, researchers developed custom specification-based intrusion detection techniques to fit different infrastructures such as mobile ad hoc networks [93–98] and WLANs [99]. Furthermore, specification-based IDSs were developed for specific use cases both for network-based (e.g., VoIP technologies [100], carrier Ethernet [101]) and host-based security (e.g., kernel dynamic data structures [102], mobile operating systems [103]).

It is worth noting that specification-based intrusion detection has also gained a main role in NCSs. Works such as [104], [105], and [106] present specification-based IDSs for Modbus, Zigbee and DNP3 respectively. Ultimately, Berthier et al. [107] and Hadeli et al. [108] take this research a step forward. Berthier et al. model not just the employed protocols (in this case C12.22) but smart meter security constraints and policies as well. This research shows the feasibility of modeling high-level properties of the infrastructures and the effectiveness of its results. Hadeli et al. take advantage from Substation Configuration Description (SCD) files to extract information related to electrical substation data exchange. The authors investigate the possibility to use this information to define different security properties of the monitored systems and, consequently, to enhance security tools such as firewalls and IDSs/IPs.

3.3 Summary

This thesis focuses on two open issues of network anomaly-based intrusion detection and their application in the context of NCSs. These two open issues are: the monitoring of network messages interrelationships and specification-based approach automation.

The former issue concerns the possibility to attack an NCS infrastructure by using a crafted sequence of individually harmless messages. The so-called “sequence attacks” target a vulnerability that state-of-the-art IDSs are not able to tackle. This is mostly due to a lack of techniques that investigate inter-message correlations. Few works in literature propose solutions that are able to detect anomalies within sequences of messages. However, *these works miss a comprehensive assessment on the feasibility and the effectiveness of monitoring NCS communications without any information related to the data carried within.* In §6, we show that these approaches usually generate many false positives and thus we discuss a suitable way to overcome the problem.

The latter issue concerns the design and development of specification-based IDSs. State-of-the-art research on specification-based intrusion detection assumes that protocol and system documentation is readily available when designing and configuring the IDS. Few thoughts are spent on where and how to retrieve this information, especially not in an automated way. Moreover, all the aforementioned

works do not explore *the possibility of autonomously and automatically extracting the information needed to build the related IDSs from documentation*, instead relying on human evaluation and translation to rules. This possibility would allow to efficiently apply specification-based approaches to entire infrastructures. In §7, we focus on this key aspect of specification-based intrusion detection with the aim of making specification-based IDS development more time-effective and accurate and, consequently, also more likely to be used in real environments.

Part II

Enhancing Intrusion Detection in NCS

CHAPTER 4

Requirements & Approach

In the introduction of this thesis we discussed three main issues that make the protection of NCSs challenging. These three open problems concern the presence of increasingly sophisticated cyber-attacks, the heterogeneity of employed (and thus monitored) devices, and the need for non-interfering (passively-monitoring) security solutions. Such issues come from an analysis of state-of-the-art research on NCS security. However, a comprehensive understanding of problems and shortcomings related to NCS security also requires a direct feedback from NCS stakeholders. For this reason, in 2013, we created a survey for experts within academia, vendors, and infrastructure operators to assess the status of employed NCSs around the world and to evaluate directions for improving their security. The survey was made available online¹ and participants could decide to compile it either anonymously or not. Non-anonymous participants were asked to follow up the survey with a face-to-face in-depth interview.

Due to the limited amount of responses (eighteen by April 2014), the results were not sufficient to pursue a quantitative analysis on NCS security. However, we compensated for the problem by focusing on a qualitative analysis mainly based on the in-depth interviews. These discussions allowed us to confirm the three open problems outlined in the introduction of this thesis. Furthermore, interviewed experts helped us to refine the research guidelines that eventually led our study to the development of custom anomaly-based IDSs and to their integration with techniques designed to automatically acquire system knowledge.

Generally speaking, the interviews showed that the increasing number of security incidents and threats involving NCSs derives from both technological and organizational weaknesses. Despite its importance, security is not always considered a major concern (e.g., with respect to service reliability or compliance with national directives) and the risk of being targeted by cyber-attacks is often underestimated.

Particularly interesting is the analysis of relevant limitations against securing NCSs. Most of the interviewees rate economical and time investments as the two major obstacles against improving the present situation. Costs always play a main

¹On Google Docs. Please refer to [10]

role in limiting the amount of resources available for NCS security. On the one hand, small organizations employ NCSs without being able to supply adequate protection. On the other hand, big corporations and critical infrastructures can usually afford to invest in security but such investment is poorly driven by a lack of information sharing between security and financial departments. Time constraints represent another crucial limitation to security and experts consider this limitation almost as important as cost constraints. Interviewees particularly emphasize the danger behind those situations in which the time available to investigate all potential security issues of an NCS is limited or even insufficient.

Furthermore, the interviews confirm and strengthen the issues discussed in the introduction of the thesis. First, a significant percentage of the experts affirm that, instead of tailored solutions, standard IT security solutions are commonly used to protect NCSs. Digging deeper on this issue shows different reasons why using standard IT security solutions may not be as effective as a custom-made solution. First, NCS security requirements can differ from standard IT infrastructures. A meaningful example relates to the so-called CIA triad. Confidentiality-Integrity-Availability (CIA) requirements usually apply to IT systems in this order of importance. For critical infrastructures (and therefore ICSs monitoring critical infrastructures) stakeholders agree on subverting this ranking and primarily support availability and integrity [30]. This is mainly due to the dangerous consequences that a lack of these two properties can cause to infrastructures such as dikes or nuclear power plants. For this reason, security solutions applied to ICSs working on critical infrastructure should always reflect this new scale of importance of the CIA triad. Second, NCSs often use custom technology. Despite the shift toward IT standards, NCSs maintain a large set of specialized (and sometimes proprietary) systems and communication protocols generally running key components of an infrastructure. Security solutions should especially target those components and, therefore, being able to deal with the employed specialized technology. Finally, NCS devices show, on average, limitations in their computational power and in their available resources with respect to devices employed in standard IT. Most NCS devices do not have the necessary capability to run standard IT security or even coexist with it. In fact, some of these components can be real-time systems playing critical roles in the infrastructure. In situations like this, even the slightest delay in their operations caused by additional communications or activities could compromise the entire infrastructure [109, 110]. For this reason, security solutions should never interfere with NCS operations.

4.1 Research Guidelines

The end of each interview focused on critically reviewing the perspectives of every stakeholder with the goal of deriving general recommendations for NCS security. This work later became a list of plausible research guidelines that academic researchers, as well as industry experts, should consider to improve NCS protection. What follows includes the main outcomes coming from the interviews.

From minimum to no interference Security solutions should not interfere with NCS operations. As discussed in the introduction, this issue was also raised multiple times during the interviews. In this regard, we believe that NIDSs represent the best trade-off between a thorough and comprehensive view of the NCS and a minimum impact on the monitored activities. Despite the value and the scientific interest of other solutions discussed in literature [111], we argue that a NIDS has the highest chance to be accepted by stakeholders and be actually deployed in real environments. This assumption relies on two main aspects. First, a NIDS passively gathers information over the network without injecting traffic and, consequently, remains invisible to the monitored infrastructure. Second, a NIDS is usually running on a standalone machine with no other purpose than performing detection, thus not interfering with any other device of the network. Differently, active solutions, such as a firewall, or even passive solutions, such as a host-based intrusion detection system (HIDS), usually perform a different, more intrusive action within an NCS and would not likely be accepted as easy by stakeholders. For example, a firewall can become a single-point-of-failure or prevent availability of key infrastructure components when not properly configured. Likewise, a HIDS can hamper devices' performance or encounter strong resistance from vendors (e.g., deploying software on a proprietary device may be against its use policy).

Reduced user effort Security solutions should decrease the involvement and the effort of the end user with respect to current solutions. This guideline plays a role in two different issues. On the one hand, we need to reduce the effort that a end user spends to deploy a NIDS. This means that the configuration of our security solution should be automated to a fairly high degree. Moreover, the time needed to setup the NIDS should be the least possible, thus avoiding users to wait unrealistic time before the system is up and running. On the other hand, we aim to reduce the effort that users spend dealing with NIDS alerts. Every time a NIDS raises an alert users must understand the reason why that alert was triggered and coherently reach to address the situation. Too many erroneous alerts (also known as false positives) make it unfeasible for the end users to properly respond to every situation. Additionally, unclear or ambiguous descriptions in the alerts make it difficult for the users to understand what the situation is about. Our security solution works toward decreasing the number of false positives to a minimum and providing as much context information as possible to the users to let them timely and effectively react to solve the related issues.

Improved in-depth analysis Security solutions should be tailored to NCS technologies. A NIDS should be able to parse the employed communication protocols and analyze, as well as interpret, the payload of the messages exchanged by the different NCS components. This capability allows to use sophisticated detection techniques and, consequently, to deal with targeted attacks such as APTs. As already discussed, existing literature confirms that a comprehensive context knowledge directly affects attack detection in a positive manner [33]. Moreover, interviewees agreed on the intrinsic difference between data carried by NCSs compared to standard IT infrastructures. The more a NIDS is able to contextualize

messages captured from the network and apply semantics to the carried information the better chances it has to correctly identify malicious behaviors and, thus, detect attacks.

Additionally, the survey suggests some specific and distinguishing characteristics of NCSs that should be considered to design new security solutions accordingly to the aforementioned guidelines.

Stable properties Compared to standard IT networks, most NCSs share several properties that rarely change over time. First of all, the behavior of an NCS depends on the implemented control algorithms. These control algorithms define the purpose of the infrastructure and are not prone to substantial changes over time. As a result, network communications among NCS components do not change as they reflect the exchange of information needed to ensure the correct implementation and execution of the control algorithms [112]. When it comes to network anomaly-based intrusion detection, stable communications are a necessary condition to lower the number of false positives. Furthermore, subverting the control algorithms represents the most attractive target for a skilled-attacker who wants to subvert the underlying physical processes [32]. For this reason, sophisticated attacks such as APTs are likely to cause a “perturbation” in network communications and, consequently, to make a NIDS raising an alert. Other stable properties of NCSs further support the use of anomaly-based intrusion detection techniques. Among them, stable network topologies and (almost) fixed sets of devices generally help to lower down the number of false positives.

Auxiliary information We have already emphasized that the development of effective security solutions often benefits from a comprehensive understanding of the monitored systems. Specifically, NCS security can take advantage of a lot of information about the infrastructures and their control processes. This assumption relies on two main aspects. First, NCS networks do not usually employ encryption [113]. This is mainly due not to undermine system performance and to let the operators promptly intervene in case of a problem [114]. The absence of encryption makes all network communications open to any detailed analysis. Second, comprehensive information about NCSs often come in the form of configuration files, reference books and user manuals [108]. This documentation can be public (e.g., vendors providing online references for the use of their products) or private (e.g., operators keeping a record of infrastructure components and implementations for maintenance). A NIDS can use this information to better understand the traffic observed in the network and to refine its detection mechanisms. Moreover, this documentation usually exploits standard templates (e.g., “Substation Configuration Language” files for ICSs, “Protocol Implementation Conformance Statement” for BASs, CAN matrixes and corresponding documentation for in-vehicle networks). A standard template allows to automatically parse the documents and extract valuable information that can enhance NIDS detection models. In conclusion, the coordinated gathering of information from heterogeneous sources such as network

traffic and documentation helps to improve NIDS capabilities and, ultimately, its effectiveness in detecting intrusions.

In what follows we include the proposed concepts and guidelines into a framework that combines the acquisition of system knowledge to anomaly-based intrusion detection.

4.2 Framework Building Blocks

Figure 4.1 illustrates the framework and its building blocks:

- **Network Traffic** represents NCS data in the form of digital communications
- **Documentation** represents NCS information in human-readable form (e.g., textual digital documents)
- **Parsing** groups all techniques aiming to extract information from network traffic
- **Document Retrieval** groups all techniques aiming to find valuable document (e.g., descriptions of infrastructure characteristics) within NCS documentation
- **Document Processing** groups all techniques aiming to extract information from a subset of chosen documents
- **Information analysis & aggregation** links and evaluate together heterogeneous information coming from both digital communications and documentation
- **Network Intrusion Detection System** represents all IDSs (general-purpose or NCS-specific) deployed within NCS networks that can benefit from acquired system knowledge
- **Rule Generation, Alert Confirmation & Evaluation, and Alert Labeling** represent three possible use of acquired system knowledge
- **Enhanced Intrusion Detection** represents the bond between system knowledge and network security as well as any solution fulfilling this paradigm

From bottom to top, each building block implements a specific task summarized in the title and relies on the notions and techniques of the blocks below. As discussed in the introduction, we consider two sources of information, such as the network traffic and the documentation related to the monitored infrastructure:

Network Traffic refers to the digital data flowing within NCS networks. This work focuses on network communications and protocols based on TCP/IP and leaves serial communications out of its scope. The reason behind this decision relies on the increasing involvement of packet switching networks for interconnecting multiple NCS devices. Even if serial communication is still employed (especially for the "last mile" toward field devices), most information is carried by standard

Figure 4.1: Approach building blocks

IT protocols such as IP and TCP. Our analysis targets both network packets’ content and their meta-data (e.g., time of arrival, relative position within a sequence of packets, etc.). The capture of the network packets relies on the work of a dedicated monitoring machine physically connected to a router or a switch through a Switched Port Analyzer (SPAN). This setup ensures that the router or switch copies all handled traffic to SPAN port and lets the monitoring machine passively read the data. Our work does not exclude the presence of multiple machines in charge of capturing network traffic at different locations within the same NCS. In fact, this usually allows better coverage of NCS communications with a little additional cost in terms of tuning the related analysis (e.g., avoiding duplicate messages, correlating packets, ensuring consistent timestamps, etc.). Finally, we assume that the rate of packets lost by the monitoring machine due to hardware limitations or connection failures remains limited and, in any case, does not involve messages of critical importance. This assumption holds in all the performed tests we discuss in the following chapters.

Documentation (Internal & External) refers to a heterogeneous set of digital documents describing several different aspects of the monitored infrastructures. We differentiate between “internal” and “external” documentation. The former identifies documents owned by NCS operators and thus referring to a single specific deployment. These documents are stored locally and are not usually publicly available. The latter includes instead a set of digital documents that are available online and describe characteristics or properties that can be common to two or more NCSs. In this regards, a document outlining functional specifications of a particular NCS device or also a functional description of a particular physical process are examples of external documentation. Both internal and external documentation can be structured or unstructured depending on how they format the carried information. Structured documents organize data into structures such as tables or “tag trees” (e.g., XML). Unstructured documents present the information as free text. Most commonly, documents include both sections of structured

and unstructured data (e.g., semi-structured documentation).

Laying above the network traffic and the documentation building blocks, we define three operational blocks aiming to extract useful data from the sources of information below. These three blocks are: Parsing, Document Retrieval and Document Processing.

Parsing represents all activities and tools devoted to analyzing and interpreting network traffic. We assume that the digital information we monitor is unencrypted. Encrypted traffic would require this building block to own extra information (e.g., a secret key) to fulfill its goal. However, as already discussed, the use of encryption is not common for NCS communications. Most of the work presented in this thesis assumes the use of parsing techniques that implement all structures and semantics used by NCS communication protocols. In case of open standards (e.g., Modbus) this possibility comes from the availability of protocol documentation. Differently, the presence of proprietary protocols implies a process of reverse engineering. If protocol structures and semantics are known, the Parsing building block implements techniques for in-depth analysis meaning that all protocol fields that are important to secure the monitored NCS are correctly interpreted. Moreover, the discussed in-depth analysis involves a stateful scan of network traffic. In fact, every network message is analyzed on its own as well as with respect to the others. Finally, we assume that parsing and interpretation of network messages happen on the fly and do not undermine the performance of the capturing. This allows any tool based on network monitoring to work on live functional NCSs.

Document Retrieval refers to techniques and tools developed to search and find valuable documents. In the case of internal documentation we consider the problem of identifying interesting sources of information within a limited set of documents. These are usually stored by NCS operators for internal use only. In the case of external documentation, we consider instead the issue of finding interesting sources of information within a much larger set of documents (e.g., the World Wide Web). Document retrieval techniques may change significantly between internal and external documentation. In fact, dealing with the former usually eases the task of selecting the right documents either because of the low number of available files (e.g., fewer documents allow to make a more in-depth analysis in the same amount of time) or because of their highly-likely relevance (a document owned by NCS operators has higher chances to be related to and useful for the monitored NCS compared to a document found online). Instead, external documentation retrieval strongly relies on the capability of primarily identifying key words with which to search for documents. As for standard online researches, this task can take advantage of web crawlers and common search engines (e.g., Google).

As we stressed the importance of key words, we argue that the Document Retrieval building block partially depends on Parsing building block. In fact, any document retrieval tool uses the outcome of the parsing to get a hint on what to search for. A document relates to the monitored NCS when the information extracted from the network matches, to some extent, the information of the document

itself (e.g., the title of a document matches the name of a device in the network). In our work, parsing traffic and recognizing devices with fingerprinting is the most significant example of this situation. Once we have this information, we begin to search for documents describing identified devices by using their model and brand names. A further example involves the use of human readable (e.g., ASCII) information found in network traffic to identify documents describing NCS control operations (e.g., in a BAS, we may be able to capture and interpret log entries related to just one specific sub-system such as water distribution and thus select just water-related documentation from operators' repositories).

Document Processing includes all procedures and tools aiming to extract valuable information from the retrieved documentation. Techniques used in this block depend on the nature of the involved documents. Unstructured documentation interpretation can take advantage of natural language processing (NLP) [115]. Differently, when it comes to structured documents, we need some more customized techniques capable of identifying specific templates, such as tables, and accordingly extracting the involved data. In all cases, the process of acquiring knowledge from documentation starts within this building block and completes in the Information Analysis & Aggregation one.

Document processing mainly relies on document retrieval as the actual processing is generally focused on just some selected documents. Nevertheless, it can happen that the document processing starts even if the document retrieval fails. Situations of this kind are limited to the case of internal documentation. In fact, if the set of documents owned by NCS operators is small, with no information coming from the document retrieval the only possibility to gain some knowledge relies on parsing the whole set and attempt to extract information exploiting a more detailed analysis.

Furthermore, document processing may rely on network traffic parsing. This is due to the need of interpreting data coming from the documents with respect to the elements found in the network. In fact, documents may include conditional information (e.g., a device behaves either in one way or another depending on an explicit choice of the operators). This condition can be resolved by observing the network (e.g., with respect to the previous example, network traffic coming from the device shows a distinctive behavior and allows document processing to make a decision accordingly). In cases like this, document processing depends on the network traffic and thus its parsing.

At this point, information coming from both sources of information comes together to generate system knowledge. **Information analysis & aggregation** represents the synthesis of the observations made from two different perspectives of the same NCS. In fact, this building block includes all techniques aiming to model and evaluate the gathered data and make it available to improve the effectiveness of a NIDS. This operation resolves into three different activities, namely Rule Generation, Alert Confirmation & Evaluation, and Alert Labeling.

Rule Generation uses the knowledge coming from the gathered information to create intrusion detection checks. This building block represents all those techniques aiming to extract new customized rules that can be included in a NIDS. For example, NCSs device can be subjected to some specific constraints described in their related documentation (e.g., device A can not communicate with device B, device A can not convey specific information, etc.). If violating these constraints means undermining the security of the device or, in any case, modifying the overall functioning of the depending NCS, the Rule Generation takes charge of building a further intrusion detection check that ensures the fulfillment of these constraints.

Alert Confirmation & Evaluation uses the knowledge coming from sources of information to review or improve the outcomes of a NIDS. As already discussed, NIDSs often suffer from too many false positives. This issue often derives from a lack of context information. As specification-based intrusion detection has lowered the amount of false positives thanks to checks directly derived by system information, Alert Confirmation & Evaluation building block can use the acquired knowledge to evaluate the importance of an alert coming from the NIDS and, in some cases, to completely drop the alert before it reaches the operators.

Alert Labeling uses the knowledge obtained from network traffic and documentation to enhance the content of an alert coming from a NIDS. Conceptually, this building block works similarly to the Alert Confirmation & Evaluation one. In fact, techniques included within Alert Labeling take an alert and make an evaluation. However, the aim of this evaluation is to improve the description and the semantics of the alert and thus to help NCS operators to understand it and take action consequently. As already discussed, cryptic or ambiguous alerts represent a significant practical limitation against the use of state-of-the-art NIDSs in real deployments.

Finally, the **Enhanced Intrusion Detection** building block represents the integration between any **Network Intrusion Detection System** and the three activities of rule generation, alert confirmation and evaluation, and alert labeling. This integration can or cannot be effortless depending on the nature of the involved network IDS. Open-source anomaly-based IDSs such as Snort [116] and Bro [117] are easy to integrate. In this cases, creating new checks means adding new rules to a dataset already employed by the network IDS. Additionally, the alerts can be intercepted before delivery as well as be suppressed or enhanced. Integration to proprietary network intrusion detection tools can be harder, especially for what concerns modifications to already-in-place detection techniques (therefore, the use of the Rule Generation).

In what follows, we discuss our approaches for acquiring system knowledge and implementing network security with respect to proposed framework and, particularly, in relation to the involved building blocks.

CHAPTER 5

Mining System Knowledge

In this chapter we present the two sources of information introduced in §4 and we describe feasible techniques to gain system knowledge. In §5.1 we discuss about information coming from network traffic. We focus on fingerprinting as this activity generally paves the way to analyze any monitored NCS. Instead, in §5.2 we focus on documentation and we investigate techniques that extract knowledge from this documentation.

5.1 Network Traffic & Fingerprinting

This section discusses concepts and techniques related to three framework building blocks, namely **Network Traffic**, **Parsing**, and **Information Analysis & Aggregation** (Figure 5.1).

Figure 5.1: Network Traffic Information Mining

5.1.1 Overview

By design, most NCS components share information over a digital network. This makes the network traffic a valuable source of information. For this reason, we consider *fingerprinting* a first important step towards acquiring system knowledge.

Fingerprinting defines a set of tasks aiming to describe several aspects of devices working in a computer network. Fingerprinting techniques play a major role in many activities related to information security (network vulnerability identification, attack analysis, risk assessments, etc.) and represents a basic approach for dealing with unknown IT environments. Device fingerprinting is the most known and well-studied kind of fingerprinting. The aim of device fingerprinting is to remotely recognize device hardware and, especially, software communicating over a digital network. Usually, this goal is achieved by exploiting compact descriptions of device features, called fingerprints or signatures, that unequivocally identify a specific hardware or software. When a fingerprint is known to belong to a specific kind of device, a fingerprinting tool (also called fingerprinter) checks if unknown devices share the same fingerprint, based on the information found in the network. If there is a match, the unknown device is “fingerprinted” and, thus, recognized to be of the same type.

Device fingerprinting can be either active or passive. Active fingerprinting implies tools that actively query a device to obtain information needed to compare and match fingerprints. Passive fingerprinting tries to acquire the same information in a less intrusive way by observing network communication only. Usually, active fingerprinting exceeds passive fingerprinting in success rate and accuracy. In fact, active fingerprinting always attempts to collect all the information needed to form a fingerprint while passive fingerprinting often deals with incomplete information available in a network at a specific moment in time. However, performing active fingerprinting is not always possible. Probing the network in the attempt to collect valuable information is a noisy operation that can easily interfere with on-going network communications.

Device fingerprinting always includes three basic activities: information gathering, storing data and creating a dataset of fingerprints, and finally updating the dataset. Information gathering concerns data employed in the analysis. This activity changes depending on the type of collected data. The most common fingerprinting technique, namely TCP/IP stack fingerprinting, relies on the inspection of TCP and IP headers [118] and, specifically, of the following fields: initial packet size, initial TTL, windows size, max segment size, windows scaling value, “don’t fragment” flag, “sackOK” flag and “nop” flag. When aggregated, these elements form a 67-bits signature that unequivocally identifies a type of device. Besides TCP/IP stack fingerprinting, other approaches use information included in different protocols at all layers of the ISO/OSI stack [119], from the application layer (protocol-based fingerprinting) to the physical layer (signal-based fingerprinting). Finally, fingerprinting may rely on meta-information such as message timing and arrangements [120, 121].

The act of storing data and creating a dataset of fingerprints makes the gathered information available for analysis. This information can be stored in different

ways. Some of these are meant to be very efficient while others focus on sophisticated data structures used for in-depth analysis. Efficiency may be important when device fingerprinting is used in conjunction with other activities (e.g., in support to intrusion detection).

Finally, updating datasets comes from the use of new hardware and software every day. For example, an old set of fingerprints may be ineffective to recognize unknown devices in a network. However, adding new fingerprints to datasets can be challenging. To be valid, any new fingerprint must go through extensive testing and must not be incoherent with old fingerprints already in the datasets. Without caution, updated datasets can easily become incongruous and lead the full fingerprinting to meaningless results.

In what follows we study fingerprinting challenges and opportunities within NCSs. To achieve this, we analyze state-of-the-art fingerprinting techniques and we investigate basic fingerprinting operations. Then, we use this analysis to define a reference model for fingerprinting tools and we discuss each model building block in the context of NCS. Finally, we evaluate the reference model by instantiating its building blocks within an NCS fingerprinting tool we call Flow Fingerprinter.

5.1.2 State of the Art

We divide fingerprinting-related state of the art into two main parts. First, we discuss all possible information sources that can be leveraged to recognize devices communicating in a network. Second, we present an overview of fingerprinting techniques. We conclude this section by discussing how fingerprinting may integrate further analyses and activities such as intrusion detection.

5.1.2.1 Information sources

Fingerprinting can have multiple targets, from physical components (e.g., simple electrical components, complex hardware systems) to software (e.g., operating systems, single applications). Different targets require different information and, thus, each fingerprinter should accordingly focus on different data sources.

Most fingerprinters focus on TCP/IP stack fingerprinting. This approach relies on storing data related to mechanisms implemented by TCP and IP (e.g., connection negotiation). Early works on TCP/IP stack fingerprinting define the basic principles of this technique. For example, Paxson et al. present in [122] a tool for TCP implementation analysis called Tcpanaly. Tcpanaly passively inspects TCP segments to spot unique characteristics derived from different TCP/IP stack implementations. Although outdated, this work clearly shows how many distinctive traits there are among TCP segments sent by devices running different operating systems. Taleck et al. further develop the same technique in [123]. The authors increase the level of confidence of TCP/IP 67-bits signatures by analyzing a more comprehensive set of characteristics related to TCP implementations.

