Data-Driven Anomaly Detection in Industrial Networks

Mikel Iturbe Urretxa

Elektronika eta Informatika Saila

Goi Eskola Politeknikoa

Mondragon Unibertsitatea

March 2017

MONDRAGON UNIBERTSITATEA

Data-Driven Anomaly Detection in Industrial Networks

Mikel Iturbe Urretxa

Advisors:

Dr. Urko Zurutuza Ortega

Dr. Roberto Uribeetxeberria Ezpeleta

Thesis submitted for the degree of Doctor of Philosophy
at Mondragon Unibertsitatea

March 2017

Alboan izanagatik,
egite honetan lagundu didazuen bidelagunei,
abentura hau gozagarri bihurtzeraino.

Lan hau, zuena (ere) (ba)da.

Originality Statement

“The purpose of computing is insight, not numbers.”
—Richard Hamming. In *Numerical Methods for Scientists and Engineers*, 1962, p. iv.

“The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards - and even then I have my doubts.”
— Eugene H. Spafford. In “Computer Recreations: Of Worms, Viruses and Core War”, *Scientific American*, March 1989, p. 110.

“Data isn’t information, any more than fifty tons of cement is a skyscraper.”
— Clifford Stoll. In *Silicon Snake Oil: Second Thoughts on the Information Highway*, 1996, pp. 193-194.

Abstract

Since the conception of the first Programmable Logic Controllers (PLCs) in the 1960s, Industrial Control Systems (ICSs) have evolved vastly. From the primitive isolated setups, ICSs have become increasingly interconnected, slowly forming the complex networked environments, collectively known as Industrial Networks (INs), that we know today. Since ICSs are responsible for a wide range of physical processes, including those belonging to Critical Infrastructures (CIs), securing INs is vital for the well-being of modern societies. Out of the many research advances on the field, Anomaly Detection Systems (ADSs) play a prominent role. These systems monitor IN and/or ICS behavior to detect abnormal events, known or unknown. However, as the complexity of INs has increased, monitoring them in the search of anomalous trends has effectively become a Big Data problem. In other words, IN data has become too complex to process it by traditional means, due to its large scale, diversity and generation speeds. Nevertheless, ADSs designed for INs have not evolved at the same pace, and recent proposals are not designed to handle this data complexity, as they do not scale well or do not leverage the majority of the data types created in INs.

This thesis aims to fill that gap, by presenting two main contributions: (i) a visual flow monitoring system and (ii) a multivariate ADS that is able to tackle data heterogeneity and to scale efficiently. For the flow monitor, we propose a system that, based on current flow data, builds security visualizations depicting network behavior while highlighting anomalies. For the multivariate ADS, we analyze the performance of Multivariate Statistical Process Control (MSPC) for detecting and diagnosing anomalies, and later we present a Big Data, MSPC-inspired ADS that monitors field and network data to detect anomalies. The approaches are experimentally validated by building INs in test environments and analyzing the data created by them. Based on this necessity for conducting IN security research in a rigorous and reproducible environment, we also propose the design of a testbed that serves this purpose.

Acknowledgements

Nowadays, virtually no research work is produced in complete isolation, and this thesis is certainly not an exception. Many people have directly or indirectly contributed to the successful conclusion of this work, both inside and outside the labs. And, though only my name makes it to the cover, they definitely deserve credit.

First, I would like to wholeheartedly thank my advisors, Urko Zurutuza and Roberto Uribeetxeberria, for their guidance and support, both at a scientific and at a humane level. Their confidence on me from the very beginning and the freedom they trusted me with has been vital on the process of carrying out this work correctly and, more importantly, doing so in a warm, comfortable, and trusting atmosphere. Their dedication, enthusiasm and positive attitude towards their everyday work is truly contagious.

Second, a big hug to the people I have been sharing work time and space with. That includes people from the R&D unit from the Telematics Group as well as the researchers from the ICT R&D lab, veterans and newcomers alike: Aitor A. & L., Alain P., Ane A., Aritz L., Dani M & R., Ekhi R., Gorka H., Iñaki A., Iñigo L., Maite B., Markel S., Mikel M., Miren I., Miriam M., Oscar S., Pablo V., Raúl R., Unai B. & I., Urtzi M.... it has really been a pleasure, you guys rock. Notably, a special mention should go to my academic brothers. To Enaitz Ezpeleta, for our conversations about work, hiking and life in general. And to Iñaki Garitano, for introducing me to the world of industrial security when I was just starting, for being there to give me a helping hand when I was stuck, and for holding the fort when I was about to finish, so I could focus more in my thesis.

Third, I am genuinely thankful to Magnus Almgren, for my visiting stay at the Chalmers University of Technology. He introduced me to a wonderful world of research, giving me the opportunity of attending interesting seminars and tutoring students. He also made time to comment on my work and for that, I am grateful. It is difficult for me to conceive a nicer working environment. I will keep fond memories from my time in Gothenburg.

Fourth, I feel deeply indebted to José Camacho from the University of Granada. His proximity and implication with this thesis work has been truly outstanding, by sharing numerous improvements and ideas for this thesis while also taking time to answer my many questions regarding them. This thesis would have been very different without his input.

Fifth, greetings to all the nice people I have met in this endeavour while I was away from home. To Alex P., Pablo B., Rafa R., Rob W. and Roberto M., for all those long evenings and nights spent in the libraries of Granada. To Pablo P., for our morning *fika* sessions, mostly devoted to sip coffee while cursing the everlasting mist in Gothenburg. And to Wissam A., for his determination to make things work and not giving up.

Outside the labs and faculties, many people deserve a big "thank you". Family, specially my parents, who instilled in me the values of education and hard work and have always been an example to look up to. Friends, who even if jokingly encouraged me "*to get a real job*" or asked questions such as "*Are you finished yet?*" or "*What was that thing you were doing?*" helped me by blowing some steam off around good food and drinks, we did have some memorable moments. And particularly, Leire Ramirez, who during all these years has coped with the ups and downs, travels, frenzy deadlines and those days burning the candle at both ends that inevitably occur during a PhD Thesis, with nothing but understanding and unconditional support from her end.

Finally, I would like to show my gratitude to the myriad of hackers, researchers and free software developers who selflessly produce and share knowledge, tools and contents and put them freely available to society as a whole, thus contributing to make the world a bit better place. Many of these resources were used in the development of this thesis.

Eskerrik asko.
Gaizki esanak barkatu eta ondo esanak gogoan hartu.
Bilbo, 2017

Contents

List of Figures

List of Tables

List of Acronyms

ADS	Anomaly Detection System
API	Application Programming Interface
ARL	Average Run Length
BDA	Big Data Analytics
CI	Critical Infrastructure
COTS	Commercial off-the-shelf
CSV	Comma-Separated Values
DoS	Denial of Service
DDoS	Distributed Denial of Service
EDA	Exploratory Data Analysis
HDFS	Hadoop File System
HMI	Human Machine Interface
ICS	Industrial Control System
ICS-CERT	Industrial Control Systems Cyber Emergency Response Team
IDS	Intrusion Detection System
IN	Industrial Network

IT	Information Technology
MitM	Man-in-the-Middle
MSPC	Multivariate Statistical Process Control
NIST	National Institute of Standards and Technology
NOC	Normal Operation Conditions
NS	Network Simulator
NTP	Network Time Protocol
OLE	Object Linking and Embedding
OPC-UA	OLE for Process Control–Unified Architecture
OS	Operating System
OSI	Open Systems Interconnection
PC	Principal Component
PCA	Principal Component Analysis
PLC	Programmable Logic Controller
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SDN	Sofware-Defined Networking
SPC	Statistical Process Control
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TE	Tennessee-Eastman
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network

Chapter 1

Introduction

Contents

This chapter introduces the work carried out in this PhD thesis. It first describes the main reasons that have motivated the election of data-driven anomaly detection in Industrial Networks as the main scope of this thesis. Second, it lists the objectives, hypotheses and contributions of the work, along the followed research methodology for its conclusion. Finally, it provides the organization of this dissertation.

1.1 Motivation

Industrial Networks (INs) refer to the networked environments where specialized, heterogeneous, interconnected components, known collectively as Industrial Control Systems (ICSs), automate, monitor and control physical processes. As such, they are responsible for running a wide range of physical processes, both in different industrial sectors and in Critical Infrastructures (CIs) [132]. The European Council [46] defines a CI as "an asset, system or part thereof (...) which is essential for the maintenance of vital societal functions, health, safety, security, economic or social well-being of people, and the disruption or destruction of which would have a significant impact (...) as a result of the failure to maintain those functions;" Examples of ICS-controlled CIs include power generation and transport, water distribution, water waste treatment and transportation systems.

Therefore, the correct functioning of CIs has a vital importance for the well being of modern societies. Miller and Rowe [107] surveyed previous security incidents that affected CIs. Nowadays, there are two main specific concerns about the impact of IN-related attacks:

1. Successful attacks against INs may have an impact on the physical process ICSs are monitoring, potentially leading to safety-threatening scenarios. Examples of such incidents include Aurora [158], Stuxnet [92], the Maroochy water breach [129], the Georgia-Pacific incident [117] and the German steel mill attack [17].

2. The proliferation of ICS-specific malware for conducting espionage. The aim of these pieces of malware is to gather information about the controlled process and/or company running it. The purpose can be twofold: to steal confidential information about the process (e.g. recipe for manufacturing a product) or to gather information to conduct attacks against a third party. Examples of such malware include Duqu [8] and Dragonfly [135].

Traditionally, INs and ICSs have relied on two main principles for their protection, often disregarding additional security measures present in Information Technology (IT) networks

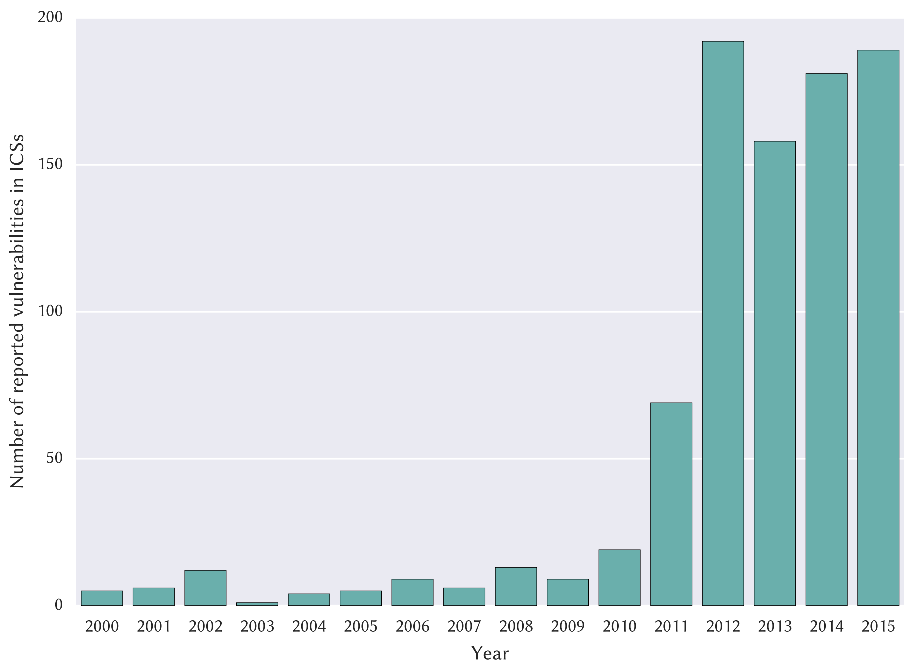


Figure 1.1: Reported ICS vulnerabilities over time [3]

(e.g. access control, authentication or encryption). The first principle is security through obscurity, where ICS vendors entrusted the secrecy of the created systems (proprietary software and hardware, private network protocols) to discourage potential attackers from finding security vulnerabilities in their products. This approach has been proven as an inefficient one [114]. While secrecy makes vulnerability discovery harder, as it is more difficult to understand the inner workings of a system, it also conveys the lack of wider verification and validation that public implementations have. Figure 1.1 shows the evolution of the publicly reported ICS vulnerabilities. While many ICS implementations are still proprietary, the number of vulnerabilities has risen since the beginning of the present decade. Some vulnerabilities even have exploits in penetration testing frameworks such as Metasploit[1]. Therefore, the intended obscurity is no longer applicable, as even if the implementation details of ICSs and INs are not known, some of their vulnerabilities and exploits are. Moreover, patching and upgrading vulnerable devices forming an IN often requires system downtime, which generally is not acceptable within the availability constraints of critical processes [32]. Thus, such processes are in many cases left unpatched and in consequence, vulnerable.

[1]`https://scadahacker.com/resources/msf-scada.html`

The other principle ICS vendors and operators have relied on for their protection is network isolation, or air-gapping. That is, INs were not to be connected with external networks. As networks would not have any outbound connections, attackers would need to obtain physical access to the facilities in order to access the IN and the ICSs in it. However, since the 1990s, pushed by the increasing demand for location-independent access to network resources, INs became progressively interconnected with external networks such as the companies' internal IT network and even the Internet [32, 69]. Project SHINE (SHodan INtelligence Extraction) discovered 2 186 971 publicly available ICSs [138] between the years 2012–2014 using the Shodan search engine. Likewise, the Shodan search engine has a public service dubbed *ICS Radar*[2] that crawls the Internet for protocols providing raw access to ICSs. Moreover, it also hosts a search category encompassing ICSs[3], that allows users to search for publicly reachable devices serving different IN protocols. The existence of these publicly available ICSs demonstrates that a large number of INs are not properly isolated and the air gap does not exist.

In the case of INs not directly connected to the Internet, they might not be publicly accessible from the Internet, but a compromise on the external IT network can provide access to the IN. For instance, in the case of the German Steel Mill incident [17], attackers were able to get access to the IT network by using a spear-phishing attack, and from there, compromise the production IN, eventually damaging a blast furnace. Even where no logical connection exists between the IT network and the IN, motivated and skilled attackers have circumvented this limitation by using alternative means, such as USB sticks (e.g. in the Stuxnet [92] incident).

The Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) provides operational capabilities to defend control systems against cyber threats [64]. In order to pursue this goal, one of its main tasks is to analyze and respond to control systems related incidents, especially the ones concerning CIs. In yearly reports detailing the number and nature of incidents regarding CIs [64–68], the ICS-CERT highlights the raise of incidents since the reporting records began. Figure 1.2 shows the evolution of the number of incidents reported to the ICS-CERT, ranging between 9 reported incidents in 2009 to 295 reported incidents in 2015.

Figure 1.3 depicts the evolution of the percentage of incidents by CI sector, based on the aforementioned incident reports. At the beginning, the sectors that accumulated most of

[2]https://ics-radar.shodan.io/

[3]https://www.shodan.io/explore/category/industrial-control-systems

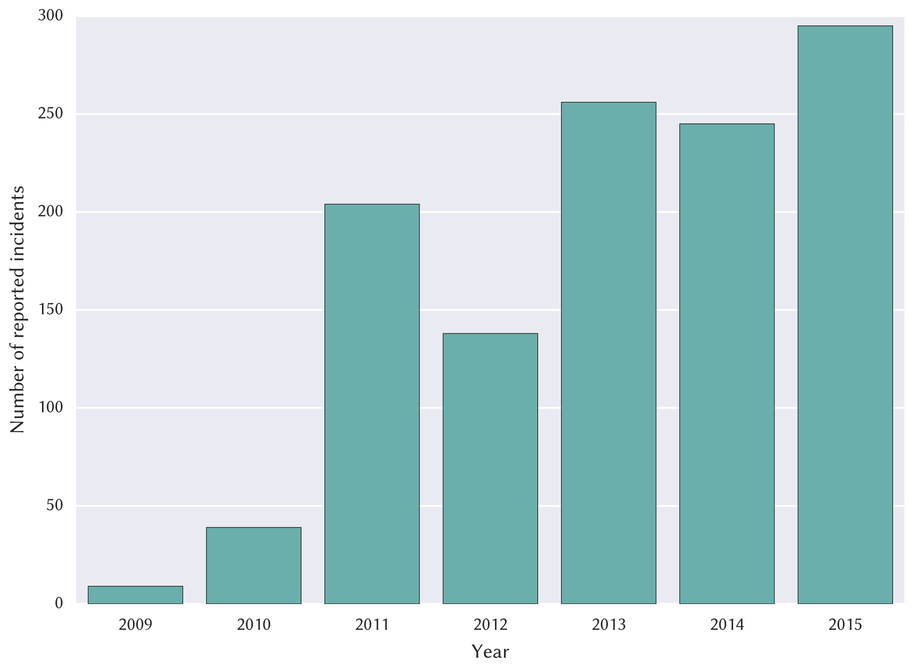


Figure 1.2: Number of reported security incidents to the ICS-CERT CIs [64–68]

the security incidents were water and energy, around 60% in total. With time, the weight of these sectors has decreased and the relevance of critical manufacturing facilities has risen to the first position, being the main target in 2015.

Due to the critical nature of CIs and the incapacity of securing them by traditional approaches, IN security is an active research field. As such, IN protection has received wide attention from both industry and the scientific community. Among the different fields of IN security research, Intrusion Detection Systems (IDSs) and, particularly, Anomaly Detection Systems (ADSs) have an important role and there are many proposals in this direction [51, 108, 164]. Previous proposals have mostly focused in either monitoring the network [51, 164], or the physical properties of the process [82, 88]. Some Anomaly Detection Systems (ADSs), especially the physical ones [88, 105, 134], are model-driven. That means that the ADS needs or creates a model of the monitored process behavior. However, modelling the nature of a complex process can be a challenging task, and outright infeasible in some scenarios. On the contrary, data-driven methods do not require to model the physical process as they make assumptions based only on data. This eases deploying data-driven ADSs in complex INs or to extend their usage in different types of INs in a simpler manner,

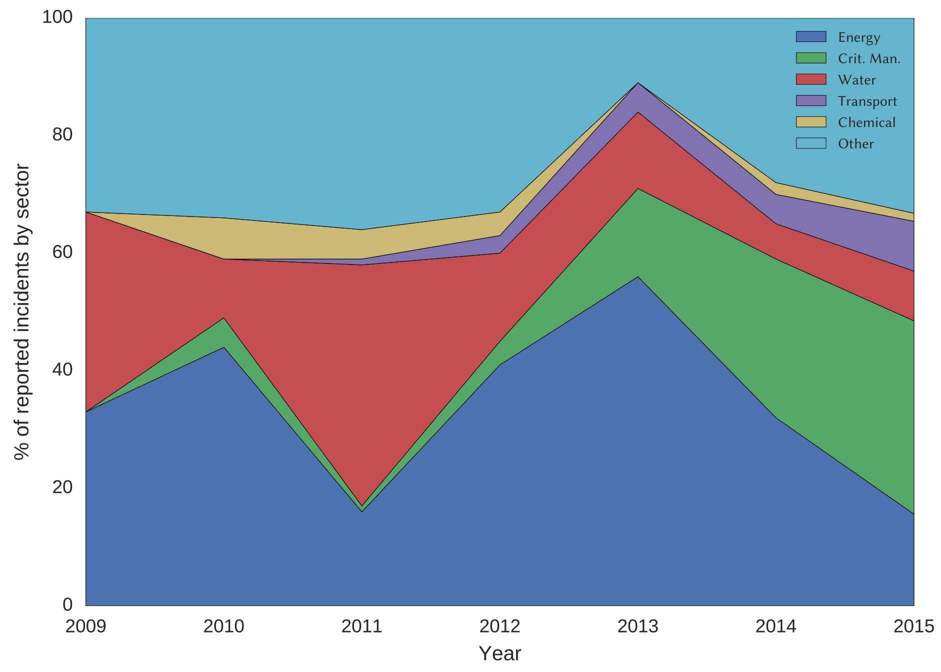


Figure 1.3: Evolution of the percentage of CI incidents by sector in time [64–68]

as it is not necessary to adapt or recreate a model.

Alternatively, since the birth of distributed computing frameworks such as MapReduce [37] and distributed file-systems such as the Hadoop File System (HDFS) [12], a new computing paradigm known as Big Data Analytics (BDA) has emerged. Big Data refers to the set of information that is too complex to process by traditional IT mechanisms within an acceptable scope [33]. Although no consensus exists, this data complexity is generally expressed in at least three qualities: the amount of data (volume), data generation and transmission pace (velocity) and diversity of data, both structured and unstructured (variety) [91]. More recently, a fourth quality is also widely mentioned: the ability to search for valuable information on Big Data (veracity) [33]. However, the term Big Data has transcended the type of information and it is also used to refer to set of methodologies and mechanisms developed to work with this type of data. BDA aims to extract valuable knowledge from Big Data by analyzing or modeling it in a scalable manner.

Among the multiple applications BDA has, Cárdenas et al. [26] and Everett [47], discussed its potential for intrusion detection, as the inherently created data in complex networks (e.g. logs, packets, flows) can be considered Big Data. Therefore, finding anomalous trends in these large datasets is not feasible with traditional mechanisms. They conclude that using BDA can lead to more efficient IDSs. However, both works center on regular, Information Technology (IT) networks, and do not examine its applicability to INs. On a related note, ICSs inherently create large amounts of heterogeneous data: process readings, where temporal values of physical properties are kept (temperatures, flows, pressures) and network data (packets, flows and logs). While the utility of BDA for analyzing large network datasets for intrusion detection has already been mentioned, it is worth noting that several proposals also exist for the analysis of process data. Specifically, BDA is considered an opportunity for improved process analysis and operation [80, 116, 125, 151, 165].

Consequently, it seems natural to link both worlds, and leverage BDA in INs to detect anomalies in these heterogeneous environments where data has very different nature. This thesis aims to fill that gap by applying BDA for anomaly detection in INs.

1.2 Research Objectives, Hypotheses and Contributions

The main objective of this thesis is to *develop data-driven Anomaly Detection Systems for Industrial Networks* that will allow the processing of large-scale, heterogeneous data to detect security events without the need of a process model. For this purpose, we focus in two different approaches: a visual monitoring approach and a holistic approach that leverages the different types of data that is created in INs, such as process data and network data, instead of just focusing on one aspect of the environment.

1.2.1 Hypotheses

The hypotheses that we will try to demonstrate are listed as follows:

- Due to the static, repetitive nature of IN traffic flows, security visualizations can aid IN operators to detect flow-related anomalies. Chord diagrams are a suitable candidate for this purpose.

- By using Multivariate Statistical Process Control (MSPC), we are able to detect anomalies and to diagnose them in ICSs, allowing the distinction between intrusions and process disturbances in a process-independent manner.

- Extending traditional MSPC models to include network and field data allows us to create an ADS that is able to monitor all data created in an IN in a scalable manner.

1.2.2 Contributions

The main contributions of this work can be summarized in the following items:

- A comprehensive literature review in the field of large-scale and heterogeneous ADSs, focusing on their applicability to INs and highlighting some open research questions that can lead to further research in the field [73].

- A IN testbed designed for the security research of INs and ICSs. This testbed is built around the software Emulab and the simulation of physical processes to ensure a rigorous environment for IN research. It supports Sofware-Defined Networking (SDN) and has an integrated data analysis module, enabling their use when conducting future research that employs these technologies [76].

- A visual monitoring system for INs representing network flows and anomalies related to them. It is built over chord diagrams that depict flows as legitimate and anomalous based on a previously created set of whitelists that describe network flows in different time frames [74, 75].

- A study on the feasibility of detecting and diagnosing intrusions in ICSs using MSPC. We conclude that MSPC is able to detect anomalies in ICSs effectively, but when diagnosing their cause it is necessary to audit the real process data. If the genuine process status is concealed from the operator, additional data (such as a network capture) is necessary to provide correct anomaly diagnosis [71, 72].

- A large-scale, heterogenenous ADS for INs that monitors field and network data to detect anomalies. For this end, we extend traditional MSPC models to include network data along with field data. The implementation is done on top of Apache Spark to ensure the scalability of the proposed system. With the presented approach, the ADS is able to detect and diagnose anomalies occurring at the field, network, or both levels.

- A Big Data tool set designed to analyze and process large multivariate datasets by using Principal Component Analysis (PCA). The components of the tool set have been designed and developed for building the previously mentioned large-scale ADS, but they can be used for other purposes as well, such as the exploration and analysis of large multivariate datasets.

Publications

Parts of the works covered in this dissertation have already been published or have been sent for review in different peer-reviewed journals and international and national conferences. We now list the scientific publications that are directly related to the work in this thesis:

Journal papers

- Mikel Iturbe, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. Towards Large-Scale, Heterogeneous Anomaly Detection Systems for Industrial Networks. *Submitted for publication to the IEEE Transactions on Industrial Informatics*, 2017.

Conference papers

- Mikel Iturbe, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. Visualizing Network Flows and Related Anomalies in Industrial Networks using Chord Diagrams and Whitelisting. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016)*, volume 2, pages 99–106, Rome, Italy, Feb. 2016.
- Mikel Iturbe, José Camacho, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. On the Feasibility of Distinguishing Between Process Disturbances and Intrusions in Process Control Systems Using Multivariate Statistical Process Control. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 155–160, Toulouse, France, Jun. 2016. IEEE.
- Mikel Iturbe, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. Sistema visual de monitorización de seguridad de flujos de red industriales. In *Proceedings of I Jornadas Nacionales de Investigación en Ciberseguridad (JNIC 2015)*, pages 59–65, León, Spain, Sep. 2015. Universidad de León.
- Mikel Iturbe, Unai Izagirre, Iñaki Garitano, Ignacio Arenaza-Nuño, Urko Zurutuza, and Roberto Uribeetxeberria. Diseño de un banco de pruebas híbrido para la investigación de seguridad y resiliencia en redes industriales. In *Proceedings of II Jornadas Nacionales de Investigación en Ciberseguridad (JNIC 2016)*, pages 3–10, Granada, Spain, Jun. 2016. Universidad de Granada.

- Mikel Iturbe, José Camacho, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. Distinguiendo entre perturbaciones de proceso e intrusiones en sistemas de control: caso de estudio con el proceso Tennessee-Eastman. In *Proceedings of the XIV Spanish Meeting on Cryptology and Information Security (RECSI 2016)*, pages 117–122, Maó, Spain, Oct. 2016. Universitat de les Illes Balears.

1.3 Methodology

The research methodology employed in this thesis is *Design and Creation*. Oates [115] defines the design and creation strategy as "the one that focuses on developing new IT products, also called *artifacts*". According to March and Smith [102, 115], these artifacts can be categorized in four main output types:

Constructs Relate to concepts, such as the notion of entities, objects or data flows.

Models The representation of a situation that is composed of constructs that are used to aid problem understanding and solution development. For instance, flow diagrams or use case scenarios.

Methods Guidance on the models to be produced and process stages to be followed to solve problems. The main examples are algorithms.

Instantiations A working system that demonstrates that constructs, models, methods, ideas, genres or theories can be implemented in a computer-based system.

Sometimes the research outcome can be a combination of these four possible outputs.

Design and creation methodology can be mistaken for normal software development process. However, this research work goes one step further than just creating a viable IT product. It is a research work with analysis explanation, argument, justification and critical evaluation of the results. As such, it aims to create new knowledge and disseminate its results through academic publications by means of building of an artifact.

Vaishnavi and Kuechler [145] describe the design and creation research process by dividing it into five iterative steps, shown in Figure 1.4.

Awareness The recognition and articulation of a problem that identifies a particular area for further research. The output of this phase is a proposal, formal or informal.

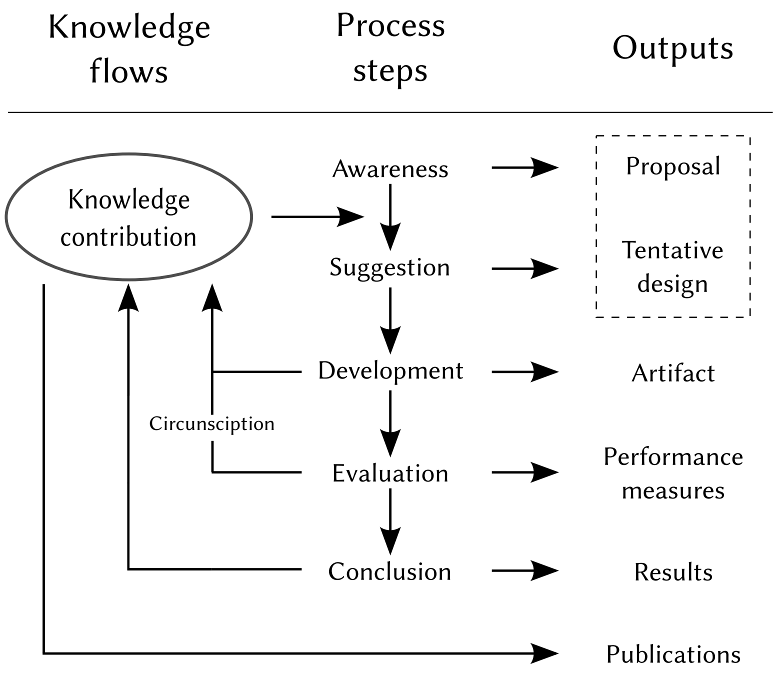


Figure 1.4: Design and Creation process used in this thesis along with its different outputs [145]

Suggestion Suggestion is an essentially creative step wherein new functionality is envisioned based on a novel configuration of either existing or new and existing elements. This new functionality aims to solve the problem identified in the Awareness phase. A tentative design is built on this phase, where the proposal is expanded and further details are given about the artifact (possible model, prototype etc.).

Development In this phase, the tentative design is further developed and implemented into an artifact. As expected, the nature of this implementation might vary depending on the artifact (e.g. piece of software, or an algorithm).

Evaluation The implementation of the artifact is examined according to the criteria defined in the proposal, implicitly or explicitly.

Conclusion This phase represents the end of the research cycle, where the results from the design and creation process meet the desired functionalities defined in previous phases. The results are consolidated and this is the phase where the acquired new knowledge is detailed and disseminated.

As it is shown in Figure 1.4, the main process is iterative, as the continuous feedback on the artifact helps to re-shape it all over again to improve results. However, there is one more task that is not related to the iterative process: the dissemination of results through scientific publications. Our approach to perform this duty has been continuous, that is, we have sent manuscripts as soon as they were elaborated enough to be published, so the entire community could benefit of the advances. The final results, however, along with the details of the research process, are written in this dissertation.

1.4 Document Overview

This dissertation is divided into seven chapters. In this section, we provide a short summary of the contents of each of the chapters.

Chapter 1 is this introduction.

Chapter 2 reviews the field of large-scale, heterogeneous ADSs, emphasizing their applicability to INs and it also identifies some open research questions that pose opportunities for further developments in the field.

Chapter 3 presents the design of a novel IN security testbed. This testbed uses Emulab and process simulation along with SDN and data analysis to provide a faithful and reproducible environment for IN security research.

Chapter 4 describes a visual monitoring system for IN flows. Based on chord diagrams and whitelists, the system shows the status of network flows, visually highlighting the anomalous ones (i.e. the ones not included in the whitelist).

Chapter 5 introduces the concept of MSPC and studies its applicability to anomaly detection and diagnosis in ICSs. It shows how MSPC is a valid method to detect anomalies in this context, also mentioning its limitations when diagnosing their cause.

Chapter 6 discusses a large-scale extension to MSPC for anomaly detection in INs. The MSPC model is enhanced to include both process and network-level data in a scalable manner. Results show that the proposed system is suitable for anomaly detection and it is scalable both in terms of monitored variables and the size of the datasets.

Chapter 7 concludes the thesis by summarizing the main findings and contributions of this dissertation. Moreover, it provides some possible lines for further work.

Additionally, the thesis contains an appendix at the end, with a summary in Basque language highlighting the context and the main contributions of the work.

Chapter 2

Literature Review

Contents

This chapter introduces the research field of anomaly detection for Industrial Networks (INs). First, it introduces the required background to provide the necessary concepts to support further analysis. Then, it studies the relevant proposals in the field of large-scale and heterogeneous anomaly detection and analyzes their applicability to INs. The chapter ends by identifying some open research areas that can lead to further development.

2.1 Technical Background

In this section we provide the necessary background to support our argumentation and to improve the understanding of the analyzed works.

2.1.1 Industrial Networks

Since the invention of the Programmable Logic Controller (PLC) in the 1960s, INs have evolved significantly from the initial primitive, proprietary and isolated environments to the complex, standard, interconnected networks that are today. Traditionally, INs were isolated environments where communication was conducted through proprietary network protocols with limited or non-existent interaction with external networks.

As mentioned in Section 1.1, the IN and Information Technology (IT) networks became more interconnected since the 1990s. On the one hand, this increased network standardization led to the start of using standard network protocols (TCP/IP) and Commercial off-the-shelf (COTS) software, laying behind proprietary, ad-hoc hardware and software solutions [132]. On the other hand, this merge significantly increased the attack surface of INs, as it exposed them to simple remote attacks and exploitation by using known vulnerabilities of COTS software. Traditional isolation and obscure characteristics that INs had relied on for security did no longer exist.

Figure 2.1 shows the network architecture of a simple IN. INs have a vertical architecture. At the bottom lays the physical process that is being controlled. The physical process has a set of sensors and actuators that are used to gather information about the state of the process and to perform actions on it. These sensors and actuators are connected to field controllers, normally PLCs, through buses or direct connections in the so-called field network. Field controllers are the workhorse of INs. They read process data from the field sensors and, based on their stored control algorithm, send orders to the actuators to interact with the process, generally trying to keep process variables' values around a set of certain setpoints. Nevertheless, except for the simplest installations, field controllers are not enough to conduct all the required tasks. Consequently, additional devices, called supervisory devices, are necessary. These devices usually run on normal IT-based hardware and software. Examples include control servers, Human Machine Interfaces (HMIs) and engineering stations. Control servers store process data and, optionally, implement second level control logic, usually involving data from different field controllers. HMIs are the graphical user interfaces operators use to interact with the process. Critical processes are monitored 24/7 by human operators. Process engineers use engineering stations to develop and test new applications regarding control logic.

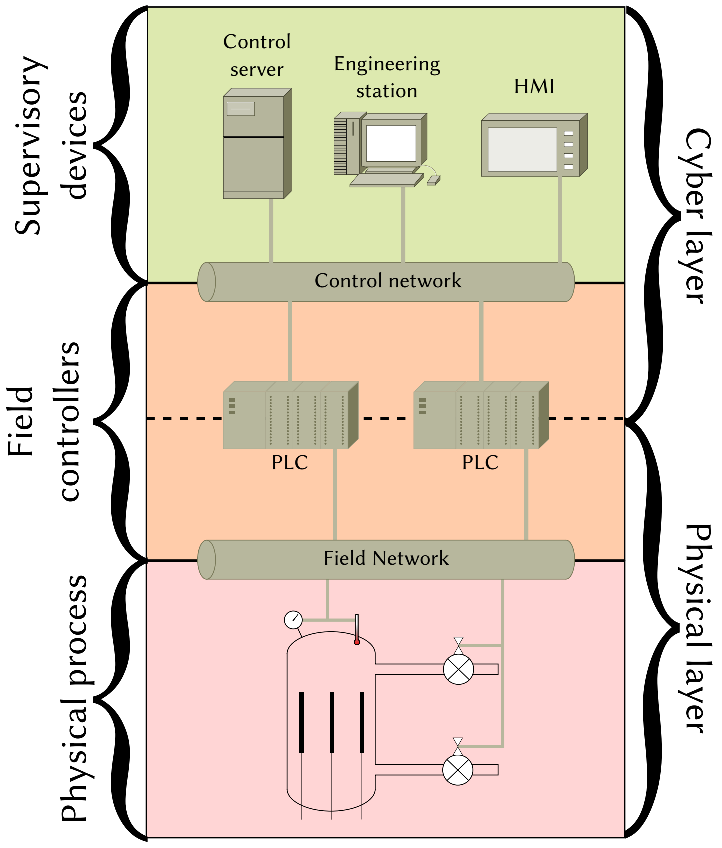


Figure 2.1: Example of a simple industrial network

INs can be further divided according to different layers. According to the definition by Genge et al. [52], on the one hand, there is the physical layer, composed of the actuators and sensors that directly interact with the physical process. On the other hand, there is the cyber layer, composed of all the IT devices and software which acquire the data, elaborate low level process strategies and deliver the commands to the physical layer. Field controllers act as the bridge between both layers, as they read field data and send local commands to the actuators, but they also forward field information to the cyber layer components while executing commands they receive from the supervisory devices.

Hence, ICSs can be considered a type of Cyber Physical Systems, as they are able to process and communicate data while also interacting with their physical environment.

There are different types of INs, such as Supervisory Control and Data Acquisition (SCADA), Distributed Control Systems (DCSs) and Process Control Systems (PCS). However, differences are getting blurred, and they can often be considered as a single entity when designing security solutions [50, 132].

Although they share a common part of technology stack, INs are inherently different to commercial IT networks. Table 2.1 shows a summary of the main differences between both network types. The main difference resides in the purpose of each of the networks: whereas in IT, the purpose is the transfer and processing of data, in the case of INs the main objective is to control a physical process. These differences mean that even when technically possible, blindly applying IT-based security mechanisms or procedures in industrial environments might lead to process malfunction or potentially safety-threatening scenarios. For instance, running anti-virus software on PLCs might compromise the PLC's ability to perform real-time operations on a process, or, conducting a penetration test can lead to dangerous scenarios [44].

However, these traits can also be leveraged to build security mechanisms for INs, that would be impractical to use in IT networks. For instance, the deterministic nature of INs and its periodic traffic between different hosts makes them suitable candidates for using Anomaly Detection Systems [108].

2.1.2 Anomaly Detection Systems

Anomaly Detection Systems (ADSs) are a subset of Intrusion Detection Systems (IDSs) [38]. IDSs are security mechanisms that monitor network and/or system activities to detect suspicious events. IDSs are classified according to two main criteria: the detection mechanism they use (signature detection or anomaly detection), and their source of information (where they collect the events to analyze).

Signature-based IDSs compare monitored data to a database with known malicious patterns (signature database). If there is a match, an alert is raised, as the activity has been identified as suspicious. Their efficiency is directly related to the completeness and accuracy of the signature database they are working with, as attacks will go undetected if their

	Industrial networks	**IT networks**
Primary function	Control of physical equipment	Data processing and transfer
Applicable Domain	Manufacturing, processing and utility distribution	Corporate and home environments
Hierarchy	Deep, functionally separated hierarchies with many protocols and physical standards	Shallow, integrated hierarchies with uniform protocol and physical standard utilisation
Failure Severity	High	Low
Reliability Required	High	Moderate
Round Trip Times	250 μs–10 ms	50+ ms
Determinism	High	Low
Data Composition	Small packets of periodic and aperiodic traffic	Large, aperiodic packets
Temporal consistency	Required	Not Required
Operating environment	Hostile conditions, often featuring high levels of dust, heat and vibration	Clean environments, often specifically intended for sensitive equipment
System lifetime	Some tens of years	Some years
Average node complexity	Low (simple devices, sensors, actuators)	High (large servers/file systems/databases)

Table 2.1: Differences between Industrial and IT networks [32, 50]

signature is not available. Among their operational characteristics, they have a low number of false positives but they are unable to detect unknown attacks. ADSs, on the other hand, identify malicious patterns by measuring their deviation from normal activity. ADSs build a model of the normal behavior of the process (through automated learning or manual specifications) and detect deviations with respect to the model [32]. Many ADSs are built using machine learning methods [15]. As opposed to signature-based IDSs, ADSs are able to detect unknown attacks, but they often yield a higher number of false positives.

Regarding the source of information, IDSs traditionally have been classified into two main categories: network-level and host-level IDSs. Network-based IDSs monitor network traffic to detect suspicious activity (suspicious connections, malicious packet payloads...), while host-based IDSs monitor local data stored in a device (system logs, file integrity...). In the case of INs, the limited processing ability of industrial devices has limited the deployment of host-based ICSs [32]. Therefore, when considering IN IDSs, the source of information criterion can be set based on the IN layer they use to gather information from: the cyber level or the physical layer. Cyber-level IDSs are similar to their IT counterparts as they generally monitor network-level data. Physical-level IDSs monitor the physical quantities of the process (pressures, temperatures, currents...) in order to detect intrusions. Physical properties of the process are constantly monitored, often polling data every few milliseconds in the case of critical variables, which with large, continuous processes can lead to a scenario where it is necessary to use Big Data Analytics (BDA) in order to process field and control data. This is further confirmed by proposals that, outside the field of security research, point to this need and propose process monitoring solutions based on BDA [80, 116, 125, 131, 151, 165].

Most IN ADSs work on the cyber layer (see surveys [51, 108, 163]). Physical-level ADSs can be divided into two main groups: ADSs where it is necessary to model the physical process [88, 134] or ADSs that do not need a specific model for the physical process [72, 82]. Few proposals combine data from both levels [53, 79].

2.1.3 Big Data Security Mechanisms

Modern and complex IT networks create and process vast amounts of data continuously. Analysis of the created data for security purposes is a daunting task, and before the advent of Big Data processing tools, data was normally sampled or only subsets of it was analyzed (e.g. only metadata). Since MapReduce [37] was introduced, several Big Data frameworks have been proposed, which allow the processing of large, heterogeneous datasets.

Traditionally, Big Data frameworks have been divided into two main groups, according to the nature of the data they work with. On the one hand, there are batch processing technologies, that work with data at rest and are usually used when doing Exploratory Data Analysis (EDA). Examples of technologies that use this approach would include Hadoop [9], Disco [113] and Thrill [10]. On the other hand, there are stream processing technologies, that are designed to work with flowing data. Gorawski et al. [56] reviewed different Big Data streaming proposals.

However, hybrid tools such as Apache Spark [156] or Apache Flink [24] are able to work both on streaming and resting data. Spark uses micro-batches to process incoming data while Flink does batch processing as a special case of stream processing.