Besides TCP/IP stack fingerprinting, studies in literature evaluate application-level fingerprinting and, thus, software recognition. In this regard, Haffner et al. exploit HTTP, FTP, and SMTP headers [124] while Moore et al. go beyond pro-

toocol implementations and investigate the possibility to build highly-reliable classification techniques based on packet payloads [125].

At physical layer, signal-based fingerprinting techniques take advantage of hardware properties to define distinguishable signatures. In [126], Gerdes et al. show that Ethernet devices can be unequivocally identified and tracked using just 25 Ethernet frames. This is achieved by analyzing variations in analog signals caused by hardware or manufacturing inconsistencies. An alternative approach for signal-based fingerprinting consists of recording hardware-related small-time deviations in processing communications, called clock skews. Kohno et al. exploit this approach in [127] and show how fingerprinters can use information included within TCP headers to estimate clock skews and, thus, fingerprint physical devices.

Several studies in literature focus on available meta-information such as network packet timing. For example, Veyssset et al. show the effectiveness of recognizing operating systems through analyses on message response time [120]. Based on this idea, the authors develop a tool, called RING, able to measure delays in TCP segment retransmissions.

Finally, some fingerprinting approaches rely on port scanning. Moore et al. analyze the use that well-known services do of standard network ports to recognize operating systems and applications [128]. However, approaches of this type face two common criticisms: the difficulty to check devices in presence of NATs (or further network components) and the use of port randomization techniques.

5.1.2.2 Fingerprinting techniques

The simplest fingerprinting techniques rely on an exact match between a stored known fingerprint and a new one derived from information gathered in the network. As even identical devices (same hardware and software) may experience small differences (e.g., software version), more advanced tool such as Nmap [129] use sophisticated techniques to compare fingerprints. Most of them fall in the field of machine learning. Different machine learning techniques have been proposed to refine a fingerprinting tool (e.g. Bayesian networks, neural networks, etc.). In [130], Beverly et al. leverage probabilistic learning for TCP/IP stack fingerprinting and develop a Naïve Bayesian classifier to passively infer host operating systems from packet headers. The approach relies on observations made over TTL Window SYN packet sizes, and “do not fragment” bits as well as the use of Bayesian matching.

For what concerns application-level fingerprinting, Haffner et al. use in [124] three different learning mechanisms to fingerprint TCP and UDP flows, namely Bayesian inference, clustering and entropy analysis. The first mechanism models a discrete distribution for each analyzed feature. The second takes advantage of the AdaBoost algorithm [131] to incrementally refine a weighted combination of weak classifiers (e.g. Bayesian classifiers). Finally, the Regularized Maximum Entropy algorithm looks for a distribution in the training set that has the maximum entropy and, at the same time, satisfies some specific constraints. The comparison process between known and unknown flows outputs a true or false value meaning that a communication flow either belongs to a fingerprinted application or is completely unrelated to it.

5.1.2.3 Applications and Considerations

Fingerprinting can enhance or complement several activities (e.g., network vulnerability identification, attack analysis). In this thesis we use device fingerprinting to gain knowledge of the monitored infrastructure and, eventually, to support intrusion detection.

Several works in literature discuss the use of fingerprinting with intrusion detection, both for misuse-based and anomaly-based IDSs. In the former case, fingerprints often help to identify devices that are not allowed to access a network. In the latter case, fingerprinting may become an additional check to help evaluating IDS alerts (e.g., discard alert related to uninteresting devices). Taleck et al. investigate the parallel use of anomaly-based intrusion detection and TCP/IP stack fingerprinting. In their work, the authors show that IDS sensors benefit from additional knowledge about the network stack implementation of target hosts to increase detection accuracy.

Different kinds of anomaly-based IDSs were proven to be more effective if coordinated with fingerprinting techniques. Jackson et al. show the positive effect of using signal-level fingerprinting to refine attack detection and analysis in [132]. Esquivel et al. use fingerprinting techniques to mitigate spamming and, specifically, implement a technique for spam detection filtering that exploits light-weight fingerprints of spam senders.

All these approaches could directly leverage fingerprinting for intrusion detection thanks to tools and techniques developed over the years for standard IT. To improve intrusion detection in the context of NCS we need to understand what challenges and limitations this context poses to fingerprinting. For this reason, we analyze aforementioned fingerprinting techniques in terms of basic operations and building blocks. Then, we link these building blocks together to build a reference architecture for fingerprinting tools. Finally, we discuss each building block in the context of NCS.

5.1.3 Reference Model

Despite the use of different network data and the aim to different targets, all fingerprinting tools share several common tasks. Specifically, it is possible to arrange these tasks into logical building blocks modeling the basic principles of fingerprinting. Each building block receives some data in input and elaborates on the this information. Eventually, each block forwards its result to another or feedback the information. Figure 5.2 illustrates the resulted reference model and its contextualization within the framework discussed in §4.2.

What follows describes every building block in detail and includes some references to mechanisms included in common fingerprinting tools such as Nmap [129], P0f [133], XProbe2++ [134], SinFP [135], etc.

Data Sources Every fingerprinting tool depends on specific data sources. The choice of using one or multiple data sources depends on the complexity of the fingerprinting tool as well as its overall target. As already discussed, information

Figure 5.2: Structural model of a fingerprinting tool

coming from TCP and IP is a common data source. Several fingerprinters go beyond the TCP/IP 67-bits signature and identify more comprehensive sets of characteristics [122, 123]. Furthermore, numerous tools exploit data belonging to different layers of the ISO/OSI stack [124]. Finally, completely different approaches take advantage of information such as network topologies, time patterns, or open network ports [128].

Gathering implements all techniques used by a fingerprinter to collect data. It relies directly on available data sources and directly captures valuable data from the network. Implementations of the Gathering building block differ according to the active or passive nature of the fingerprinter. Active fingerprinting tools directly query target systems to gather information needed to describe their fingerprints. Instead, passive tools try to acquire the same information in a less intrusive way by simply waiting for valuable data to appear in the network. When it comes to active fingerprinting, Gathering takes charge of probing the network. This means creating suitable network packets, sending them and handling the related responses (e.g., such as in the case of Nmap and XProbe2++). Collected information, once labeled and organized, will form the dataset of the fingerprinter. It is worth noting that not all network traffic contains valuable data. For this reason, Gathering behaves as a filter dropping useless or malformed network messages.

Model Generation organizes and stores data captured by Gathering building block. Every fingerprinter needs some basic knowledge to rely on. For this reason, over the training phase, a user links the model (also, “Decision Model”) derived by the captured data to a textual description of the system it belongs to. However, sophisticated fingerprinting tools are able to automatically refine their datasets (e.g., thanks to machine learning techniques [124, 130]). Finally, Model Generation ensures that every model is consistent and unambiguous and avoids overlaps in the dataset.

Decision Models are the outputs of Model Generation building block. These decision models represent the knowledge of a fingerprinter. Every fingerprinter uses

its knowledge to recognize unknown systems. In the simplest case, decision models are signatures stored in text files (such as for Nmap and P0f). More sophisticated fingerprinters use complex decision models or even multiple ones at the same time. For example, XProbe2++ use different models depending on available network messages [134] while SinFP exploits two different datasets of active and passive signatures at the same time [135].

Pre-processing organizes and arranges captured data for comparison with the information stored in the decision models. Most of the times, Pre-processing leverages the same techniques used by Model Generation building block and, thus, create temporary or sometimes incomplete Decision Models. However, a few tools such as XProbe2++ add some extra data manipulation to ease and speed up the classification process.

Classification implements the comparison between collected data of unknown systems and Decision Models. Classification can employ different comparison techniques. For example, P0f performs a simple matching among signatures. Instead, tools such as Nmap and XProbe2++ leverage smarter comparison methods such as fuzzy signature matching. Finally, other tools such as XProbe2++ make their comparison more efficient by using decision trees. Sophisticated fingerprinters enhance Classification building blocks with mechanisms aiming to neutralize fingerprinting mitigation mechanisms. Commonly, these mechanisms can detect if some information was artificially manipulated to avoid fingerprinting (e.g., by a scrubber [136]).

Decision represents the outcome of the Classification building block as well as the result of a fingerprinting process. This result can be of different kinds: a binary value, a likelihood, or a comprehensive information providing several measurements and percentages of uncertainty. Some fingerprinters use these results to improve their datasets either manually or automatically.

As discussed, each fingerprinting building block specializes according to the type of fingerprinting we want to develop. In what follows, we use the reference model as a guideline to discuss fingerprinting within NCS.

5.1.4 Fingerprinting NCSs

When it comes to NCSs, standard fingerprinting techniques are not always effective. In our preliminary work, we performed tests to determine what standard fingerprinting tools can recognize on an NCS network. Tests showed that tools such as Nmap, P0f, etc. were not able to recognize most of devices deployed within NCSs (e.g., PLCs). For example, P0f recognized only devices running windows (e.g., some engineering workstations) but was not able to provide any valuable information about controllers. Also exploiting active probing, Nmap, Xprobe2++ and SinFP were not able to correctly identify any industrial or building automation device (an extract of the performed tests on NCS controllers can be found at [9]).

Tests with standard fingerprinters failed not just because of a lack of signatures within the datasets. An in-depth analysis showed that some of the problems could not have been fixed even by simply improving the involved datasets. For example, we created a custom signature for a Siemens PLC and added it to the Xprobe2++ database. Despite this new information, the tool was not able to recognize the device. In fact, we noticed that signature's field "icmp echo tos bit" was filled in randomly by the PLC invalidating the signature (in this specific case, the PLC was fingerprinted as a "HP JetDirect" printer). Furthermore, performed tests showed some intrinsic ambiguities in dealing with NCS-related information. For example, Nmap succeeded to recognize Siemens PLC's web server running on port 80 and labeled the service as "Siemens Simatic S7-1200 PLC httpd". Despite that, the tool fingerprinted the device as a QEMU because of the other collected information (e.g., TCP/IP headers) showing all its limitations in merging and leveraging multiple information.

Due to the failing of standard fingerprinters, we addressed the issue by deepening on the challenges that NCSs pose to fingerprinting and investigating on the opportunities that can be leveraged instead.

5.1.4.1 Challenges

NCSs share a set of properties that makes device fingerprinting more challenging compared to regular IT:

Critical Processes NCSs often monitor and control physical processes in which a failure may have disastrous consequences (or may be otherwise very undesirable). For this reason, active probing (e.g., scanning for available communication ports, opening arbitrary TCP connections, etc.) should generally be avoided. Several researches on NCSs argue that passive analyses are always preferable with respect to active ones because they minimize the potential risk of interfering with any underlying processes [110]. This is less of a concern in standard IT networks such as corporate LANs or the Internet where active fingerprinting tools such as Nmap are widely used. The main reason for this relies on the difference among the employed devices. Computational resources available within NCSs (e.g. controllers) are not comparable with standard IT (e.g., personal computers) and NCS devices are not usually capable of handling intrusive information gathering techniques [109].

Device Heterogeneity NCSs deal with countless physical and control processes. For every one of them, there are devices performing a wide range of different activities such as data collection, information sharing, coordination control, etc. Furthermore, every NCS vendor usually designs and develops its own devices for all aforementioned activities. In this situation, creating unique and reliable fingerprints is hard. Fingerprinters need to access a huge amount of different installations to reach ubiquitous coverage. Differently, standard IT networks such as corporate LANs mostly involve personal computers running standard operating systems. Tools such as Nmap and P0f provide comprehensive sets of fingerprints for these systems while they do not have any information related to industrial or

building automation embedded devices. It is worth noting that, with the spread of the Internet of Things, device heterogeneity will likely become more and more of a challenge for fingerprinting.

Proprietary Protocols The employment of numerous proprietary protocols is an issue related to device heterogeneity. In fact, NCS devices mainly implement vendor-specific application level protocols. Moreover, some of these implementations are not published or completely documented. Fingerprinting tools may not be able to leverage unknown protocols to gather information about unknown devices.

Long-running TCP Sessions Most popular fingerprinting tools (both active and passive) take advantage of TCP SYN and SYN-ACK packets that are exchanged during connection establishment. When it comes to NCS passive fingerprinting, there is no assurance to observe these packets over a limited period of time. Typically, once an embedded device (e.g., a PLC) sets up a connection, the underlying TCP session remains open for a very long time (e.g., days or even weeks) and application data keeps flowing over the same TCP connection. For this reason, the only option to trigger a TCP connection setup is to actively connect to the device, which is undesirable for the critical process issue discussed above. This situation severely reduces the applicability of TCP/IP stack fingerprinting.

5.1.4.2 Opportunities

Besides the challenges, NCSs share some characteristics that fingerprinting tools can leverage:

Long Life-cycle of Devices NCSs and especially ICSs have been used since the '70s and most system operators do not usually update hardware and software components [137]. Therefore, device life cycles are usually long enough to guarantee a fingerprint to remain valid for years. In contrast, personal computers and common operating systems are frequently modified and updated making it difficult to maintain fingerprinter datasets.

Stable Network Topologies and Communications Communication patterns do not usually change in NCS networks [138]. This is mostly due to the fact that, once established, control processes remain the same and thus devices behave accordingly by continuously performing the same tasks [139]. Fingerprinting tools can exploit this notion of stability by adopting communication patterns as device fingerprints. Differently, standard IT networks do not allow the same. Behaviors of personal computers strongly depend on the users and the related communication patterns are likely to change over time depending on user current needs (e.g. browsing or downloading)

Protocol Specifications Some NCS protocols such as Modbus, MMS or BACnet already carry information useful to fingerprint devices. Moreover, a few of

these protocols provide explicit mechanisms to query devices and collect information about employed hardware and software. As showed in [140] few tools exist that already take advantage of this opportunity (e.g., such as PLCscan [141] and Modbuspatrol). For this reason, fingerprinters can exploit these mechanisms to enhance their results. It is worth noting that this opportunity marginally contradict the Critical Processes challenge and the danger to probe NCS devices. However, we observe that exploiting mechanisms made available by the underlying protocols lower the risk of any dangerous interference with the infrastructure compared to the unexpected probing discussed above.

5.1.4.3 NCS Fingerprinter

Designing a comprehensive fingerprinter for NCS relies on mapping challenges and opportunities of the environment on the reference model discussed in §5.1.4. In what follows, we rediscuss each building block in the NCS scenario:

Sources of Information We showed already the difficulties to collect information coming from TCP and IP because of the long-running TCP sessions. Furthermore, a still extensive use of proprietary protocols makes it hard to use standard approaches of application-layer fingerprinting. However, we observed that some of these protocols can provide mechanisms to support fingerprinting techniques or, in any case, provide useful data to identify devices responsible for the monitored communications. For this reason, messages exchanged with specific protocols such as Modbus or BACnet are valuable sources of information. Furthermore, NCS fingerprinters should take advantage of the Stable Network Topologies and Communications opportunity by deriving and extracting valuable meta-information and thus performing device recognition through behavior analysis.

Gathering Compared to standard IT, NCS fingerprinting needs to reverse the balance between active and passive fingerprinting. Passive fingerprinting is the safest choice for NCSs. The only exception to passive fingerprinting techniques rely on the use of already-in-place mechanisms provided by NCS protocols. As already discussed, this activity is not as dangerous as injecting unusual traffic in NCS networks and fingerprinting tools such as PLCscan and Modbuspatrol work along this line. Depending on the selected data source, the effectiveness of Gathering building block relies on sensor (e.g., network sniffer) positions within NCS networks. Information such as communication flows captured over different part of an NCS network may be hardly comparable (e.g., comparing ICS communications coming from the field network and the process network). These situations requires Gathering building block to label captured data with extra information (e.g., information on the observed network).

Model Generation Compared to standard IT, information captured from an NCS network can be rarely combined into simple signatures. Missing information from TCP and IP headers and complex multidimensional information coming from

communication flows are two reasons why NCS fingerprinting requires more comprehensive decision models. Accordingly, the process of generating models and, ultimately, acquiring knowledge about a network goes through merging and harmonizing heterogeneous information together.

Decision Models Every decision model represents a device and list a set of related characteristics. Due to the difficulty of testing multiple NCSs and thus derive comprehensive information, it is important to link each decision model to the general characteristics of the NCS in which that model has been created. For example, when it comes to device behaviors, we assume that the likelihood to correctly fingerprint two devices relies on the similarity of the two related decision models as well as the overall similarity of the two NCS in which these devices are deployed. The simultaneous presence of information from the device and the NCS within the same fingerprint contextualizes the decision model. This context-aware decision model represents a comprehensive description of an NCS device working in a specific NCS infrastructure and ensures to attempt fingerprinting only when an overall contextual similarity among different NCSs is found.

Pre-processing According to the decision models, the Pre-processing building block is in charge of the information needed to compare unknown devices. This building block does not substantially change compared to the description proposed in §5.1.3 as it formats valuable network data coming from Gathering building block while dropping worthless one and, ultimately, forwarding the information ahead to Classification building block.

Classification Techniques for comparing NCS devices depend on the chosen decision models and can inherit limitations coming from the chosen data structures. Coming back to the example of context models, classification techniques cannot likely obtain strong evidences in fingerprinting NCS devices unless the contexts are similar. Most of the times, comparison techniques should be able to identify device types (e.g., determining who is a SCADA server and who is a PLC) and roles (e.g., determining if a PLC is coordinating others or just monitoring a physical-process as a field PLC). However, the presence of detailed information coming from “fingerprinting-friendly” protocols such as Modbus and BACnet can ultimately allow a fingerprinter to gather detailed information and comprehensively describe a device.

Decision Finally, NCS fingerprinters are meant to output likelihoods rather than true/false responses. This is due to the complexity of the decision models. According to the similarity of the overall NCSs and the amount of device characteristics of a device that matched with a decision model, a fingerprinter outputs an estimation of a device to be of a specific kind, to have a specific role, etc. For the same reason, it is unlikely for an NCS fingerprinter to automatically update its dataset. This may be allowed only with a clear result supported by information coming from “fingerprinting-friendly” protocols. In any case, automatically generated models

could be added to a different dataset (compared to the one manually evaluated) and consequently used in a limited number of situations (e.g., solving ambiguities when the main dataset does not provide reliable results).

To put into practice concepts and ideas discussed on the reference model, we now instantiate an NCS fingerprinter, namely the Flow Fingerprinter.

5.1.5 Use Case: the Flow Fingerprinter

The Flow Fingerprinter (FF) attempts to identify NCS devices by analyzing network communication patterns.

The approach focuses on device behaviors. FF builds decision models based on observations of known device communication patterns and characteristics. The acquired knowledge is later used to compare decision models with new models generated by unknown device communications. The tool does not aim to recognize employed hardware and software but uses information about known behaviors to guess unknown device types and roles. Therefore, the primary goal of FF is to recognize which device is a field device (e.g., a PLC), a SCADA server, an engineering workstations, etc.

FF works under the assumption that, within the same NCS, devices with equivalent roles share common characteristics in their network traffic. Furthermore, FF uses the concept of “context model” outlined in §5.1.4.3 to describe the traffic of a device in relation to the overall traffic observed in the NCS network. From this, we derive the possibility to compare traffic generated by two different NCSs. If two NCSs operate in the same domain (e.g., two infrastructures for house energy monitoring) and shows similar traffic patterns (e.g., same network topologies), we assume that two devices with equivalent roles share the same common characteristics even when they belong to the different NCSs.

5.1.5.1 Description

We now describe FF using the reference model building blocks presented in §5.1.4.3. In what follows, we omit Pre-Processing building block because it follows the same mechanisms of Model Generation.

Sources of Information & Gathering FF passively captures the traffic flowing in an NCS network. The follow-up analysis focuses on three layers of the ISO/OSI stack: datalink, network and transport. Specifically, FF considers communications happening on Ethernet, IP ver. 4, IP ver. 6, TCP, and UDP protocols.

The choice of limiting the analysis to the five protocols listed above comes from the challenge of Proprietary Protocols and guarantees the possibility to run the tool in any NCS network. To avoid the Long-running TCP Sessions challenge, FF does not parse protocol headers but stores statistics about each communication (e.g., data exchanged on every protocol, number of network frames, etc.).

Model Generation FF uses available data to build: 1) a model of the NCS network and 2) a behavior representation for every single device communicating

in that network. To achieve this, we leverage directed multi-graphs where:

- A **graph** represents the whole NCS network or a section of the network
- A **node** represents a device communicating in the NCS network (or in the chosen section)
- An **edge** represents a communication between two devices over one of the five protocols mentioned above

For every available NCS, FF stores three models of different meanings. The “datalink model” has got a node for each MAC address observed in the network and an edge for each Ethernet communication. The “network model” has got a node for each IP address observed in the network and an edge for each TCP and UDP communication. The “communication model” has got a node for each IP address that sends traffic in the network and, again, an edge for each TCP and UDP communication. Models coming from known NCSs are manually labeled by linking every node to the actual device and its related information. This information forms FF dataset.

Classification When it comes to devices deployed in different NCSs, Classification goes over two phases. The former phase concerns a comparison between decision models of the same kind (e.g., two “datalink models”). This means comparing graphs (entire NCS networks) or sub-graphs (NCS sub-networks). The latter phase focuses on single devices and fulfills the goal of identifying unknown device types and roles. This last phase runs only if the first confirms a similarity between the two NCS networks (or sub-networks). Model comparison relies on both standard graph properties (e.g. number of nodes and edges) and network measurements (e.g. exchanged bytes and packets). What follows is a list of metrics FF uses to compare two models:

- Number of packets per second
- Number of bytes per second
- Ratio between the number of nodes and the number of edges
- Ratio between the number of nodes and the number of packets per second
- Ratio between the number of edges and the number of packets per second
- Ratio between the number of nodes and the number of bytes per second
- Ratio between the number of edges and the number of bytes per second
- Ratio between the number of packets and the number of bytes
- Ratio between the number of bytes exchanged over Ethernet connections and the total number of bytes
- Ratio between the number of bytes exchanged over IPv4 connections and the total number of bytes
- Ratio between the number of bytes exchanged over IPv6 connections and the total number of bytes

- Ratio between the number of bytes exchanged over TCP connections and the total number of bytes
- Ratio between the number of bytes exchanged over UDP connections and the total number of bytes

Model comparison results depend on the number of nodes included in the two networks and the amount of matching metrics. This result represents the likelihood of two NCSs being similar. Similarly, FF evaluates a further list of metrics for every unknown device and compares them with known information within decision models. What follows is a list of metrics FF uses to compare two nodes:

- Ratio between the number of edges and total number of edges in the model
- Ratio between the number of incoming edges and total number of edges in the model
- Ratio between the number of outgoing edges and total number of edges in the model
- Ratio between the number of exchanged packets per second and the average number of packets per second
- Ratio between the number of incoming packets per second and the average number of packets per second
- Ratio between the number of outgoing packets per second and the average number of packets per second
- Ratio between the number of exchanged bytes per second and the average number of bytes per second
- Ratio between the number of received bytes per second and the average number of bytes per second
- Ratio between the number of sent bytes per second and the average number of bytes per second

FF evaluates the same metrics for each one of the five protocols mentioned before and compares all results. Furthermore, FF assigns bonuses when all metrics of the same kind share similarities (e.g., all Ethernet metrics of the two devices show similar results). This mechanism promotes comparisons of devices sharing affinities on specific protocols with respect to multiple matches coming from unrelated metrics.

Decision Decision building block puts comparison results together and computes the likelihood of two devices being of the same type and role. Given two devices A and B, belonging to M_a and M_b respectively, FF computes the similarity by using the following formula :

$$\text{similarity}(M_a^*, M_b^*) \cdot \text{similarity}(a, b) \quad (5.1)$$

computed on all possible sub-networks M_a^* and M_b^* .

to a “main PLC” in the ENEL facility. Furthermore, the decision models created in the training phase were probably insufficient to describe a comprehensive set of possible communication patterns happening between PLCs and SCADA servers. Nevertheless, the use of Nmap, P0f, XProbe2++, and SinFP on the same traffic did not give any valuable information (P0f scored the best and correctly fingerprinted one of the SCADA Servers as a Windows machine).

5.1.6 Conclusions

Network traffic is the most valuable source of information to build system knowledge. This is due by the amount of available information (most NCS components share information on a network) and ease of access this information (e.g., passively sniffing network traffic on a router). In this section, we focused on fingerprinting techniques and we evaluated what kind of information about NCSs can we derive by passively looking at the generated traffic. To achieve this, we analyzed NCS fingerprinting challenges and opportunities and we mapped related limitations and advantages to a fingerprinting reference model we derived from an analysis on standard IT fingerprinting.

FF is a possible instantiation of the reference model and, thus, a plausible approach to NCS fingerprinting. Nonetheless, FF showed limitations in its results, namely the possibility to identify device types and roles only and low reliability when it came to different devices in the network carrying out similar tasks. However, it is worth noting that FF evaluation would need a comprehensive set of decision models and, thus, a broad training over multiple NCS facilities. Nevertheless, FF exceeded results coming from standard fingerprinters such as Nmap and P0f paving the path for further improvements.

In Sec. 7.3.1.2 we come back to NCS fingerprinting and we discuss a further instantiation of the reference model in the context of building automation.

5.2 Documentation

This section discusses concepts and techniques related to four building blocks, namely **Documentation (Internal & External)**, **Document Retrieval**, **Document Processing** and **Information Analysis & Aggregation** (Figure 5.4).

In what follows, we investigate which NCS documents can integrate the knowledge we acquire with fingerprinting techniques. Furthermore, we give an overview on the challenges related to documentation retrieval and processing. This thesis does not include any new general technique for information retrieval and text processing. However, we show in §7 how concepts discussed in this section turn into custom solutions applied in the context of building automation systems.

5.2.1 Overview

NCSs are usually large-scale distributed infrastructures operated by several people. Both the complexity of the systems and personnel’s shared responsibilities create the need for extensive documentation describing multiple aspects of these

Figure 5.4: Documentation Information Mining

infrastructures. For this reason, description and configuration documents are often available to operators for maintaining their systems under control and trace changes and events related to system devices, control procedures and monitored physical processes.

As a key aspect of this research relies on gaining knowledge about NCSs, we argue that *documentation created and used by operators to monitor and manage their infrastructures can be leveraged to improve security solutions employed in the protection of the same infrastructures*. This approach comes both from the infeasibility of collecting more detailed information through meetings with the operators and from the limitations of acquiring knowledge from the network directly. Meeting operators and thus manually customizing a security system for each single deployment (and after each modification of the monitored infrastructure) is a solution that does not scale. Despite being helpful when it comes to tuning a security system, an extensive application of this approach faces an effort that operators cannot guarantee for the reasons discussed in §4 (e.g., time constraints). On the other hand, we already discussed in this chapter the possibility to understand device roles and purposes thanks to observations made on the generated network traffic. However, these observations are not always sufficient to describe the correct behavior of the devices in full details. Anomaly-based detection approaches always assume a trusted learning phase that safely shows instances of normal behavior. Nevertheless, this learning phase may sometimes not be granted and, most important, observed behavior always includes an arbitrary small subset of allowed operations leaving the description of a correct behavior incomplete. All this ultimately generates false positives and false negatives.