Extracting insight from the large amount of information that could be leveraged for security event detection (e.g. logs, network flows or packets) in a network can be considered a Big Data problem [26, 47]. Consequently, different types of Big Data security mechanisms have been proposed:

- Intrusion detection (see survey [167])
- Botnet detection ([35, 49, 100, 128])
- Malware detection ([31, 70, 99, 118]) and analysis ([60, 78, 161])
- Distributed Denial of Service (DDoS) detection ([36, 98, 109, 144, 162])
- Spam detection ([27, 28, 94])

On a related note, other resources have been developed that even if they are not security mechanisms *per se*, they have been designed to handle large volumes of network data, and thus, can be useful to build security mechanisms on top of them:

- Frameworks for analyzing network flows ([96, 141])
- Frameworks for analyzing network packets ([6, 97])
- Frameworks for analyzing logs ([90, 153])

However, in this chapter we will limit the scope of the literature review to Big Data ADSs that could potentially be applicable to the industrial domain.

2.2 Taxonomy

In this section, we describe the taxonomy or classification method that will be used in Section 2.3 for existing large-scale industrial ADSs. Figure 2.2 shows the created taxonomy tree. When classifying IDSs in general, two main criteria are used: the detection method and the scope of the IDS [5, 32, 38, 108]. We can apply these criteria to build an IN ADS classification method.

2.2.1 Detection Method

The main criterion to classify IDSs resides on the detection method. While the difference between signature-based IDSs and ADSs was already covered in Section 2.1.2, ADSs can be further classified based on their detection technique. According to Axelsson [5] and Mitchell and Chen [108] ADS detection techniques belong in two categories:

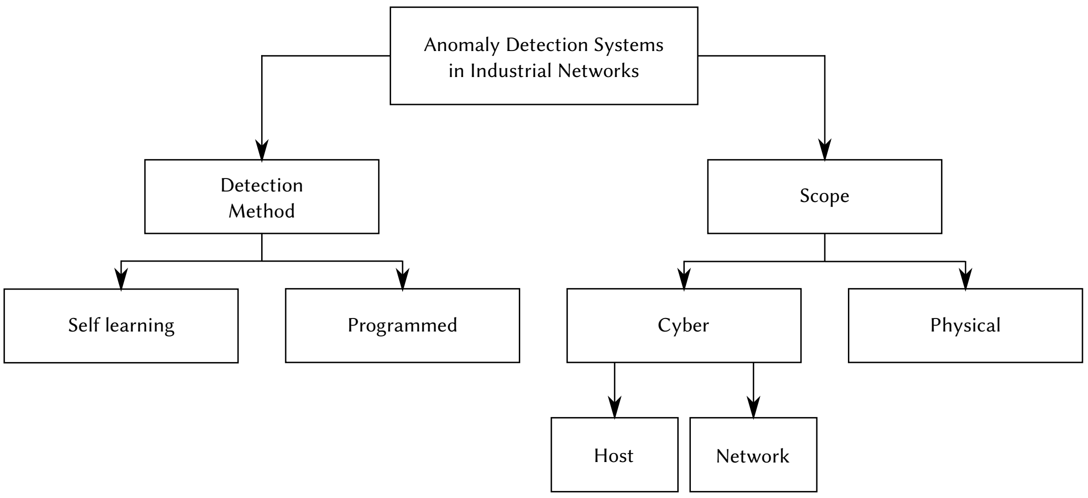


Figure 2.2: A taxonomy for Anomaly Detection Systems in Industrial Networks

1. *Self Learning or Behavior-Based ADSs.* The ADS detects anomalous features that are distinct from normal system behavior. Normal system behavior can be retrieved in a unsupervised (e.g. clustering historical data) or in a semi-supervised manner (e.g. collection of training, generally attack-free, data).

2. *Programmed or Behavior-Specification-Based ADSs.* Using expert knowledge, a human defines legitimate behaviors and implements them on the ADS. The ADS detects anomalies by detecting deviations from the specified behavior.

2.2.2 Scope

Apart from the detection method, the other main criterion for IDSand ADSs is their scope, that is, the source and nature of the data used for audit. In IT ADSs, there are two main types of ADSs depending on the data they use.

1. *Network ADSs.* ADSs monitor a network without focusing on individual hosts. The most prominent data sources for these ADSs are network flows and packets.

2. *Host ADSs.* The ADS monitors data from an individual host to check anomalies. Examples of host data include logs, files or system calls.

While this split was conceived for IT-based ADSs, this classification has also held for IN ADSs [32, 108, 164]. And indeed, most IN ADS proposals can be classified in one of the two above categories. Nevertheless, due to the cyber-physical nature of INs, this classification is not complete enough, as it only tackles the cyber part of INs, while not considering the physical dimension of INs that handles field data. Field data mainly consists of sensor signals that monitor physical quantities (temperature, pressure...) although other process-based variables (counters, setpoint values...) might be present. There are several examples of IN ADSs that leverage field-level data. [72, 82, 88, 134]. This data can come from logs on a control server, direct process measurements, simulated data, or can be scattered across different hosts or devices. Therefore, ADS proposals that leverage process data for anomaly detection do not fit well in the above classification. Consequently, we have created a novel taxonomy where the physical dimension of IN ADSs is taken into account as a proper data source. This taxonomy can be leveraged to classify IN ADSs, both conventional and Big Data proposals, as it encompasses more data sources and types that are present in INs, than previous presented taxonomies that do not acknowledge the existence of ADSs based on the physical layer of INs.

2.3 Anomaly Detection Systems

In this section, we survey existing Big Data ADSs that could be used in INs. Proposals are divided according to the taxonomy described in Section 2.2.

2.3.1 Cyber-level ADSs

Cyber-level, Self Learning ADSs

The proposal of Xu et al. [152] is an ADS based on host log mining. System logs are first parsed to provide a unified data-structure from different log formats, by getting log format templates from the application source code. Then, they build features from the extracted log data, focusing in state ratio vector (a vector representing a set of state variables on a time window) and the message count vector (a vector representing a set of related logs with different message types) features. These vector features are later mined using an algorithm based on Principal Component Analysis (PCA) for anomaly detection. The results are finally visualized in a decision tree to aid operators to find the root cause of an anomaly. The analysis is carried out in a Hadoop cluster to increase computing speed.

Yen et al. [154] introduce Beehive, a large scale log mining tool that uses Hive [142] to detect suspicious activities in corporate networks. For that purpose, Beehive mines logs coming from different sources and, especially, web proxy logs. Beehive clusters log data and identifies misbehaving hosts as cluster outliers, as they show a unique behavioral pattern. The clustering is done by an adapted version of the k-means algorithm. The incidents related to the outliers were labeled manually by using other system logs and showed that many of these outliers where not detected by traditional security mechanisms.

Ratner and Kelly [121] conduct a case study of network traffic anomalies in a corporate network. For this end, they extract packet metadata from a set of captured packets, and they perform specific queries in the gathered data to detect attacks, mainly IP scans. In order to process the large dataset, they use Apache Hadoop. They find a large number of IP scans and conclude that roughly half of the packets arriving from external IP addresses are anomalous. Those anomalies were found by comparing each packet's IP metadata to the average values for each day.

Therdphapiyanak and Piromsopa [140] expose an anomaly detection system based on host log analysis. First, the system parses log data and later clusters it by using k-means. Once the clusters are formed, the authors extract major characteristics from the clusters to examine differences and similarities. Minor clusters with important differences when compared to others are flagged as anomalous. While the system has been tested with Apache Web Server logs, the authors address aggregating logs from different network agents in future steps. Log parsing and clustering is performed in a Hadoop cluster.

Camacho et al. [19] use a PCA-based solution to detect anomalies in computer networks. The workflow of the approach can be seen on Figure 2.3. The anomaly detection is accomplished in two separate phases: a model building phase, where the ADS is tuned based on training data, and a monitoring phase, where the ADS analyzes incoming data and determines it as anomalous or legitimate based on the model built during the previous phase. In the first phase, incoming data (generally, IDS and Firewall logs) is pre-processed and converted into feature vectors. Later, this data is used to create a PCA model, where the original features are transformed into a new variable subspace. This dimensionality reduction helps to discard anomalies that made into the training data, by filtering outliers. The PCA model can also be used to create two different statistics that are widely used for for process monitoring: Hotelling's T^2 [62], comprising the leverages of the PCA model and the SPE [77], involving the residuals of the model. The proposed approach calculates the statistics for each of the observations in the training set and based on it, calculates a control limit based on an arbitrary confidence level, where a given percentage of the training observations should be below the control limit. Once the control limits have been set, the training phase has ended

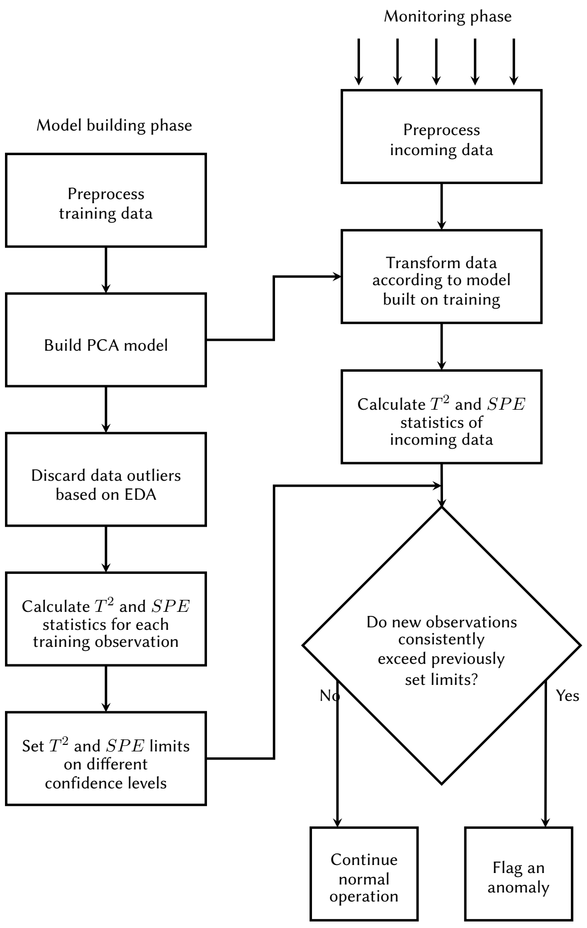

Monitoring phase
Model building phase
Preprocess training data
Build PCA model
Discard data outliers based on EDA
Calculate T^2 and SPE statistics for each training observation
Set T^2 and SPE limits on different confidence levels
Preprocess incoming data
Transform data according to model built on training
Calculate T^2 and SPE statistics of incoming data
Do new observations consistently exceed previously set limits?
No
Yes
Continue normal operation
Flag an anomaly

Figure 2.3: Anomaly Detection System proposed by Camacho et al. [19], based on Principal Component Analysis (PCA)

and the ADS is now prepared to work in the monitoring phase. In this phase, incoming data is pre-processed and transformed using the previously created PCA model and it calculates the T^2 and SPE values for each incoming observation. If several consecutive observations surpass either of the set control limits (the necessary number of out-of-bounds observations depends on the confidence level), an anomaly is flagged. The process can be parallelized using hierarchical PCA and the workload shared through several slaves. The ability of PCA to work with high dimensional data ensures that the approach can be extended to a wide range of incoming data.

Hashdoop [48] is a MapReduce-based framework designed to run anomaly detection systems in a distributed manner. It does not provide enhanced anomaly detection capacity to the original ADS but it speeds up its execution. First, it splits and hashes network traffic, preserving traffic structures. Later, each of the hashed traffic subsets is analyzed by an instance of the ADS to detect anomalies. Finally, the generated information about the subsets is summarized in a single output report. Results show that processing time is reduced when using Hashdoop-powered ADS compared to their single-node counterparts.

Marchal et al. [103] propose an intrusion detection system that uses honeypot data to detect similar intrusions in networks. First, it collects Domain Name System (DNS) replies, HTTP packets and IP flow records from the network, along with honeypot data. Based on the collected data, three different scores are computed in order to quantify the maliciousness of the recorded DNS, HTTP and flow communications. This quantification uses other gathered or publicly available data such as domain blacklists or the data compiled by the in-house honeypot. When one of these maliciousness indices reach a certain threshold, a flag is raised to inform about the anomaly. The authors test different data-intensive frameworks that are designed to work with potentially very large data volumes. According to their tests, Apache Spark and its subproject, Shark, are faster than Hadoop, Hive or Pig. However, several concerns arise with this mechanism: the performance of the proposed system is directly related to the performance of the honeypot. If an attacker does not interact with the honeypot and their domain is not explicitly blacklisted, the mechanism will not be able to raise an alert, even in the case of known attacks.

MATATABI [137] is a threat analysis platform that stores data from different sources (DNS captures and querylog, Network flows and spam email) in a Hadoop cluster and organizes it in Hive [142] tables. Later, different modules query this data via a Javascript Object Notation (JSON) Application Programming Interface (API). Although the exact implementation details of each of the analysis modules are vague, each module queries the stored data looking for anomalous patterns, such as hosts receiving or sending a large number of packets, specific port scans by counting the number of packets to a specific port number, or botnet activity through abnormal DNS activity. While the gathered data is varied, the modules are designed to query a single type of data. If suspicious activity is detected, it is in the operator's hand to query other types of data to find additional evidence of the attack.

TADOOP [143] is a network flow ADS that implements an extension of the Tsallis Entropy [166] for anomaly detection, dubbed DTE-FP (Dual q Tsallis Entropy for flow Feature with Properties). In short, TADOOP gathers network flows and computes a pair of q values aiming to accentuate high and low probability feature distributions, usually linked to traffic anomalies. TADOOP is based on four main modules (i) The *Traffic Collector* gathers network flow packets and decodes them. (ii) The *Entropy Calculation Module* extracts flow features from each flow and it computes the DTE-FP q values for each flow feature distribution. (iii) The *Semi-Automatic Training Module* is the responsible for setting optimal q pair detection thresholds for each of the distribution. The criterion is keeping false positive rate below an arbitrary maximal threshold. (iv) The *Detection module* calculates entropy values for all the flows in a given time window and compares them to the thresholds computed by the training module to detect anomalies. TADOOP uses Hadoop for storing and processing historical flow data. TADOOP is evaluated using the flow data of a university network.

Gonçalves et al. [55] present an approach for detecting misbehaving hosts by mining server log data. In the first phase, they extract features from DHCP, authentication and firewall logs, and for each host a feature vector is created. These vectors are later clustered using the Expectation-Maximization (EM) algorithm which are later used to build a classification model. Smaller clusters in the set correspond to anomalous host behavior. In the second phase, once the classification model is built, incoming data is clustered in a similar way as in the first phase, however, these newly created clusters are classified with the previously created model in order to detect if they are anomalous. While the feature extraction from the log data is done in Hadoop, clustering and classification of the data is carried out with the Weka [59] data mining tool.

Dromard et al. [43] extend the UNADA [29] ADS to detect anomalies in Big Data network environments. UNADA is a three-step unsupervised ADS. (i) *Flow Change Detection.* Flows gathered in a given time window are aggregated on different levels defined by network masks. For each level, UNADA computes a simple metric or feature of the aggregated flows: number of bytes, number of packets, number of IP flows...Then, when a new set of flows is gathered, these metrics are recomputed for the new flow and compared to the previous set. If there is a change in the values, the time window is flagged and further computed. (ii) *Clustering.* In this phase, UNADA clusters the feature vectors from the previously flagged flow sets using DBSCAN [45]. Network flow feature vectors can have numerous variables and DBSCAN does not perform well in multivariate environments. In order to overcome this issue, UNADA splits the feature space into smaller, two-dimensional subspaces and computes DBSCAN independently on each of them. (iii) *Evidence accumulation.* In the last phase, data from each of the subspaces is aggregated to identify anomalies. In each subspace, independently, data points that do not belong to a cluster are flagged as anomalous and UNADA records the distance to the nearest cluster centroid. A dissimilarity vector is built with the accumulated abnormality scores for each flows across all subspaces. To ease anomaly detection, dissimilarity vectors are later sorted and a threshold is defined to finally flag flows as anomalous. The authors evaluate the performance of UNADA over Apache Spark [156] to compute the ADS over the network data gathered on a core network of an Internet Service Provider. Results show that the approach is able to detect flow anomalies while speeding up execution time in regards to the original UNADA proposal.

The proposal of Rathore et al. [120] is a flow ADS built on four layers. (i) *Traffic capturing*. The traffic is captured from the network and forwarded to the next layer. (ii) *Filtration and load balancing*. This layer checks whether the flow has been previously registered as a legitimate or anomalous in a database. If it has not, data is forwarded to the next layer. (iii) *Hadoop layer*. This layer extracts the features from the gathered data. It uses Apache Spark [156] on top of Hadoop for faster computation. (iv) *Decision Server*. The extracted features are classified as legitimate or anomalous sets by a set of classifiers implemented in Weka. The authors use the well-known intrusion detection NSL-KDD dataset for result evaluation and conclude that the C4.5 and REPTree are the best performing classifiers for this task.

Wang et al. [149] propose a continuous, real-time flow ADS based on Apache Storm. In order to reach this objective, they combine three different detection methods: (i) Network flows. They count the number of flows in a small enough time slot that allows online processing. After, they compute the standard deviation and mean of this count and calculate a confidence interval based on them. Later, they perform a set of operations over the flows involving hashing into groups and calculating Inter-group Flow Entropy [148]. In all steps, the system checks that the observations are inside the confidence interval, otherwise an alarm is raised. (ii) Intuitive Methods based on Traffic Volume. The system applies the same approach as in network flows but taking into account the number of packets in a time window instead the number of flows. (iii) Least-Mean-Square-based detection. The system uses a Least-Mean-Square-based (LMS) filtering method that aims to find inconsistencies between the inter-group flow and packet entropies, which should be strongly correlated. LMS also operates in an online manner. They evaluate they approach by replaying a capture of an Internet backbone while introducing in parallel two types of anomalies that where not present in the capture: An attack involving a large number of small network flows and an attack involving a small number of large flows.

Gupta and Kulariya [57] compare a set of feature extraction and classification algorithms for anomaly detection. They benchmark the different approaches using the popular intrusion detection KDD'99 and NSL-KDD datasets and the algorithms implemented in Spark's MLlib library. They evaluate correlation based feature selection and hypothesis based feature selection for feature extraction. For classification they measure the performance of Naïve Bayes, Logistic Regression, Support Vector Machines, Random Forests and Gradient Boosted Decision Trees. They conclude that hypothesis based feature selection helps to achieve a better classification score. Among the classifiers, Random Forests and Gradient Boosted Decision Trees yield better results than the rest.

Cyber-level, Programmed ADSs

The work presented by Giura and Wang [54] uses large-scale distributed computing to detect APTs. First, they model the APT using an Attack Pyramid, a multi-plane extension of an attack tree [2, 124] where the top of the pyramid represents the asset to be protected. The planes of the pyramid represent different environments where attack events can be recorded (e.g., user plane, application plane, physical plane...). The detection method groups all potential security events from different planes and maps the relevant events that are related to a specific attack context. This context information is later leveraged to detect a security incident if some indicators surpass a set of user-defined thresholds. The method uses MapReduce to consider all the possible events and related contexts.

Bumgardner and Marek's approach [16] consists in a hybrid network analysis system that uses both stream and batch processing, capable of detecting some network anomalies. First, it uses a set of probes that collect network traffic to build and send network flows to the specified processing unit. Then, the created flows are stream processed through Storm to enrich it with additional data (e.g. known state of the internal network) and anomalies are detected based on previously defined event detection rules (bot activity, network scans). Once the flows have been processed, they are stored in a HBase table, a column oriented database, to perform EDA to get further insight that it is not explicitly stated in each of the flows. This batch data processing is executed on top of Hadoop. The main drawback of Bumgardner and Marek's approach is that the system's anomaly detection capability is directly related to the capability of describing network events or anomalies using rules when doing stream processing.

2.3.2 Physical-level ADSs

Hadžiosmanović et al. [58] present a log mining approach to detect process-related threats from legitimate users' unintentional mistakes. They identify unusual events in log data to detect these threats. In order to extract the unusual events from the potentially large log data, they first use a FP-growth algorithm to count matching log entries. Later, unusual events are defined as the ones whose number of occurrences is below of a user-set, absolute threshold. FP-growth algorithms do not use candidate generation, and thus, are able to effectively count occurrences in two data scans.

Difallah et al. [40] propose a scalable ADS for Water Distribution Networks. Specifically, they use Local Indicators of Spatial Association (LISA) [4] as a metric for anomaly detection, by extending the metric to consider temporal associations. In the proposal, wireless sensors send process data to a set of base stations that perform part of the anomaly detection process by computing a limited set of LISA calculations on the streaming data they receive. Thus, it uses a distributed approach for a first phase of anomaly detection. Later, data is sent to a central Array Database Management System (ADBMS). The ADBMS allows global analytics of the distribution network as a whole. Evaluation of the proposal is done using Apache Storm for the stream processing in the base stations and SciDB [14] as the ADBMS, analyzing data from a simulated environment created after the Water Distribution Network of a medium-sized city.