The use of documentation to gain knowledge about monitored infrastructures works when involved documents own the following properties:

- **Availability** Documentation about monitored systems must be accessible. No analysis would be possible without direct access to information describing infrastructure components, mechanisms and constraints. To improve analysis

capabilities, documentation should be provided in an electronic form to allow automated knowledge extraction.

- **Authenticity** Information about infrastructure components, mechanisms and constraints should be verified or verifiable. Document analysis must rely on authentic information to avoid building knowledge—and, ultimately, tuning security systems—based on bogus data. For this reason, documentation retrieval should always be performed on private repositories (e.g., owned by operators) or reputable third parties (e.g., vendors’ websites).
- **Integrity** Even when the source of information is reliable, the process of gathering documentation should employ secure retrieval techniques. Document analysis must rely on the integrity of the collected information for the whole process of knowledge extraction.
- **Linkability** Finally, information retrieved from the documentation must be linkable to what a security system can observe. For example, on a network, information about components, mechanisms and constraints needs to map to features monitored by an IDS (e.g., information on a whitelist of network services should link to the correct IP address).

In §7, we will show how these properties hold for NCSs and, specifically, how they instantiate in the context of building automation systems.

5.2.2 NCS Documentation

NCS documentation comes from different sources (external/public vs. internal/private) and with different formats (structured vs. unstructured). In what follows, we describe some notable examples of documents that can be used to gain knowledge about the related monitored infrastructures.

5.2.2.1 External Documentation

External documentation refers to all documents that are publicly available. These documents come with the advantage of being easily accessible as they do not require any agreement with infrastructure operators or customized retrieval techniques. On the other hand, these documents usually include less valuable information than others as they broadly refer to general devices and technologies without describing any detail of their implementation and purposes within specific infrastructures.

Protocol Standard Protocol standards refer to public/private technical documents describing rules and mechanisms of communication protocols. “Requests for comments” (RFCs) are major examples of this kind (e.g., “793: Transmission Control Protocol” and “791: Internet Protocol”). In the same way, standard bodies such as the Internet Engineering Task Force (IETF), the International Organization for Standardization (ISO), the International Electrotechnical Commission

(IEC) provide open standards defining thousands of different protocols. Two notable examples for this work are MMS and IEC 60870-5-104.

Protocols standards are usually unstructured documents that aim to inform users implementing or deploying the related communication protocols within their infrastructures. Document templates as well as the kind of information provided within the standards follow rules defined by the standard bodies. However, these documents usually lack of a fixed structure and this makes it difficult to automate the extraction of knowledge. Nevertheless, most specification-based approaches use these standards to define intrusion detection rules (e.g., [57]).

PICS The Protocol Implementation Conformance Statements (PICSs) are structured documents defining the set of requirements that a specific implementation of a protocol standard meets or lacks. These documents are usually the output of formal protocol conformance testings on devices. In fact, at the end of this analysis, the tester fills in a pre-defined PICS template writing down all the information concerning the implementation of a specific communication protocol within the device under analysis. Ultimately, the PICS allows an end user to understand if the related device meets his or her needs and requirements.

PICSs are valuable sources of information to gather knowledge about infrastructure components. These documents allow to refine the information given by the standards and to narrow down the description of device mechanisms and functioning procedures. Furthermore, a structured template allows to automate the extraction of the information to a fairly high degree. In this regard, it is worth noting that PICSs support the use of the Testing and Test Control Notation v.3 (TTCN-3). This strongly typed scripting language used in conformance testing of communicating systems bridges the gap between abstract test scripts and concrete communication infrastructures. The possibility to combine TTCN-3 scripts with ASN.1, IDL and XML type definitions makes a PICS adopting TTCN-3 easily integrable with other software such as an IDS.

5.2.2.2 Internal Documentation

Internal documentation refers to all documents that belong to infrastructure operators. These documents are often used to maintain a record of infrastructure components and keep trace of changes and updates of the monitored systems. Accessing internal documentation may not always be possible due to the sensitivity of the involved information. In most cases, connecting and extracting information from internal documentation relies on signing a non-disclosure agreement (NDA) with infrastructure operators. Despite these disadvantages, accessing this kind of documents substantially improves the acquisition of knowledge about infrastructure processes. The fact that the documents refer only to a specific infrastructure (rather than a protocol or a class of devices) and that these documents are maintained by the operators makes sure that included information is authentic and reliable. Furthermore, the property of linkability described above holds due to the high quality of information and details.

EDE Engineering Data Exchange (EDE) files allow to organize data sharing among different devices within multi-vendor infrastructures. This documents define which device talks to another, message formats, data exchanged etc. Information included in the EDE files refines information coming from protocol standards and PICSs. In fact, if a standard defines a communication protocol and a PICS defines its implementation within a class of devices, the EDE file describes the instantiation of specific mechanisms and data structures of that protocol within a unique device. In other words, EDE files define the internal characteristics of a device and the way it interacts with the outside world.

As for PICSs, EDE files come with standard templates. Spreadsheets and Comma-Separated Values (CSV) are common examples of EDE templates. This characteristic makes EDE files especially suitable for automated knowledge extraction. Furthermore, detailed information allows to effectively model the expected behavior of a device that is correctly fulfilling its purposes. In this regard, it is worth noting that information within EDE files is rarely prone to human error. In fact, it is common for operators to set up and run a device using a dedicated software and lately automatically generate the related EDE file. This makes the EDE file the most precise description of device inner functioning.

SCL Files Substation Configuration description Language (SCL) is a language specified by IEC 61850 to formally define and describe electrical substation device functioning and activities. SCL files are examples of the highest level of information retrievable for knowledge extraction. Like EDE files, SCL files describe functions and data structures of single devices. However, these files also integrate information about expected communications among devices (e.g., which devices can connect with each other, when these devices can connect, and what they can exchange).

SCL files are structured documents (e.g., XML files) of different types. According to the carried information there can be: Substation Configuration Description or SCD files (describing complete substation details), IED Capability Description or ICD files (defining complete capability of Intelligent Electronic Devices), etc. The use of this information within a process of knowledge extraction guarantees a comprehensive and detailed overview of infrastructure devices and their connections. As for the EDE files, operators can automatically generate SCL files by using software employed to setup their infrastructures. For this reason, included information is rarely prone to inaccuracies.

5.2.3 Analysis

In what follows we briefly outline techniques aiming to correctly identify infrastructure-related documentation and to effectively extract information from this documentation with the purpose of acquiring knowledge. This thesis does not focus on data mining and we do not propose any new general technique to select and parse NCS documentation. Nevertheless, we describe the most important challenges as well as plausible solutions to document retrieval and processing.

5.2.3.1 Document Retrieval

Document retrieval generally defines a process of selection, within a set of documents, of those that match a specific query. Sometimes referred as a branch of text or information retrieval, document retrieval is today a cornerstone of the Internet and, specifically, of Internet search engines.

For what concerns NCSs, document retrieval plays a major role when it comes to identify useful infrastructure-related documents for knowledge extraction. This process changes depending on the nature of the documentation. On the one hand, external documentation requires retrieval techniques able to scan huge collections of documents (including those that are not related to monitored infrastructure). On the other hand, internal documentation requires less time-efficient techniques but improved accuracy as most documents may refer to the same systems. In both cases, the research starts with a query that includes information coming from network traffic. As already discussed, fingerprinting techniques provide some insights on devices communicating on the network. This information represents the seed for a research over available documentation that can enhance infrastructure view and, thus, operators knowledge.

Our research for effective document retrieval techniques relies on the state-of-the-art and, specifically, on available tools and does not propose any custom solution. Document retrieval state of the art differentiates between two categories of techniques, namely form-based and content-based [142]. The former exploits syntactic properties of document's text and compares it with the information coming from the network. In this case, the text is generally unstructured. The latter makes a semantic connections between documents and retrieved information going beyond a simple string matching. Furthermore, the effectiveness of employed document retrieval techniques drastically changes depending on the way documents are ordered and indexed [143, 144]. Form-based techniques often exploit suffix tree arrangements while content-based techniques mostly work with inverted indexes.

For the development of our security solutions we decided not to make any assumption on the nature of available documents. We considered both internal and external documentation as heterogeneous datasets including structured and unstructured documents with different formats (e.g., extensions such as ".txt", ".pdf", etc.). In the case of external documentation, our solutions focused on selecting documents over the Internet. This approach mostly demanded the creation of appropriate search queries for web search engines (e.g., Google, Bing). To achieve this, we relied on network traffic. We employed fingerprinting techniques to extract valuable information from the network, thus identifying communicating devices, and we parsed ASCII payloads to derive further knowledge about the infrastructure. Differently from external documentation, internal documentation is a collection of documents stored within general-purpose computers or dedicated network storages. In those cases, we independently implemented custom inverse indexes using document information such as titles, abstracts, and key words. As for the previous case, we used information coming from fingerprinting to match words over the indexes and, thus, pinpoint documents likely to be related to devices communicating in the network. In §7 we discuss an instantiation of these

techniques in the context of building automation systems.

5.2.3.2 Document Processing

Once one or more documents are retrieved, we automate the process of information extraction. This process changes if the document under analysis is structured or unstructured. The former is easier to handle. Documents such as CSV and XML files allow effective parsing and labeling of included information. The latter needs more sophisticated approaches such as Natural Language Processing (NLP) [115] to extract valuable knowledge. Furthermore, document processing often deals with semi-structured documents. Files such as PDF include both structured sections (e.g., tables) and free text. In this case, information extraction can leverage layout information to enhance results provided by NLP [145].

PICS, EDE, and SCL files are examples of NCS structured documentation. Processing XML files such as SCLs is a well-known and widely studied issue in literature. There are numerous possible processing tasks related to XML. Schwentick et al. defines four categories in [146] based on the languages supported by the World Wide Web Consortium (W3C). First, schema languages such as Document Type Definition (DTD) and XML schema apply to the process of XML document validation. Then, navigation languages such as XPath [147] focus on the identification of relationships among XML tags. Query languages such as XQuery apply the process of data and knowledge extraction. Finally, transformation languages such as XSLT [148] focus on data manipulation. In the context of NCSs, Hadeli et al. show a practical application of query and transformation languages to gain knowledge from SCL files related to power transmission and distribution systems [108].

As for XMLs, CSV files (e.g., EDE files) make the process of extracting information quite easy. Differently, PDF files such as PICSs require a semi-structured approach leveraging data arrangement and textual content (e.g., the presence and identification of key words). Even when data arrangements are not clearly defined by meta-information, studies show that it is possible to derive data structures and tables by clustering analysis. Ramakrishnan et al. discuss the possibility to identify text blocks from PDF-formatted files and classify these blocks into logical units [145]. Burcu et al. specifically focus on PDF tables and describe methods for effective data extraction [149]. Finally, several studies present tools to automate PDF parsing and data mining; among them, “pdf2txt” [150] and “pdftables” [151].

Protocol standards and vendors’ user manuals are examples of unstructured NCS documentation. As discussed, numerous studies on free-text analysis rely on text mining or NLP. Early works such as Strzalkowski et al. generally adopt NLP to process documents and extract content-carrying terms as well as to discover dependencies among terms and build conceptual hierarchies [152]. Later, these approaches have specialized within different domains. For example, Saneifar et al. apply NLP to log files generated by digital systems [153]. In their work, the authors discuss how to effectively extract and define a domain terminology from these log files. Differently, Sadeh et al. develop specific techniques for extracting privacy requirements from natural language policies [154]. Over the years, literature studies have proposed numerous text mining algorithms have been proposed based

on different techniques such as categorization, clustering, etc. Also, several tools have been developed to fulfill text analysis and information extraction. Among the most important, Carrot2 [155] and GATE [156] are widely-known open source tools freely available for software integration. A few tools address NCS-related documentation directly. Among them, we mention software coming from the rfc-analysis project [157]. Project's goal is to develop code for RFC automated analysis.

In §7 we discuss knowledge extraction from PICS and EDE files related to building automation devices.

5.2.4 Conclusions

With respect to information coming from network traffic, NCS documentation brings a more comprehensive and precise view of monitored infrastructures and components. The presence of this documentation as well as the quality of included information is guaranteed to a fairly high degree by operators that use such resources as a way to keep their systems under control. Analysis of the state of the art shows that standard approaches for document retrieval and processing are feasible and suitable techniques to deal with NCS documentation examples identified in §5.2.2.

5.3 Section Conclusion

In this chapter, we discussed two main possible approaches to gain system knowledge about NCSs, namely from network traffic (with fingerprinting) and from documentation. The former exploits available information in the network (e.g., network messages) to identify communicating devices and their distinctive properties. The latter uses available description and configuration documents to enhance results given by the first approach. Gaining information from documentation relies on the results provided by network analysis and fingerprinting. In fact, document retrieval and processing use fingerprinting information to select interesting documentation as well as extract valuable information from its content. In the following two chapters we show how the use of these two sources of information and, more in general, the acquisition of knowledge about NCSs can enhance security solutions monitoring these infrastructures.

CHAPTER 6

Intrusion Detection in ICS

In this section we present a new example of APT, namely the “sequence attack”, and we design and develop a *sequence-based intrusion detection system* (S-IDS) to tackle this kind of attacks. Furthermore, we show how the S-IDS can integrate with techniques for acquiring system knowledge and how this integration decreases the number of false positive. The architecture of the S-IDS fits the framework proposed in §4. Concepts and techniques discussed in this chapter relate to the building blocks emphasized in Figure 6.1.

Figure 6.1: SIDS and improvement related building blocks

6.1 Sequence Attacks

As discussed in the introduction, we define “sequence attacks” meaning cyber-attacks that rely on the possibility of crafting a sequence of individually harmless network messages that is capable of disrupting the correct functioning of an NCS. We identify four kinds of sequence attacks classified according to the target com-

ponent of an NCS (i.e. device or physical process), and the type of misuse carried out (i.e. manipulate timing or order of messages). Figure 6.2 illustrates the four kinds of sequence attacks.

Figure 6.2: Sequence attack classes

Looking at the first dimension, we distinguish between attacks targeting a device protocol stack implementation, and attacks targeting the physical process controlled by such devices. The first, more traditional, type of attacks consists of exploiting vulnerabilities in the protocol stack implementations. The second type of attacks is specific to NCSs, and aims at diverting or taking control over the industrial process by leveraging the lack of integrity and authentication mechanisms. To take control of the process, an attacker may either re-program the logic executed by controllers (e.g., PLCs [158] such as it happened in the case of Stuxnet), or directly control the process from the network by using the control messages defined by the NCS protocols for operators. In this chapter we focus on this second, less-explored, scenario.

Looking at the second dimension, we distinguish between attacks in which messages or commands are sent with an incorrect/malicious order, and attacks in which messages or commands are sent with an incorrect/malicious timing. The first type of attacks consists of violating the state machine of application protocols, or sending sequence of commands that put the process in an unsafe state. The second type of attacks consists of leveraging limitations of embedded NCS devices in processing input data, or exploiting weaknesses of process equipment (e.g. motors or valves) when changing their operational state in ways not originally foreseen by manufacturers.

Based on these two dimensions, we derive four attack scenarios and provide for each one of them a simple yet realistic example.

Message order-based control device compromise The majority of application-level protocols used for controlling PLCs have a set of functionalities to load/store the logic program from an engineering workstation with network connectivity to the PLC. The load/store functionalities are typically achieved by specific sequences of messages. At a high level, we could describe this sequence of messages for uploading logic program to the PLC as: (1) lock the PLC, (2) stop the running program(s), (3) delete the existing program(s), (4) transfer the program's code to the PLC, (5) create new program(s) with the new code, (6) start the new program(s) and (7) unlock the PLC. Each step is achieved by sending a particular message to the PLC. From experiments on ABB controllers, we have found that one can attack some PLCs by just sending some of these messages in an inconsistent order. For example, sending a (valid) start program message when the program is still running causes an error which is not properly handled by the PLC firmware and causes the PLC to crash.

Message order-based physical process compromise Carcano et. al. introduced a scenario comprising a pipe with high pressure steam flowing on it [159]. The pressure is regulated by two valves (V1 and V2). An attacker, capable of sending control messages on the process network, sends a write message to the PLC controlling the valves in order to force V2's complete closure, and a write message in order to force V1's complete opening (thus maximizing the incoming steam). Each of these commands are perfectly legal when considered individually, while sent in the specified order will bring the system to a critical state.

Message timing-based control device compromise Many embedded devices used in NCS have restricted computing capabilities, and suffer from poor protocol stack implementations (because they are based on the wrong assumption that all devices in the network will behave according to the intended use of the protocol). As a result, it is fairly easy to mount attacks to exhaust the resources of these embedded devices by simply flooding them with (valid) protocol messages (e.g. TCP SYN-flood). However, from experiments, we have found that one can achieve the same effect by using application-level messages. This is particularly true for PLCs using UDP-based application protocols. Flooding these devices with application-level messages that require expensive operations (e.g. diagnostic functions) quickly exhausts the resources of the PLC, compromising their ability to complete their scan-loop in real-time, and eventually leading to a complete reset of the device.

Message timing-based physical process compromise A report from the US President's Commission on Critical Infrastructure Protection [160] presents an attack scenario for the water distribution sector. In this scenario, major control valves on the water pipeline are rapidly opened and closed causing a so-called "water hammer effect", which could result in a large number of simultaneous main breaks in the pipeline. Such an attack could be carried out by rapidly sending a sequence of write messages, issuing open and close commands to the PLCs controlling these valves.

6.2 State of the Art

There are only few examples of sequence-aware IDSs in literature. Sekar et al. take advantage of a sequence analysis to profile IP, ARP, TCP and UDP communications in [57]. The authors discuss the concept of specification-based anomaly detection and describe the steps to implement a detection system based on extended finite state automata (EFSA). This work proves the feasibility of a sequence analysis based on comprehensive protocol specifications but it does not extend the approach to application-layer protocols or domains beyond TCP/IP.

Krueger et al. leave specification-based approaches aside and implement a method for protocol inspection and state machine analysis called PRISMA [161]. Their approach uses n-grams analysis to group network messages into clusters. Then, the authors use Markov chains to link the clusters together and identify rules defining the behavior of a communication. Also in this case, performed tests support the feasibility of the approach. However, the authors focus on textual protocols and do not broaden the testing to binary protocols. As already discussed, works such as [74] highlight the difficulties of using n-gram analysis for binary protocols. As NCSs mainly deal with these protocols, every approach to sequence analysis should take this characteristic into account.

Works such as [162, 163] focus on this last issue. Goldenberg et al. analyze Modbus messages and use deterministic finite automata (DFAs) to build communication models from real traffic traces [162]. The authors rely on the high periodicity of Modbus traffic and performed tests on data coming from real infrastructures confirm this assumption. However, it is worth noting that the modeling does not take into account effects of human interaction in the stability of the traffic. Moreover, our tests show that Modbus traffic is periodic only if we are able to filter out random delays (e.g., due to network components such as routers). Therefore, the proposed approach seems insufficient to capture a more comprehensive set of events happening in real scenarios.

In the same way, Yoon et al. focus on Modbus but they model communications using dynamic Bayesian networks (DBNs) and probabilistic suffix trees (PSTs) [163]. When a network communication is observed, the system looks at the likelihood of generating the related sequence of messages from the PST model. With respect to [162], this approach attempts to solve the problem of random delays. The algorithm works as follows: if the action of restoring a delayed (or missing) message makes the communication acceptable for the PST, the system considers it as “normal traffic” and proceeds. However, even if this mechanism is capable of identifying and filtering out false positives caused by delays, the authors do not discuss the impact of the approach on the number false negatives (e.g. real attacks opportunely filtering specific messages on the network).

Differently, Hadžiosmanović et al. approach the problem from a further perspective by analyzing ICS process variables [164]. The authors do not model network messages but focus on process variable values extracted from packet payloads and use auto-regression modeling and control limits to monitor their behavior over time. When a value does not fit the model or exceeds the control limits, the IDS raises an alert and suggest the expected correct behavior used as reference. This

approach shows promising results but faces several difficulties in modeling specific classes of variables (e.g. floating points). This is due to the way ICS applications deal with variables. When the size of a process variable exceeds the memory unit used by an application (e.g., 16 bits in Modbus) the value is split and saved within two or more memory locations. The proposed IDS is unable to follow such mapping and treats every memory location independently from the others. This ultimately causes the employment of erroneous models.

Finally, it is worth noting that there are several examples of sequence-aware approaches employed for host-based intrusion detection. Some of these works focus on profiling user activities [165]. Others focus instead on profiling programs or sequences of system calls [63, 64, 75, 166, 167].

State-of-the-art research on sequence attack detection within NCSs still faces three main challenges. First, *employed approaches dealing with large sets of features produce inefficient and sometimes unmanageable models* (e.g., in the case of Goldenberg et al. the DFAs dealt with a high number of states). Second, *the effect of communication delays and human operations decrease model accuracy and consequently hamper the detection of malicious activities*. Third, *detection of semantic attacks such as sequence attacks in NCS requires knowledge about employed protocols and messages* (e.g., as discussed, pure machine learning approaches such as Krueger et al. in [161] do not properly handle this kind of threats).

6.3 Security Solution

6.3.1 Sequence-based Intrusion Detection

Figure 6.3 illustrates the reference architecture of our sequence-aware intrusion detection (S-IDS) and contextualizes it within the diagram of Figure 4.1. The S-IDS relies on a layered structure. Each layer receives data from a lower layer, makes an evaluation, and finally forwards the results to the following layer. A layered structure allows to abstract from the different inputs/outputs and thus improve usability, maintainability and, foremost, the extensibility of the S-IDS.

Reader is the first layer of the architecture. Generally, the Reader is in charge of capturing raw information (in this case, network packets) and generates a uniform and identically formatted input stream for the S-IDS. Furthermore, Reader filters out redundant or corrupted data that will not be needed by the following layers.

Sequencer is the core of the architecture. This layer is in charge of transforming data flowing through the Reader into temporal sequences of events. This activity relies on a precise definition of what an event is. In fact, every event is defined by a set of characteristics or features used by the Sequencer to organize incoming information in formalized data units. For example, in NCSs, network messages concerning variable readings may be considered one type of event, messages including variable writings is another, etc. If needed, a more fine-grained categorization of events can be in place (e.g., instead of one group of messages, differentiating

Figure 6.3: S-IDS general architecture

read messages depending on the target variables). Ultimately, the Sequencer sorts the data units according to a timestamp that represents the moment in which the event has occurred. Obtained sequences of events are, thus, forwarded to the next layer.

Modeler is the last layer that performs data manipulation. This layer is in charge of reading temporal sequences coming from Sequencer and building models that represent system behaviors over time. A modeling step in the architecture is important for two reasons. First, it is unfeasible to store and analyze large amounts of long sequences of events without exhausting computational resources. It is common for an IDS to investigate long-time behaviors of infrastructures and thus capture information for days or weeks. For this reason, the Modeler should arrange data in compact data structures. Second, the detection can focus on a subset of features included in the sequences. For this reason, the Modeler should emphasize such features and provide an efficient interface to access them.

Detection includes all detection algorithms used to identify sequence attacks. Differently from the underlying layers, Detection does not completely abstract from Modeler outputs. Implementation of detection mechanisms may depend on the characteristics of provided models. Detection investigates and evaluates “detection models” with respect to a set of “training models” used as references. Furthermore, Detection is in charge of alerting users when the “detection models” show malicious patterns.

In what follows we discuss a possible implementation of an S-IDS. We use a Reader that can gather information from three different ICS protocols, namely Modbus, MMS and IEC104. Then, we use a Sequencer to analyze communication behavior and model observed sequences of messages through discrete-time Markov chains. Finally, we implement a detection mechanism computing weighted-distance

statistics among Markov chain states and transitions based on transition probability values.

6.3.1.1 Sequencing

Detecting “sequence attacks” relies on the possibility to extract sequences of events from the network and identify information needed to instruct an IDS. In what follows, we discuss the definition of a sequence of events and we start the process that brings a set of network frames to become device communication models.

The first and simplest approach to this process considers every traffic frame as an event per se, such as:

Definition 1. *A sequence of events is a time-ordered list of messages $\{m_{t_n}\}$ (with $n \in [0, \infty[$ and $t_n < t_{n+1}$) that are observed in a network.*

However, not all traffic frames are equally important for sequence analysis. Particularly, we aim to detect sequence attacks that work at ISO/OSI application layer and we do not want to consider any message concerning underlying network mechanisms (e.g., TCP ACKs). Furthermore, we decide to focus on sequence attacks that involve single connections (e.g., one TCP stream) between a sender and a receiver (all sequence attacks discussed in §6.1 are of this kind). For this reason, we group traffic frames into communication streams. From these assumptions, we derive that:

Definition 2. *A sequence of events is a time-ordered list of application-level messages $\{m_{t_n}\}$ (with $n \in [0, \infty[$ and $t_n < t_{n+1}$) that are exchanged over a network channel established between two devices.*

The last definition encompasses all possible network protocols. However, protocols have different characteristics (e.g. synchronous vs. asynchronous communications, pushing vs. polling schemes, etc.). For this reason, Sequencer uses a customized definition for every different protocols. To show this, we analyze and describe properties of three widely-used ICS protocols that encompass heterogeneous characteristics: Modbus, MMS, and IEC104. We define sequences of Modbus, MMS and IEC104 events¹ as follows:

Definition 3 (Modbus). *A sequence of **Modbus** events is a time-ordered list of events $\{e_{t_n}\}$ where e is a 3-tuple $\langle ID, Code, Data \rangle$ derived from a couple of messages $(m^{Req}_{t_n}, m^{Res}_{t > t_n})$.*

In a Modbus event, *ID* is the “Transaction Identifier”, *Code* indicates the type of operation performed, and *Data* represents the information carried by Modbus requests m^{Req} and responses m^{Res} . This last information depends on the “Function code” and can include variable addresses, number of coils or registries, numerical values, etc.

¹All following definitions comply with the conditions of $n \in [0, \infty[$ and $t_n < t_{n+1}$ that define a time-ordered list.

Definition 4 (MMS). A sequence of **MMS** events is a time-ordered list of events $\{e_{t_n}\}$ where e is a 4-tuple $\langle ID, PDU, Service, Data \rangle$ derived either from a couple of messages $(m^{Req}_{t_n}, m^{Res}_{t > t_n})$ in case of synchronous communications, or messages m_{t_n} in case of asynchronous ones.

In an MMS event, *ID* is the “Invoke ID”, *PDU* indicates the type of communication (e.g. Initiate, Request, Response, Error, etc.), *Service* describes the type of operation (e.g. read, write, etc.), and *Data* represents the information carried by MMS requests and responses. This last information depends on the previous two attributes and can represent variable addresses, numerical values, general information, etc.

Definition 5. A sequence of **IEC104** events is a time-ordered list of events $\{e_{t_n}\}$ where e is a 3-tuple $\langle Format, Service, Data \rangle$ derived from a couple of messages $(m^{Req}_{t_n}, m^{Res}_{t > t_n})$ in case of synchronous communications, or messages m_{t_n} in case of asynchronous ones.

In an IEC104 event, *Format* is the format of the messages, *Service* defines the performed service, and *Data* represents the information carried by IEC104 messages. This last information specifies the Service attribute and can represent values, errors, etc.

6.3.1.2 Modeling

Once the Sequencer crafts a sequence of events, the Modeler takes care of arranging the information in a data structure that is suitable for detection.

In what follows, we use discrete-time Markov chains (DTMC). Modeling NCS communication with DTMCs has some advantages compared to using other modeling techniques such as n-grams or Deterministic Finite Automata (DFA). On the one hand, n-gram analysis would require to split sequences of events into subsequences of length n . Therefore, a communication model would be defined by the statistical distribution of the n-grams included in whole sequence of events. This analysis would miss to identify subsequences of events larger than n that remain constant over time. Nevertheless, works such as Wespi et al. show that countermeasures to this problem are possible by using more sophisticated modeling techniques based on variable-length patterns [63, 64]. On the other hand, modeling communications with DFA would allow to identify sequences of any length but it would not suit the analysis of rare events. Without considering the probability of an event to occur, it would be impossible to effectively identify stable sequences and, consequently, to model the actual behavior of a network communication.

The modeling process maps every event of a sequence in a DTMC state and transforms the property of “being consecutive” in a DTMC transition. Every state represents a set of events that share the same semantic meaning. Some of the attributes used in the definitions of events to build the sequences do not change this meaning and, thus, may not be considered in this phase. For example, two Modbus events that differ in their transaction identifiers belong to the same DTMC state as transaction identifiers do not modify the semantic content of the messages.

Building the DTMCs is a protocol-independent and completely automated process. Modeler reads a sequence of events coming from the Sequencer and progressively populates the model. For every event in the sequence, Modeler either creates a new state or updates an existing one. At the same time, the Modeler adds or updates a transition that links the previous visited state to the current one. For this reason, every transition between two states indicates that at least an event belonging to the first state comes immediately before an event of the second one.

Algorithm 1 formalizes the modeling process. For every event e_{t_n} , the algorithm either adds it to an existing state (“updateS”) or creates a new one in case the semantic meaning does not fit any other state in the model (“addS”). For each step, the algorithm adds or updates a transition that links the previous visited state to the current one (“addT” and “updateT” respectively).

Algorithm 1: ModelingDTMC(*sequence*)

```

forall  $e_{t_n} \in \text{sequence}$  do
   $\text{State}_{DTMC} \leftarrow \text{extract\_meaning}(e_{t_n});$ 
  if  $\text{State}_{DTMC} \in DTMC$  then
    |  $\text{updateS}(\text{State}_{DTMC});$ 
  else
    |  $\text{addS}(\text{State}_{DTMC}, DTMC);$ 
  if  $\text{Transition}_{\text{previousState}, \text{State}_{DTMC}} \in DTMC$  then
    |  $\text{updateT}(\text{Transition}_{\text{previousState}, \text{State}_{DTMC}});$ 
  else
    |  $\text{addT}(\text{Transition}_{\text{previousState}, \text{State}_{DTMC}}, DTMC);$ 
   $\text{previousState} \leftarrow \text{State}_{DTMC};$ 

```

Every DTMC state S is defined by a 5-tuple $\langle \text{Data}, \text{Type}, \# \text{Events}, \text{FTS}, \text{LTS} \rangle$ where:

- **Data:** is the information univocally identifying S and describing the meaning of the events included in S
- **Type:** is a property indicating if S represents request/response pairs, just requests, or just responses
- **#Events:** is the number of events that belong to S
- **First Time Seen (FTS):** is the timestamp of the first element included in S
- **Last Time Seen (LTS):** is the timestamp of the last element included in S

Every DTMC transition δ from a source state (Src) to a destination state (Dst) is defined by a 6-tuple $\langle \text{Probability}, \# \text{Jumps}, \text{FJ}, \text{LJ}, \text{ATE}, \sigma \text{ATE} \rangle$ where:

- **Probability:** is the ratio between the number of jumps (or passages) from Src to Dst and the total number of jumps from Src to any other state of the DTMC

- **#Jumps**: is the number of jumps from Src to Dst
- **First Jump (FJ)**: is the timestamp of the first jump between Src and Dst
- **Last Jump (LJ)**: is the timestamp of the last jump between Src and Dst
- **Average Time Elapsed (ATE)**: is the average time elapsed between two consequent events of Src and Dst
- **Standard Deviation on Time Elapsed (σ ATE)**: is the standard deviation computed over all jumps included in δ

Figures 6.4 and 6.5 illustrate two steps of the modeling process of a Modbus communication including the following events:

1. **Modbus “Request/Response” element**: Tran. ID = “134”, Unit ID = “1”, Function Code = “1” (Read Coils), Data = “Starting address = 0, Quantity of coils = 1”, Timestamp = “1st Sep. 2013 10:15:32.032”
2. **Modbus “Request/Response” element**: Tran. ID = “135”, Unit ID = “1”, Function Code = “3” (Read Holding Registers), Data = “Starting address = 100, Quantity of registers = 10”, Timestamp = “1st Sep. 2013 10:15:33.126”
3. **Modbus “Request/Response” element**: Tran. ID = “136”, Unit ID = “1”, Function Code = “1” (Read Coils), Data = “Starting address = 0, Quantity of coils = 1”, Timestamp = “1st Sep. 2013 10:15:34.983”
4. **Modbus “Request/Response” element**: Tran. ID = “137”, Unit ID = “1”, Function Code = “3” (Read Holding Registers), Data = “Starting address = 100, Quantity of registers = 10”, Timestamp = “1st Sep. 2013 10:15:35.033”

Figure 6.4: DTMC modeling (phase 1)

As shown in Figure 6.4, the first event of the sequence causes the creation of a state of type “Request/Response” representing Modbus messages used to read 1 coil from address 0 (state A). The second event results in a further state that shares the same type but represents Modbus messages used to read 10 registers from address 100 (state B). At the same time, the modeling process adds a transition linking the two states and describing the sequentiality between the first two events (δ_1). Figure 6.5 illustrates what follows with the last two events. The third event shares the same meaning of the first one and thus increases the $\#Events$ attributed of state A. Moreover, a new transition connects back state B to state A representing the new relationship between the second and the third events (δ_2). Finally, the

Figure 6.5: DTMC modeling (phase 2)

fourth event belongs to state B as it shares the same meaning of the second one. Since the DTMC already has a transition linking the two states, the modeling process increases attribute *#Jumps* as well as the other attributes of $\delta 1$.

6.3.1.3 Analysis

Before presenting the detection mechanisms, we discuss the feasibility of sequencing and modeling processes for network communications observed in real NCS infrastructures.

As outlined at the beginning, we test our approach on traffic samples captured within three different utilities. Modbus traffic belongs to the control network of a water treatment and purification plant. MMS traffic belongs to a network controlling gas storage facilities and pipelines. IEC104 traffic belongs to a network monitoring a gas distribution infrastructure and specifically refers to a few distribution secondary substations. The time frame of observation goes from one day (MMS) to five days (Modbus and IEC104). It is worth noting that all captures refer to situations in which operators and engineers work normally and perform any regular or occasional routines on their systems. These samples represent the most realistic use case in which a security system can be deployed, tested and tuned.

From an implementation point of view, Reader building block of Figure 6.3 works on top of Tshark parsers [168]. Compared to an ad-hoc solution, this approach allows to avoid problems related to TCP stream reconstruction (e.g. re-transmissions, segments reordering). Every TCP stream identified by Tshark is forwarded to the Reader that works as an interface to the Sequencer. This building block takes care of transforming Modbus, MMS, and IEC104 messages into events and concatenating events to form sequences. At the same time, Modeler builds the DTMCs that represent the sequences handled by the Sequencer.

Modbus Figure 6.6 illustrates the Modbus network. From this network we cap-

Figure 6.6: Modbus network

ture 47 GB of traffic and more than 260M frames. Every communication among PLCs and RTUs include a single “Read Holding Register” command sent in a loop. For this reason, we especially focus our analysis on the connections among PLCs and SCADA Server. These TCP streams contain about 43M frames. The related sequences contain about 10M events each. Generated models consist of 21-22 states and a variable number of transitions (between 172 and 291). All operations involve “Read Holding Register” commands over several different sets of addresses.

These models look similar with each other and do not seem including sequences with periodical behavior (e.g., Figure 6.7). However, a more in-depth analysis reveals the following outcomes:

- Given the total number of possible transitions within a DTMC², the observed ones cover 40% – 60% of the total. This means that, on average, half of the transitions have not occurred even after five days of analysis.
- Not all observed transitions have the same probability. Figure 6.8 shows transitions with probabilities higher than 0.7. These transitions include more than 150K jumps each. This analysis brings to light a hidden sequentiality among the states that is occasionally broken by random delays. Just a third of the states has more than one transition with probability higher than 0.25

² $\#state$ to the power of 2, coming from $\#states \cdot (\#states - 1)$ transitions between couples of states plus $\#states$ transitions starting and ending in the same state

Figure 6.7: Screenshot of a DTMC representing the Modbus communication between SCADA Server 140 and PLC 203

(thus, including more than 130K jumps) and the 30 transitions showed in Figure 6.9 cover 99.8% of the total number jumps in the model.

- There are no transitions starting and ending in the same state. This means that no “Read Holding Register” operation is performed twice in a row.
- Transition distribution is not uniform as some states have a higher number of transitions than others.

Furthermore, Figure 6.9 shows that specific subgroups of states share the following two properties:

1. each group has one or more transitions connecting an inner state to an outer state.
2. each group has few transitions connecting among inner states. Moreover, these transitions form a path that connects all the states of the group and include the majority of the jumps ($\sim 99.9998\%$).

A possible explanation to these properties suggest the presence of a multi-threading OS process handling Modbus requests in the SCADA server. In fact, each thread is responsible for a subset of messages and, specifically, of an interval of Modbus registers that the SCADA Server wants to read. This hypothesis explains why it was impossible to find a clean sequence from the beginning (each thread can be randomly scheduled by the CPU) and why some transitions never appear in the model (threads are always created in a loop and operations within the same thread are always sequential).

Figure 6.8: Screenshot of the same DTMC of Figure 6.7 considering transitions with probability > 0.7 only

MMS Figure 6.6 illustrates the MMS network. From this network we capture 4 GB of traffic with more than 18.3M frames. TCP streams connecting the main PLC (coordinating all field devices) to the other PLCs involve on average 2M frames. The related sequences consist of about 600K events that result into models of 2 to 7 states. DTMCs have a variable number of transitions that cover between 35% and 75% of all possible transitions. The highest percentage relates to small DTMCs with two states (“Read Request” commands on two variables of field PLCs) and three transitions. Other DTMCs include a higher number of states. Figure 6.11 illustrates an example of this kind. As discussed for Modbus, there is a subset of transitions (8 out of 17) that covers 99.99% of the performed jumps and forms a clean path in the DTMC (Figure 6.12). Instead, differently from the Modbus case, the DTMC shows episodes of “strict consequentiality” (e.g., Figure 6.11 between the two states at the bottom of the figure). Although we do not know whether altering the order of these two states (and thus the two messages on the network) can cause problems to the control process, it seems their sequentiality is enforced by the system and, for this reason, it is likely to remain the same unless the control process changes.

Communications among the two SCADA Servers and the main PLC include almost the same number of frames (1.9M) and sequence events (570K) found in the communications discussed before. However, the related models change completely. These two DTMCs include about 280 states and 1280 transitions (Figure 6.13). This is due to two reasons:

- The variety of operations performed by the SCADA Servers is higher compared to PLCs.
- Most of the states of these DTMCs represent “Read Unconstrained Address” operations whose implementation is not defined by the standard. This makes

Figure 6.9: Screenshot of the same DTMC of Figure 6.7 considering transitions with probability > 0.25 only

every state identifiable by a byte string rather than its semantic meaning and, ultimately, allows to group together events that share the same identical data only.

It is worth noting that correctly parsing, and thus interpreting, “Read Unconstrained Address” operations would reduce the number of states. However, the analysis on these DTMCs shows the following:

- The percentage of observed transitions corresponds to the 0.02% of the total meaning that white-listing transitions could be more effective than in the Modbus case.
- About 45% of the states define “univocal relationships” between two other states. This relationship is described in the following terms: Y is a state that defines an “univocal relationship” between X and Z if transitions from X to Y are always followed by transitions from Y to Z. Moreover, these states are often chained together forming structures with strong consequentiality. For this reason, we know that specific subsets of MMS events always follow each other.
- DTMCs divide into two parts: a densely connected core and smaller groups of states at the edge of the model. Most of these small groups link to the core with few transitions that often involve only one jump. We argue these clusters are one measurable effect of human intervention. This hypothesis is supported by an analysis made on a process log file storing information about performed operations in the system. In fact, there is a match between human operations timestamps (from the log) and transition starting times (from the DTMCs). About 50% of the log entries labeled as “human operations” were recorded at the same time in which transitions involving the small groups

Figure 6.10: MMS network

of states discussed before were performed. Also, there is a match between the absence of human operations in the log and a the absence of transitions related to the same small groups of states. Finally, it is worth noting that the two time frames with the highest number of transitions and states from these small groups correspond to the only two operations recorded as “Suppress” in the log file. We know from the operators that these operations correspond to a manual stop of system alarms with a consequent reset of several system control parameters to a “normal” state likely generating numerous MMS messages in the network.

In the end, both analyses on Modbus and MMS emphasize a huge difference between PLC-to-PLC/RTU-to-RTU and server-to-PLC/server-to-RTU communications. In fact, the related DTMCs show that the former exhibits fewer states and transitions than the later because of a lower variety of events.

IEC104 Figure 6.6 illustrates the IEC104 network. From this network, we capture 51 GB of traffic with more than 203M frames. TCP streams connecting the SCADA Server and the RTUs include 70K to 96K frames and 13K to 24K IEC104 events. Most DTMCs have a number of transitions that covers 20 – 33% of the total. Figures 6.15 and 6.16 illustrate two examples of TCP streams with the highest number of frames. Both DTMCs can be logically divided in two parts according to the number of transitions. On the left, DTMCs show high variety of states and high density of transitions without distinctive paths. On the right, there are states including “single point information” commands only and clear single paths. As dis-

Figure 6.11: Screenshot of a DTMC representing the MMS communication between main PLC 151 and PLC 211

cussed for MMS, the right side of the models allows to make strong assumptions on the expected sequences of events.

Instead, Figure 6.17 illustrates a further server-to-RTU communication with fewer events (13K) but a greater number of states and transitions. Shown transitions covers just 0.04% of the total number (lower than previous cases) but maintains some distinctive paths. The analysis on IEC104 communications leads to the following outcomes:

- All DTMCs present sections of the models with no clear paths. These sections include often the same type of IEC104 events such as, “TESTFR” functions, “Measured Value - Normalized Value” commands, etc.
- At the same time, there are sections of the models that exhibit sequentiality. Despite the presence of multiple paths, an in-depth analysis shows that involved states overall refer to the same semantic information. In fact, every state include “Single Point Information” commands with a different number of object addresses. However, the set of object addresses at the end of every path is always the same. Moreover, the order in which the SCADA server reads those object addresses always remains the same.

It is worth noting that IEC104 traffic samples owns, on average, a lower number of events with respect to Modbus or MMS. This is mainly due to long delays among the involved messages (compared to an average of hundredths of seconds, for IEC104 this value increases of 3-4 orders of magnitude). Furthermore, σ_{ATE} of IEC104 transitions are higher compared to Modbus and MMS cases. This is due to IEC104 communications pushing scheme that reduces the number of messages on the network and makes transition timing less precise.

Figure 6.12: Screenshot of the same DTMC of Figure 6.11 considering transitions with probability > 0.19 only

Discussion Despite the increased stability of NCS communications compared to standard IT (e.g., long-running TCP sessions, limited set of messages sent in a loop), our tests on real ICSs communications reveal some challenges to sequence analysis. Tests show that the claimed stability does not always ensure clear paths in the generated models. Furthermore, specific examples such as the MMS server-to-PLC communication show the effect that a lack of knowledge about the involved messages has on the number of states and transitions. Nevertheless, we identify some properties that should be considered in the development of an S-IDS. All tests confirm that the number of transitions never goes close to the maximum permitted by a DTMC. This result suggests that network events do not combine into every possible sequence but a specific subset of allowed sequences exist. This is a necessary condition for an S-IDS. Furthermore, tests show that only a small fraction of the transitions appears most frequently. This information emerges from the DTMCs by filtering out transitions with a low number of jumps. Remaining transitions often form clear paths through the states, and thus, strengthen the hypothesis of a substantial sequentiality in the events observed in a network. This sequentiality can be occasionally weakened by random delays and human interferences.

This analysis leads to two sequence detection mechanisms. The first enforces strict sequentiality constraints defined by the models (e.g. exploiting the “univocal relationships” of Figure 6.11). The second focuses on the probability distributions with which these constraints occur (e.g. enforcing transition probabilities to always remain the same).

The first mechanism will tackle the “order-based” attacks described in §6.1 as these attacks involve sequences of events that are not in the observed models. Instead, the second mechanism will tackle “timing-based” attacks as these attacks involve allowed sequences that occur too frequently or rarely with compared to

the observed models.

6.3.1.4 Detection

Targeted anomalies can be of three types:

- **Unexpected Probability:** relates to known DTMC states and transitions whose related probabilities differ (by at least a value of θ) compared to the one observed during the training phase. This anomaly can reveal a time-based

Figure 6.14: IEC104 network

sequence attack when the repetition (or the lack) of occurrences of two or more events cause damage to either the control or the physical process (e.g., the repetition of two conflicting events shown in [160]).

While identifying the aforementioned anomalies, detection algorithms should avoid the following false positives:

- **Unknown but Correct State:** relates to an unknown state anomaly caused by an event that is not dangerous for the monitored infrastructure. This false positive can occur in case of uncommon user operations or changes either in the control or physical parameters. In situations like this, an S-IDS should identify the anomaly but label it as “harmless”.
- **Unknown but Correct Transition:** relates to an unknown transition anomaly caused by a new link between two events that does not interfere with the correct functioning of the monitored infrastructure. This false positive can occur in case of event delays or user interferences within control or physical processes. As in the previous case, an S-IDS should identify when these situations are not dangerous for the monitored infrastructures.

We argue that it is unlikely to have false positives related to unexpected probability anomalies. This mainly comes from the robustness of this parameter. Substantial changes in the probability values of a DTMC mean a considerable modification in the way events are correlated within the monitored infrastructure. Actions of this kind are not frequent in NCSs. For this reason an S-IDS should always notify unexpected probability anomalies.

Figure 6.15: Screenshot of a DTMC representing the IEC104 communication between SCADA Server 102 and RTU 122

Figure 6.16: Screenshot of a DTMC representing the IEC104 communication between SCADA Server 102 and RTU 177

Finally, it is worth noting that the impact of false positives also relates to the available training time. The more an S-IDS is able to train (the more data it acquires on correct events and patterns) the more it will be able to distinguish between normal and malicious activities.

Detection Mechanism In the detection phase we evaluate differences between training and detection models. The detection mechanism identifies any state of a detection DTMC that does not belong to any training DTMC. The same check is performed on new transitions that belong to known states (transitions belonging to unknown states are considered part of an unknown state anomaly). Finally, for each known transition, the detection algorithm computes the difference, or “distance”, between the probability values measured in the learning phase and the ones measured in the detection phase. This result identifies unexpected probability

Figure 6.17: Screenshot of a DTMC representing the IEC104 communication between SCADA Server 102 and RTU 105

anomalies related to single transitions or computed on an overall set of transitions belonging to the same state. We compute the distances between states and between transitions as follows:

$$d_S = \sum_{\delta \in T} \frac{|p_{\delta_{detection}} - p_{\delta_{learning}}|}{2} \quad (6.1)$$

$$d_\delta = |p_{\delta_{detection}} - p_{\delta_{learning}}| \quad (6.2)$$

where:

- T is the set of transitions belonging to state S
- $p_{\delta_{detection}}$ is the probability value of transition δ in the detection phase
- $p_{\delta_{learning}}$ is the probability value of transition δ in the learning phase in case this transition exists (0 otherwise)

Both d_S and d_δ are between 0 and 1.

A detection mechanism based on probability distances needs a definition of a threshold θ . If d_S and d_δ exceed θ , the detection mechanism triggers an alert including information about involved states and transitions. The definition of the threshold strongly impacts the performance of a S-IDS. If θ is too high the accuracy of the S-IDS increases (fewer false positives) but its completeness decreases (more false negatives). On the other hand, if θ is too low the high number of false positives is likely to make the S-IDS unusable. Also, the threshold value depends on the nature of the monitored infrastructure. NCSs that show higher variability in the communication patterns (e.g., BASs) need higher thresholds as well.

Finally, it is worth noting that this detection mechanism can work either online (i.e., running on live systems) or offline (e.g., on an early captured network trace).

Offline detection takes place only at the end of the analysis on detection traffic. Once the S-IDS has gone through all the network frames, it builds the DTMC and computes all distances. Instead, online detection requires a slightly different setup. In this case, DTMCs are progressively populated while the network frames reach S-IDS sensors. At the beginning, the number of jumps and transitions in the DTMCs is necessarily low. This likely causes the probability values to diverge from those observed during the learning phase. For this reason, online detection needs to wait a reasonable amount of time τ such that the number of network frames observed in the detection phase is comparable with the number of frames captured and used in the learning phase.

6.3.1.5 Gaining System Knowledge

Proposed modeling and detection mechanisms work the same on any NCS communication. However, there are several aspects that can improve the effectiveness of the S-IDS and, consequently, the quality of the overall monitoring. In what follows, we discuss possible improvements of the S-IDS detection mechanism.

Fingerprinting NCS fingerprinting may interact and improve an S-IDS in several ways. First, a fingerprinter can detect a malicious intrusion in a network before an S-IDS takes action. Based on its own analysis, a fingerprinter may know which devices belong to a network. Any new device that does not match any previous fingerprint could be alerted to the operators without any further activity carried by the S-IDS. Furthermore, the operators may decide to focus the analysis of the S-IDS on a subset of devices only. This may be due to the importance of a specific category of devices (e.g., all SCADA servers) compared to the others as well as some limitations of the detection approach. This last aspect concerns the work of the S-IDS on irregular and unpredictable communications. For example, devices such as engineer's workstations are used by operators to check on NCS devices in the network. These devices and the related network communications do not usually follow any standard pattern and may have a negative impact on the S-IDS (too many false-positives). Finally, NCS fingerprinting can support the S-IDS with respect to the choice of the right detection threshold. It is not mandatory to use the same detection threshold everywhere in the NCS network. In fact, a more precise approach may take into account the nature of each monitored device and consequently set up the right threshold improving the effectiveness of the S-IDS. As we observed in the tests, PLCs and RTUs show, on average, more stable patterns compared to SCADA servers. For this reason, the identification of a device could be forwarded from the fingerprinter to the S-IDS before the detection mechanism takes action.

Event Importance Not all the events have the same importance within an infrastructure. First, different messages, commands, and values have different roles in the control processes and the physical processes beyond. Furthermore, some messages, commands, and values are more likely to be involved in an attack against an infrastructure. If this information is known, it is possible to enhance the proposed

detection mechanism by focusing the analysis on just the “important” events. An S-IDS that leverages this information reacts differently when an anomaly relates to an “important” or an “unimportant” event. At the same time, the S-IDS triggers an alert on anomalies related to “unimportant” events only if the change in the communication is consistent. This mechanism, if correctly implemented, can significantly reduce the number of false positives.

Evaluating the importance of an event requires in-depth knowledge of the monitored infrastructure. For this reason, the easiest way to assign importance values to the events is leaving this to the operators. However, the number of choices that an operator can take is usually limited compared to the number of events an S-IDS processes. Therefore, we propose two different (yet inter-operable) approaches to apply and use importance values within DTMC models.

The first approach relies on making a priori assumptions on the importance of some specific events. For what concerns NCS communications, we can roughly divide NCS events into two categories, namely “monitoring” and “control”. The former category includes network messages used to gather information about the status of a process or a device. The latter category includes all messages used to modify this status and, thus, actively interact with the processes of an NCS. From attackers’ perspectives, “command” events are more likely to be important than “monitoring” ones. If the attacker wants to damage the infrastructure by subverting the control process.

The second approach relies on computing the importance of DTMC events. If we have information about some “important” events we can try to derive information about other events of the same DTMC based on a concept of proximity. This means that: the closer an event is to an “important” event the higher its importance score will be. To achieve this, we distribute the importance of the events among all the states of the DTMC according to transition probabilities.

Algorithms 2 and 3 describe an implementation of this approach. In this example, the importance of a given state S in the DTMC is computed as the probability to go from S to an “important” state. The higher probability p becomes the more important a state S is.

Computing the importance of the states changes detection score computations of 6.1 and 6.2 as follows:

$$d_S = w_s \cdot \sum_{\delta \in T} w_{s_\delta} \cdot \frac{|p_{\delta_{detection}} - p_{\delta_{learning}}|}{2} \quad (6.3)$$

$$d_\delta = w_s \cdot |p_{\delta_{detection}} - p_{\delta_{learning}}| \quad (6.4)$$

where:

- w_s is the importance of state S
- w_{s_δ} is the importance of a neighbor state linked to S by transition δ
- w_{source} is the importance of the state whose δ belongs

Both results from 6.3 and 6.4 are still included between 0 and 1 if the maximum value of importance is 1 (e.g., we assign one to every “important” state).