Hurst et al. [63] introduce a Big Data classification system for anomaly detection on CIs. They extract process data from a simulated nuclear power plant and extract relevant features from it, by selecting a number of variables that best describe the overall system behavior. However, this feature extraction relies on expert knowledge to identify the subset of variables that are most suitable. Moreover, the needed features will vary between different types of processes, even different installations, and thus, the approach is process-dependent. They do not specify the used criteria for feature selection. After feature extraction, they perform anomaly detection using five different classifiers by splitting the gathered data into two halves for the training and testing. They demonstrate that increasing both dataset size and the number of features used for anomaly detection yields better classification results. They do not specify the framework they used for this large-scale classification.

Kiss et al.'s system [83] is designed to detect field-level anomalies in Industrial Networks. By leveraging the field data that sensors and actuators periodically send, they classify normal and abnormal operation cases. To this end, field parameters are used to build feature vectors that are later clustered using k-means to identify operation states and anomalous states of the physical process. In order to deal with the growing field data, the system uses Hadoop to create the different clusters. As the vectors to be clustered are built using field data, these feature vectors depend on process nature. Furthermore, in case of complex physical processes, building the features and identifying different operation states can be a challenging problem that can complicate the deployment of the proposal.

Wallace et al. [147] propose a Smart Grid ADS by mining Phasor Measurement Unit (PMU) data. The overview of their proposal is depicted in Figure 2.4. The system first models normal grid operation by measuring voltage deviation from each of the PMUs and creating a cumulative probability distribution to represent the likelihood for a signal to have a given voltage deviation. After the distribution function has been created, the likelihood of a given divergence of two voltage signals can be estimated. The system evaluates this calculated likelihood in order to classify an incoming observation as anomalous or legitimate. In detail, the system calculates the voltage deviation from two consecutive signals, and then, using the probability distribution function constructed with the historical data, establishes an event as anomalous if this deviation is unlikely to happen. That is, consecutive signals with high discrepancies in voltage values are more unlikely to arise, and therefore, when they happen they can be classified as anomalous situations in the grid. After an anomaly has been flagged, further analysis of the data can explain the nature of the anomaly. This anomaly identification is carried out by a classification decision tree algorithm, that infers the type of anomaly based on three hand-coded events, developed with expert knowledge. The evaluation is done using real PMU data of an electrical grid and using Apache Spark for data computation.

2.4 Discussion

In this section, we discuss the proposals presented in Section 2.3, pointing to the advantages and disadvantages of the proposals, stressing their applicability to INs.

Table 2.2 shows a comparison of the presented works, according to different criteria:

- Domain: Refers to the network type the proposal has been defined to work in: IT or IN.
- Granularity: Axelsson [5] defines granularity of data processing as a "category that contrasts systems that process data *continuously* with those that process data in *batches* at a regular interval."
- Time of detection: Axelsson [5] defines this category by defining the two main groups that compose it: systems that give results in *real-time* or near real-time and those that process data with some delay (*non-real).* Though related to the previous category, they

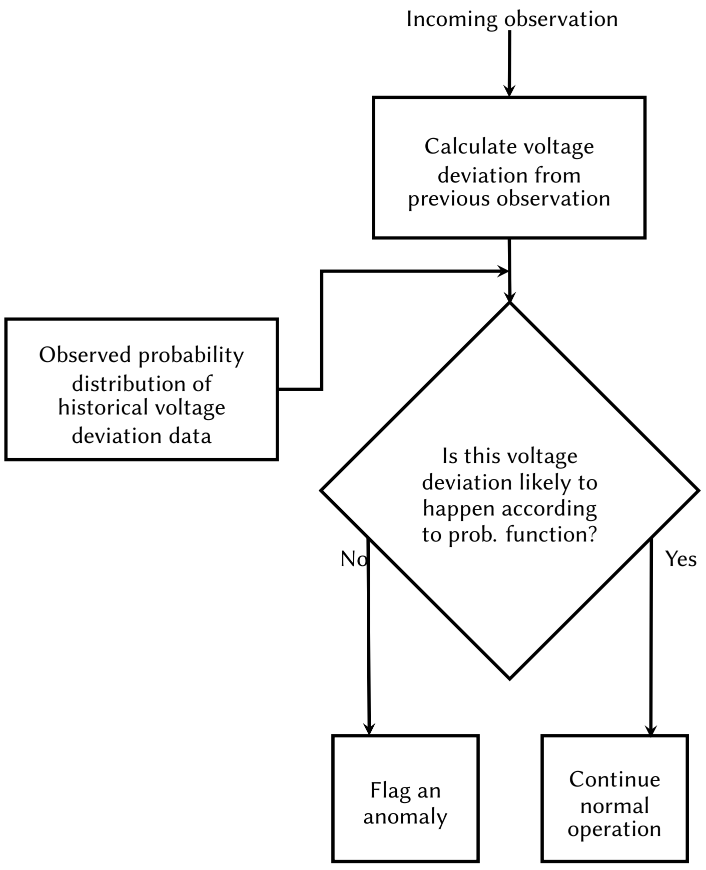

Incoming observation
Calculate voltage deviation from previous observation
Observed probability distribution of historical voltage deviation data
Is this voltage deviation likely to happen according to prob. function?
No
Yes
Flag an anomaly
Continue normal operation

Figure 2.4: Flowchart of the Smart Grid Anomaly Detection procedure proposed by Wallace et al. [147]

do not overlap, as some real-time systems might process micro batches, thus giving real-time or almost real-time performance.

- Source of information: Refers to the type of input data the ADS collects and audits for anomaly detection.
- Main detection technique: Refers to the main technique the ADS leverages to detect anomalies in the gathered information.

Table 2.2: Comparison of the surveyed works

Name	Ref.	Dom.	Granul.	Time of detect.	Sources	Main Detect. technique
Beehive	[154]	IT	Batch	Non-real	Proxy logs	k-means
Bumgardner & Marek	[16]	IT	Both	Real	Network flows	Threshold establishing
Camacho et al.	[19]	IT	Both	Non-real	Firewall & IDSlogs	PCA
Dromard et al.	[43]	IT	Batch	Non-real	Network flows	DBSCAN
Difallah et al.	[40]	IN	Both	Real	Process data	LISA
Giura and Wang	[54]	IT	Batch	Non-real	Network and aplication data	Threshold establishing
Gupta and Kulariya	[57]	IT	Batch	Non-real	Network captures	Several feature extraction and classification algs.
Gonçalves et al.	[55]	IT	Batch	Non-real	DHCP, Authentication and Firewall logs	EM
Hadžiosmanović et al.	[58]	IN	Batch	Non-real	SCADA logs	FP-Graph
Hashdoop	[48]	IT	Batch	Non-real	Network traffic (textual format)	None

Continued on next page

Table 2.2 – *Continued from previous page*

Name	**Ref.**	**Dom.**	**Granul.**	**Time of detect.**	**Sources**	**Main Detect. technique**
Hurst et al.	[63]	IN	Batch	Non-real	Process data	Multiple classification algs.
Kiss et al.	[83]	IN	Batch	Non-real	Process data	k-means
Marchal et al.	[103]	IT	Batch	Non-real	Honeypot, DNS, HTTP and Network flow data	Threshold establishing
MATATABI	[137]	IT	Batch	Non-real	DNS records, Network flows, Spam email	Multiple
Rathore et al.	[120]	IT	Batch	Non-real	Network flows	C4.5, RepTree
Ratner and Kelly	[121]	IT	Batch	Non-real	Network packets	Manual data querying
Therdphapiyanak & Piromsopa	[139]	IT	Batch	Non-real	Network logs	k-means
TADOOP	[143]	IT	Batch	Non-real	Network flows	DTE-FP
Wallace et al.	[147]	IN	Cont.	Real	Process data	Cumulative Probability Distribution
Wang et al.	[149]	IT	Cont.	Real	Network flows	Inter-group entropy, LMS

Continued on next page

Table 2.2 – *Continued from previous page*

Name	**Ref.**	**Dom.**	**Granul.**	**Time of detect.**	**Sources**	**Main Detect. technique**
Xu et al.	[152]	IT	Batch	Non-real	Console logs	PCA

As Table 2.2 shows, most of the proposals are both batch and in non-real-time. Moreover, a similar set of proposals use a single type of data input as the source for audit information. Thus, it can be stated that the majority of these proposals focus on handling large, resting data volumes for anomaly detection (one of the *V* Big Data dimensions) while the other dimensions (mainly velocity and variety) are not as relevant.

Table 2.3 shows the Big Data adoption level of the proposals, by listing the following metrics:

- Locus of Data Collection (LDC): Axelsson [5] notes that "audit data can be collected from many different sources in a distributed fashion, or from a single point using the centralised approach".
- Locus of Data Processing (LDP): Similarly, Axelsson states that "audit data can either be processed in a central location, or is collected and collated from many different sources in a distributed fashion ."
- Underlying solution: Lists the underlying Big Data technology the ADS uses for Big Data computing.
- Evaluation environment: Shows the nature of the evaluation data used to test the performance of the ADS.

Table 2.3 shows that most of the proposals use distributed computing for data processing. However, distributed data collection, where data from different sources is analyzed is not as widespread. Hadoop and Spark are the most prominent Big Data frameworks that are used for anomaly detection. It is worth mentioning that in some proposals, although using these mechanisms, they are only used in a part of the data pipeline. For instance, they use the Big Data tools for feature extraction, while once the features have been extracted into a smaller feature dataset, other conventional tools are used for the data classification.

Table 2.4 summarizes the suitability of the IT-based solutions to be used in INs. For that end, it defines the following metrics:

Name	**Ref.**	**LDC**	**LDP**	**Solution**	**Eval. environ.**
Beehive	[154]	Dist.	Dist.	Hadoop, Hive	Operational network
Bumgardner & Marek	[16]	Dist.	Dist.	Storm, HBase, Hadoop	Operational network
Camacho et al.	[19]	Dist.	Unknown	Custom	Public dataset
Dromard et al.	[43]	Dist.	Dist.	Spark	Operational network
Difallah et al.	[40]	Dist.	Dist.	Storm	Simulated process data
Giura and Wang	[54]	Dist.	Dist.	Hadoop	Operational network
Gupta and Kulariya	[57]	Cent.	Dist.	Spark	Public dataset
Gonçalves et al.	[55]	Dist.	Dist.	Hadoop, Weka	Operational network
Hadžiosmanović et al.	[58]	Cent.	Cent.	Custom	Operational network
Hashdoop	[48]	Cent.	Dist.	Hadoop	Public dataset
Hurst et al.	[63]	Cent.	Unknown	Custom	Simulated process data
Kiss et al.	[83]	Cent.	Dist.	Hadoop	Simulated process data
Marchal et al.	[103]	Dist.	Dist.	Hadoop, Hive, Pig, Spark	Operational network
MATATABI	[137]	Dist.	Dist.	Hive	Operational network
Rathore et al.	[120]	Cent.	Dist.	Spark, Weka	Public Dataset
Ratner and Kelly	[121]	Cent.	Dist.	Hadoop	Operational network
Therdphapiyanak & Piromsopa	[139]	Cent.	Dist.	Hadoop, Mahout	Public Dataset
TADOOP	[143]	Cent.	Dist.	Hadoop	Operational network
Wallace et al.	[147]	Dist.	Dist.	Spark	Operational network
Wang et al.	[149]	Dist.	Dist.	Storm	Operational network
Xu et al.	[152]	Cent.	Dist.	Hadoop	Operational network

Table 2.3: Big Data comparison of the surveyed works

- OSI Layer: Refers to the corresponding layer of the Open Systems Interconnection (OSI) model the network data belongs to. In the case of logs, it shows the layer of the network application that created the logs.

- IN interoperability: Refers to the performance of running the IT ADS, out-of-the-box in an industrial environment. Low interoperability means that the ADS would not be usable. Medium means that the ADS is expected to run on INs and to detect anomalies to some extent. High means that the ADS is also tailored to work in IN environments.

- Response type. Categorizes the ADSs in two categories, not related to the detection mechanism, but to their response when an anomaly is flagged. Passive responses consist of logging and sending alerts, without interacting with the traffic, while active responses try to tackle the source of the intrusion or anomaly. Active response mechanisms are often referred to as Intrusion Prevention Systems (IPSs). All the reviewed works have passive passive responses. The usage of active responses that fit well into the availability constraints of INs is still an undeveloped field [108].

- Self Security. Zhu and Sastry [164] define self-security to "whether the proposed ADS itself is secure in the sense it will fail-safe." Availability is an important concern in INs. As such, redundant and fail-safe mechanisms are widespread in INs.

Name	Ref.	OSI layer	IN interoperability	Self-security
Beehive	[154]	7	Low	Medium
Bumgardner & Marek	[16]	3,4	Medium	Medium
Camacho et al.	[19]	3,4,7	Medium	Unknown
Dromard et al.	[43]	3,4	Medium	Medium
Giura and Wang	[54]	3,4,7	Medium	Medium
Gupta and Kulariya	[57]	3,4,7	Medium	Medium
Gonçalves et al.	[55]	3,4,7	Low	Medium
Hashdoop	[48]	Packet captures	Dependent on implementation	Medium
Marchal et al.	[103]	3,4,7	Medium	Medium
MATATABI	[137]	3,4,7	Medium	Medium
Rathore et al.	[120]	3,4	Medium	Medium
Ratner and Kelly	[121]	Packet captures	Medium	Medium
Therdphapiyanak & Piromsopa	[139]	3,4,7	Medium	Medium
TADOOP	[143]	3,4	Medium	Medium
Wang et al.	[149]	3,4	Medium	Medium
Xu et al.	[152]	7	Medium	Medium

Table 2.4: Suitability of IT-based solutions for their use in INs

As Table 2.4 lists, most proposals, especially the ones that work with network flows, are able to work in INs, as nowadays IT networks and the cyber layer of IT networks share the same network stack at the OSI 3 and 4 layers (Network and Transport) and similar network infrastructure coexists in both types of networks (e.g. firewalls). However, even if technically possible, it is yet to be seen how they would perform.

It is worth mentioning that even if not listed in Table 2.4, the Time of Detection feature (covered in Table 2.2) becomes a relevant aspect of the ADSs when measuring their suitability for INs, as their real-time nature requires fast detection to raise alerts as fast as possible and to perform mitigation actions if necessary [108].

Furthermore, although Big Data ADSs listed were not designed for the availability constraints of INs, the usage of distributed file-systems for data storage and the distributed nature of Big Data processing, gives most solutions a relative defense against faults, as shown by the self-security field. It might not be enough for the high availability requirements on which ICSs and INs have, but still, it makes Big Data ADSs better candidates in this aspect than their conventional counterparts.

2.5 Open research areas

There are several open research lines in the area of Big Data ADSs for INs. We categorize them based on the different Big Data dimensions.

2.5.1 Dealing with volume

Most surveyed Big Data ADSs have dealt with large volumes of data, and in many cases it has been the main focus of the Big Data ADS. Indeed, some of the surveyed works go no further than applying conventional algorithms and approaches using Big Data mechanisms.

Therefore, the volume requirement for Big Data ADSs can be considered as partially fulfilled. However, there is still room for improvement that can lead to further research:

- No large-scale cyber-level ADSs for IN specific protocols. Some IT counterparts deal with application-level data (7th OSI layer) but no proposals exist for INs. While lower OSI level proposals exist and could be applied to INs, these kind of mechanisms have been more studied and attackers expect related defensive measures [108]. Therefore, it is necessary to develop large-scale ADSs that will gather information from IN-specific protocols, opening the way of analyzing packet payload information.
- Big Data IN storage. Though process data has been traditionally stored in historian servers, novel approaches for the storage of IN related data are necessary: not only process readings, but wider types of data (for instance, network traces, or alerts). This can help not only with Anomaly Detection but also for other fields of research regarding INs and Big Data.

2.5.2 Dealing with velocity

As stated in Section 2.4, the vast majority of the proposals are not continuous nor real-time. This presents the issue that the mentioned approaches are only capable of finding anomalies over historical data, and when new data arrives, a new, larger version of the original dataset that contains the new data is computed again in order to find anomalies. In some of the proposals historical data is divided in time bins and only data corresponding to an specific time bin is executed.

However, this is an impractical approach for a realistic ADS, more so in INs where, as previously stated, real-time detection is an important aspect. It is necessary to develop streaming models where incoming data is treated on arrival in order to detect anomalies. An issue regarding streaming models is that it is not possible to perform EDA on them. EDA and the building of several models require data at rest, so relationships between different observations can be defined. Similarly, most streaming models need well-defined models for acting on incoming data.

A solution to this problem might lay in building hybrid models based on a two-phase approach where (i) A model is defined based on gathered historical data at rest. (ii) After building a model, this model is applied to compute incoming streaming data. INs have the advantage over IT networks that they are more static and deterministic by nature, so two-phase ADSs seem a viable solution, as once an ADS model is built, it will seldom require an update.

It is necessary to mention, that to encourage and compare different contributions in the area of real-time ADSs for INs, it is necessary to create and use a set of metrics where latency should be taken into account [108].

2.5.3 Dealing with variety

Bompard et al. [11] state that "In order to make IDSs effective in protecting this kind of systems, it is then needed a set of multilayer aggregation features to correlate events generated from different sources (...) in order to detect large scale complex attacks. This probably represents the next research challenge in this field". Still, this is an open issue, as most proposals reviewed deal with a single, or few data sources for anomaly detection.

ICSs are multivariate and heterogeneous by nature, they deal with very diverse types of data, both at the network level (packets, flows, logs...) and specially at the field level where they keep track of a large number of different physical quantities simultaneously. However, existing large-scale ADSs do not leverage data from both levels. This issue is extensible to most conventional ADSs as well, as only a few proposals deal with both process-level and network-level data [53, 79] to detect anomalies.

BDA gives the opportunity to use this heterogeneous data and leverage it in a unified manner to detect anomalies. In this direction, the work of Camacho et al. [19, 20] gives promising insight. The usage of multivariate algorithms, such as PCA, can help to build a model where parametrized cyber and process data can be used to build a single ADS that leverages data from both levels. PCA-based techniques scale well horizontally and are used in fields such as genomics where they are used to handle massively dimensional data.

2.5.4 Dealing with veracity

From our point of view, Big Data veracity for Anomaly detection is not only related to correctly flagging a relevant anomaly on a large dataset, but also to communicate and alert the anomaly correctly, instead of overwhelming the operator with too much alert noise. In a related note, we believe that properly testing different Big Data ADSs on neutral, relevant environments such as using public datasets, is also ensuring the veracity of ADSs in Big Data. Therefore we can identify the following research areas:

- Closing the semantic gap. Sommer and Paxon [130] define the semantic gap as the lack of actionable reports for the network operator. In other words, the ADS does not provide sufficient diagnosis information to aid decision making for the operator. In INs, it is necessary for an operator to know what is the cause for an anomaly, as successful attacks or serious disturbances could have potentially catastrophic outcomes. BDA can help to provide useful information about the cause of the anomaly. Big Data visualization techniques or Visual Analytics might play a significant role in this matter.

- Necessity to have realistic, large-scale datasets. Few datasets exist for Anomaly Detection evaluation in INs and existing datasets [111] are too small to evaluate Big Data ADSs. Therefore, it is necessary to have public, realistic, large-scale IN datasets that would allow the evaluation of the ADS performance independently.

- Integration of honeypots. When trying to find anomalies in Big Data, it is important to keep the value of false positives and false negatives low. The task of finding anomalies is equivalent to find a needle in a haystack. Trusted data sources can help in this endeavor. Honeypots can consitute such a trusted information source, as by definition do not yield any false positives [146]. The field of IN-oriented honeypots is still maturing [146], but the possibility of feeding and correlating IN honeypot data to a Big Data IN ADS, in a similar fashion as Marchal et al. [103], opens the way to a new field of research.

2.6 Conclusions of the Chapter

We have presented a literature review comprising of three main contributions: i) a review of current proposals of Big Data ADSs that can be applied to INs, ii) a novel taxonomy to classify existing IN-based ADSs, and, iii) a collection of possible future research areas in the field of large-scale, heterogeneous and real-time ADSs for INs.

Big Data Anomaly Detection in Industrial Networks is still a developing field. Few proposals exist for INs exclusively, but some IT-based solutions show that it is possible to have similar counterparts on INs. Nevertheless, while most proposals focus on large-volume solutions for anomaly detection, other aspects, such as dealing with data with high velocity or variety is still largely untackled. We have offered some future research work areas regarding these open issues.

Chapter 3

A hybrid testbed for industrial network security and resiliency research

Contents

In the previous chapter, we have surveyed the field of Big Data anomaly detection from the point of view of Industrial Networks (INs). After discussing some of the issues regarding current research proposals, we have detected room for improvement in the field of data-driven, large-scale and heterogeneous Anomaly Detection Systems (ADSs) for INs.

However, before pursuing further development in this field, it is necessary to create a safe testing environment that will provide the means for the development of the ADS by generating data and can also serve as a realistic platform for ADS evaluation. This chapter covers the design of a hybrid IN testbed that aims to fulfill such purposes, with the additional objective of being useful for future research, not necessarily coupled with the objective of developing novel ADSs. Therefore, it supports additional technologies, such as Sofware-Defined Networking (SDN) or an integrated data analysis module.