Algorithm 2: computeImp()

```

forall  $S \in DTMC$  do
   $visitedStates \leftarrow S$ ;
   $p = 1$  //Current probability;
  return computeImp( $S, p, visitedStates$ );

```

Algorithm 3: computeImp($A, p_{past}, visitedStates$)

```

 $p = 0$ ;
forall  $s \in neighbors(S)$  do
  if  $s \in importantStates$  then
     $p += p_{past} * probability(S, s)$ ;
  else if  $s \notin visitedStates$  then
     $visitedStates \leftarrow s$ ;
     $p += computeImp(s, p_{past} * probability(S, s), visitedStates)$ ;
return  $p$ ;

```

Event semantics Creating events from network messages and arranging those events in a model take advantage of semantics. In fact, we look at the meaning of a message to understand the kind of event it represents as well as we group events together in a DTMC state when they relate to the same activity. However, when it comes to alerting anomalies, the detection mechanism just verifies the presence of states and transitions or computes the differences with respect to the observed probabilities.

A further, more in-depth, semantic analysis of an anomalous state (or transition) has a twofold value. First, this analysis may lead the S-IDS to revise its results and drop alerts (e.g., anomalies within DTMC may still fit infrastructure expected behavior at a different level of abstraction). Second, the same analysis allows the operator who is reacting to the alert to better understand the context in which such alert has been raised.

For example, Figure 6.18 models events related to the monitoring of a temperature. Each state defines intervals of n degrees (e.g., State A represents $[0^\circ, 10^\circ]$, State B represents $[10^\circ, 20^\circ]$, etc.). The value of the variable ranges between 0° and 100° . Furthermore, let us suppose two anomalous transitions appear in the model during detection (Figure 6.19).

According to the detection algorithm, the two anomalies have the same detection score. However a further step in the analysis could help the S-IDS to rank the two anomalies and thus give a better suggestion to the operator. In fact, the two transitions are very different in meaning as one concerns a temperature increase of $\sim 20^\circ$ while the other represents an increase of $\sim 100^\circ$. While an imprecise measurement of a sensor or a missing message can explain the first anomaly, the same considerations do not apply for the second one.

A comparative analysis among all transitions would easily show that the aver-

Figure 6.18: Example of a DTMC built over a learning phase

Figure 6.19: Example of a DTMC built over a detection phase

age temperature variation is $\sim 10^\circ$. This analysis shows that the first anomaly is quite close to a normal behavior and emphasizes the distance between the second anomaly and the rest of the known transitions. For this reason, an S-IDS could decide to ignore the first anomaly or to forward comparison information to the operator.

The heterogeneity of infrastructures and events an S-IDS must face may undermine the applicability of this last enhancing approach. However, there are situations where the presence of a further in-depth semantic analysis on the observed events may be supported by available information coming from external or internal sources (e.g., documentation or additional data in the messages).

Transition timing Effective monitoring of sequences of events strongly relies on stable transition timing. The more events follow standard time patterns the easier identifying anomalies is. Attributes ATE and σ ATE provide information on the likelihood of a transition changing over time and may suggest when to perform detection on the related involved events. In fact, transitions with low σ ATE always relate to a stable behavior of an automated control process. On the other hand, transitions with high σ ATE are usually the measurable effect of human behavior. An S-IDS should not consider transitions of the second type over a learning phase as this may create meaningless models.

6.3.2 Evaluation & Discussion

In order to evaluate the effectiveness of our detection approach, we tested the S-IDS again on Modbus traffic coming from the water treatment and purification facility. First, we analyzed the clean traffic. The primary purpose of this test was to evaluate whether the proposed detection mechanism generated false positives. Then, we introduced some artificial sequence attacks into the trace to show S-IDS effectiveness in detecting sequence attacks.

6.3.2.1 Sequencing and Modeling

We split 24 hours traffic into 4 hours training and 20 hours detection. Over the training, we observed 20 different Modbus connections: 9 PLC-to-RTU, 3 PLC-to-PLC, and 6 PLC-to-SCADA Server, 1 SCADA Server-to-SCADA Server and 1 HMI-to-SCADA Server.

As shown in §6.3.1.3, most Modbus connections involved just one or two Modbus requests and responses sent periodically (e.g., once every second). Connections between PLCs and SCADA Server had instead higher variety of messages. As discussed before, not all observed transitions have the same probability. For example, Figure 6.20 shows that most transitions have low probabilities (thinner lines) while a small set of them have a probability greater than 0.9 and form a clear path through all the states of the model (thicker lines). Once again, from this example, we know that the sequence of events given by transitions with high probability breaks occasionally due to random delays or human interventions.

6.3.2.2 Detection Mechanism

We compute our detection scores with 6.1 and 6.2 on the rest of the traffic captured on the same network. We minimized the threshold θ to 0.1 to alert any significant change within the transition probability sets. No malicious traffic was included in the dataset, therefore this test verified the resilience of the detection mechanism against false positives. The S-IDS raised 211 alerts (Table 6.1) with a rate of 1 alert every ~ 40 minutes. A follow-up analysis revealed that 96% of the false positives relate to new transitions caused by network delays on “read” operations within one of the aforementioned connections between PLCs and SCADA server. Other anomalies are: six new states (“write” messages not observed in the training phase), two anomalous states and an anomalous transition (belonging to one of the previous anomalous states). The two anomalous states relate to a “read” and a “write” operations respectively. The “read” is alerted because of small changes within its transition set (transition probabilities changed within a maximum of $d_\delta = 0.05$). Instead, the “write” operation is alerted because of the detection score of the anomalous transition ($d_\delta = 0.2$). The information given by the operators confirms that this anomaly relate to a change in the status of a pump. It is worth noting that this change is observed in the models built over the training phase. However, the low number of events in that specific Modbus communication makes the probability value of the transitions increase substantially.

Figure 6.20: Screenshot of a DTMC representing the Modbus communication between a SCADA Server and a PLC

	Unknown State	Unknown Transition	Unexpected Probability
# Alerts	6	202	3

Table 6.1: Alert distributions with non-malicious traffic

To prove the effectiveness of the S-IDS we injected two sequence attacks into the traffic. First, we inverted two write messages concerning the control of two different pumps (Attack 1). Then, we triggered several start and stop commands on the same pump (Attack 2). The choice of these two attacks follows the two threat scenarios presented at the beginning of this chapter. Due to the criticality of the water infrastructure, it was not possible to test the attacks in the actual network. For this reason, we used the same 20 hours of real traffic plus the attacks to the S-IDS. Also in this case, the S-IDS raised the previous 211 false positives for the reasons discussed above. In addition, the S-IDS raised also 8 correct alerts detecting both the attacks. The first attack generated two new transitions (one following the other on the targeted “write” events). Figure 6.21 shows the first attack modeled in the DTMC. The second attack caused two anomalous states, a new transition and three anomalous transitions with a substantial change in the probability values (the transition linking the “start” event to the “stop” one increases from ~ 0.001 to ~ 0.99). Figure 6.22 shows the second attack and its effects on the DTMC. It is worth noting that there is a difference between the new

transitions generated by the two sequence attacks. In the first case, transitions have low probabilities compared to the new transition created in the second case. This is due to the differences between the two attack scenarios. In fact, the first scenario relies on subverting just a specific operation known to be potentially dangerous while the other scenario relies on repeating the same operation multiple times.

Figure 6.21: Screenshot of a DTMC representing the Modbus communication targeted by the first attack

6.3.2.3 Enhancing Detection

To reduce the rate of false positives we used one of the enhancing methods proposed in §6.3.2.2. In that section, we argued that an attacker is likely to target communications that involve commands or write operations. Therefore, we considered “important” every command and “write” operation observed in the network and assigned to each related state in the DTMCs a value of “importance” equal to 1. Furthermore, we run algorithms 2 and 3 to compute the “importance” of every other event in the DTMCs.

With this new setup we repeated the two tests discussed before. In this case, the test raised just 9 false positives instead of 211, with a rate of 1 alert every two hours and a half. This result drastically improves the usability of the S-IDS even if, from a practical point of view, this rate may be still difficult to handle for an operator. Likewise, with the malicious traffic, we got 17 alerts. The 8 correct alerts identified before being effects of the sequence attacks were confirmed with the enhanced detection. These results show the impact that some knowledge about the “importance” of the events has on detection.

Figure 6.22: Screenshot of a DTMC representing the Modbus communication targeted by the second attack

6.4 Section Conclusion

In this section we discussed a new kind of APT, namely the sequence attack, and its effect within NCSs. We observed that state-of-the-art security solutions against sequence attacks still face limitations. This is due to several challenges such as identifying and analyzing the correct sets of information and modeling them into proper data structures. Moreover, such analysis must be robust against message delays and human operations that can occasionally break the stability of ICS communications.

To improve detection of sequence attacks we presented our S-IDS. The layered architecture of the S-IDS encompasses in-depth analyses of NCS data and leads to a comprehensive description of device behavior over time. Furthermore, we tested the feasibility of our analyses and modeling techniques on data traces coming from real critical infrastructures. Finally, we tested our detection techniques by using traffic from a water treatment and purification facility and by crafting synthetic attacks. Results showed that our S-IDS is able to correctly identify sequence attack instances as well as keeping the number of false positives low.

Future research steps encompass the identification of an even more comprehensive set of detection mechanisms that can be used in a broader variety of NCSs. As differences among NCSs can be substantial (e.g., physical processes, control strategies) we need to consider more data from a variety of sources. Furthermore, anomalies in communication patterns do not always relate to attacks but can be generated by a number of other causes (e.g., device failures). For this reason, the S-IDS could benefit from further temporal and statistical analysis specifically aiming

to recognize the causes of identified anomalies.

Finally, it is worth noting that leveraging semantics of NCS communications is a powerful way to enhance the S-IDS and improve its effectiveness. In the following chapter we will keep investigating on this concept and we will automate knowledge extraction to further improve intrusion detection.

CHAPTER 7

Intrusion Detection in BAS

In this section we present the *document-based intrusion detection system* (D-IDS). The D-IDS implements all building blocks of the framework illustrated in figure 7.1 and inherently combines system knowledge acquisition and network security together in one comprehensive solution.

Figure 7.1: DIDS building blocks

In what follows, we show possible attacks against BACnet infrastructures and we present the evaluation environments used to investigate our approach. Then, we discuss design and implementation of the D-IDS. Finally, we test the D-IDS, analyze results and draw some general guidelines on how to apply the same approach to different NCSs.

7.1 Attacks on BACnet

BACnet defines a limited security architecture providing peer and operator authentication along with data confidentiality and integrity ("Clause 24 — Network

Security” [48]). However, none of this is implemented in available products [169]. This leaves BACnet infrastructures vulnerable to numerous cyber-threats [170–173].

We categorize attacks on BACnet into three main groups: snooping, denial of service (DoS), and process control subverting. This categorization derives from the list of BACnet protocol threats described in [170].

Snooping attacks concern stealing information about a specific building automation system. To achieve this goal, these attacks require access to the building automation system network. Once inside, attackers can take advantage of BACnet services such as `ReadProperty` and `ReadPropertyMultiple` to gain knowledge of the BACS. This includes device models, locations, status, and information on their BACnet support (e.g., which BACnet services and objects they implement). Attackers may need this information to understand the infrastructure and pave the way to further intrusions. However, snooping attacks do not disrupt any process of the building automation system.

Differently, DoS attacks try to interfere with control processes by making controllers unreachable for operators. This category only considers DoS attacks that are performed through the use of BACnet routing features (e.g., malicious modifications to the BACnet routing tables) and leaves other kinds of DoS out of its scope. As for the snooping attacks, DoS attacks need malicious users to have access to the network. Moreover, this kind of attacks requires information about the network layout. Attackers can achieve their goal by sending BACnet messages, such as `Initialize-Routing-Table`, to modify a BACS’ routing tables. In this way, operators lose visibility on single devices or even entire sections of the building automation system.

Finally, process control subverting includes those attacks that directly modify control processes and, consequently, interfere with physical operation. This kind of attacks requires more skilled attackers with sufficient knowledge about the building automation system functioning. In this scenario, attackers exploit specific controllers by using several different BACnet services, such as `WriteProperty` or `DeleteObject`, to change the BACSs’ structures and operations. This leads to a loss of control by the operators and, consequently, leads to risks for components and people.

7.2 Evaluation Environments and Settings

For this work, we analyzed two different building automation installations over more than two months of constant operation. The first building automation system belongs to the University of Twente in the Netherlands and is in charge of supervising utilities and services provided to the university campus. Its duties encompass energy consumption control, HVAC, and room monitoring and management (e.g., pressure and temperature control, shading, etc.). The second building automation system belongs to the Lawrence Berkeley National Laboratory (LBNL) and supervises several services on its premises. The LBNL process control focuses mostly on room monitoring and energy consumption for the Lab facilities. Both infras-

tructures deal with hundreds of BACs from several different vendors (including Siemens, Automated Logic, Kieback&Peter).

The IDS we deployed at the University of Twente linked to a SPAN port on a switch directly connected with the SCADA servers monitoring the whole building automation system. The same switch is responsible for routing most of the traffic of the building automation network. This allowed us to capture and analyze most of the BACnet messages exchanged by BACs. Differently, at LBNL we could monitor only a subset of the building automation system by linking to a switch in charge of connecting BACs inside one building. However, this was sufficient to automatically gather the information needed for our approach to craft the specification rules.

The two infrastructures generally showed similar traffic patterns. Several BACs shared the same sets of objects and used the same kind of messages to exchange information. Furthermore, both UT's and LBNL's traffic samples included numerous BACnet routing messages (e.g., *Who-Is*, *I-Am*, *Who-Has*, and *I-Have*) organizing communication paths within the two networks. However, the two infrastructures presented some differences related to communication and control strategies (e.g., all BACs deployed at UT used confirmed services thus requiring acknowledgments from message recipients while some devices at LBNL used just unconfirmed ones). Particularly, the employment of BACs from different vendors led operators to employ individual procedures implemented through the use of **ConfirmedPrivateTransfer** BACnet services. Such services are used to invoke proprietary or non-standard routines in remote BACs.

Once deployed, we assume that our system is able to capture real-time traffic of the monitored building automation system in a completely passive fashion and to retrieve documentation publicly available on the Internet. This allows to gather the information we need to build specification rules and implement effective detection.

On the other side, we assume attackers can gain full access to the network as well. We consider this happening in a way that is similar to standard IT environments (e.g., phishing, software vulnerability exploitation). Tools such as Shodan [174] show how easy it is to find building automation networks exposing their devices to the Internet. Once inside, attackers can obtain a convenient viewpoint on the building automation control processes. Two key factors support this assumption. First, most building automation protocols take advantage of broadcast communications to exchange information among devices (e.g., routing notifications). This already allows attackers to easily observe a large part of the traffic. Secondly, the hierarchical structure of common building automation networks steers valuable information messages towards servers such as DCS servers. By gaining access to one of these servers, attackers can observe most of the traffic within the building automation system.

Within a building automation network, attackers may use attacks outlined in §7.1 to gain knowledge on, or subvert, the correct functioning of the building automation system. In this last scenario, any safety feedback in place can usually be overridden [175]. Therefore, attackers can put infrastructure components under stress, possibly threatening human safety when it comes to devices such as electrical equipment.

7.3 Security Solution

7.3.1 Documentation-based Intrusion Detection

Our approach works towards automated development of specification rules for network security monitoring. Setting up and customizing a specification-based IDS for a particular infrastructure requires a large amount of information about the monitored system, implying a substantial manual effort in gathering and refining the specification rules. As details of the infrastructures are often described within specs and configuration files, especially in many NCS environments, the process of collecting this information—and, consequently, the development of the actual IDS—can be automated to a fairly high degree through the following steps (see also Figure 7.2):

Figure 7.2: Documentation-mining approach based on the framework proposed in §4

(1) System Discovery gathers information about the monitored NCS. In this step, our system analyzes the network traffic in order to: 1) identify devices communicating on the network (e.g., models, brands); 2) determine role and purpose of each identified device (e.g., a device is a controller, an HMI, etc.).

Every time the system collects enough information about a specific device it proceeds with the next step.

(2) Feature Lookup implements a set of information retrieval techniques to gather knowledge about devices identified during System Discovery. The purpose of this step is to: 1) find verified information (e.g., specs, configuration files) about the infrastructure devices; and 2) select features and constraints from the retrieved documents and arrange results in a structured form.

A successful Feature Lookup relies on the assumptions of *availability* and *linkability* outlined in the introduction. The assumption of *availability* implies the existence of documents about infrastructures and components that are automatically retrievable. This requires the information to be provided in electronic form and being suitable for parsing. Also, this assumption includes an assurance on the authenticity of the retrieved information (e.g., by the use of reliable sources, by the employment of secure retrieval techniques). The assumption of *linkability* guarantees that information derived by retrieved documentation can be checked by the system against observations within the traffic (e.g., messages, variables, etc.). Particularly, after the identification of network devices and the successful retrieval

of their related constraints from the documentation, the assumption of *linkability* enables assigning effective specification rules to the right targets.

(3) Rule Definition uses the knowledge obtained in the Feature Lookup to craft the specification rules. To achieve this goal, the system needs to: 1) select identified information from Feature Lookup; 2) translate this information to specification rules.

We focus our specification-based intrusion detection on *controllers* (e.g., BACs for building automation systems, PLCs for ICSs, ECUs for in-vehicular networks). This decision comes from the key role these components have within NCSs. In fact, controllers are involved in any monitoring and control operation of the infrastructure either autonomously or accessed by operators. Furthermore, controllers are likely targets for attackers (as illustrated in §7.1). We observe that NCS controllers share a number of properties. First, every controller *employs a limited set of variables to fulfill its function*. These variables can go from simple memory addresses to complex objects but *often have predetermined types*. Moreover, all controllers *use a limited set of methods (or services) to access and manipulate variables of other controllers*. Finally, *each variable can assume a limited range of values according to its type or the physical characteristic it represents*. We leverage these shared properties to define a set of general constraints, or *abstract rules*, checking NCS variables' *types, values* and access *methods*. These abstract rules are the seeds we use to automatically generate specialized specification rules. To achieve this, we define a mechanism that maps information retrieved in the Feature Lookup step to the abstract rules. This process automatically completes the abstract rules and, as a result, customizes detection for the monitored NCS.

Once a rule is defined, it becomes active and, thus, part of the detection mechanism. During detection, an active rule verifies if its related constraint is fulfilled or not. When this last condition becomes true, the system triggers an alert for the user.

(4) Semantic labeling involves the action of collecting human readable information to label IDS alerts. This information may relate to: 1) an assessment on the criticality of the event signaled by the alert; 2) the identification of the possible causes. Enhancing the alerts allows to improve user's understanding of the situation and increases his/her chances to react in a timely manner.

Having presented the phases in a generic way, we now describe our experimental setup and, then, how we have instantiated them to build a specification-based IDS for BACnet-based building automation systems.

7.3.1.1 Implementation Background

We implement our approach using the Python programming language [176] and Bro [117]. Bro is a network traffic analyzer employed in different domains such as security, performance measurement and network monitoring. The system comes with comprehensive built-in functionalities for traffic analysis and supports several

network protocols ranging from standard (e.g., HTTP, FTP) to domain-specific (e.g., Modbus [37] or DNP3 [45]). Bro provides a Turing-complete scripting language that allows users to select and analyze network events (e.g., connection establishments). We choose to describe specification rules through the “Bro scripting language” because of its efficiency and expressiveness. We developed a BACnet parser for Bro using Spicy [177], a parser generator whose specification language allows users to effectively define protocols’ syntax and semantics.

We publish the BACnet parsing code for Spicy, as well as the Python scripts, as open source software.¹ However, we cannot open-source the Bro code containing the rule checks due to the sensitivity of the information about the two building automation systems sites.

Finally, in the network of the University of Twente, we capture BACnet traffic through PF_RING network sockets [178]. PF_RING greatly increases the packet capture speed with respect to other software solutions such as the use of “mmap” or “device polling”. As we rely on gathering as much information of the infrastructure as possible during System Discovery we exploit PF_RING to lower BACnet packet loss rate to a minimum.

7.3.1.2 System Discovery

To identify BACs we implement three different techniques that we term: “*BACnet Device Object analysis*”, “*BACnet Address linking*”, “*BACnet Property set fingerprinting*”.

The first technique directly follows from the protocol standard and relies on the mandatory presence of a `Device` object in every BACS device. The `Device` object defines “a standardized object whose properties represent the externally visible characteristics of a BACnet device”. Among these properties there are: `Object_Name`, `Vendor_Name`, `Vendor_Identifier`, `Model_Name`, `Firmware_Revision`, `Application_Software_Version`, `Location`, and `Description`. Most of these properties are set by vendors and provide information on a device’s identity (e.g., `Model_Name`) and role (e.g., `Description`). BACnet services such as `ReadProperty` and `ReadPropertyMultiple` can access those properties. As these services are widely employed by user interfaces and logging servers to automatically update data related to infrastructure’s components, information on `Device` objects regularly passes through the network and, thus, is available to System Discovery. As the `Object_Identifier` property of a `Device` object is a parameter that uniquely identifies a device in a BACnet network, a message such as the one in Figure 7.3 allows us to identify a BACS and understand its purpose. In the Wireshark screenshot example, BACS with identifier “17001” is a “Blue ID S10 Controller”.

For BACnet objects of other types, since no information can be extracted from the IP address (multiple BACnet devices may share the same IP address), a further parameter allows to identify message sources and destinations: the BACnet address. As for the `Device` object’s `Object_Identifier`, the BACnet address (together with the `Network_Identifier`) is unique within a BACnet network. In

¹<https://github.com/specification-mining-paper-usenix-2016/specification-mining>

Figure 7.3: “BACnet Device Object analysis” example

the “BACnet Address linking” technique, the BACnet address bridges the gap between a known **Device** object and any BACnet object included in the same BACS. Figure 7.4 shows an example of this analysis. When “device 4001” is known (as a result of the previous technique), any message carrying both the related **Device** object’s **Object_Identifier** and the BACnet Address allows us to link the two parameters (Figure 7.4a). Any later message then carrying the BACnet address along with a further object (e.g., “Analog Value 171”) enables linking to the corresponding device (Figure 7.4b). This technique works well because **I-Am** messages pass the network frequently to ensure visibility of all BACnet objects.²

Finally, if no information can be extracted from **Device** objects or BACnet addresses, System Discovery can benefit from observations of the BACnet properties. As discussed in §2.2.2.1, the BACnet property set is extensible. Every object of a BACS has a set of standard and proprietary properties that form a “fingerprint” of that object and device. The third technique assumes that two objects sharing the same fingerprint are likely to be of the same kind. During System Discovery, it is possible to create a database of identified fingerprints each one pointing to the corresponding BACS (identified with the previous two techniques). Whenever an unknown object presents a property set already in the database, the system infers the most likely related device.

Experiments Previous work by us shows that traditional fingerprinting techniques are usually ineffective on most NCSs [7]. In our tests, tools such as Nmap [129] and P0f [133] were able to identify just a limited number of Windows and Linux workstations. The techniques presented above proved more effective. Thanks to frequent **ReadPropertyMultiple**, our system was able to gather information on most BACSs. Moreover, BACnet address linking and BACnet property set fingerprinting allowed the system to link most of the observed BACnet objects to identified devices. At the end of System Discovery, we gathered information on

²The technique can also directly use messages carrying **Device** object information if the source BACnet address is present in the header. However, for this kind of messages, having the BACnet Address fields is not mandatory.

(a) I-Am message

(b) ReadProperty message

Figure 7.4: “BACnet Address linking” example

~15k BACnet objects belonging to the 445 devices shown in Tables 7.1 and 7.2.

Thanks to the information from the operators, we know that we correctly identified 98.2% of the BACnet actually deployed (445 out of 453 devices). Eight devices did not link to any useful BACnet message or identifiable property set. However, these devices convey almost no information over the network (a few hundreds BACnet messages over two months of capturing compared to an average of tens of thousands) and did not involve any notable equipment. Identifying the aforementioned 445 devices took just a few hours of monitoring.

7.3.1.3 Feature Lookup

Searching for documentation on identified BACnet is possible because the two assumptions of *availability* and *linkability* hold for BACnet-based building automation systems. Verified information about BACnet is *available* within PICs and EDE files. This information includes BACnet’s vendors, models and even refers to specific BACnet objects, thus is *linkable* to what we observed over System Discovery.

Feature Lookup targets both online and offline documentation. On the one hand, we use Google APIs to search and retrieve publicly available documents such as PICs on the Internet. On the other hand, we retrieve EDE files from private repositories in the installations. Both cases allow for document authenticity. In the former case, we narrow the search to a subset of reliable sources such as vendors’

# of devices	Vendor	Model	Role
5	Kieback&Peter	DDC4000	DCS
15	Priva	HX 80E	Router
7	Priva	Compri HX	Controller
25	Priva	Compri HX 3	Controller
36	Priva	Compri HX 4	Controller
12	Priva	Compri HX 6E	Controller
85	Priva	Compri HX 8E	Controller
2	Priva	Blue ID S10	Controller
16	Priva	Comforte CX	HMI
2	Delta Controls	eBCON	Controller
3	Siemens	PXG80-N	Controller
3	Siemens	PXC64-U	Controller
3	Siemens	PXC128-U	Controller
3	Siemens	PXR11	Controller
3	Siemens	PXC00-U + PXA30-RS	Controller
1	Unknown	Unknown	-

Table 7.1: University of Twente - BACS device list

# of devices	Vendor	Model	Role
23	Automated Logic	LGR	Router/Gateway
14	Automated Logic	ME	Controller
11	Automated Logic	SE	Controller
159	Automated Logic	ZN	Controller
1	Automated Logic	WebCTRL	HMI
9	Johnson	NAE	Controller
1	Johnson	NIE	Controller
4	Paragon Controls Inc.	EQ	Controller
4	Sierra	BTU Meter	Energy meter
4	Sierra	FFP	Controller
1	Tracer	UC400	Controller
2	Niagara	AX Station	SCADA server
7	Unknown	Unknown	-

Table 7.2: LBNL - BACS device list

websites and reputable third parties (e.g., BACnet International Laboratories³). In the latter case, we assume a secure connection to a trusted dataset managed by the operators.

Once a BACS links to one or more of these documents, our system parses the documents looking for useful information. According to BACnet specifications, a PICS has a standard template and we observe that most PICSs are closely modeled to it. Figure 7.5 shows three extracts from the PICS of the “Blue ID S10 Controller” mentioned in the previous section.

As outlined in §5.2.2.1, each PICS provides a description of the related BACS and the BIBBs it implements (Figure 7.5a). Moreover, PICSs include information about supported BACnet objects and properties, as well as their characteristics (Figures 7.5b and 7.5c).