3.1 IN Security Testbed Research

Due to the critical mission of some industrial processes and the difficulty to recreate tests in real, live environments, both the scientific community and industry have employed testbeds to design and test different security measures and scenarios. On the one hand, Holm et al. [61] survey existing Industrial Control System (ICS) security testbed proposals. They identify thirty different contributions, mainly devoted to vulnerability analysis, security education and the creation of environments to test and evaluate the performance of detection mechanisms, such as firewalls or Intrusion Detection Systems (IDSs). On the other hand, Di Pietro and Panzieri [39] develop a taxonomy that allows the classification of tested proposals into a set of categories. Likewise, they perform a comparison of seven different testbeds according to the presented taxonomy.

Based on these two works, it can be inferred that the main classification criterion for security ICS testbeds resides on the methods and solutions used for their implementation. Holm et al. [61] define these methods as virtualization, simulation, emulation and hardware. Most testbeds do not uniquely rely on a single method of implementation, and they combine different methods for different tasks. For instance, it is possible to simulate a physical process, while the controller runs inside a virtual machine.

However, among the mentioned implementation approaches, three different testbeds can be chosen as representative examples of each method: the proposals by Reaves and Morris [122], Candell et al. [23] and Siaterlis et al. [127].

Reaves and Morris [122] present a testbed centered on virtual devices, that work as controllers. These virtual devices are responsible of controlling a physical process that is simulated. Therefore, the control logic is implemented in them and they possess communication abilities that they use to gather data from the process and issue commands to it. Depending on the settings, these devices can act as masters or slaves and are interoperable with real control systems. The testbed has two different simulated processes: a water tank and laboratory-scale gas pipe.

The testbed presented by Candell et al. [23] is mainly based on hardware. The controllers and the rest of the agents in the industrial networks are physical. As for the physical process, this testbed has two variants: (i) Simulation of the Tennessee-Eastman process [42] and (ii) The usage of a physical robotic arm. In addition to these two different processes, the testbed has third component that contains the network hardware and it is responsible for the capture and modification of the network's packets.

The proposal of Siaterlis et al. [127], known as EPIC, is the most flexible of the three proposals here mentioned. EPIC's architecture is founded in the Emulab software [150], which employs to recreate the cyber layer of the IN (network topology and nodes). Emulab's flexibility allows to use physical and virtual controllers dynamically, along with different network topologies and states. Like the previous proposals, EPIC simulates the physical process under control.

Even if the focus of the aforementioned testbeds are fundamentally different, in practice, they share some shortcomings when handling cybersecurity research related to INs.

On the one hand, the proposals lack SDN support. The potentiality of these type of networks for their use in INs has already been mentioned in the literature [160]. Specifically, in the field of cybersecurity, SDN can provide resiliency and additional security mechanisms to INs [41, 110]. For this reason, Dong et al. [41] argue that SDN should be incorporated in IN testbeds, proposing an applied use case to the Smart Grid.

On the other hand, the different types of analyzed testbeds also lack a module for storing and processing the data created in it. As covered in Chapter 2, storing and processing data from an IN can pose a Big Data problem. Reaves and Morris [122] point to the lack of a module with this characteristics and consider the creation of a repository for storage "a natural extension" to their testbed. In the case of Candell et al. [23], they mention the existence of a module for network packet storage, but the module has no additional functionalities.

The testbed presented in this chapter aims to overcome these two issues, by providing support for SDNs and having an integrated module for large-scale data processing.

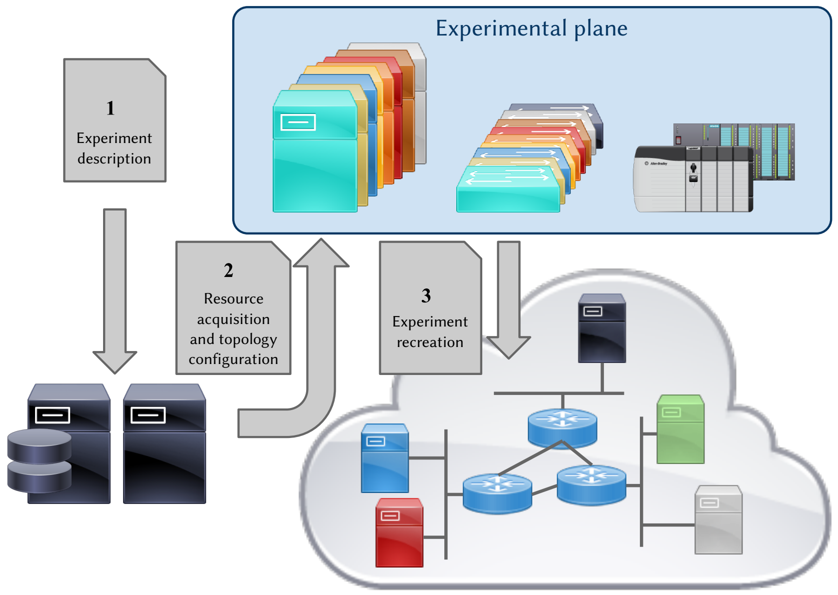


Figure 3.1: Experiment creation workflow in Emulab

3.2 Testbed structure

In this section we explain the structure of the proposed IN testbed.

3.2.1 Overview

The structure of the testbed is based on using the Emulab software [150]. Emulab is a software system which provides a platform for the research, education and development of networks and distributed systems. It has been designed with three main objectives: simple usage, environment control and realism. It achieves the objectives by utilizing abstraction and virtualization systematically.

Emulab's basic architecture is composed by two control servers, a set of physical resources that are used as experiment nodes and a set of network devices whose aim is to interconnect the different nodes.

Figure 3.1 shows the experiment creation process of Emulab in different steps:

1. Using a Network Simulator (NS) file, the user specifies the details of the experiment, such as the number of participating nodes (physical or virtual), the Operating System (OS) that powers each node, the network topology or the details of each of the network links (packet throughput, packet loss etc.)

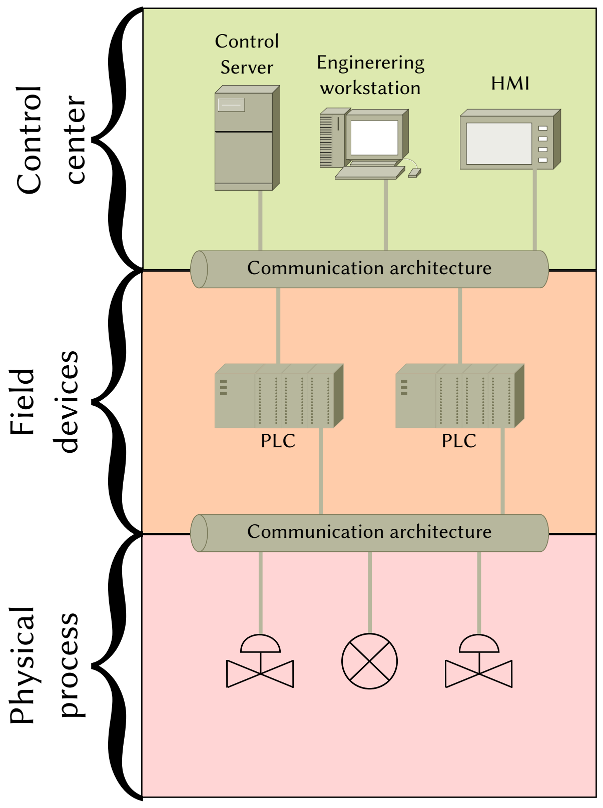


Figure 3.2: Simplified architecture of an IN testbed as defined by Holm et al. [61]

2. Once the details have been specified, the NS file is loaded in the servers that manage the testbed, and Emulab books and configures the necessary resources for the experiment.

3. Later, Emulab recreates the network topology, configures the nodes, creates the system users in the hosts and gives the final user the possibility of connecting to each of the nodes and interact with them. The experiment is ready to run.

Based on the guidelines given in the National Institute of Standards and Technology (NIST) Special Publication 800-82 [132], Holm et al. [61] consider four types of elements that every IN testbed should cover (Fig. 3.2):

1. **The Communication Architecture** encompasses the components that allow the communication between different devices that form the IN. Examples include switches, routers and communication lines.

2. **The Control Center or Supervisory Devices** refer to the severs and workstations that

are used to monitor and control the physical process remotely. Examples include process historians, Human Machine Interfaces (HMIs) and control servers.

3. **The Field Devices or Controllers** concern the devices that link the physical and the cyber world, and thus, generally interacting directly with the process itself. Examples include Programmable Logic Controllers (PLCs) or Remote Terminal Units (RTUs).

4. **The Physical Process** involves the physical reality that the ICSs monitor and control.

We now focus in each of the aspects and detail its implementation in our testbed.

3.2.2 Communication Architecture

In the testbed, the communication architecture is built around two main concepts: the Emulab software and SDN.

Emulab

Previously, we have mentioned Emulab as the software system that powers experiment execution. Figure 3.3 shows Emulab's architecture.

Emulab is installed in two different servers. These two servers manage all the hardware and software available for experiment recreation. The first server, dubbed *Boss*, controls the hardware, hosts the user interface and manages the name system and the deployment service that loads the disk images containing the OSs and applications that will run on the experiment nodes. The second server, named *Ops*, manages and stores user data, which generally is the one to be used in the experiments and the data created in them.

As shown in Figure 3.3, the two servers are connected into the control plane network. This network is used to have a direct control over the experiment nodes. Therefore, each experiment node has a dedicated network interface with the sole purpose of enabling management from Emulab, while the rest are used in the experiment. Moreover, *Boss* is connected to the hardware control plane. This network comprises all the switches and routers present in the network, along with the power controller. Switches and routers are configured to create the necessary Virtual Local Area Networks (VLANs) that will create the designed network topology for the experiment, whereas the power controller is used to remotely reboot experiment nodes and to shut down idle nodes that are not actively taking part in any experiment, allowing to save energy. *Boss* also needs outside network access, thus providing remote

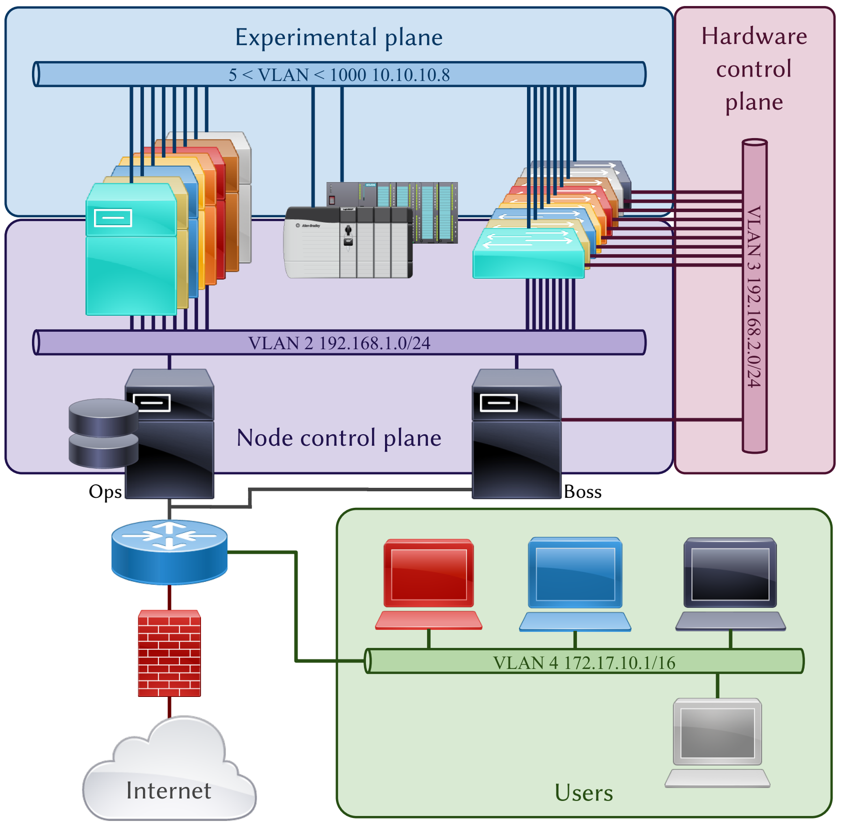


Figure 3.3: Emulab architecture

users the ability to interact with experiment nodes and uploading the NS file describing the experiment.

The experimental plane is used to recreate all the defined experiments. As stated previously, each experiment node has at least two network interfaces, one for its control and the rest connected to the experiment switches, where the experiment traffic is going to flow. Emulab maintains a database where it lists which node interface is connected to which switch port. This information is important when designing the VLANs.

Apart from recreating different network architectures, Emulab is also able to recreate different network conditions. That is, it can emulate packet loss, latency or bandwidth for each of the connections defined in the experiment. A study conducted by Siaterlis et al. [126] concludes that Emulab is an efficient and realist tool to emulate different network conditions and as such, it is a valid tool for scientific research.

Software Defined Networking

Software Defined Networking (SDN) constitutes a novel networking paradigm that aims to overcome the existing limitations of current networking infrastructures by unifying network policies in a single software system called controller [81].

Software defined networks separate the network control plane from the data transmission plane. Data transmitting devices such as switches or routers on traditional networks become simple packet forwarding devices, while the control logic and the decision making process are implemented in a single controller, belonging to the control plane. Therefore, the network agents responsible for network configuration are not the ones responsible for packet forwarding, as the controller instructs the network agents how to behave by sending commands that the agents integrate in their forwarding tables.

The communication between data and control planes can be conducted by using well defined network protocols. In that sense, OpenFlow [106] is one of the most important protocols serving this purpose. Devices capable of communicating using OpenFlow can have one or more packet forwarding tables containing different rules for packet handling. These tables contain rules where each of the rules describes a subset of the network traffic and an action to perform over it, such as removing it, forwarding it and/or modifying the packets. Therefore, depending on the rules that a controller sets on a SDN-capable device, this device can behave as a data forwarder, a firewall or can perform other actions such as load balancing or packet shaping.

This testbed uses OpenFlow as the protocol for the construction of different software defined networks. In order to prevent collisions between the Emulab software and the SDN controller, we identify the node containing the SDN controller as unmanaged, and as a consequence, its network connection would be independent and Emulab would not interfere with the node or the connection. Moreover, it is necessary to isolate each of the experiment nodes in different VLANs, to force packet routing to the SDN-enabled device, as the Emulab-managed switch would simply forward the packet based on the configuration loaded by Emulab, without first asking to the SDN controller. This way, the traffic is shaped according to the filters and rules defined in the SDN controller.

3.2.3 Control Center

The control center encompasses the set of devices that perform a supervisory control of the process, or enable its remote control or analysis. Examples of these devices include control servers, engineering workstations and HMIs. In contrast to the field devices, the devices present in the control center run over standard hardware and Commercial off-the-shelf (COTS) software. Consequently, for experimentation purposes, it is possible to virtualize these devices and have them run in virtual machines instead of running directly over hardware.

Emulab is able to work with virtual hosts, and this approach is more resource efficient, as physical hardware is shared between several virtual hosts. However, independently of the medium used, in order to employ control center devices in Emulab, it is first necessary to build disk images with the software that will be used in the experiment over the pre-built Emulab images. This process can be described in the following steps:

1. Load an Emulab pre-built image into a host.
2. Install and/or configure the necessary software in the host.
3. Create a new image from the host comprising the OS and the applications.
4. Deploy the newly created image in experiments.

3.2.4 Field Devices or Controllers

The existence of field controllers is one of the main characteristics of INs. Based on the data they receive from their multiple input ports and according to the implemented control algorithm, field devices act on the process by sending orders through their output ports to the actuators. They also send data and receive orders from the supervisory devices in the control center. As explained in Section 2.1, field devices act as a bridge between the physical and the cyber layer of the IN.

As with the control center devices, it is possible to use physical or virtual controllers.

Physical Controllers

Physical controllers are the hardware-based devices that are used in production INs. However, with Emulab, it is not possible to manage physical controllers in the same manner as the control center devices, as the hardware and software controllers run are much more specialized. However, it is possible to configure these controllers offline and to add them as unmanaged nodes to the experiment, where the Emulab software does not administer the controller, in a similar fashion to the SDN controller mentioned previously.

The main advantage of using physical controllers in the testbed relies on having a more realistic testing environment where it can be possible to perform vulnerability analysis or to work with proprietary software, hardware and protocols that can be challenging to replicate in a virtual or emulated environment.

However, the usage of physical devices presents the drawback of cost. Physical devices are expensive, and replicating a complex IN with a high number of controllers can result infeasible for this reason.

Virtual Controllers

The alternative to using physical controllers is the virtualization of controller functions. Nowadays, it is not possible to completely virtualize field devices, especially in the case of proprietary controllers [61]. In spite of it, it is possible to emulate the functions of a controller by using software, a practice that has been used previously in IN testbeds [122, 127].

In our testbed proposal, we use software-based controllers, that emulate field controller behavior by acquiring data from the physical process and forwarding it to the control center, while also relaying orders from the control center to the physical process. The existence of libraries for general purpose programming languages for industrial communications such as PyModbus[4] or Snap7[5], eases the task of building software that emulates controller behavior.

3.2.5 Physical Process: Tennessee-Eastman

As stated previously, the actual physical process is the physical reality that ICSs aim to automate, monitor and control. Therefore, it lays at the core of INs and without a physical process there would be no IN. When integrating physical processes into ICS and IN security research, the use of real processes is often not feasible.

Recreating realistic and complex physical processes (e.g. a city-level water distribution system, or a large chemical plant), with their equipment would require vast amounts of resources. Moreover, when evaluating attacks where the intent of the adversary is to cause physical impact (e.g. closing valves or increasing the pressure in a vessel) the equipment in the testbed can result damaged, and repairing it can be costly, both in time and money.

Consequently, when integrating physical processes into them, security IN testbeds have relied on two different approaches:

[4]https://github.com/bashwork/pymodbus
[5]http://snap7.sourceforge.net/

- Simple laboratory-level model processes [112].
- Complex, real-world, simulated processes [88, 127]

In our testbed, we have chosen to use the simulation approach for three main reasons. First, as mentioned previously, simulation simplifies embedding complex physical processes within a testbed. Second, it also eases the procedure of establishing a set of standard physical processes that can be used to reproduce findings across different testbeds. And third, the use of common simulation platforms offers a standard evaluation ground to evaluate and compare different proposals in a common environment.

The simulation model used in this work is the popular Tennessee-Eastman (TE) process, proposed by Downs and Vogel [42]. The TE process is modelled after a real chemical process of the Eastman Chemical Company. Though a real process, Downs and Vogel modified some of the aspects of the chemical process, such as the components, kinetics and operating conditions in order to protect its proprietary nature.

The TE process produces two liquid products (G, H) from four gaseous reactants (A, C, D, E). In addition, there are also a byproduct (F) and an inert (B), making a total of eight chemical components, coded after the first eight letters of the alphabet. Four irreversible, exothermic reactions take place in the modeled process:

$$\begin{aligned}
A(g) + C(g) + D(g) &\rightarrow G(liq), && \text{Product 1},\\
A(g) + C(g) + E(g) &\rightarrow H(liq), && \text{Product 2},\\
A(g) + E(g) &\rightarrow F(liq), && \text{Byproduct},\\
3D(g) &\rightarrow 2F(liq), && \text{Byproduct}.
\end{aligned}$$

Figure 3.4 shows a general overview of the process. It has five main operation units: the reactor, the condenser, the recycle compressor, the vapor-liquid separator and the stripping column. The gaseous reactants, fed by three different feeds, react in the reactor to form liquid products. The products, along with residual reactants, leave the reactor as vapors. Later, the vapors are cooled in the condenser to return to liquid state. Then, the vapor-liquid separator isolates the non-condensed vapors, which are fed again to the reactor by using a centrifugal compressor. The condensed components, on the other hand, move to a stripping column to remove the remaining residual reactants. The final product (Mix of G and H) exits the stripper and head to a refining section that separates them. This final refining section is not included in the model. Similarly, the inert and the byproduct are purged in the vapor-liquid separator as vapor.