EDE files also follow a standard template but they use a simpler “comma-separated values” (CSV) format. Each EDE file presents details of a specific BACS (Figure 7.6 shows an extract of Device 4001 EDE file). Data includes all implemented BACnet objects (e.g., “device 4001” owns “Analog Value 171”, “Multi-state Value 15”, etc.) and their descriptions. Furthermore, EDE files include information about `Present.Value` properties with value ranges (e.g., “Analog Value 171” can vary from `min-present-value 0` to `max-present-value 100`).

Experiments The program we implemented to search for online documentation uses the outputs of System Discovery (vendors and models) and further keywords such as “PICS” to retrieve information about identified BACSs. The system ranks Google results coming from public repositories (e.g., www.bacnetinternational.net) and the web by quantifying the presence of the keywords in document titles. For example, the “Blue ID S10 Controller” links to a PDF document titled “BACnet.PICS.Blue.ID.S10.Controller.pdf” (Figure 7.5). With this technique we identified a PICS for 99.3% of the devices deployed in the two building automation systems (442 out of 445 among the devices identified in the System Discovery step). Two ‘Siemens PXR11’ and one ‘Paragon Controls Inc. EQ’ were the only devices that did not link to any PICS. However, we could not find the related PICSs even by a manual search either.

Offline research targeted specific devices directly. While online documentation always provides general information about BACSs of a certain kind (e.g., all “Blue ID S10 Controller”), offline repositories provide detailed information related only to devices deployed in the monitored building automation system. For this reason, instead of vendors and models we searched through the available documents using device `Object.Instances`. For example, starting from “Device object 4001” from System Discovery, we found an EDE file titled “Controller 4001_EDE.csv”. While LBNL did not provide any configuration file, operators from the University of Twente shared with us 10 files of this kind. While they confirmed that there was indeed an EDE file for every deployed device, they could not grant us unlimited access to all of them due to information sensitivity. For this reason, the operators chose the 10 files based on roles and purposes of the related devices. Each file we

³<http://www.bacnetinternational.org/>

Product information	
Date	2013-07-23
Product name and model number	Blue ID S10 Controller
Application software version	1.0
Firmware revision	1.00
BACnet protocol revision	9
Product description	The Blue ID S10 Controller contains a powerful microprocessor. It can be easily programmed for building automation purposes. The processing speed and computing power mesh seamlessly with the requirements of modern and integrated systems. The controller uses a reliable operating system that ensures quality and operational security. It is fully controllable via BACnet including commissioning, writing, reading, alarm and event handling.
BACnet standardized device profile	BACnet Building Controller (B-BC)

Vendor information	
Vendor name	Priva
Vendor ID	105
Contact information	PO Box 18 2678 ZG The Netherlands www.priva.co.uk

Supported BIBBs	
DS-RP-A	Data Sharing - Read Property - A
DS-RP-B	Data Sharing - Read Property - B
DS-RPM-A	Data Sharing - Read Property Multiple - A
DS-RPM-B	Data Sharing - Read Property Multiple - B

(a) PICS extract 1

Standard object types supported ¹ Dynamically creatable and deletable	
Accumulator	no
Analog Input	no
Analog Value	no
Binary Input	no
Binary Output	no
Binary Value	no
Calendar	yes
Device	no

(b) PICS extract 2

Supported properties per object type	Required or optional	Readable (R) or readable/writable (R/W)	Additional comments
Accumulator			
Object_Identifier	required	R	
Object_Name	required	R	
Object_Type	required	R	
Present_Value	required	R/W	
Description	optional	R	
Status_Flags	required	R	

(c) PICS extract 3

Figure 7.5: PICS example

PROJECT_NAME		UTWENTE4\Hoofdschema						
VERSION_OF_REFERENCEFILE		126						
TIMESTAMP_OF_LAST_CHANGE		14-04-2015, 11:11:59,890						
AUTHOR_OF_LAST_CHANGE		Regel Partners						
VERSION_OF_LAYOUT		2						
device obj.-instance	object-name	object-type	object-instance	present-value-default	min-present-value	max-present-value	unit-code	description
4001	AV - 171	2	171	0	0	100	98	setp. KM1\Y1 - stuursi
4001	MV - 15	19	15	1				Selectie nieuw\Select
4001	AV - 11	2	11	100	0	32767	71	Selectie nieuw\Bedrij
4001	AV - 42	2	42	0	0	2	95	Selectie nieuw\1e ge
4001	AV - 43	2	43	0	0	2	95	Selectie nieuw\2e ge

Figure 7.6: EDE file example

obtained described a BACS identified over System Discovery.

The aforementioned concerns about information sensitivity refer to the initial manual analysis we had to perform over the EDE files and would not hamper the applicability of our approach. In an ideal deployment, one would have a secure connection between the IDS and the machine storing the EDE files, without any human activity involved for retrieval operations and processing. However, both University of Twente and LBNL operators store infrastructure documentation on computers also used for other purposes than building automation, and direct connections to those resources were infeasible.

Finally, we implemented two programs to parse PICSs and EDE files respectively. In the first case, the program goes from document’s top to bottom guided by the diagram shown in Figure 7.7. For every available PICS, the program first selects all implemented BIBBs and BACnet objects (Figures 7.5a and 7.5b). Each object can be creatable/deletable and this information follows the object as a “yes/no” or equivalent symbols (Figure 7.5b). Finally, for every object, the script selects a list of properties that can be writable or not (Figure 7.5c). Figure 7.8 shows

Figure 7.7: PICS parsing diagram

parsing results of the “Blue ID S10 Controller” coming from the PICS showed in Figure 7.5.

Most of the retrieved PICS did not have any information about property values. Instead, this information was included in the EDE files. A further program went through all EDE files selecting **Present.Value** minimum and maximum values for every listed object. This new information was structured as shown in Figure 7.9.

Figure 7.8: Parsing PICS example results

Figure 7.9: Parsing EDE example results

7.3.1.4 Rule Definition and Detection

Next we describe how the information gathered in previous steps is used to define specification rules. In §7.3.1, we motivated our focus on variables’ *types*, *values* and related access *methods* as basis for our specification rules. From this, we derive three *abstract rules*: 1) a “Type” rule checks if a variable of a specific type is allowed; 2) a “Value” rule checks which values a variable may assume; and 3) a “Method” rule checks which methods can be used to access a specific variable. All rules have the same structure: each element (type, value, method) is evaluated against a set of allowed possibilities. For example, in the “Type” rule, a variable’s type is evaluated against all the allowed types of variable a controller may implement (Algorithm 4).

Algorithm 4: Abstract “Type” rule

```

if  $Variable_{type} \notin ControllerAllowedVariableTypes$  then
  | Alert(“Variable type not permitted”)

```

We use a Python program to automate the process of mapping information retrieved over Feature Lookup to the abstract rules. In the following, we discuss how we map these abstract rules into specification rules for monitoring for each type.

Type Rule: The “Type” rule checks which BACnet objects and properties each BACS can use. This information comes from the PICSs (Figures 7.5b and 7.5c) and, thus, is included in the results of Feature Lookup (Figure 7.8). Therefore, a script selects allowed objects and properties of each identified BACS and transforms the “Type” rule into the two specification rules shown in Algorithm 5.

In the case of the “Blue ID S10 Controller”, the $ControllerAllowedObjectTypes$ set contains all the objects found in the PICS such as **Accumulator**, **Analog Input**,

Algorithm 5: BACnet “Type” rules

```

if  $BACnet\ Object \notin Controller_{AllowedObjectTypes}$  then
  |  $Alert("Forbidden\ Object")$ 
else
  if  $BACnet\ Property \notin Controller_{Object_{AllowedPropertyTypes}}$  then
    |  $Alert("Forbidden\ Property")$ 
  else
    —

```

etc. In the same way, the $Controller_{Accumulator_{AllowedPropertyTypes}}$ set of a “Blue ID S10 Controller” contains properties $Object_Identifier$, $Object_Name$, etc. Whenever the system captures a BACnet message including an object and some property, the two rules check object and property types respectively and alert if these types are not included in the defined sets. This allows the system to detect snooping attacks and any other attack dealing with unexpected objects and properties.

Value Rule: The “Value” rule checks which values BACnet properties may assume. This information comes from the EDE files (Figures 7.6 and 7.9) and, thus, is automatically mapped to the concrete rule as shown in Algorithm 6.

Algorithm 6: BACnet “Value” rule

```

if  $BACnet\ Property\ value \notin Controller_{(Object,Property)_{AllowedPropertyValues}}$  then
  |  $Alert("Forbidden\ Value")$ 
else
  —

```

For example, when it comes to “Device 4001”, the system alerts if “Analog Value 171” is below 0 or above 100. This rule protects the infrastructure against process control subverting scenarios, and thus attacks attempting to modify parameters of the physical and control processes.

Method Rule: The “Method” rule validates the BACnet services each BACS can use. This information comes from the PICSs in the form of a list of BIBBs (Figures 7.5a) and is included in the results of Feature Lookup (Figure 7.8). BIBBs can be replaced with corresponding services by a simple lookup operation. Therefore, Algorithm 7 checks if a BACnet service belongs to the set of allowed services.

Algorithm 7: BACnet “Method” rule

```

if  $BACnet\ Service \notin Controller_{AllowedServices}$  then
  |  $Alert("Forbidden\ Service")$ 
else
  —

```

In the case of the “Blue ID S10 Controller”, the `ControllerAllowedServices` includes services from BIBBs `DS_RP_A` (`ReadPropertyRequest`), `DS_RP_B` (`ReadPropertyResponse`), etc. This rule allows the system to detect attackers misusing BACnet services to fulfill their goals.

Furthermore, we use the “Method” rule to check which BACnet object is creatable/deletable and which BACnet property is writable. Following the standard, we compile three sets of BACnet services with services that create objects, that delete objects, and that write properties respectively. Then, the system uses the information from Feature Lookup to define checks on non-creatable/deletable objects and non-writable properties by using Algorithm 8.

Algorithm 8: BACnet additional “Method” rules

```

if BACnet Service  $\in$  CreateObjectServices then
  | Alert(“Forbidden object creation”)
else
  if BACnet Service  $\in$  DeleteObjectServices then
    | Alert(“Forbidden object deletion”)
  else
    if BACnet Service  $\in$  WritePropertyServices then
      | Alert(“Forbidden property writing”)
    else

```

For example, the first two rules alert the presence of services attempting to create or delete `Accumulator` objects belonging to a “Blue ID S10 Controller”. The third rule reports any service attempting to write to a non-writable property, such as `Accumulator`’s `Object_Identifier`.

7.3.1.5 Semantic Labeling

As for Rule Definition, Semantic Labeling leverages the semantic information retrieved from available documentation. We start from the BACnet messages that trigger the specification rule. Involved BACnet services, objects or properties are scanned against specs and configuration files. Compared to the previous step, within Semantic Labeling, we can select more verbose textual pieces of information, because alerts are later read and interpreted by human users. While a specification rule needs precise information to enforce a check or a constraint, the alert can be more descriptive and present to the user a broader variety of information (e.g., images supporting the explanation of the monitored activities). Moreover, the alert can directly reference to used documentation to allow operators further analyses of the situation.

Experiments In our prototype implementation we were able to provide a reference to the device description supplied by the PICS and the related configuration files. Moreover, we provided a link to any other related document retrieved over

Feature Lookup for every alert triggered by the system pointing at relevant sections within those documents. This substantially enhanced the feedback to the operator and provided an immediate view of the involved BACS components. Furthermore, we could find in the retrieved documentation additional information about possible causes of detected problems (e.g., descriptions of specific variables' behaviors and misbehaviors). In those cases, our system directly copied the information from the documents into the related alerts.

7.3.2 Evaluation

Our system filled the abstract rules with the information coming from Feature Lookup crafting hundreds of specification rules. To improve efficiency we arrange the specification rules in an order that avoids meaningless checks (e.g., we do not want to check a BACnet property if we already know that the BACnet object it belongs to is not allowed). For every captured BACnet message, the system checks if the BACnet service is allowed; then, if involved BACnet objects can be used, created or deleted; then, if involved BACnet properties are allowed and writable. Finally, the system examines properties' actual values. Only a small set of specification rules are of this last type due to the limited number of EDE files that operators provided us with.

As outlined in §7.2, we tested our approach against more than two months of real traffic. Over the two months of capturing, our system triggered 237 unique alerts; 226 at the University of Twente and 11 at LBNL. The two results differ because of the different views we achieved over the two infrastructures (as already described, at LBNL we could monitor only a subset of the building automation system and thus a subset of the traffic). Table 7.3 shows the three abstract rules, the corresponding specification rules and whether or not a rule raised an alert.

Abstract Rule	Specification Rule	# Alerts
Type Rule	Forbidden object	2
	Forbidden property	234
Value Rule	Forbidden value	0
Method Rule	Forbidden service	0
	Forbidden object creation	0
	Forbidden object deletion	0
	Forbidden property writing	1

Table 7.3: Detection results

We did not find any evidence of malicious activities over the time span of the captures. However, our approach still provided interesting insights. At the University of Twente, the system raised alerts on two BACs using forbidden objects. Both cases involved a “proprietary” object never described within the PICS. According to the available documentation, the two devices (two Siemens controllers PXC128-U) should not include anything that was not defined within the BACnet standard. Nevertheless, a device probed the two controllers (Figure 7.10) and received back correct BACnet responses about an unknown object. A meeting with the operators revealed that this object is vendor-defined and gathers information

on parameters of the BACs recognizable and understandable by vendors only. Operators confirmed that vendors have access to the building automation system to monitor their devices, and use of an unknown BACnet object happens even though the documentation does not mention this possibility because of its internal nature. However, operators did not know that involved BACs provide specific functionalities and attackers can potentially exploit such circumstances.

Figure 7.10: Unexpected object ReadProperty_Request

Detection on BACnet properties provided the highest number of alerts (all alerts at LBNL were of this kind). Our system generated several alerts on ReadProperty and ReadPropertyMultiple messages attempting to retrieve non-existing properties. As a matter of fact, these properties were not defined by the PICs and, for most cases, we could eventually confirm the non-existence of these properties by observing some BACnet errors carried in the responses to those read requests (Figure 7.11).

(a) Unexpected property ReadProperty_Request

(b) ReadProperty_Response confirmation of the alert

Figure 7.11: Unexpected property read operation

A BACS asking for unimplemented properties is not necessarily a violation of the specs. In fact, all PICs define what a BACS implements without defining what other BACSs may ask for. A situation in which a BACS sends back a BACnet-error response to warn about a non-existing property (Figure 7.11b) is in line with the specs and should be of no harm for the system. However, the reason to

alert on situations of this kind is twofold. First, this situation may be of interest from a security perspective. Despite being handled by the BACnet protocol, these circumstances may hide a “network discovery” scenario where an attacker tries to gain knowledge of the infrastructure by randomly probing BACs. As described in §7.1, snooping is one plausible attack in building automation systems. Secondly, the same situation shows a common side-effect of the joint use of different BACnet software solutions. As servers and workstations do not usually know in advance which BACs they will connect to, predefined BACnet discovery messages exist in order to gather general information of building automation components. These messages do not consider which BACnet properties are defined for each device and simply use large sets of them. This consequently generates several error responses on the network.

To dig deeper into property-related issues, we extended the “unknown property” specification rule to also check if properties enforced by PICSs were always implemented. Therefore, we created a further instance of “Type” rule checking all BACnet error messages to detect missing properties that were supposed to be used by the BACs. The system revealed several messages reporting “unknown-property” errors about properties declared to be part of devices’ BACnet implementations. All these mismatches between implementation and specification are particularly relevant for what concerns interoperability. In fact, software solutions that define their interactions with a BACS based on its public documentation can incur into inconsistencies caused by incorrect or lacking implementations.

Finally, the system triggered an alert corresponding to an unexpected write operation on a BACnet property supposed to be readable only. A Priva controller received a BACnet `WriteProperty` request on the `Exception_schedule` of an object `Schedule` (Figure 7.12). Despite what we knew from the related PICS, the BACS sent back a `SimpleACK` message, acknowledging the success of the operation (the actual writing was confirmed by later read operations). These kinds of situations are especially dangerous due to the unpredictability of their results. As no indication is provided by the vendor, the write operation can either succeed or fail, and may generate a response or not (even independently from the actual modification of the value within the property). Meeting with the operators revealed that this write operation was due to a human mistake during the configuration of the Priva controller. However, the same situation could fit the “process control subverting” scenario described in §7.1.

Figure 7.12: Unexpected property write request

7.3.2.1 Discussion

By construction, our IDS is able to detect events that do not match the specifications coming from retrieved documentation. This aspect leads to two considerations:

- On the one hand, an alert raised by the system does not necessarily refer to a security-relevant event as the related mismatches may not directly harm the monitored devices. However, all findings revealed network activities otherwise invisible to operators. Over the two months of analysis, every alert identified either an actual mismatch between device documentation and implementation (e.g., unimplemented BACnet Properties) or an operator mistake (e.g., the unexpected writing operation). As already discussed, these issues can cause significant gaps in the knowledge operators have about their infrastructures and may potentially lead to dangerous misconfigurations of the involved systems. The meeting with the operators at the University of Twente confirmed that employed HMIs were not able to signal any of the misconfigurations found or even notify the users on generated BACnet errors. As a result, the University of Twente asked to deploy our system into the building automation system continuously and let operators receive notifications of the generated alerts. So while our datasets did not include actual attacks, we were able to reliably detect notable deviations from the specifications at zero false-positives. This result is in line with the work of Uppuluri et al. [179] showing that specification-based intrusion detection works towards optimal detection rate while substantially decreasing the number of false positives compared to anomaly-based detection.
- On the other hand, our approach does not necessarily detect all possible attacks threatening the monitored infrastructure. In fact, any attack operating within the boundaries defined by employed specifications would not be caught by our IDS. However, our solution substantially narrows down what a malicious user can do and covers most of the attack scenarios defined within the categories listed in §7.1. Furthermore, our solution does not exclude the use of other approaches such as pure anomaly-based intrusion detection either improving the obtained rule set or working in parallel.

Each one of the implemented phases effectively achieved the defined goals. Thanks to the numerous read operations, System Discovery took just a few hours of network sniffing to gather all the information needed to describe the whole set of BACSSs. With this information, our approach was able to rapidly and automatically identify available sources of information and craft effective specification rules.

Feature Lookup focused just on structured documents such as PICSs and EDE files. In some of the tests, we further extended online research to documents such as BACS user manuals. Our system was able to download 10 manuals related to components deployed in the monitored infrastructures. However, we decided to not further employ manuals because an analysis showed they were fully overlapping with the information found in the PICSs. Nevertheless, one way to improve

our documentation-mining approach is to enable handling heterogeneous documentation and, especially, unstructured information. To this regard, we observe that Feature Lookup should abstract from domain-specific parsing scripts and generalize the process of mining and structuring infrastructure features. Correctly selecting information can take advantage of standard data mining and natural language processing. Our work did not present a general approach to this activity. However, works such as [152, 153, 180] may fulfill this goal.

With more general techniques capable of extracting knowledge from heterogeneous documentation, the effort of deploying the system completely converges on mapping retrieved information to the abstract rules. According to the monitored infrastructures, operators should identify the related concepts of variable type, value and access method and, eventually, let the system interpret data coming from Feature Lookup and instantiate the specification rules.

Even without such a general approach, our solution drastically reduced the time needed to deploy intrusion detection into a BACnet-based building automation system. Obtaining the same set of specification rules by hand would have required substantial effort, making it infeasible for larger infrastructures. Furthermore, the obtained system comes with the intrinsic capability to update according to the changes of the monitored infrastructure. In fact, whenever new BACSs are deployed, our system transparently reads the new information over the network and goes through the three steps all over again. In the end, this solution makes the implemented system directly applicable to any other BACnet infrastructure with no further effort on configuration or deployment.

The proposed approach works likewise for different building automation technologies. As discussed, this would mostly require a modification of the mapping process linking retrieved information and abstract rules but would leave the core concept unchanged. Other building automation infrastructures such as KNX [49] and LonWorks [51] also meet the requirements of *availability* and *linkability*. These widely used protocols present characteristics similar to the ones observed for BACnet. Moreover, both KNX and LonWorks promote and support the use of documents describing protocol implementation details (although not as formal as BACnet PICS).

To show the generality of our approach beyond building automation systems we outline how the same documentation-mining technique applies to two different domains of NCSs, namely ICS and in-vehicle networks.

Industrial Control Systems Specification-based intrusion detection for ICSs is not new. Works such as [107] show the effectiveness of this approach applied to electrical grids. However, applying a set of specification rules to a real deployment still requires manually crafting all parameters on specific needs. Again, our research can improve the use of specification-based intrusion detection by leveraging available information of the deployments. For example, in the smart grid scenario, we would focus on Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTUs), as these play a main role in the infrastructure. We would analyze variables handled by these controllers, their types, values and access methods and then use the abstract rules defined in §7.3.1.4.

Regarding the assumption of §7.3.1.3, verified information about the smart grid is *available* within configuration files that use the “Substation Configuration Language” (SCL). SCL files usually provide formal representations of modeled data and communication services. The information included in these files is *linkable* thanks to the included detail descriptions of the involved infrastructures (e.g., “Substation Configuration Description” files). Besides SCL files, operators usually store additional documentation describing physical and control processes as in the building automation use case. An IDS can leverage this documentation to gather further information and derive specification rules.

The three steps of the approach remain unchanged. System discovery will passively gather data about devices communicating over the ICS network. According to the verbosity of the involved protocols, an IDS will eventually collect enough information to identify infrastructure components and start the Feature Lookup step. Once information about PLCs and RTUs functioning is retrieved, Rule Definition will use it to define the actual specification rules.

In-Vehicle Networks Similar argumentation can be applied to communication of “Electronic Control Units” (ECUs) over automotive bus systems like the “Controller Area Network” (CAN) or FlexRay found in all of today’s cars.

CAN is a network where connected ECUs communicate by means of small messages with a payload of only 8 bytes. CAN uses a content-based addressing where messages only carry a 11 (or 21) bit message identifier and receiving ECUs will select messages relevant to them based on this message identifier. Message identifiers also serve as prioritization, as the employed CSMA/CR medium access scheme will always grant priority to the message with the lowest message identifier avoiding collisions on the bus. Transport layer protocols like ISO-TP allow for transfer of longer messages fragmented into smaller network packets and more complex forms of addressing crossing gateways connecting multiple CAN segments.

In order to maintain and manage the assignment and semantics of message identifiers, the design phase of an automotive network involves setup of a so-called CAN-Matrix that lists exactly which ECU is supposed to send which message identifier, which ECUs will receive messages of certain type and also the payload syntax and semantics. This design is done using sophisticated tools like Vector Informatics CANoe.⁴ The data provided by such tools is a perfect data source for specification-based IDS and for our approach, so the criteria of *availability* is met. *linkability* is more of a concern, as messages per se do not contain information on their source or type and a recipient needs to know (part of) the CAN matrix to identify how to decode a certain message ID. However, with the CAN matrix, we do have information on the types of ECUs available and can therefore conduct System discovery. This information can then be used to conduct Feature Lookup. A lot of relevant information (which messages are supposed to be seen on which bus segment) is again contained in the CAN matrix. Unfortunately, documentation in vehicular networks is not as standardized as the PICSS are in BACnet. So feature lookup would probably require more detailed investigations and more complex

⁴http://vector.com/vi_canoe_en.html

document parsing. Rule definition is then straightforward. However, having no source or destination addresses in packets, one would have to focus on message IDs, bus segments, and payload for detection.

While specification-based intrusion detection has been proposed many times especially for CAN-based networks [181, 182], a structured approach to rule-mining is missing in this domain so far and we see this as a promising field of application for our approach.

7.4 Section Conclusion

In this section we discussed attacks on BACnet and their consequences within NCSs. As for the previous chapter, we observe that state-of-the-art security solutions still face some limitations. Particularly, we emphasize the lack of security solutions that take advantage of BACnet information to detect BACnet protocol-based attacks.

To enhance detection of this kind of attacks we presented our D-IDS. The D-IDS is the result of a novel approach to specification-based intrusion detection and inherently bonds together system knowledge acquisition and network security. While state-of-the-art solutions for NCS security employ manually-crafted specification rules, we showed that it was feasible to automatically mine these rules from available documentation. The tests performed on real building automation systems showed the effectiveness of the approach and confirmed the improvements in the detection processes.

Part III

Final Remarks

CHAPTER 8

Conclusion And Outlook

We now outline the main contributions of our work in relation to the research questions described in the introduction. Furthermore, we discuss the lesson learned and we frame interesting future research directions that would enhance the results presented in this thesis.

8.1 Summary of Contributions

Since the spread of digital networks and the Internet, NCSs have acquired a significant role in our daily life. In parallel, NCS-related security has increasingly gained attention both for the criticality of specific systems (e.g., critical infrastructures) and the appearance of sophisticated cyber-attacks such as APTs. Over the years, both security researchers and stakeholders have worked on improving this situation and enhancing NCS protection mechanisms. However, most attempts only focused on applying standard IT solutions to NCSs with poor results due to the differences between the two environments. Nonetheless, NCS-specific security solutions have also faced the challenge of interpreting observed NCS operations and making a distinction between acceptable activities and malicious ones. Our work goes along this line of research and paves the way to solving the issue through deepening and improving the acquisition of knowledge about the monitored infrastructures by leveraging network traffic information and available documentation. Furthermore, we designed and discussed two novel NIDSs, namely the *Sequence-based* and the *Document-based* IDSs, specifically tuned on NCS requirements and characteristics. Finally, we tested our concepts and solutions on data coming from real operating facilities such as critical infrastructures and building management systems.

The main research question of §1.2 represents the focal point of this thesis:

Main RQ: *Can we defend NCSs from APTs by developing more effective anomaly-based IDSs that leverage system knowledge?*

To answer this question, we defined a general framework combining the process of knowledge acquisition with the use of custom intrusion detection solutions. The

framework guided our study through the analysis of different building blocks, each one addressing a specific aspect of the main research question. This path eventually led us to tackle four different subquestions. The first one investigated the feasibility of gaining infrastructure information from network traffic:

RQ1: *To what extent can we infer system knowledge from NCS network traffic?*

This question mainly focused on fingerprinting and its feasibility within NCSs. In §5.1, we showed that standard fingerprinting techniques and tools were not suitable for NCS environments and, as a result, failed to select valuable information in the network as well as to identify any device besides common ones (e.g., servers running Windows). For this reason, we addressed NCS fingerprinting as follows:

1. We studied state-of-the-art fingerprinters to outline a general architecture and define all basic building blocks needed to implement fingerprinting in NCSs.
2. We identified NCS fingerprinting main challenges and opportunities and we instantiated all aforementioned building blocks in the context of NCSs.
3. We developed and tested a proof-of-concept NCS fingerprinter based on the general architecture and the NCS fingerprinting challenges and opportunities (e.g., purely passive, exploiting NCS stable communications, etc.)