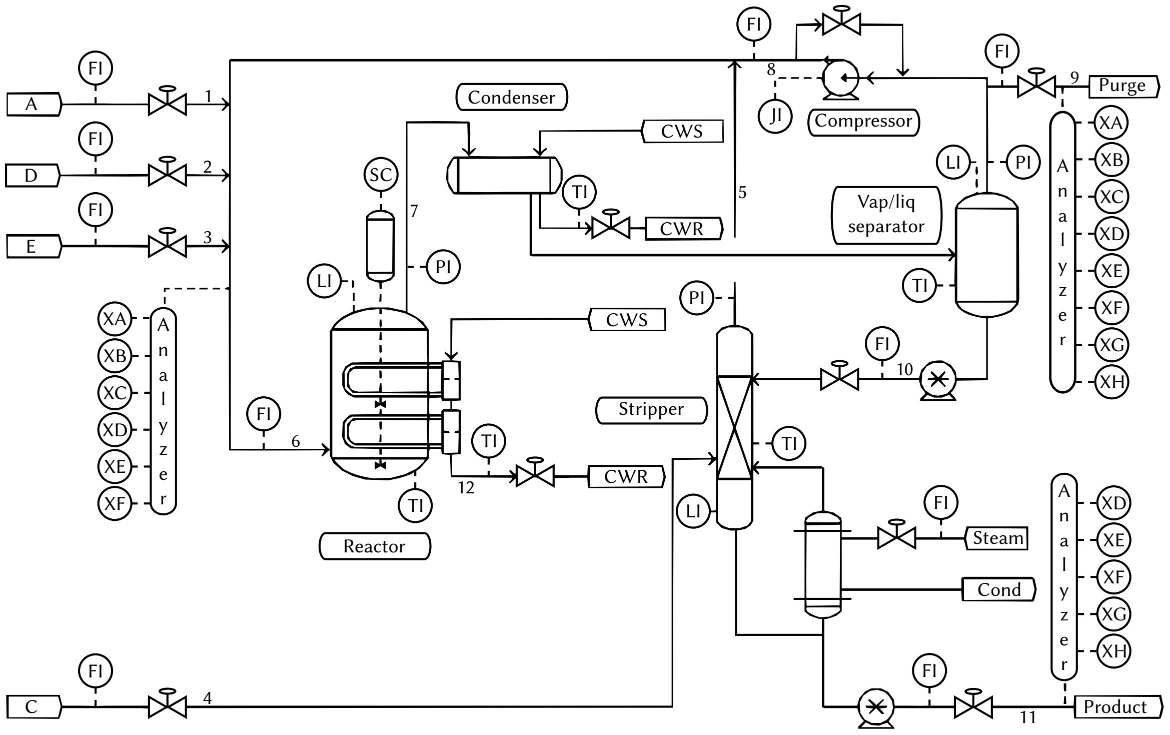


Figure 3.4: Tennessee-Eastman process flowsheet

The process can produce different ratios of G/H in the finished product. This aspect is determined by the operation mode of the plant. The model supports six different operation modes, at three different ratio of products and two different production rates.

Originally, the model was released with no embedded control approach, as its aim was to benchmark different control strategies. As such, several control strategies have been proposed for the TE process [93, 123]. Moreover, the TE process has also served as a standard ground for fault diagnosis proposals [155]. However, more recently, the TE process has increasingly developed as the main physical scenario for IN and ICS security research [25, 82, 85, 88, 105] and has become a *de facto* standard for this purpose.

In this thesis, we use the DVCP-TE implementation presented by Krotofil and Larsen [87], publicly available on GitHub[6]. This model is oriented towards security research. It is developed as a Simulink model, where the process itself is implemented as a Simulink S-function and the control approach is described using Simulink blocks. We use the control strategy presented by Larsson et al. [93]. Apart from the control strategy, the model also has implemented a framework to perform attacks on sensor and actuator signals that we will discuss

[6]https://github.com/satejnik/DVCP-TE

with further detail in other sections of this document.

3.2.6 Data Analysis

As covered in Chapter 2, IN data monitoring can certainly be considered a Big Data problem, as long, continuous experiments can generate large amounts of data, fast and of different nature (e.g. variable readings, logs or packet captures). This capture data can be later used for different purposes inside the testbed: process monitoring, anomaly detection or process optimization, among others.

We mentioned in Section 3.1 that the necessity of data storage and analysis in testbeds has already been noticed by some proposals.

In our testbed, we use Apache Spark [156, 157] for the analysis of the created data in the testbed. Built over a Hadoop File System (HDFS) layer, Spark can process both data at rest and continuously in a distributed manner.

The integration method for this module is similar to the control center; as Apache Spark is designed to work over standar OSs and hardware, it is possible to use the image creation process to deploy Spark instances in the experiment as needed.

3.3 Conclusions of the Chapter

The development of novel IN testbeds for security research is an active research field. As such, several testbeds have been proposed, each with different methods of implementation for evaluating various aspects of INs.

In this chapter, we have presented a testbed that dynamically emulates network behavior, including its topology and the devices that form it that additionally simulates a physical process. Together, the testbed is able to build the existing functionalities of an IN, both at the cyber/network and at the process level.

For this end, the testbed uses the Emulab software and the Tennessee-Eastman process, both previously used for IN and ICS security research.

Additionally, the testbed supports SDN, allowing further research in IN security, such as the development of advanced security mechanisms for attack detection and response and network resilience, an aspect that has not been covered in previous proposals. Besides, the testbed has an integrated data analysis module that allows the large-scale processing of resting and streaming data.

Chapter 4

Visualizing Network Flows and Related Anomalies in Industrial Networks using Chord Diagrams and Whitelisting

Contents

This chapter introduces a novel visual flow monitoring system for Industrial Networks (INs). First, it introduces chord diagrams and their previous uses for security visualizations. Next, it describes the proposed system, introducing the methodology of building the visual chord diagrams based on flow data. Finally, the chapter shows the evaluation of the proposal, where the system is validated using real industrial traffic.

4.1 Chord diagrams

Chord diagrams, also known as Circos diagrams, are circular diagrams that represent relationships between different entities. Though originally conceived for genomics [89], the usage of diagrams has expanded into a wide variety of fields.

Typically, the visualized entities are arranged in a circular manner. Each entity occupies a given arc length of the circle mentioned. This length is proportional to the weight the entity has compared to the rest.

Chords are links that match the entities that jointly form the circle all together. Each chord generally links two different entities, and the width of the chord at both ends denotes the nature of the link. The wider chord end belongs to the entity that is dominant in the relationship between both entities linked by the chord. For instance, in the case that the chord represents a trade relationship between two countries, the country with the wider chord end sells more goods to the other country and vice versa.

The main advantage in the usage of chord diagrams to represent network flow data, even under normal network operation conditions, is that diagrams can provide situational awareness to operators in a direct manner, whereas traditional text-based alert systems cannot. This way, network operators can easily check how each host is interacting with the rest of the network.

Moreover, when using chord diagrams, it is not only possible to visualize relationships between different entities, but also their prominence when compared to the rest of the network. When visualizing network flows, it is possible to represent their activity through the size of the chords. For instance, active flows can be depicted using larger chords. Other types of visualizations, such as bi-partite graphs, lack this magnitude feature.

Communication patterns between hosts are fixed in Industrial Networks, as in this type of networks each host usually only communicates with a small subset of the hosts present in the network. Therefore, few chords are necessary to represent all possible flows, and diagrams are kept simple enough to be meaningful, even in large networks.

Chord diagrams are considered to scale well [89, 104]. However, if an industrial network is complex enough to render a unique chord diagram too confusing, simpler chord diagrams can be computed for each of the network segments. Industrial Networks are hierarchical, vertical and segmented by nature [50], so it is possible to use different chord diagrams to represent the traffic in a network segment. Another approach to tackle potential scalability issues might be to use the multi-scale approach proposed by Zeng et al. [159].

In the field of network security, chord diagrams have been used in diverse types of visualization systems, but its usage is not as widespread as other types of diagrams.

Mazel et al. [104] use chord diagrams to perform a visual comparison of different Anomaly Detection Systems and their detection performance.

The work of Layton et al. [95] gives a representation of the relationships between clusters of phishing websites with chord diagrams.

OCEANS [34] uses chord diagrams (dubbed as Ring Graphs) for visualizing network flows between subnets. However, OCEANS is centered in traditional IT networks and lacks the additional information that can be gathered from industrial networks, where whitelisting policies can not be as strict as in industrial networks. Moreover, the color code used in OCEANS' chord diagrams is by the logical location of the host or subnet (internal or external IP), not by the nature of the connection (normal or anomalous).

To the best of our knowledge, no flow and security-oriented visualization system has been developed for industrial networks, let alone using chord diagrams. Nevertheless, some advances have been made to ease process monitoring visualization [136].

4.2 Proposed visualization system

Figure 4.1 shows the workflow of the flow monitoring and visualization system.

First, flow-enabled networking devices inside an industrial network send network flow packets to a flow collector.

Once flow collection has started, the flow collector is queried to generate offline, a model of detected flows. The model contains a whitelist of allowed network traffic flows. We call this phase the learning phase. Once a model has been created, the system queries the collector for new flow records and compares them to the model online, detecting flows that do not comply with the policies and tagging each individual detected flow as valid or anomalous. This corresponds to the detection phase. Finally, once the tagged dataset is available, the system builds a set of chord diagrams to represent the results. This is the visualization phase.

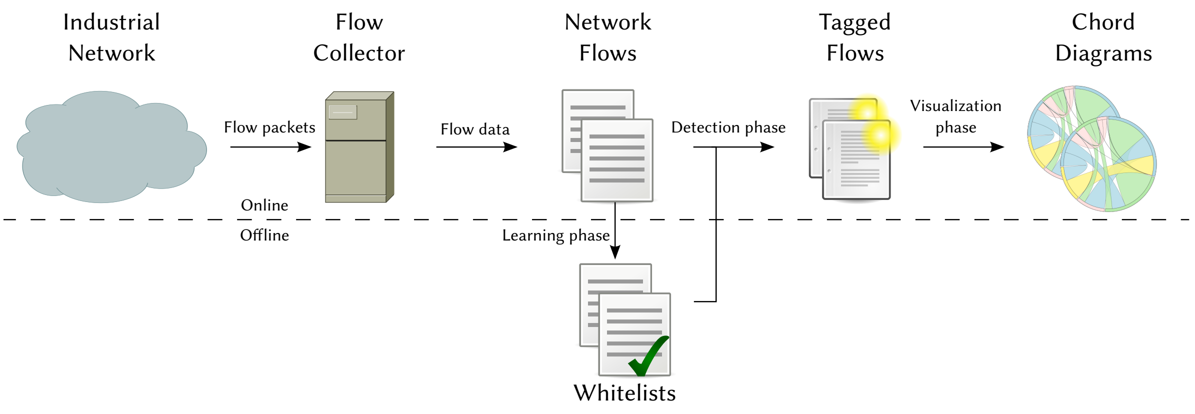


Figure 4.1: Overview of the flow monitoring system

While the learning phase happens once per network, the detection and the visualization phases occur periodically.

4.2.1 Learning phase

In this phase, a model is automatically created from the flows that have been detected in the network in a given time frame. The length of this time frame to build the model depends on the nature of the controlled process. For instance, a process that consists in small batches will require shorter learning time than longer, continuous processes, as the cyclical network patterns will be shorter. The collected network flows in this time window are considered legitimate and are used to build the model.

The whitelist that models the network flow behavior is stored in a human-readable Comma-Separated Values (CSV) file. This way, it is possible for an operator to add missing flows to the modeled whitelist, or, on the contrary, to delete flows that should be considered anomalous.

In our approach, we store the following data on the whitelist per flow: source IP address, destination IP address, server port, IP protocol and registered number of packets in the flow in the given time. For whitelisting purposes, the client port is not registered as it is assigned randomly and taking it into account would yield false positives. Barbosa et al. [7] do not take the number of packets in the flow into account. However, we consider packet number an important aspect to be recorded for two main reasons: (1) it is a good metric to be used with chords in the visualization (e.g. to depict the more active flows as wider chords), allowing the operator the identification of the main network flows. (2) this approach allows the system to detect flow anomalies that relate to its size (e.g. Denial of Service attacks or a downed host).

Whitelisting with time-dependent flow data

As useful as might be, taking into account the number of packets complicates the usage of whitelists. As the number of packets in a flow is time-dependent (the longer the time, the higher the number of registered packets), it is necessary to establish the time frame in which the whitelist is valid when comparing this value. In other words, a whitelist is only relevant if the capture time that has been used to build it is the same as the time length of the incoming flow data. For instance, if a whitelist records the first ten minutes of the flow data from an industrial network, it is necessary to poll the network in intervals of ten minutes in order to be able to correctly compare packet numbers.

There are two approaches that can be followed:

1. A single whitelist is created, with recorded flow data from a specific time frame. All incoming flow data is collected and later, when querying it, it is divided in chunks where the capture duration of each chunk is the same as the time the whitelist has used upon creation. The latest chunk of flow data and the whitelist are compared and a single visualization is created.

2. Various whitelists are created, each containing data belonging to different time frames. Flow data is collected, and when querying it, the chunk size varies to the duration of the specific whitelist it is being compared to. Latest chunks of flow data are compared with each correspondent whitelist and different visualizations are created, each showing the information of the last time frame belonging to the chunk and whitelist.

The second option is a better option, as it offers more granularity and increases the ability to detect flow anomalies that might not be easy to detect with a single, fixed-length whitelist. For instance, let us assume a host that sends a large number of packets in short bursts but within the packet number limits of the whitelist with short time frames. If this bursts should decline after a short time, but for whatever reason they do not, the unique whitelist system will not be able to detect the anomaly, as it is not able to check the system in the long run and the packet number is correct in each of the short time frames.

If the opposite case, where the whitelisting time frame is too long, we might not be able to detect short bursts of a high number packets that might not change much the whole number of packets in the long run.

Therefore, in our approach we propose a system where whitelists of different time length are considered. Nonetheless, the optimal number of whitelists and the time length of each of them is process-dependent and should be studied for each network. However, it is important to note that with longer learning periods, the probability of whitelisting malicious traffic gets higher.

4.2.2 Detection phase

In this phase, the different created whitelists are used to evaluate new flow data. This flow data is queried from the flow collector with different time lengths in order to match each of the time lengths registered with the whitelists. Later, this new flow data is compared to the whitelist corresponding to the same time frame. This way, the packet number of each flow is kept consistent, as comparing data collected in different time lengths would raise a high number of false positives. This process is repeated constantly in a batch manner.

The mechanisms checks if the flow data matches the one in the whitelist. In the case of source and destination addresses, server port and protocol, the flow information must match exactly. In the case of the registered number of packets in the time frame there is an exception: both numbers do not have to match exactly, but do not have to differ vastly either. The detector gives the possibility of setting a user-defined threshold for packet number tolerance in terms of percentage. Flows that are above or below this percentage threshold are considered anomalous, while the ones that are within the limits are considered valid.

If the flow is whitelisted, no alert is raised and the flow is tagged as legitimate. Still, if a non-whitelisted flow is detected, the system raises an alert and the flow is tagged as anomalous. In addition, the system also checks if all the flows registered in the whitelist also happen during the given time frame. If a flow registered in the whitelist has not been detected in the given time frame the flow is tagged as missing and an alert is raised. This gives the opportunity of detecting a downed host or connection.

We have created the following tags in the detector, based on the comparisons the system does between whitelists and new flow data:

Whitelisted flow The flow is considered legitimate according to the whitelist.

Anomalous network flow Two hosts communicate between them but according to the whitelist, these two hosts are not allowed to do so. All flows regarding a previously unknown host are marked as such.

Incorrect port A host tries to access a different port than the usual on a host it is allowed to communicate with.

Incorrect protocol A network flow is detected using a different IP protocol to the whitelisted one.

Missing flow A flow contemplated on the whitelist has not been detected on the collected flow data.

Anomalous flow size The packet number on the designated flow is either higher or lower than the defined threshold when compared to the whitelist.

Each of this tags is used to give information about the cause of the anomaly both in the raised alerts and in the rendered chord diagram.

Once the data has been tagged, the system translates known IP addresses into host names in the tagged dataset in order to make flow data easier to understand to the user.

Finally, after the detection phase, we have a fully tagged flow dataset. This tagged information is later used in the visualization phase to build the chord diagram that depicts the network flows and related anomalies in the industrial network.

4.2.3 Visualization phase

In this phase, each of the tagged flow datasets is rendered visually in the form of a chord diagram.

First, each of the active hosts in the network is given an arc section of the circle of the chord diagram. The arc length is given by the number of packets the host has sent on the measured time frame; more active senders have wider arcs than more silent hosts. The nature of the host determines its color; each type of host has an identifying color (e.g. PLCs are blue) while individual hosts are differentiated by having a different shade of the same color.

In our case, Programmable Logic Controllers (PLCs) are depicted with blue colors, control servers are green, Human Machine Interfaces (HMIs) are purple and, finally, different network devices (gateways, switches etc.) are colored in orange.

Once the hosts have been located, it is necessary to represent network flows between them. This is achieved by using chords: each bidirectional network flow is rendered as a single chord that links two distinct hosts. If two hosts have different network flows (for instance, a host communicates with two different services offered by another host), only a single chord is created in the diagram. The width of each chord end is given by the number of packets the related host sends. For example, if in a given flow Host A sends more packets to Host B than vice versa, the chord will be wider at the Host A's end. Similarly, more busy flows are depicted as wider chords than the almost-inactive counterparts. Later, each of the legitimate flow chords is filled with the color of the more active host in the communication.

Figure 4.2 shows a completed chord diagram where all the registered flows have been tagged as legitimate. Note that hosts of the same type share similar colors. In chord diagrams where network flow data is shown, all chords will link distinct hosts, as when a host accesses a local service, the network communication is carried out through the loopback interface and the data does not travel over the network. When the user hovers over an specific flow, the visualization shows basic information about the flow, such as the name of the involved hosts and the number of packets that take part in each direction of the network flow.

Since under normal operation conditions legitimate traffic flows represent most of the traffic of an industrial network, the reproduction of each traffic flow by a selection of a color-range makes easier to distinguish between different flows. Thus, the network operator can determine if the traffic tagged as legitimate is behaving as expected.

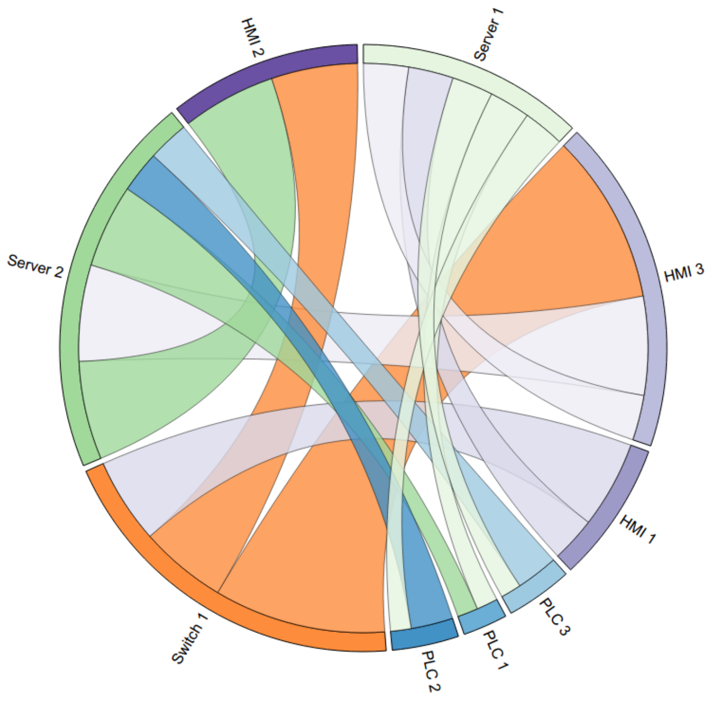
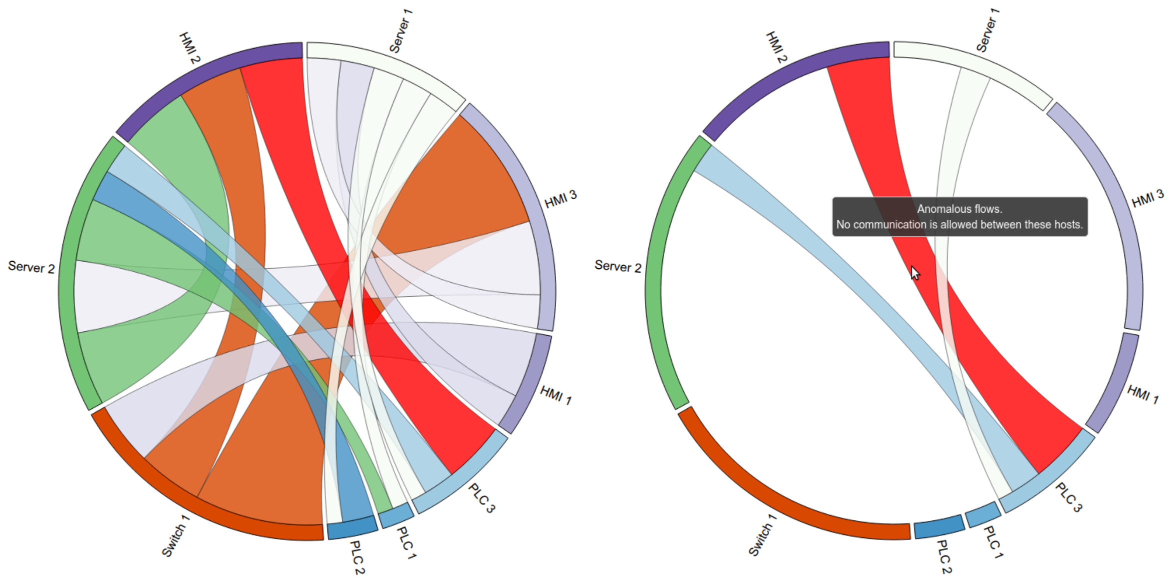


Figure 4.2: Chord diagram depicting a set of legitimate network flows.