The Flow-Fingerprinter allowed us to recognize most of the devices communicating over NCS networks and provided a baseline for further analyses. In fact, we used results coming from fingerprinting to keep searching for valuable information about the monitored infrastructure. This leads us to the second research question:

RQ2: *Can we automate the process of gaining system knowledge from NCS documentation?*

In §5.2, we argued that NCS operators use heterogeneous documentation to maintain their infrastructures and keep trace of any activity or update. We investigated the advantages of using this documentation for security and we studied which techniques can be leveraged to achieve this goal. Ultimately, we answered RQ2 as follows:

1. We identified which kind of documents are available to provide valuable information of the monitored NCSs.
2. We discussed possible document retrieval techniques for both public documentation (e.g., protocol standards) and private documentation (e.g., configuration files)
3. We outlined plausible parsing techniques to analyze the content of identified documents and infer valuable knowledge in case of both structured and unstructured text.

This work showed the feasibility of gaining knowledge from NCS documentation and laid the foundation for developing security solutions based on information of the monitored infrastructure, its control processes and the physical processes beyond.

The last two subquestions related to cyber-attack detection and investigated the benefits of using information coming from network traffic and documentation (thus, the results of RQ1 and RQ2) to improve detection of sophisticated attacks and APTs within NCSs. RQ3 focused on a new kind of attacks called sequence attacks:

RQ3 *How do we detect APTs such as sequence attacks against NCSs?*

In §6 we showed that an effective countermeasure against sequence attacks relies on both the development of a custom anomaly-based IDS and the parallel use of knowledge about the monitored infrastructure gathered from network messages. More in details, we tackled RQ3 as follows:

1. We identified four categories of sequence attacks according to the target component of an NCS and the type of misuse carried out
2. We developed a reference architecture for sequence-based intrusion detection and we implemented an S-IDS in the context of industrial control testing its effectiveness against the attacks identified before on traffic coming from real facilities
3. We integrated the implemented S-IDS with techniques of knowledge acquisition such as fingerprinting and we showed the improvement that this approach brought in terms of lowering the number of false positives

Sequence attacks had never been properly defined before and few state-of-the-art security solutions had attempted to model sequences of messages within NCS communications. However, none of these solutions developed an effective mechanism to detect sequence attacks. Our work comprehensively discussed the development of sequence-based intrusion detection solutions and tested the S-IDS against traffic coming from real critical infrastructures. Furthermore, we showed the advantages of leveraging the knowledge about the monitored infrastructure to enhance our S-IDS and avoid the misclassification of normal operations as malicious activities.

Finally, RQ4 focused on specification-based intrusion detection and, thus, the concept of building intrusion detection out of descriptions of monitored devices and processes in a system or a network:

RQ4 *Can we develop specification-based intrusion detection that relies on automatically acquired system knowledge?*

In §7, we argued that a major disadvantage of specification-based intrusion detection is the effort needed to manually define rules correctly describing the behavior of a system. Our work aimed at automating the definition of these rules on the basis of the identified mechanisms for knowledge acquisition. Therefore, we engaged in the following steps:

1. We instantiated techniques defined for RQ1 and RQ2 in the context of building automation
2. We defined a process that maps information coming from documentation to specification rules we automated this process within a specification-based IDS for BACnet-based building automation systems
3. We tested this approach against traffic coming from real building automation systems and we showed its effectiveness in identifying infrastructure misconfigurations and further potentially malicious activities

The documentation-based intrusion detection system exceeds state-of-the-art specification-based solutions in terms of effectiveness and ease of use. The D-IDS we developed was able to automatically craft hundreds of specification rules inferred by heterogeneous documentation retrieved on the Internet and from private storages. The checks and the related alerts spotted undesirable activities that had been overlooked by other deployed solutions such as HMIs. As a result, infrastructure operators asked to permanently deploy the D-IDS and let them receive notifications of the generated alerts.

Summarizing, this thesis presents the study results that brought us from understanding NCSs, their control operations and the physical processes beyond to secure these infrastructures leveraging the same information. We showed that the adoption of custom security solutions is needed to protect NCSs due to their specific properties and characteristics and we motivated the use of system knowledge to achieve this purpose. In the end, we tested and evaluated this concept on two major examples of NCS, namely industrial control and building automation. In the former case, we tackled a specific kind of attack and showed the advantages of integrating the S-IDS with information coming from the network. In the latter case, we designed a security solution inherently binding intrusion detection and infrastructure knowledge acquisition. The D-IDS showed the intrinsic potential of this link and thus positively answers the main research question fulfilling the ultimate goal of the thesis.

8.2 Limitations

All results presented in this thesis were supported by thorough discussions with stakeholders and testing on real data. Despite our concerns, we emphasize the difficulties of achieving comprehensive validation of the proposed tools and techniques. In fact, the very nature of NCS poses several challenges to studying their security weaknesses and the related countermeasures. What follows is a list of the most critical obstacles to our research:

- **Lack of attack data:** Both the S-IDS and the D-IDS missed facing real attack scenarios. We evaluated the S-IDS synthesizing sequence attacks based on tests made on stand-alone components (e.g., the message order-based control device compromise) and information given by operators (e.g., the message timing-based physical process compromise). However, we did not have

access to any real sequence-attack to validate the implemented intrusion detection techniques (at the time of writing of this thesis, no description of known cyber-attacks against real facilities seems leading back to a sequence attack). Similarly, we were not aware of any actual attack performed against the building automation systems while the D-IDSs were operating. Cases such as the unexpected writing operation were the closest circumstances resembling a dangerous situation even if they only related to operators mistakes rather than cyber-attacks. The presence of real attack data available for testing still remains a necessary condition for a comprehensive evaluation of NCS security solutions and, more in general, for improving intrusion detection research.

- **Lack of public datasets:** As for attack data, public NCS datasets would have helped to improve the evaluation of our intrusion detection approaches and to evaluate their performances with respect to other security solutions. Many researches on NCS security face the challenge of generalizing results beyond the testing on a specific use case (e.g., a single traffic trace coming from a particular infrastructure). In this thesis, we attempted to widen the evaluation both for the S-IDS and the D-IDS exploiting three different critical infrastructures and two distinct building automation systems respectively. Nevertheless, a comprehensive evaluation would have required a much larger testing. Unfortunately, NCS data such as traffic traces are hard to find as no organization openly provides that kind of resources due to the sensitivity of the involved information. For this reason, the presence of NCS open datasets is always limited by stakeholders' privacy concerns. Likewise, all organizations collaborating to our research asked not to share the data we used or explicitly asked for signing non-disclosure agreements.
- **Lack of collaborations among stakeholders:** In a broader sense, improving NCS security relies on strengthening cooperation among all NCS stakeholders. We observe that public/private partnerships encouraged by governmental organizations (e.g., international research projects founded by the European Commission) are one valid answer to this issue. Nevertheless, improving the situation requires a greater effort and support that would eventually bring mutual benefits for all participants. Among all possible ways to collaborate, we notably remark the sharing of data (as for the two previous points of this list), the sharing of equipments and tools (e.g., devices, software, testbeds), and the sharing of expertise either through formal or informal practices (e.g., CERTs or round tables respectively).

As for our case, all future studies on NCS security will likely have to deal with the limitations described before.

8.3 Future Research Directions

This thesis links to several still-open research directions and paves the way to new ones that would further improve the bond between security and system knowledge.

What follows is a list of plausible future developments in the context of NCS security:

- **The “time component” and time anomalies:** time is an important component of process control. NCSs often encompass a number of coordinated operations occurring on a tight schedule. Latencies in the communications are often undesirable and can even result in dangerous consequences (e.g., delaying network messages that carry information about malfunctions of a nuclear power plant reactor). Some research in literature discusses the topic and presents solutions aiming to model system operations timing and using these models to spot discrepancies. However, these solutions still face many false positives despite the overall periodicity of NCS operations. Most of NCS security solutions discussed in literature still do not encompass the notion of time or fail to properly model operations timings and, thus, spot discrepancies with respect to an accepted behavior. Our work does not tackle time issues directly although the S-IDS can detect some time anomalies by observing changes in DTMC transition parameters such as “average time elapsed” and “standard deviation on time elapsed”. Nevertheless, detecting time anomalies still misses an extensive analysis on which operations (or kinds of operations) are stable enough to justify the use of a time-based IDS. A viable solution to the problem includes the use of techniques such as discrete Fourier transformations to differentiate between highly predictable communications—where it is possible to perform time-anomaly detection—from noisier communications that would instead undermine any approach of this kind.
- **Distributed sensors and data integration:** in this thesis we proposed two network intrusion detection solutions working with anomaly-based and specification-based techniques respectively. In the same way, other researches in literature present a number of different approaches to intrusion detection leveraging further techniques (e.g., signature-based) and data sources (e.g., host-based detection). We observe that each solution may bring awareness on a specific security aspect of the monitored infrastructure and, thus, a proper integration of these solutions may improve the overall capability of a system (and its operators) to identify several distinct malicious activities. Integrating the work of different IDSs comes with the price of handling a highly complex system that aims to merge together disparate alerts and data streams into one single consistent piece of information to the operators. Security Information and Event Management (SIEM) systems are a plausible answer to this need. However, most solutions of this kind do not go beyond simple statistics computed on IDS alerts (e.g., evaluating the number of simultaneous alerts). As the complexity of IDSs increases, SIEMs should evolve to systems capable of judging alerts based on a deeper knowledge and a more complete view of the monitored infrastructure as well as employ advanced analysis techniques to weight the importance of each alert on the basis of the system that generated it and the concurrent information coming from the others (e.g., frameworks based on theoretical schema such as subjective

logic have already been proposed as a feasible solution to this issue [183]).

- **A security specification language:** intrusion detection solutions such as the D-IDS presented in this thesis show the feasibility of mapping data coming from the network traffic to information retrieved from documentation. This approach eventually improves the quality of the alerts sent to the operators in an attempt to ease the understanding of security-related events in the network and improve the overall visibility on the monitored infrastructure. Generally speaking, this mapping allows to connect two different domains: the first includes network messages and data, the second encompasses meaningful events understood by operators. Bridging the semantic gap between the two domains means allowing to go from one to another (both ways) and, thus, also translating meaningful events to their related network messages and data. This concept paves the way to defining high-level policies or constraints and automate a process that translates them into low-level checks on the observed traffic. The definition of a language for security specifications would formalize this process by organizing the events that are valuable to the operators, the acquired knowledge, and the network traffic data into categories of unambiguous meaning. This approach would improve the applicability of solutions such as the document-based IDS by clearly defining the meaning of automatically generated specification rules and by supporting the creation and application of new ones from the operator side.

In the end, with the spread of technologies such as the Internet of Things, NCSs and further cyber-physical systems will increasingly gain space in tomorrow's cities and societies. In a few years, new technologies will arise and the research on security, as well as the future research directions discussed in this section, will develop and twist in multiple ways which are sometimes difficult to foresee. This thesis draws attention to the bond between system knowledge and network security and the key role that this bond plays to enhance NCS-specific security solutions in the field of intrusion detection. We believe that this paradigm will also stand still in the future and it will eventually help to drag every "smart" environment to inherently become a secure environment.

Author's Publications

- [1] Marco Caselli, Emmanuele Zambon, Johanna Amann, Robin Sommer, and Frank Kargl. "Specification Mining for Intrusion Detection in Networked Control Systems." In: *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*. 2016, pp. 791–806.
- [2] Marco Caselli, Emmanuele Zambon, and Frank Kargl. "Sequence-aware Intrusion Detection in Industrial Control Systems." In: *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, CPSS 2015, Singapore, Republic of Singapore, April 14 - March 14, 2015*. 2015, pp. 13–24.
- [3] Marco Caselli, Emmanuele Zambon, Jonathan Petit, and Frank Kargl. "Modeling Message Sequences for Intrusion Detection in Industrial Control Systems." In: *International Conference on Critical Infrastructure Protection IX - 9th IFIP WG 11.10, ICCIP 2015, Arlington, VA, USA, March 16-18*. 2015, pp. 49–71.
- [4] Marco Caselli and Frank Kargl. "A Security Assessment Methodology for Critical Infrastructures." In: *The 9th International Conference on Critical Information Infrastructures Security, CRITIS 2014, Limassol, Cyprus, October 13-15*. 2014, pp. 332–343.
- [5] Emiliano Casalicchio, Marco Caselli, Alessio Coletta, and Igor Nai Fovino. "DNS as critical infrastructure, the energy system case study." In: *International Journal of Critical Infrastructures IJCIS* 9.1/2 (2013), pp. 111–129.
- [6] Emiliano Casalicchio, Marco Caselli, and Alessio Coletta. "Measuring the global domain name system." In: *IEEE Network* 27.1 (2013), pp. 25–31.
- [7] Marco Caselli, Dina Hadziosmanovic, Emmanuele Zambon, and Frank Kargl. "On the Feasibility of Device Fingerprinting in Industrial Control Systems." In: *The 8th International Conference on Critical Information Infrastructures Security, CRITIS 2013, Amsterdam, The Netherlands, September 16-18*. 2013, pp. 155–166.
- [8] Emiliano Casalicchio, Marco Caselli, Alessio Coletta, Salvatore Di Blasi, and Igor Nai Fovino. "Measuring Name System Health." In: *International Conference on Critical Infrastructure Protection VI - 6th IFIP WG 11.10, ICCIP 2012, Washington, DC, USA, March 19-21*. 2012, pp. 155–169.

- [9] Marco Caselli, Frank Kargl, and Valentin Tudor. *CRISALIS D4.4 Device Fingerprinting*. Tech. rep. European Community's Seventh Framework Programme, 2014.
- [10] Marco Caselli, Frank Kargl, and Tobias Limmer. *CRISALIS D5.1 Security Testing Methodology*. Tech. rep. European Community's Seventh Framework Programme, 2014.
- [11] Valentino Angeletti, Erwin Kooi, Marco Caselli, Tobias Limmer, and Corrado Leita. *CRISALIS D4.4 Setup Detailed Description*. Tech. rep. European Community's Seventh Framework Programme, 2013.
- [12] Emmanuele Zambon, Marco Caselli, and Magnus Almgren. *CRISALIS D6.4 Device Fingerprinting*. Tech. rep. European Community's Seventh Framework Programme, 2015.

Bibliography

- [13] *Electrical Railway Review*. v. 46. The Wilson Company, 1906.
- [14] Alberto Bemporad, Maurice Heemels, and Mikael Johansson. *Networked control systems*. Vol. 406. Springer, 2010.
- [15] Peng Zhang. *Industrial control technology: a handbook for engineers and researchers*. William Andrew, 2008.
- [16] National Joint Apprenticeship & Training Committee. *Building Automation: Control Devices and Applications*. American Techn. Publ., 2008.
- [17] *Framework for Improving Critical Infrastructure Cybersecurity*. Tech. rep. National Institute of Standards and Technology (NIST), 2014.
- [18] Roberto Baldoni and Luca Montanari. *Italian Cyber Security Report. A National Cyber Security Framework*. Tech. rep. 2015.
- [19] Peter Bisson, Jacques Bughin, Michael Chui, Richard Dobbs, James Manyika, Alex Marrs, and J. Michael. *Disruptive Technologies: Advances that will transform life, business, and the global economy*. Tech. rep. McKinsey Global Institute, 2013.
- [20] Cori Winslow. *Smart Homes*. Tech. rep. New York State Energy Research and Development Authority (NYSERDA), 2015.
- [21] Nicolas Falliere, Liam O. Murchu, and Eric Chien. *W32. stuxnet dossier*. Tech. rep. Symantec Corporation, Security Response, 2011.
- [22] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Márk Félegyházi. “Duqu: Analysis, detection, and lessons learned.” In: *ACM European Workshop on System Security (EuroSec)*. 2012.
- [23] Chris Bronk and Eneken Tikk-Ringas. “Hack or attack? Shamoon and the Evolution of Cyber Conflict.” In: *Social Science Research Network* (2013).
- [24] Aunshul Rege-Patwardhan. “Cybercrimes against critical infrastructures: a study of online criminal organization and techniques.” In: *Criminal Justice Studies* 22.3 (2009), pp. 261–271.
- [25] S.J. Rockefeller. *A kill chain analysis of the 2013 target data breach*. Tech. rep. Committee on Commerce, Science and Transportation. United States Senate, 2014.

- [26] Building Services Research and Information Association (BSRIA). *Should Building Managers worry about scary movies?* 2015. URL: <https://blogs.bsria.co.uk/2015/01/20/should-building-managers-worry-about-scary-movies/> (visited on 11/01/2016).
- [27] Zero Hedge. *Air Conditioner Failure At CenturyLink Data Center Takes ITG Dark Pool Offline*. 2015. URL: <http://www.zerohedge.com/news/2015-08-27/air-conditioner-failure-centurylink-data-center-takes-itg-dark-pool-offline> (visited on 11/01/2016).
- [28] Digital Trends. *Thousands of smart gadgets hacked to send out spam email*. 2014. URL: <http://www.digitaltrends.com/computing/thousands-smart-gadgets-hacked-send-spam-email/> (visited on 11/01/2016).
- [29] Colin Tankard. “Advanced Persistent threats and how to monitor and deter them.” In: *Network Security* 2011.8 (2011), pp. 16–19.
- [30] Keith Stouffer, Suzanne Lightman, Victoria Pillitteri, Marshall Abrams, and Adam Hahn. *Guide to industrial control systems (ICS) security*. Tech. rep. Special Publication 800-82. NIST, 2011.
- [31] Alvaro A. Cárdenas, Saurabh Amin, and Shankar Sastry. “Research Challenges for the Security of Control Systems.” In: *Proceedings of the 3rd USENIX Workshop on Hot Topics in Security, HotSec’08, San Jose, CA, USA, July 29*. 2008, p. 6.
- [32] Igor Nai Fovino, Andrea Carcano, Thibault De Lacheze Murel, Alberto Trombetta, and Marcelo Masera. “Modbus/DNP3 State-Based Intrusion Detection System.” In: *24th IEEE International Conference on Advanced Information Networking and Applications, AINA 2010, Perth, Australia, 20-13 April*. 2010, pp. 729–736.
- [33] Robin Sommer and Vern Paxson. “Outside the Closed World: On Using Machine Learning for Network Intrusion Detection.” In: *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berkeley/Oakland, California, USA*. 2010, pp. 305–316.
- [34] Theuns Verwoerd and Ray Hunt. “Intrusion detection techniques and approaches.” In: *Computer Communications* 25.15 (2002), pp. 1356–1365.
- [35] Mohamed Endi, Y.Z. Elhalwagy, and Attalla Hashad. “Three-layer PLC/SCADA system architecture in process automation and data monitoring.” In: *The 2nd International Conference on Computer and Automation Engineering (ICCAE)*. Vol. 2. IEEE. 2010, pp. 774–779.
- [36] Igor Nai Fovino, Alessio Coletta, and Marcelo Masera. *Taxonomy of security solutions for the SCADA sector*. Project ESCORTS - Security of Control Land Realtime Systems. Public Deliverable. 2010.
- [37] Modbus-IDA. *Modbus application protocol specification v1.1b3*. Online at: <http://www.modbus.org/specs.php>. 2006.
- [38] Andy Swales. *Open Modbus/TCP Specification*. 1999.

- [39] Andy Swales and Chris Gray. “Transparent factories through industrial internets.” In: *Canadian Conference on Electrical and Computer Engineering*. Vol. 2. IEEE. 1999, pp. 931–936.
- [40] Bruno Dutertre. “Formal Modeling and Analysis of the Modbus Protocol.” In: *Critical Infrastructure Protection, Post-Proceedings of the First Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection, Dartmouth College, Hanover, New Hampshire, USA, March 19-21, 2007*. Springer, 2007, pp. 189–204.
- [41] ISO. *Industrial automation systems 9506 – Manufacturing Message Specification – Part 1: Service definition*. International Standard. 2003.
- [42] ISO. *Industrial automation systems 9506 – Manufacturing Message Specification – Part 2: Protocol specification*. International Standard. 2003.
- [43] IEC. *Telecontrol equipment and systems 60870 – Part 5-101: Transmission protocols – Companion standard for basic telecontrol tasks*. International Standard. 2003.
- [44] IEC. *Telecontrol equipment and systems 60870 – Part 5-104: Transmission protocols – Network access for IEC 60870-5-101 using standard transport profiles*. International Standard. 2006.
- [45] Control Microsystems. *DNP3 User and Reference Manual*. 2007.
- [46] PROFIBUS Nutzerorganisation. *Application Layer protocol for decentralized periphery and distributed automation, Specification for PROFINET, Version 2.3*. 2010.
- [47] Wolfgang Kastner, Georg Neugschwandtner, Stefan Soucek, and H Michael Newman. “Communication systems for building automation and control.” In: *Proceedings of the IEEE* 93.6 (2005), pp. 1178–1203.
- [48] ANSI/ASHRAE Standard 135-2012. *A Data Communication Protocol for Building Automation and Control Networks*. 2012.
- [49] KNX Association. *KNX System Specifications, v3.0*. 2009.
- [50] Siemens. *Communication in Building Automation*. 2014.
- [51] LonWorks Echelon. *The LonTalk protocol specification*. 2003.
- [52] James P. Anderson. *Computer security threat monitoring and surveillance*. Tech. rep. James P. Anderson Company, 1980.
- [53] Dorothy E. Denning. “An Intrusion-Detection Model.” In: *Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 7-9. 1986*, pp. 118–133.
- [54] Hervé Debar, Marc Dacier, and Andreas Wespi. “Towards a taxonomy of intrusion-detection systems.” In: *Computer Networks* 31.8 (1999), pp. 805–822.
- [55] Pedro Garcia-Teodoro, Jesús E. Díaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. “Anomaly-based network intrusion detection: Techniques, systems and challenges.” In: *Computers & Security* 28.1-2 (2009), pp. 18–28.

- [56] Camen Torrano-Giménez, Alejandro Perez-Villegas, and Gonzalo Álvarez Marañón. “An anomaly-based approach for intrusion detection in web traffic.” In: *Journal of Information Assurance and Security (JIAS)* 5.4 (2010), pp. 446–454.
- [57] R Sekar, Ajay Gupta, James Frullo, Tushar Shanbhag, Abhishek Tiwari, Henglin Yang, and Sheng Zhou. “Specification-based anomaly detection: a new approach for detecting network intrusions.” In: *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22. 2002*, pp. 265–274.
- [58] Phillip A. Porras and Alfonso Valdes. “Live Traffic Analysis of TCP/IP Gateways.” In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 1998, San Diego, California, USA. 1998*.
- [59] Harold S. Javitz and Alfonso Valdes. “The SRI IDES Statistical Anomaly Detector.” In: *IEEE Symposium on Security and Privacy. 1991*, pp. 316–326.
- [60] Wenke Lee and Salvatore J. Stolfo. “Data Mining Approaches for Intrusion Detection.” In: *Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, USA, January 26-29. 1998*.
- [61] Aleksandar Lazarevic, Vipin Kumar, and Jaideep Srivastava. “Intrusion detection: A Survey.” In: *Managing Cyber Threats*. Springer, 2005, pp. 19–78.
- [62] Nong Ye, Syed Masum Emran, Qiang Chen, and Sean Vilbert. “Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection.” In: *IEEE Transaction on Computers* 51.7 (2002), pp. 810–820.
- [63] Andreas Wespi, Marc Dacier, and Hervé Debar. “Intrusion Detection Using Variable-Length Audit Trail Patterns.” In: *Proceedings of Recent Advances in Intrusion Detection, Third International Workshop, RAID 2000, Toulouse, France, October 2-4. 2000*, pp. 110–129.
- [64] Hervé Debar, Marc Dacier, Mehdi Nassehi, and Andreas Wespi. “Fixed vs. Variable-Length Patterns for Detecting Suspicious Process Behavior.” In: *Proceedings in Computer Security - ESORICS 98, 5th European Symposium on Research in Computer Security, Louvain-la-Neuve, Belgium, September 16-18. 1998*, pp. 1–15.
- [65] Christoph C. Michael and Anup K. Ghosh. “Two State-based Approaches to Program-based Anomaly Detection.” In: *The 16th Annual Computer Security Applications Conference (ACSAC 2000), 11-15 December 2000, New Orleans, Louisiana, USA. 2000*, pp. 21–30.
- [66] Christopher Krügel, Darren Mutz, William K. Robertson, and Fredrik Valeur. “Bayesian Event Classification for Intrusion Detection.” In: *The 19th Annual Computer Security Applications Conference (ACSAC 2003), 8-12 December 2003, Las Vegas, NV, USA. 2003*, pp. 14–23.

- [67] Dit-Yan Yeung and Yuxin Ding. “Host-based intrusion detection using dynamic and static behavioral models.” In: *Pattern Recognition* 36.1 (2003), pp. 229–243.
- [68] Hervé Debar, Monique Becker, and Didier Siboni. “A neural network component for an intrusion detection system.” In: *1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA, May 4-6, 1992*, pp. 240–250.
- [69] Loye Lynn Ray. “Training And Testing Anomaly-Based Neural Network Intrusion Detection Systems.” In: *International Journal of Information Security Science* 2.2 (2013), pp. 57–63.
- [70] John E. Dickerson and Julie A. Dickerson. “Fuzzy network profiling for intrusion detection.” In: *The 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS)*. IEEE. 2000, pp. 301–306.
- [71] Susan M. Bridges and Rayford B. Vaughn. “Fuzzy data mining and genetic algorithms applied to intrusion detection.” In: *Proceedings of 12th Annual Canadian Information Technology Security Symposium*. 2000, pp. 109–122.
- [72] Karlton Sequeira and Mohammed Javeed Zaki. “ADMIT: anomaly-based data mining for intrusions.” In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. 2002, pp. 386–395.
- [73] Yihua Liao and V. Rao Vemuri. “Use of K-Nearest Neighbor classifier for intrusion detection.” In: *Computers & Security* 21.5 (2002), pp. 439–448.
- [74] Dina Hadžiosmanović, Lorenzo Simionato, Damiano Bolzoni, Emmanuele Zamboni, and Sandro Etalle. “N-Gram against the Machine: On the Feasibility of the N-Gram Network Analysis for Binary Protocols.” In: *Proceedings of the Research in Attacks, Intrusions, and Defenses - 15th International Symposium, RAID 2012, Amsterdam, The Netherlands, September 12-14, 2012*. Elsevier, 2012, pp. 354–373.
- [75] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. “Intrusion Detection Using Sequences of System Calls.” In: *Journal of Computer Security* 6.3 (1998), pp. 151–180.
- [76] Arati Baliga, Vinod Ganapathy, and Liviu Iftode. “Automatic Inference and Enforcement of Kernel Data Structure Invariants.” In: *The 24th Annual Computer Security Applications Conference, ACSAC 2008, Anaheim, California, USA, 8-12 December*. IEEE. 2008, pp. 77–86.
- [77] Brendan Dolan-Gavitt, Abhinav Srivastava, Patrick Traynor, and Jonathon T. Giffin. “Robust signatures for kernel data structures.” In: *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, pp. 566–577.