In case of non-legitimate flows, the chord is filled with red color, as it can be seen in



Anomalous flows.
No communication is allowed between these hosts.

(a) Anomalous flow between PLC 3 and HMI 2. **(b)** Detail of the flow when hovering the mouse.

Figure 4.3: Representation of an anomalous network flow.

Figure 4.3. On the one hand, Figure 4.3a represents how the red color stands out over the rest of chords when the diagram is rendered. On the other hand, Figure 4.3b shows how the diagram filters the information concerning a single host when hovering the mouse over it, to highlight related information and ease visualization. As it is also shown, when hovering over the anomalous flow, the diagram shows additional information about the flow, regarding the reason why it has been flagged as such. As stated before, this information is contained in the tag assigned in the detection phase. In this case, no traffic between the PLC 3 and HMI 2 is allowed according to the whitelist.

With the exception of the "Missing flow" tag, all detected non-legitimate network flows are dyed in red to visually highlight it from the rest of the flows. However, due to the different nature of the "Missing flow" tag, these flows are rendered in black (see Figure 4.7). These flows are as well the only flows that are rendered with the data from the whitelist instead of the collected flows, as no data regarding them has been retrieved from the network.

4.3 Application in an Industrial Network monitoring dashboard

This section tests the previously described system within an industrial network.

4.3.1 Test network

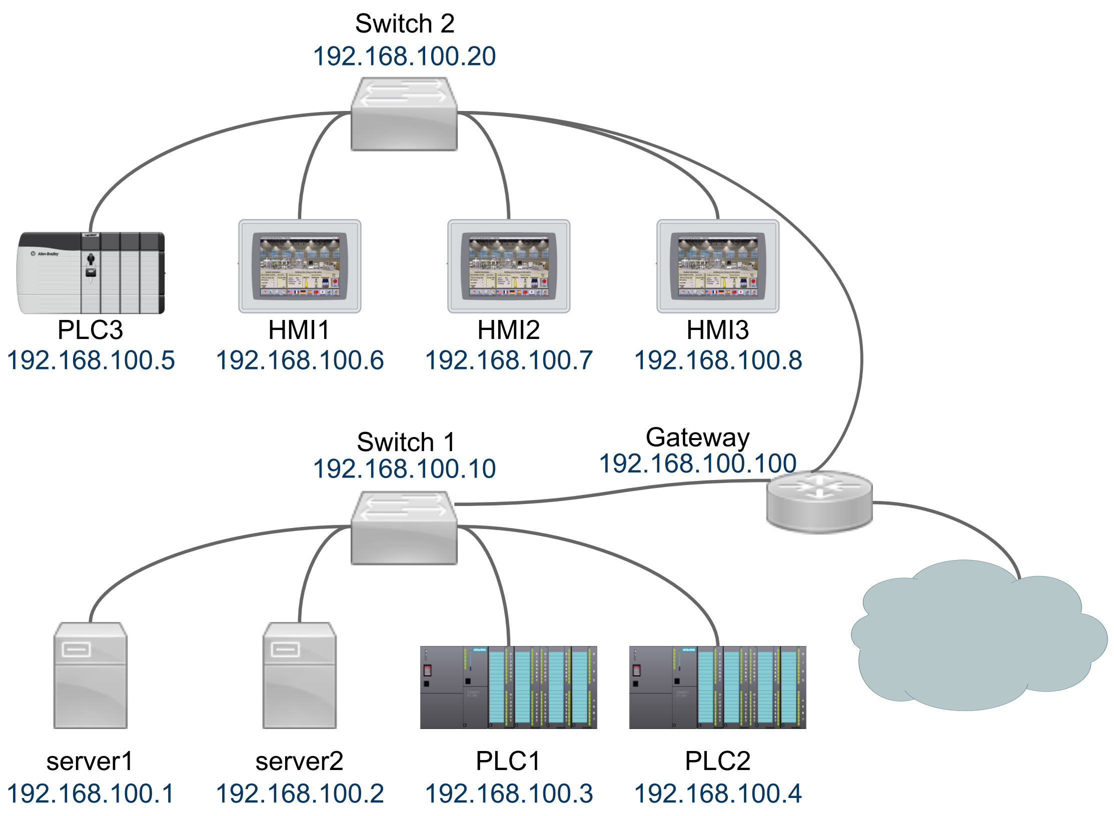


Figure 4.4: Network topology of the test industrial network.

As security testing on a live network can have unexpected consequences, such as malfunctioning or safety issues [44], and currently, to the best of our knowledge, there is no network flow data for industrial networks, we have duplicated the network topology of a real industrial installation. The original network is the control network of a car painting line in a manufacturing facility.

Figure 4.4 shows the topology of our test network. Both network switches are the network agents that send flow packets to the collector. In our case, we use Cisco's NetFlow, version 5. Moreover, Switch 1 is also the DNS Server of the network.

There are three Programmable Logic Controllers (PLCs) in the network that are responsible for controlling the industrial process. Two supervisory control servers poll process data from all the PLCs. Communication between servers and PLCs is done through the Modbus/TCP protocol.

There are also three Human Machine Interfaces (HMIs) present in the network, that enable operators to overview the process through the representation of process data in a visual, accessible manner. HMI 1 gathers data from Server 1, HMI 2 renders data from Server 2 and finally, HMI 3 visualizes data from both servers. Communication between HMIs and the servers is done using the OLE for Process Control–Unified Architecture (OPC-UA) protocol.

A gateway gives the industrial network access to external hosts, such as the network flow collector.

4.3.2 System Implementation

For our tests, we use Cisco's Netflow (version 5) as network flow system to send data to the flow collector. The switches from the network send flow data to a Logstash[7] agent that receives it, parses it and later indexes it in an ElasticSearch[8] cluster. This approach allows potential large-scale usage of the monitoring system and fast querying of the flow data to render visualization. The visualization system that builds these chord diagrams has been developed using the D3 [13] library.

4.3.3 Cases

In this section we show rendered chord diagrams in three different anomalous cases: a Denial of Service (DoS) attack, a network scan that aims to enumerate hosts in the network, and a network outage where a host goes down. For test purposes, all the next chord diagrams have been created with data taken at ten minute intervals, using their equivalent whitelist and with a threshold of 20% variation tolerance in the number of packets in the flow.

Denial of Service

Denial of Service (DoS) attacks occur when an attacker tries to obstruct the normal functioning of a host or service by making it unavailable to legitimate users. In industrial networks, where availability is the primary security concern and latency issues can create significant network problems, DoS attacks are a real problem. In our case we mimic a DoS attack from the HMI 3 to Server 1 by making a great number of illegitimate network requests.

Figure 4.5 shows the rendered result. The flow with the attack is painted in red, as it has surpassed the established threshold for network packages. As the sent number of packets

[7]https://github.com/elastic/logstash
[8]https://github.com/elastic/elasticsearch

gets higher, HMI 3 also gets a wider arc in the chord diagram circle, as well as the chord's end in its side.

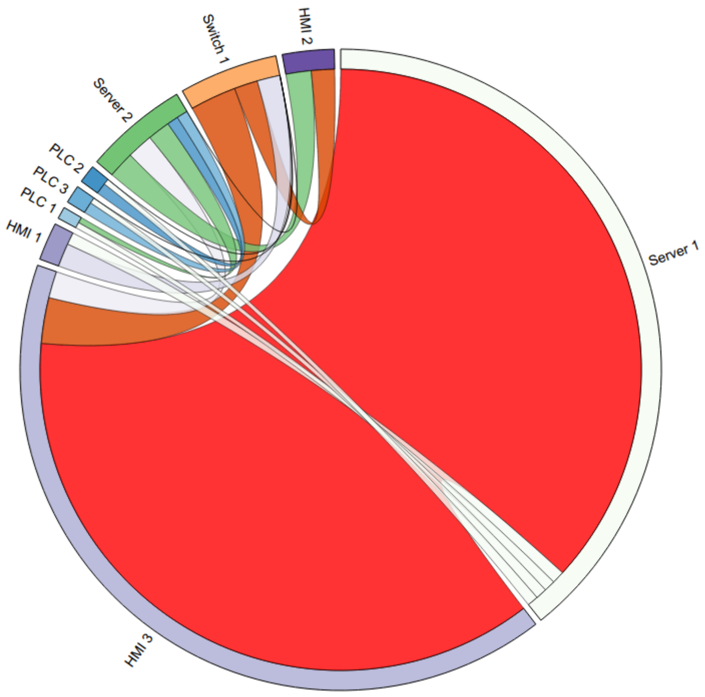


Figure 4.5: Visualization of a Denial of Service attack.

Host discovery

Host discovery is one of the first steps an attacker performs when obtains access to an unknown network in order to gather insight about it. Port scanning is one of the most used techniques for host discovery. For our test, conducted a TCP Connect scan with Nmap from the host HMI 3.

Figure 4.6 shows the chord diagram depicting the attack. All flows regarding HMI 3 are flagged as malicious, either because it is communicating with non whitelisted hosts (e.g. PLCs) or because it uses different protocols and/or ports with hosts that it is actually allowed to communicate with.

Host down

Finally, we consider the case when a host goes down from the network and it is not able to receive or send packets. In this case, we have physically disconnected Server 1 from the

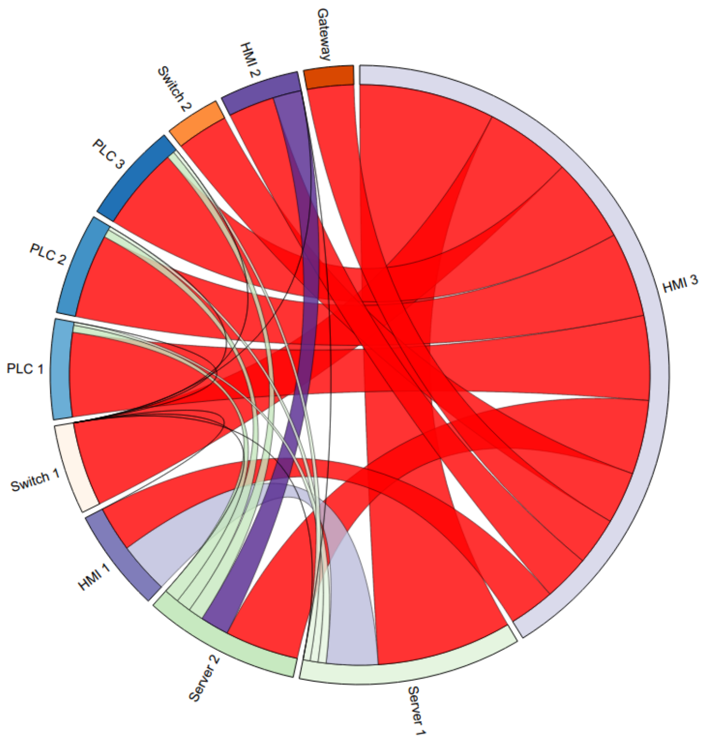


Figure 4.6: Visualization of a port scan.

network.

Figure 4.7 shows how the system shows the downed host, with black chords representing that we are dealing with missing flows. In order to be able to render the diagram, data is taken from the whitelist, as no real data has been collected from the network regarding these flows.

4.4 Conclusions of the Chapter

We have presented a novel pipeline for network data analysis that enables to visually monitor industrial networks by using whitelists and chord diagrams. To do so, first we build a time-based industrial traffic model which whitelists allowed network flows. Moreover, the model considers packet throughput, in addition to host addresses, server ports and IP protocols that makes possible to detect additional flow-related anomalies (DoS attacks and downed hosts). Each entry of the model whitelists an specific duration of gathered flow data. In the same way, every new flow data is compared against the traffic model to see if it fits an entry. All flows are tagged according to its nature (legitimate, anomalous, incorrect port or protocol,

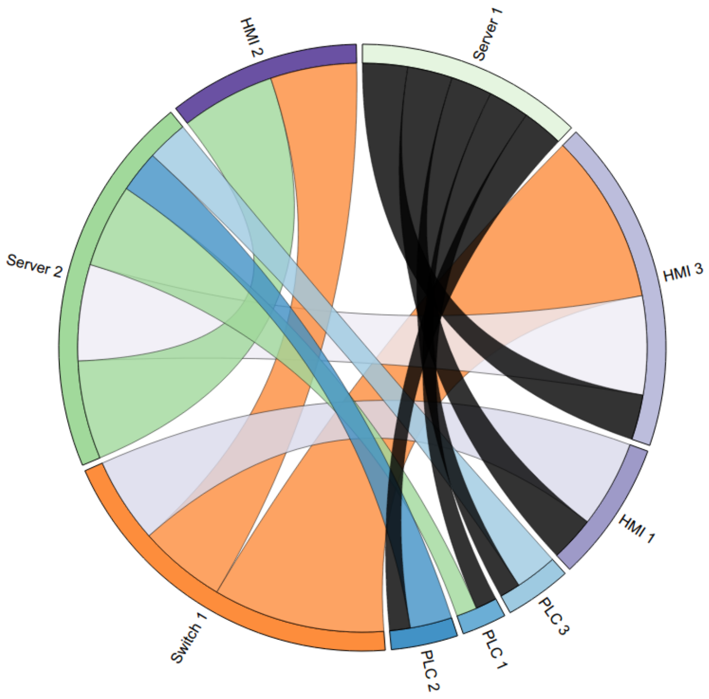


Figure 4.7: Visualization of a downed host.

missing and anomalous flow size).

This tagged data is used to build chord diagrams that represent network flow relationships between different hosts. The size of the chords represents the amount of network packets in the flow, used as the main metric to build the diagram. The tagging system provides a color code to highlight anomalous flows (in red and black) and also provides feedback about its nature. Historical flow data is stored in a scalable search server to store large amounts of data and to perform fast queries over it.

Chapter 5

On the Feasibility of Distinguishing Between Process Disturbances and Intrusions using Multivariate Statistical Process Control

Contents

This chapter introduces a different data-driven methodology for anomaly detection and diagnosis in the physical layer of Industrial Networks (also known as the field level). This technique, Multivariate Statistical Process Control (MSPC), previously used in the field of process control, simplifies variable monitoring in multivariate environments. The chapter further analyzes the performance of MSPC for anomaly detection, specifically focusing in the diagnosis of events that can be misread due to other similar situations with a related outcome.

5.1 Anomaly Detection based on Process data

We introduced Anomaly Detection Systems (ADSs) in the context of Industrial Networks (INs) in Section 2.1 of this work. Anomaly Detection is an active research field and we mentioned that different proposals regarding anomaly detection can be categorized into network-level or field-level ADSs.

While most of the approaches leverage network level data to detect anomalies in INs (see surveys [51, 108, 164]), other proposals, such as the one presented in this chapter, address this task by leveraging field or process data.

When dealing with field level data, proposals can be further classified in two subgroups: (i) solutions that require a model of the monitored process to detect anomalies and (ii) approaches where modelling the process is not necessary. Process model dependant contributions include the work of McEvoy and Wolthusen [105] and Svendsen and Wolthusen [134]. While effective to detect anomalies, these approaches require accurate modelling of the physical process. This requirement poses an important obstacle for implementing detection systems of this nature, especially in complex processes. More process-independent approaches on the other hand, include the work of Kiss et al. [82] and Krotofil et al. [88].

Kiss et al. [82] present an anomaly detection technique based on the Gaussian mixture model clustering of the field-level observations. Later, they use silhouette examinations to interpret the results. Nevertheless, they only consider attacks as possible factors for abnormal situations in the process, without considering process faults or disturbances. Therefore, process related anomalies could be mislabeled as attacks and vice versa.

Krotofil et al. [88] propose a method to detect when attackers tamper with sensor signals. To this end, they use entropy to detect inconsistent sensor signals among a cluster of correlated signals. Although they consider scenarios with process disturbances, there is no direct comparison between tampered sensor signals and similar process disturbances.

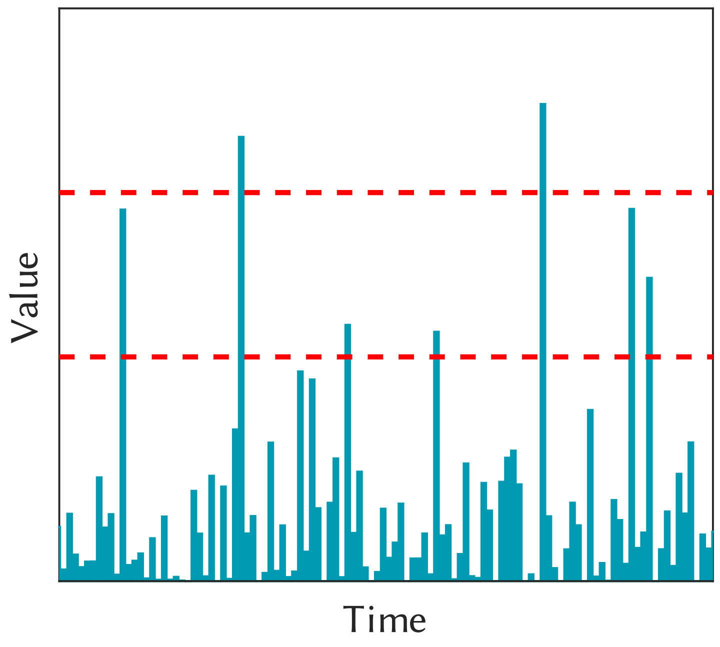


Figure 5.1: Example of a control chart. Control limits are presented for 95% (lower dashed line) and for a 99% (upper dashed line) confidence levels

In this approach, we go beyond the state of the art by presenting a novel, data-driven, security anomaly detection and diagnosis technique for field data. Additionally, we also analyze the effect of process disturbances and its effect when detecting security anomalies.

5.2 Multivariate Statistical Process Control

Multivariate Statistical Process Control (MSPC) [101] is a process monitoring methodology that relies on the use of multivariate control charts to detect unexpected changes in the monitored process. It is an extension of the univariate Statistical Process Control (SPC) approach.

Stoumbos et al. [133] define SPC as a "set of statistical methods used extensively to monitor and improve the quality and productivity of manufacturing processes and service operations. SPC primarily involves the implementation of control charts, which are used to detect any change in a process that may affect the quality of the output."

Figure 5.1 shows an example of a control chart with two control limits given by two different confidence levels. Under normal process operating conditions, 99% of all the points will fall under the upper control limit. In that case, we consider that the process is in a state of *statistical control.* It is important not to confuse this term with other similar expressions, such as control loop or automatic feedback control, as they refer to different concepts. Statistical control refers to the state of the process where only common causes of variation are present [101].

The existence of consistent observation series over the established control limit is likely to be attributed to a new special cause. In the case of a physical process, this variation source may be attributed to attacks or process disturbances, i.e. an anomaly.

The univariate nature of SPC means that only a single variable is monitored and visualized in a control chart. However, industrial processes are multivariate by nature, as many process variables are observed in a plant (e.g. temperatures, pressures, volumes or distances). As monitoring all variables with SPC would be impractical, only a few of them are monitored, generally the ones related to product quality (e.g. purity of the produced chemicals).

Nevertheless, the monitoring of a few quality-related variables is impractical. The approach does not take into account the information that other process variables give. For instance, the diagnosis of an anomalous event is complicated, as it relies on expert knowledge and a one-at-a-time inspection of process variables [84].

MSPC aims to solve these problems by providing tools to monitor all measured variables in an efficient manner. In that sense, MSPC does not only monitor the evolution of variable magnitude but also the evolution of the relationship it has to other variables. For this end, a main technique that MSPC uses is Principal Component Analysis (PCA).

5.2.1 PCA-based MSPC

Let us consider process historical data as a $\mathbf{X} = N \times M$ two-dimensional dataset, where M variables are measured for N observations. PCA transforms the original M-dimensional variable space into a new subspace where variance is maximal. It converts the original variables into a new set of uncorrelated variables (generally fewer in number), called Principal Components (PCs) or Latent Variables.

For a mean-centered and auto-scaled[9] $\mathbf{X}$ and A PCs, PCA follows the next expression:

$$\mathbf{X} = \mathbf{T_A}\mathbf{P_A^t} + \mathbf{E_A} \tag{5.1}$$

where $\mathbf{T_A}$ is the $N \times A$ score matrix, that is, the original observations represented according to the new subspace; $\mathbf{P_A^t}$ is the $M \times A$ loading matrix, representing the linear combination of

[9]Normalized to zero mean and unit variance

the original variables that form each of the PCs; finally, $\mathbf{E_A}$ is the $N \times M$ matrix of residuals.

In PCA-based MSPC, both the scores and the residuals are monitored, each in a separate control chart [22]. On the one hand, to comprise the scores, the D-statistic or Hotelling's T^2 [62] is monitored. On the other hand, in the case of the residuals, the chosen statistic is the Q-statistic or SPE [77].