- [78] Martín Abadi, Mihai Budiu, Úlfar Erlingsson, and Jay Ligatti. “Control-flow integrity.” In: *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, pp. 340–353.
- [79] S.H.C. Haris, Ghossoon M. Waleed Al-Saadoon, R.B. Ahmad, and Maha Ghani. “Anomaly detection of IP header threats.” In: *International Journal of Computer Science and Security (IJCSS)* 4.6 (2011), p. 497.
- [80] Ke Wang and Salvatore J. Stolfo. “Anomalous Payload-Based Network Intrusion Detection.” In: *Proceedings of Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004*. Springer. 2004, pp. 203–222.
- [81] Damiano Bolzoni, Sandro Etalle, Pieter H. Hartel, and Emmanuele Zambon. “POSEIDON: a 2-tier Anomaly-based Network Intrusion Detection System.” In: *Proceedings of the 4th IEEE International Workshop on Information Assurance (IWIA 2006), 13-14 April 2006, Egham, Surrey, UK*. 2006, pp. 144–156.
- [82] Walter J. Scheirer and Mooi Choo Chuah. “Network intrusion detection with semantics-aware capability.” In: *The 20th International Parallel and Distributed Processing Symposium (IPDPS 2006), Proceedings, 25-29 April 2006, Rhodes Island, Greece*. IEEE. 2006, 7–pp.
- [83] Marc Damashek. “Gauging similarity with n-grams: Language-independent categorization of text.” In: *Science* 267.5199 (1995), pp. 843–848.
- [84] Myung-Sup Kim, Hun-Jeong Kang, Seong-Cheol Hong, Seung-Hwa Chung, and James W. Hong. “A flow-based method for abnormal network traffic detection.” In: *Proceedings of Managing Next Generation Convergence Networks and Services, IEEE/IFIP Network Operations and Management Symposium, NOMS 2004, Seoul, Korea, 19-23 April 2004*. Vol. 1. 2004, pp. 599–612.
- [85] Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras. “Flow whitelisting in SCADA networks.” In: *International Journal of Critical Infrastructure Protection IJCIP* 6.3-4 (2013), pp. 150–158.
- [86] Calvin Ko, George Fink, and Karl N. Levitt. “Automated detection of vulnerabilities in privileged programs by execution monitoring.” In: *The 10th Annual Computer Security Applications Conference, ACSAC 1994, 5-9 December, 1994 Orlando, FL, USA*. IEEE. 1994, pp. 134–144.
- [87] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. “A Sense of Self for Unix Processes.” In: *IEEE Symposium on Security and Privacy, May 6-8, 1996, Oakland, CA, USA*. 1996, pp. 120–128.
- [88] Calvin Ko, Manfred Ruschitzka, and Karl N. Levitt. “Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-Based Approach.” In: *IEEE Symposium on Security and Privacy, May 4-7, 1997, Oakland, CA, USA*. 1997, pp. 175–187.

- [89] Calvin Ko, Paul Brutch, Jeff Rowe, Guy Tsafnat, and Karl N. Levitt. "System Health and Intrusion Monitoring Using a Hierarchy of Constraints." In: *Proceedings of Recent Advances in Intrusion Detection, 4th International Symposium, RAID 2001 Davis, CA, USA, October 10-12*. Springer. 2001, pp. 190–204.
- [90] Ivan Balepin, Sergei Maltsev, Jeff Rowe, and Karl N. Levitt. "Using Specification-Based Intrusion Detection for Automated Response." In: *Proceedings of Recent Advances in Intrusion Detection, 6th International Symposium, RAID 2003, Pittsburgh, PA, USA, September 8-10*. Springer. 2003, pp. 136–154.
- [91] R. Sekar, Yong Cai, and Mark Segal. "A specification-based approach for building survivable systems." In: *Proceedings of the 1998 National Information Systems Security Conference (NISSC'98)*. Vol. 11. Citeseer. 1998, pp. 338–347.
- [92] R. Sekar and Prem Uppuluri. "Synthesizing Fast Intrusion Prevention/Detection Systems from High-Level Specifications." In: *Proceedings of the 8th USENIX Security Symposium, Washington, D.C., August 23-26, 1999*. 1999.
- [93] Chin-Yang Tseng, Poornima Balasubramanyam, Calvin Ko, Rattapon Limprasittiporn, Jeff Rowe, and Karl N. Levitt. "A specification-based intrusion detection system for AODV." In: *Proceedings of the 1st ACM Workshop on Security of ad hoc and Sensor Networks, SASN 2003, Fairfax, Virginia, USA*. 2003, pp. 125–134.
- [94] Jean-Marie Orset, Baptiste Alcalde, and Ana R. Cavalli. "An EFSM-Based Intrusion Detection System for Ad Hoc Networks." In: *Proceedings of Automated Technology for Verification and Analysis, Third International Symposium, ATVA 2005, Taipei, Taiwan, October 4-7*. Springer, 2005, pp. 400–413.
- [95] Hoda M. Hassan, Mohy Mahmoud, and Sherif El-Kassas. "Securing the AODV protocol using specification-based intrusion detection." In: *Proceedings of the 2nd ACM Workshop on Q2S and Security for Wireless and Mobile Networks (Q2SWinet'06), Terromolinos, Spain, October 2*. 2006, pp. 33–36.
- [96] Tao Song, Calvin Ko, Chinyang Henry Tseng, Poornima Balasubramanyam, Anant Chaudhary, and Karl N. Levitt. "Formal Reasoning About a Specification-Based Intrusion Detection for Dynamic Auto-configuration Protocols in Ad Hoc Networks." In: *Formal Aspects in Security and Trust, Third International Workshop, FAST 2005, Newcastle upon Tyne, UK, July 18-19, 2005, Revised Selected Papers*. Springer, 2005, pp. 16–33.
- [97] Chinyang Henry Tseng, Tao Song, Poornima Balasubramanyam, Calvin Ko, and Karl N. Levitt. "A Specification-Based Intrusion Detection Model for OLSR." In: *Proceedings of Recent Advances in Intrusion Detection, 8th International Symposium, RAID 2005, Seattle, WA, USA, September 7-9*. Springer. 2005, pp. 330–350.

- [98] Jimmi Grönkvist, Anders Hansson, and Mattias Sköld. “Evaluation of a specification-based intrusion detection system for AODV.” In: *The 6th Annual Mediterranean Ad Hoc Networking Workshop*. 2007, pp. 121–128.
- [99] Rupinder Gill, Jason Smith, and Andrew J. Clark. “Specification-Based Intrusion Detection in WLANs.” In: *22nd Annual Computer Security Applications Conference (ACSAC 2006), 11-15 December 2006, Miami Beach, Florida, USA*. IEEE, 2006, pp. 141–152.
- [100] Phu Truong, Dennis Nieh, and Melody Moh. “Specification-based intrusion detection for H.323-based voice over IP.” In: *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology*. 2005, pp. 387–392.
- [101] Pan Jieke, João Redol, and Miguel Correia. “Specification-based Intrusion Detection System for Carrier Ethernet.” In: *WEBIST 2007 - Proceedings of the Third International Conference on Web Information Systems and Technologies, Volume IT, Barcelona, Spain, March 3-6*. 2007, pp. 426–429.
- [102] Nick L. Petroni Jr., Timothy Fraser, Aaron Walters, and William A. Arbaugh. “An Architecture for Specification-Based Detection of Semantic Integrity Violations in Kernel Dynamic Data.” In: *Proceedings of the 15th USENIX Security Symposium, Vancouver, BC, Canada, July 31 - August 4*. 2006.
- [103] Ashwin Chaugule, Zhi Xu, and Sencun Zhu. “A Specification Based Intrusion Detection Framework for Mobile Phones.” In: *Proceedings of Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10*. Springer. 2011, pp. 19–37.
- [104] Steven Cheung, Bruno Dutertre, Martin Fong, Ulf Lindqvist, Keith Skinner, and Alfonso Valdes. “Using model-based intrusion detection for SCADA networks.” In: *Proceedings of the SCADA Security Scientific Symposium*. Vol. 46. 2007, pp. 1–12.
- [105] Paria Jokar, Hasen Nicanfar, and Victor C. M. Leung. “Specification-based Intrusion Detection for home area networks in smart grids.” In: *IEEE Second International Conference on Smart Grid Communications, SmartGridComm 2011, Brussels, Belgium, October 17-20*. IEEE. 2011, pp. 208–213.
- [106] Hui Lin, Adam J. Slagell, Catello Di Martino, Zbigniew Kalbarczyk, and Ravishankar K. Iyer. “Adapting Bro into SCADA: building a specification-based intrusion detection system for the DNP3 protocol.” In: *Cyber Security and Information Intelligence, CSIIRW '13, Oak Ridge, TN, USA, January 8-10*. ACM. 2013, p. 5.
- [107] Robin Berthier and William H. Sanders. “Specification-Based Intrusion Detection for Advanced Metering Infrastructures.” In: *17th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2011, Pasadena, CA, USA, December 12-14*. IEEE. 2011, pp. 184–193.

- [108] Hadeli Hadeli, Ragnar Schierholz, Markus Braendle, and Cristian Tuduce. “Leveraging Determinism in Industrial Control Systems for Advanced Anomaly Detection and Reliable Security Configuration.” In: *Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2009, September 22-25, 2008, Palma de Mallorca, Spain*. IEEE. 2009, pp. 1–8.
- [109] David P. Duggan, Michael Berg, John Dillinger, and Jason Stamp. *Penetration testing of industrial control systems*. Tech. rep. Sandia National Laboratories, 2005.
- [110] Abdun Naser Mahmood, Christopher Leckie, Jiankun Hu, Zahir Tari, and Mohammed Atiquzzaman. “Network Traffic Analysis and SCADA Security.” In: *Handbook of Information and Communication Security*. Springer, 2010, pp. 383–405.
- [111] Ali Abbasi. *Ghost in the PLC: stealth on-the-fly manipulation of programmable logic controllers’ I/O*. Tech. rep. TR-CTIT-16-02. 2016.
- [112] Dina Hadziosmanovic, Damiano Bolzoni, Sandro Etalle, and Pieter H. Hartel. “Challenges and opportunities in securing industrial control systems.” In: *Complexity in Engineering, COMPENG 2012, Aachen, Germany, June 11-13*. IEEE. 2012, pp. 1–6.
- [113] Cristina Alcaraz, Javier Lopez, Jianying Zhou, and Rodrigo Roman. “Secure SCADA framework for the protection of energy control systems.” In: *Concurrency and Computation: Practice and Experience* 23.12 (2011), pp. 1431–1442.
- [114] Mark Fabro and Trent Nelson. *Control systems cyber security: Defense-in-depth strategies*. Tech. rep. INL/CON-07-12804. Idaho National Laboratory (INL), 2007.
- [115] Aravind K Joshi. “Natural language processing.” In: *Science* 253.5025 (1991), pp. 1242–1249.
- [116] Martin Roesch. “Snort: Lightweight Intrusion Detection for Networks.” In: *Proceedings of the 13th Conference on Systems Administration (LISA-99), Seattle, WA, November 7-12*. 1999, pp. 229–238.
- [117] Vern Paxson. “Bro: A System for Detecting Network Intruders in Real-Time.” In: *Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, USA, January 26-29*. 1998.
- [118] Fyodor Yarochkin. “Remote OS detection via TCP/IP stack fingerprinting.” In: *Phrack Magazine* 17.3 (1998), pp. 1–10.
- [119] Hubert Zimmermann. “OSI reference model—The ISO model of architecture for open systems interconnection.” In: *IEEE Transactions on Communications* 28.4 (1980), pp. 425–432.
- [120] Franck Veyssset, Olivier Courtay, Olivier Heen, et al. *New tool and technique for remote operating system fingerprinting*. Tech. rep. Intranode Software Technologies, 2002.

- [121] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. “Traffic classification through simple statistical fingerprinting.” In: *ACM SIGCOMM Computer Communication Review* 37.1 (2007), pp. 5–16.
- [122] Vern Paxson. “Automated Packet Trace Analysis of TCP Implementations.” In: *Proceedings of the ACM SIGCOMM 1997 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, September 14-18, 1997, Cannes, France*. ACM. 1997, pp. 167–179.
- [123] Greg Taleck. “Ambiguity Resolution via Passive OS Fingerprinting.” In: *Proceedings of Recent Advances in Intrusion Detection, 6th International Symposium, RAID 2003, Pittsburgh, PA, USA, September 8-10*. Springer. 2003, pp. 192–206.
- [124] Patrick Haffner, Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. “ACAS: automated construction of application signatures.” In: *Proceedings of the 1st Annual ACM Workshop on Mining Network Data, MineNet 2005, Philadelphia, Pennsylvania, USA, August 26*. ACM. 2005, pp. 197–202.
- [125] Andrew W. Moore and Konstantina Papagiannaki. “Toward the Accurate Identification of Network Applications.” In: *Proceedings of Passive and Active Network Measurement, 6th International Workshop, PAM 2005, Boston, MA, USA, March 31 - April 1*. 2005, pp. 41–54.
- [126] Ryan M. Gerdes, Thomas E. Daniels, Mani Mina, and Steve Russell. “Device Identification via Analog Signal Fingerprinting: A Matched Filter Approach.” In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2006, San Diego, California, USA*. 2006.
- [127] Tadayoshi Kohno, Andre Broido, and Kimberly C. Claffy. “Remote Physical Device Fingerprinting.” In: *2005 IEEE Symposium on Security and Privacy (S&P 2005), 8-11 May 2005, Oakland, CA, USA*. IEEE, 2005, pp. 211–225.
- [128] David Moore, Ken Keys, Ryan Koga, Edouard Lagache, and Kimberly C. Claffy. “The CoralReef Software Suite as a Tool for System and Network Administrators.” In: *Proceedings of the 15th Conference on Systems Administration (LISA 2001), San Diego, California, USA, December 2-7*. USENIX Association. 2001, pp. 133–144.
- [129] Gordon Lyon. *Nmap security scanner*. 2013. URL: <http://nmap.org/> (visited on 11/01/2016).
- [130] Robert Beverly. “A Robust Classifier for Passive TCP/IP Fingerprinting.” In: *Proceedings of Passive and Active Network Measurement, 5th International Workshop, PAM 2004, Antibes Juan-les-Pins, France, April 19-20*. Springer, 2004, pp. 158–167.
- [131] Yoav Freund and Robert E. Schapire. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.” In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

- [132] Edward Alan Jackson. *Detecting intrusions at layer one: device fingerprinting for network access authorization*. 2006.
- [133] Michal Zalewski. *P0f: Passive OS Fingerprinting tool*. 2006. URL: <http://lcamtuf.coredump.cx/p0f.shtml> (visited on 11/01/2016).
- [134] Fedor V. Yarochkin, Ofir Arkin, Meder Kydyraliev, Shih-Yao Dai, Yen-nun Huang, and Sy-Yen Kuo. “Xprobe2++: Low volume remote network information gathering tool.” In: *Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2009, Estoril, Lisbon, Portugal, June 29 - July 2*. IEEE. 2009, pp. 205–210.
- [135] Patrice Auffret. “SinFP, unification of active and passive operating system fingerprinting.” In: *Journal in Computer Virology* 6.3 (2010), pp. 197–205.
- [136] David Watson, Matthew Smart, G. Robert Malan, and Farnam Jahanian. “Protocol scrubbing: network security through transparent flow modification.” In: *IEEE/ACM Transactions on Networking* 12.2 (2004), pp. 261–273.
- [137] Jason Stamp, John Dillinger, William Young, and Jennifer DePoy. *Common vulnerabilities in critical infrastructure control systems*. Tech. rep. SAND2003-1772C. Sandia National Laboratories, 2003.
- [138] Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras. “A first look into SCADA network traffic.” In: *2012 IEEE Network Operations and Management Symposium, NOMS 2012, Maui, HI, USA, April 16-20*. 2012, pp. 518–521.
- [139] Dina Hadziosmanovic, Robin Sommer, Damiano Bolzoni, and Pieter Hartel. *Improving SCADA Security with Context-aware Network Profiling*. Poster at IEEE Security and Privacy poster abstracts.
- [140] Sergey Gordeychik. *SCADA Strangelove or: How I Learned to Start Worrying and Love Nuclear Plants*. 2013. URL: <http://scadastrangelove.blogspot.nl/> (visited on 11/01/2016).
- [141] SCADA Strangelove. *PLCScan*. 2012. URL: <https://code.google.com/archive/p/plcscan/> (visited on 11/01/2016).
- [142] Gerard Salton and Michael E. Lesk. “The SMART automatic document retrieval systems - an illustration.” In: *ACM Communications* 8.6 (1965), pp. 391–398.
- [143] Justin Zobel, Alistair Moffat, and Kotagiri Ramamohanarao. “Inverted Files Versus Signature Files for Text Indexing.” In: *ACM Transactions on Database Systems* 23.4 (1998), pp. 453–490.
- [144] Ben Carterette and Fazli Can. “Comparing inverted files and signature files for searching a large lexicon.” In: *Information Processing and Management* 41.3 (2005), pp. 613–633.
- [145] Cartic Ramakrishnan, Abhishek Patnia, Eduard H. Hovy, and Gully A. P. C. Burns. “Layout-aware text extraction from full-text PDF of scientific articles.” In: *Source Code for Biology and Medicine* 7 (2012), p. 7.

- [146] Thomas Schwentick. “Automata for XML - A survey.” In: *Journal of Computer and System Sciences* 73.3 (2007), pp. 289–315.
- [147] “W3C XML Path Language.” In: *Encyclopedia of Social Network Analysis and Mining*. 2014, p. 2337.
- [148] “XSL Transformations.” In: *Encyclopedia of Social Network Analysis and Mining*. 2014, p. 2432.
- [149] Burcu Yildiz, Katharina Kaiser, and Silvia Miksch. “pdf2table: A Method to Extract Table Information from PDF Files.” In: *Proceedings of the 2nd Indian International Conference on Artificial Intelligence (IICAI), Pune, India, December 20-22. 2005*, pp. 1773–1785.
- [150] AKS-Labs. *PDF2TXT*. 2001. URL: <http://www.pdf2txt.com/> (visited on 11/01/2016).
- [151] The Sensible Code Company. *PDFTables*. 2010. URL: <https://pdftables.com/> (visited on 11/01/2016).
- [152] Tomek Strzalkowski. “Natural Language Information Retrieval.” In: *Information Processing and Management* 31.3 (1995), pp. 397–417.
- [153] Hassan Saneifar, Stéphane Bonniol, Anne Laurent, Pascal Poncelet, and Mathieu Roche. “Terminology Extraction from Log Files.” In: *Proceedings of Database and Expert Systems Applications, 20th International Conference, DEXA 2009, Linz, Austria, August 31 - September 4. 2009*, pp. 769–776.
- [154] Norman Sadeh, Alessandro Acquisti, Travis D. Breaux, Lorrie Faith Cranor, Aleecia M. McDonald, Joel R. Reidenberg, Noah A. Smith, Fei Liu, N. Cameron Russell, Florian Schaub, et al. *The usable privacy policy project: Combining crowdsourcing, machine learning and natural language processing to semi-automatically answer those privacy questions users care about*. 2013.
- [155] Stanislaw Osinski and Dawid Weiss. “Carrot²: Design of a Flexible and Efficient Web Information Retrieval Framework.” In: *Proceedings of Advances in Web Intelligence Third International Atlantic Web Intelligence Conference, AWIC 2005, Lodz, Poland, June 6-9. 2005*, pp. 439–444.
- [156] Hamish Cunningham. “GATE, a General Architecture for Text Engineering.” In: *Computers and the Humanities* 36.2 (2002), pp. 223–254.
- [157] Nick Doty. *Automated text analysis of Requests for Comment (RFCs)*. 2015. URL: <https://github.com/npdoty/rfc-analysis> (visited on 11/01/2016).
- [158] Stephen E. McLaughlin and Patrick McDaniel. “SABOT: specification-based payload generation for programmable logic controllers.” In: *the ACM Conference on Computer and Communications Security, CCS’12, Raleigh, NC, USA, October 16-18. 2012*, pp. 439–449.

- [159] Igor Nai Fovino, Andrea Carcano, Thibault De Lacheze Murel, Alberto Trombetta, and Marcelo Masera. “Modbus/DNP3 State-Based Intrusion Detection System.” In: *24th IEEE International Conference on Advanced Information Networking and Applications, AINA 2010, Perth, Australia, 20-13 April. 2010*, pp. 729–736.
- [160] Edward M Roche. “Critical Foundations: Protecting America’s Infrastructures.” In: *Journal of Global Information Technology Management 1.1 (1998)*. President’s Commission on Critical Infrastructure Protection, pp. 49–50.
- [161] Tammo Krueger, Hugo Gascon, Nicole Krämer, and Konrad Rieck. “Learning stateful models for network honeypots.” In: *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence, AISec 2012, Raleigh, NC, USA, October 19. ACM. 2012*, pp. 37–48.
- [162] Niv Goldenberg and Avishai Wool. “Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems.” In: *International Journal of Critical Infrastructure Protection (IJCIP) 6.2 (2013)*, pp. 63–75.
- [163] Man-Ki Yoon and Gabriela F Ciocarlie. “Communication pattern monitoring: Improving the utility of anomaly detection for industrial control systems.” In: *NDSS Workshop on Security of Emerging Networking Technologies. 2014*.
- [164] Dina Hadziosmanovic, Robin Sommer, Emmanuele Zambon, and Pieter H. Hartel. “Through the eye of the PLC: semantic security monitoring for industrial processes.” In: *Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC 2014, New Orleans, LA, USA, December 8-12. ACM. 2014*, pp. 126–135.
- [165] Terran Lane and Carla E. Brodley. “Sequence matching and learning in anomaly detection for computer security.” In: *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management. 1997*, pp. 43–49.
- [166] Guojun Mao, Jing Zhang, and Xindong Wu. “Intrusion detection based on the short sequence model.” In: *7th World Congress on Intelligent Control and Automation (WCICA 2008)*. IEEE. 2008, pp. 1449–1454.
- [167] Christina Warrender, Stephanie Forrest, and Barak A. Pearlmutter. “Detecting Intrusions using System Calls: Alternative Data Models.” In: *IEEE Symposium on Security and Privacy, Oakland, California, USA, May 9-12. 1999*, pp. 133–145.
- [168] Gerald Combs. *Tshark: the wireshark network analyser*. 1998. URL: [URLhttp://www.wireshark.org](http://www.wireshark.org) (visited on 11/01/2016).
- [169] Michael Newman. *BACnet: The Global Standard for Building Automation and Control Networks*. Momentum Press, 2013.
- [170] David G Holmberg and D Evans. *BACnet wide area network security threat assessment*. Tech. rep. US Department of Commerce, National Institute of Standards and Technology (NIST), 2003.

- [171] Steffen Wendzel, Benjamin Kahler, and Thomas Rist. “Covert Channels and Their Prevention in Building Automation Protocols: A Prototype Exemplified Using BACnet.” In: *2012 IEEE International Conference on Green Computing and Communications, Conference on Internet of Things, and Conference on Cyber-Physical and Social Computing, GreenCom/iThings-/CPSCoM 2012, Besancon, France, November 20-23, 2012*. IEEE. 2012, pp. 731–736.
- [172] Sebastian Szlósarczyk, Steffen Wendzel, Jaspreet Kaur, Michael Meier, and Frank Schubert. “Towards Suppressing Attacks on and Improving Resilience of Building Automation Systems - an Approach Exemplified Using BACnet.” In: *Sicherheit 2014: Sicherheit, Schutz und Zuverlässigkeit, Beiträge der 7. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), Vienna, Austria, March 19-21, 2014*, pp. 407–418.
- [173] Jaspreet Kaur, Jernej Tonejc, Steffen Wendzel, and Michael Meier. “Securing BACnet’s Pitfalls.” In: *ICT Systems Security and Privacy Protection*. Springer, 2015, pp. 616–629.
- [174] John C Matherly. *SHODAN the computer search engine*. 2009. URL: <http://www.shodanhq.com/help> (visited on 11/01/2016).
- [175] Timothy I. Salsbury. “The Smart Building.” In: *Springer Handbook of Automation*. Springer, 2009, pp. 1079–1093.
- [176] Michel F. Sanner. “Python: a programming language for software integration and development.” In: *Journal of Molecular Graphics and Modelling* 17.1 (1999), pp. 57–61.
- [177] Robin Sommer, Johanna Amann, and Seth Hall. “Spicy: A Unified Deep Packet Inspection Framework for Safely Dissecting All Your Data.” In: *The 32th Annual Computer Security Applications Conference, ACSAC 2016, Los Angeles, California, USA, USA, 5-9 December 2016*. (to appear). IEEE. 2016.
- [178] Luca Deri. “Improving passive packet capture: Beyond device polling.” In: *Proceedings of System Administration and Network Engineering (SANE)*. Vol. 2004. 2004, pp. 85–93.
- [179] Prem Uppuluri and R. Sekar. “Experiences with Specification-Based Intrusion Detection.” In: *Proceedings of Recent Advances in Intrusion Detection, 4th International Symposium, RAID 2001 Davis, CA, USA, October 10-12*. Springer, 2001, pp. 172–189.
- [180] Daniel Gildea and Daniel Jurafsky. “Automatic Labeling of Semantic Roles.” In: *Computational Linguistics* 28.3 (2002), pp. 245–288.
- [181] Ulf E. Larson, Dennis K. Nilsson, and Erland Jonsson. “An approach to specification-based attack detection for in-vehicle networks.” In: *Intelligent Vehicles Symposium*. IEEE. 2008, pp. 220–225.

- [182] Pierre Kleberger, Tomas Olovsson, and Erland Jonsson. “Security aspects of the in-vehicle network in the connected car.” In: *IEEE Intelligent Vehicles Symposium (IV), 2011, Baden-Baden, Germany, June 5-9, 2011*, pp. 528–533.
- [183] Stefan Dietzel, Rens Wouter van der Heijden, Hendrik Decke, and Frank Kargl. “A flexible, subjective logic-based framework for misbehavior detection in V2V networks.” In: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2014, Sydney, Australia, June 19*. IEEE. 2014, pp. 1–6.