For an n observation, both statistics are computed as follows:

$$D_n = \sum_{a=1}^{A} \left(\frac{t_{an} - \mu_{\mathbf{t}_a}}{\sigma_{\mathbf{t}_a}} \right)^2 ; \; Q_n = \sum_{a=1}^{A} (e_{nm})^2 \tag{5.2}$$

where t_{an} is the score of the observation in the a-th PC, $\mu_{\mathbf{t}_a}$ and $\sigma_{\mathbf{t}_a}$ represent the mean and standard deviation of the scores of the a-th PC in the training data respectively and e_{nm} stands for the residual value corresponding to the m-th variable.

D and Q statistics are computed for each of the observations in the anomaly-free training data, and control limits are set for each of the two charts. Training data is previously inspected through Exploratory Data Analysis (EDA) to remove existing outliers that could change D and Q values. Later, these statistics are also computed for incoming data and plotted in the control chart. When an unexpected change occurs in one (or more) of the original measured M variables, one (or both) of these statistics will go beyond control limits. Thus, an M-dimensional monitoring scenario is effectively converted into a two-dimensional one.

An event is considered anomalous when three consecutive observations surpass the 99% confidence level control limit in either of the monitored statistics [119]. Leaving some of the observations out of bounds (1% of the observations with a control limit set on the 99% confidence level) improves the performance of the control charts in the monitoring phase [21, 119].

Once an anomaly has been detected, anomaly diagnosis in MSPC is generally carried out using contribution plots [84]. These plots show the contribution of the original measured variables to an anomalous event. Details of the calculation and analysis of contribution plots can be found in the work of Alcala and Qin [1].

In this work, we use oMEDA plots [18] to diagnose the anomaly causes by relating anomalous events to the original variables. In essence, oMEDA plots are bar plots where the highest or lowest values in a set of variables reflect their contribution to a group of observations. Therefore, when computed on a group of observations within an anomalous event, the most relevant variables related to that particular event will be the ones with the highest and lowest bars. Though similar, one of the main differences of oMEDA plots with traditional contribution plots is that the oMEDA plots are capable of comparing different sets of observations whereas traditional plots can only compute a single set of them. In that sense, oMEDA plots can be considered an extension to the contribution plots. In this case, to compute oMEDA we first define a dummy variable, **d**, a vector of length N, in which the anomalous observations that are to be computed are marked with 1, leaving the rest as 0.

For a set of observations marked in **d**, oMEDA is computed as follows:

$$d^2_{A,(i)} = \frac{1}{N} \cdot \left(2 \cdot \sum_{(i)}^{d} - \sum_{A,(i)}^{d}\right) \cdot \left|\sum_{A,(i)}^{d}\right| \tag{5.3}$$

where $\sum_{(i)}^{A}$ and $\sum_{A,(i)}^{A}$ represent the weighted sum of elements for variable i in $\mathbf{X}$ and its projection $\mathbf{X}_A$ according to the weights in **d**, respectively. Larger absolute values of d^2 will indicate a larger contribution of that variable in causing the anomalous observation.

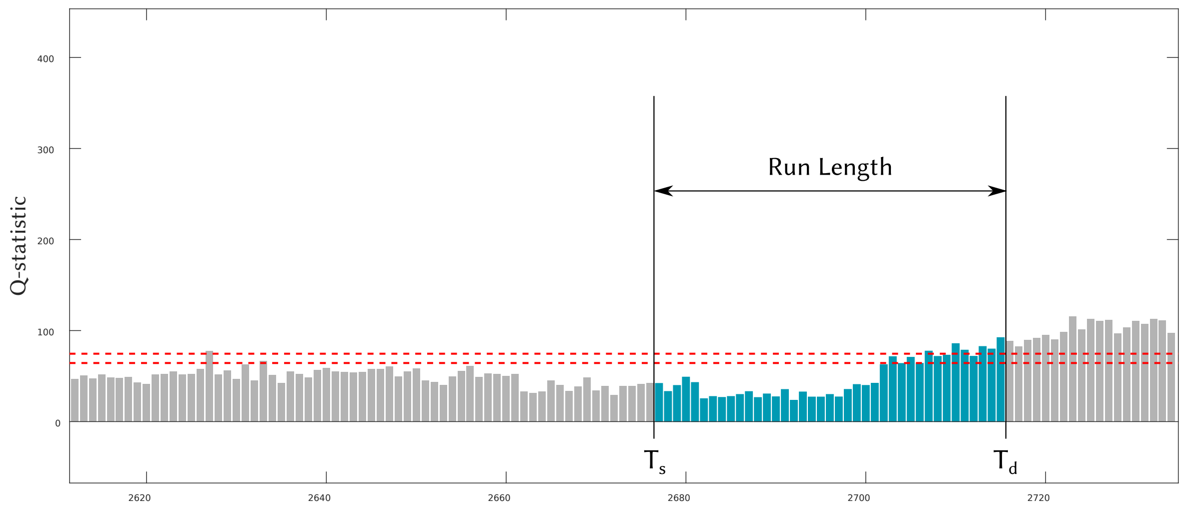


Figure 5.2: Run Length example

In the field of process control, Average Run Length (ARL) is an evaluation metric that measures the performance of control approaches in different situations. Chakraborti [30] defines run length as a “variable that represents the number of control statistics that must be plotted in order for the chart to first detect a shift from a stable or in-control process”.

In our case, we treat run length as the elapsed time between the start of an anomalous situation, either a disturbance or an attack (T_s) and its detection when three consecutive observations go out of bounds of the 99% confidence level (T_d). Figure 5.2 shows the example of the run length when monitoring the Q-statistic or SPE. The ARL is computed by averaging run lengths of various executions.

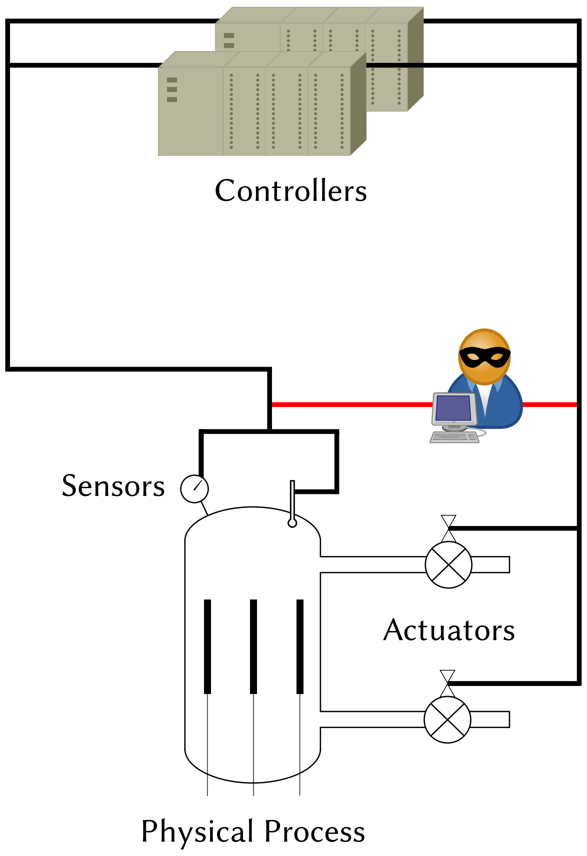


Figure 5.3: Example of a field network and used attack model.

5.3 Proposed approach

Figure 5.3 shows an example of a field network. At the core of the system resides a physical process, with a fixed number of sensors and actuators. These sensors and actuators are the input/output devices that controllers use to interact with the process. Controllers read process data from the sensors, and according to the control algorithm implemented in them, they decide what is the next step to be taken on the actuators. Once the actuators act on the process, the process reacts to the change and evolves. And as the process evolves, so do the sensor readings. Then, sensor data is fed to the controllers again, thus repeating the process.

However, the communication between process controllers and sensor/actuators is often performed over insecure transmission lines, frequently using unencrypted, unauthenticated, legacy protocols. Thus, it is possible for an attacker to interact with the communication, performing Man-in-the-Middle (MitM) attacks or Denial of Service (DoS) attacks.

This can lead to situations where the data fed to the controller is not the genuine value read by the sensors, or that the actuators receive data that was not sent as such by the controllers.

In the work presented in this Chapter, we apply MSPC over a simulated industrial process, the Tennessee-Eastman, to detect anomalies and diagnose their cause distinguishing between natural (disturbances) and human induced (attacks) factors.

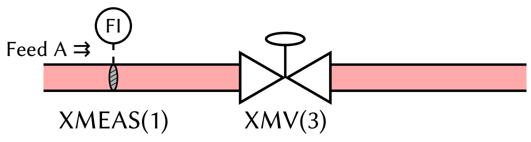
5.3.1 Attacks and Disturbances in the Tennessee-Eastman process

The Tennessee-Eastman (TE) process has already been introduced in Section 3.2.5. As already stated, the TE is a well-known benchmark model, modeled after a real chemical process. We also noted that, while more recently it has become a prominent choice among security research works, the original aim of the TE model was to evaluate different process control approaches.

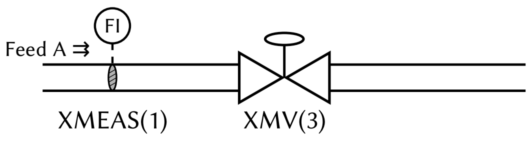
For this purpose, the TE model has 41 measured variables (XMEAS), 12 manipulated variables (XMV) and 20 process disturbances (IDV) implemented. On the one hand, XMEAS variables correspond to sensor readings, where 22 of them are measured in real time and the rest are variables measuring the amount of chemical components at different stages of the process, which are gauged in fixed intervals. On the other hand, XMV variables represent the actuators. Therefore, the XMEAS values are read by the controllers, and then controllers interact with the process by sending commands to the XMV actuators. In a physical process, the aim of the control strategy is to keep field values as close to a set of previously defined setpoints, where the process is supposed to work under certain desirable conditions, such as maximum throughput, minimal cost or specified output purity. Process disturbances are unexpected and undesired changes in process conditions that can affect the process normal operation. A good control algorithm will withstand the impact of a disturbance and keep the process running as close to the control setpoints as possible. In the TE model, IDV disturbances allow to evaluate the performance of different control strategies against adversities. For a full description of the variables and disturbances, refer to the original Downs and Vogel paper [42] introducing the model.

Out of the modelled disturbances, IDV(6) is one of the most difficult to handle. It simulates a loss of reactant in an input feed (Feed A). As A is a necessary chemical reactant to produce the product, the process stops the production due to a too low liquid level in the stripping column. The input flux of feed A is measured by XMEAS(1), whereas XMV(3) is the manipulated variable that controls the valve of feed A.

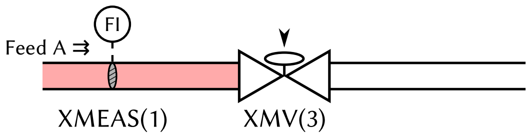
Figure 5.4 shows three different scenarios for Feed A. The reactant flows into the input



(a) NOC



(b) IDV(6)



(c) Closing XMV(3)

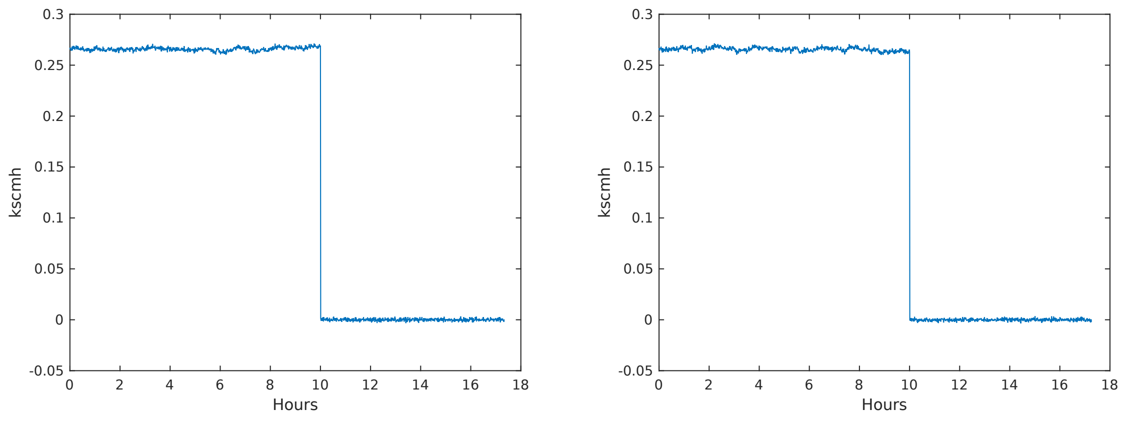
Figure 5.4: Stream 1 (Feed A) of the TE process under different conditions

pipe from the left, XMEAS(1) measures the quantity of reactant that goes through it and XMV(3) controls the flow that goes into the plant. Figure 5.4a, shows Feed A under Normal Operation Conditions (NOC). There are no contingencies and the A reactant flows into the plant and thus XMEAS(1) reports normal readings. Figure 5.4b depicts an IDV(6) disturbance: the Feed A is empty (probably due to a problem upstream) and therefore, no reactant enters the pipe. Opening the valve will have no effect in increasing A levels. Hence, XMEAS(1) will equal to zero. Finally, Figure 5.4c illustrates the case where an attacker closes the XMV(3) valve. The reactant flows normally to the pipe, but as the valve is closed, Feed A does not reach the plant. As there is no running flow, XMEAS(1) will read zero values again.

Therefore, it is to be expected that attacks on closing the valve XMV(3) and the existence of disturbance IDV(6), will affect similarly to XMEAS(1), and thus, to the process.

Figure 5.5 shows both situations. When monitoring XMEAS(1), there is almost no difference between IDV(6) and an integrity attack on XMV(3) where the attacker commands closing the valve controlling feed A, as the flow decreases abruptly in both cases. Both the disturbance and the attack occur at the tenth hour. After 17 hours and 43 minutes, the process shuts down in both cases as the stripper liquid level becomes too low to continue plant operation.

Hence, we can conclude that having a process disturbance and a potential attack on a process variable that react almost identically provides a sound setup to test the performance of techniques that try to distinguish them.



(a) IDV(6) **(b)** Attack on XMV(3)

Figure 5.5: Comparison of the evolution of XMEAS(1) under disturbance IDV(6) or an integrity attack on XMV(3).

5.3.2 Adversary modelling

The adversary and attack models considered in this scenario are the ones proposed by Krotofil et al. [88]. We consider that the adversary is able to read and manipulate network traffic, between controllers and the physical process as depicted in Figure 5.3. Therefore, the attacker is capable of manipulating input data both at the controllers' (forged XMEAS data) and/or the physical process' (forged XMV data) end, performing an integrity attack.

Following the model of Krotofil et al. [88], we consider an attacked variable $Y_i'(t)$ at time $t, 0 \leq t \leq T$ as follows, where T is the duration of the simulation and T_a the arbitrary attack interval. An integrity attack is defined as follows:

$$Y_i'(t) = \begin{cases} Y_i(t), & \text{for } t \notin T_a \\ Y_i^a(t), & \text{for } t \in T_a \end{cases} \tag{5.4}$$

where $Y_i^a(t)$ is the modified variable value injected by the attacker.

Similarly, during the DoS, the attacker effectively stops communication, and no communication reaches the actuator or the controller. Krotofil et al. [88] define as a DoS attack starting at t_a as:

$$Y_i^a(t) = Y_i(t_a - 1) \tag{5.5}$$

where Y_i^a is the last value received before the DoS attacks.

5.4 Experimental results

In order to evaluate our approach, we conduct a set of experiments where the randomized TE model is run ten times per anomalous situation. The model we used for the set of experiments is the DVCP-TE model presented by Krotofil and Larsen [87], freely available on Github[10]. The time length of each simulation is 72 hours, except in the cases where the process shuts itself down due to safety constraints. For each simulation hour, variable data is recorded 2000 times, that is, every 1.75 seconds. The training dataset consists of 30 runs, and this data is used to build the MSPC model and establish the control limits of the D and Q statistics.

All anomalies start at the 10th hour of simulation. For each of the anomalous situations, we calculate the ARL, that refers to the averaged run length across different executions. As previously stated, an event is flagged as anomalous when three consecutive observations surpass the 99% control limit.

Once an anomaly is flagged, oMEDA charts are computed for the set of the first three observations that surpass control limits in each of the ten runs in either of the two control charts (monitoring D and Q-statistic).

For each anomalous event two plots are created, one with real process data (data the process receives and sends), and the other with controller level data. Both data sets will be identical in case of an attack free environment. But, in the case of attacks, both data sets will diverge.

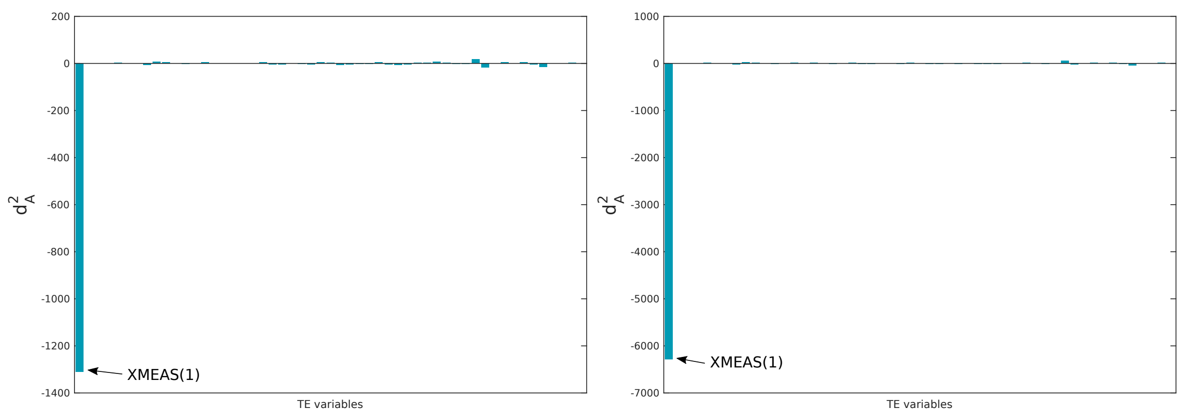
For the analysis of the process data, and plotting purposes, we used the MEDA toolbox [22].

We set five different scenarios based on the similar anomalous cases presented in Section 5.3.1: *a)* Disturbance IDV(6), *b)* Integrity attack on XMV(3), *c)* Integrity attack on XMEAS(1), *d)* DoS attack on XMEAS(1) and *d)* DoS attack on XMV(3).

Disturbance IDV(6)

Figure 5.6 shows the oMEDA charts for the case of the IDV(6) disturbance. As the A feed level, measured by XMEAS(1), is much lower than expected, XMEAS(1) stands out as the major contributing variable to this anomaly in both levels. As there is no interference from an attacker, the oMEDA charts are the same at both ends.

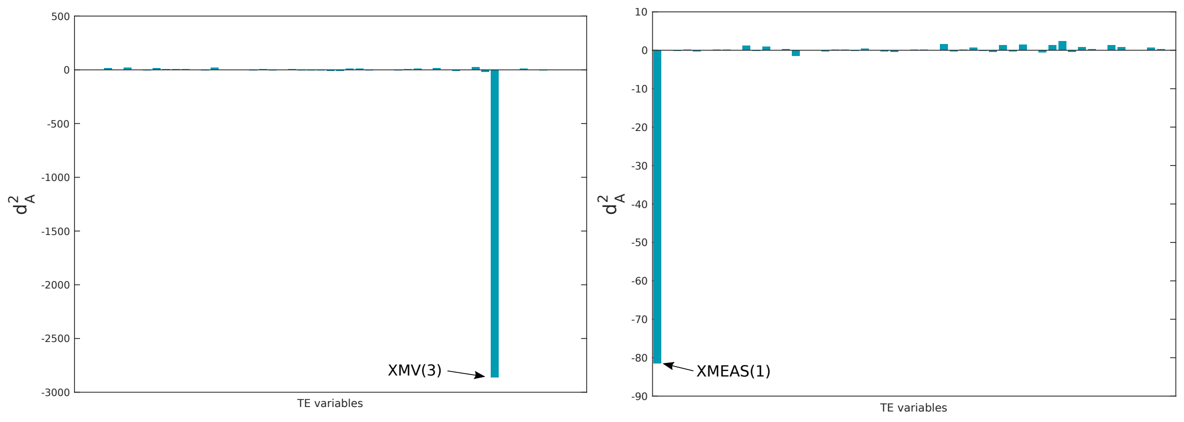
[10]http://github.com/satejnik/DVCP-TE



(a) Field-level **(b)** Controller-level

Figure 5.6: oMEDA plots of process disturbance IDV(6)

Integrity attack on XMV(3)



(a) Field-level **(b)** Controller-level

Figure 5.7: oMEDA plots of an integrity attack to XMV(3)

Figure 5.7 shows the oMEDA charts for the case of where the attacker performs an integrity attack to XMV(3), setting its value to zero and effectively closing the valve of feed A. In this case, from the controllers point of view, the anomaly is similar to the one with IDV(6). It is when we look at process-level data that we see that the real concerned variable is not XMEAS(1). Rather, the attacker is manipulating XMV(3) to perform the attack.

Integrity attack on XMEAS(1)

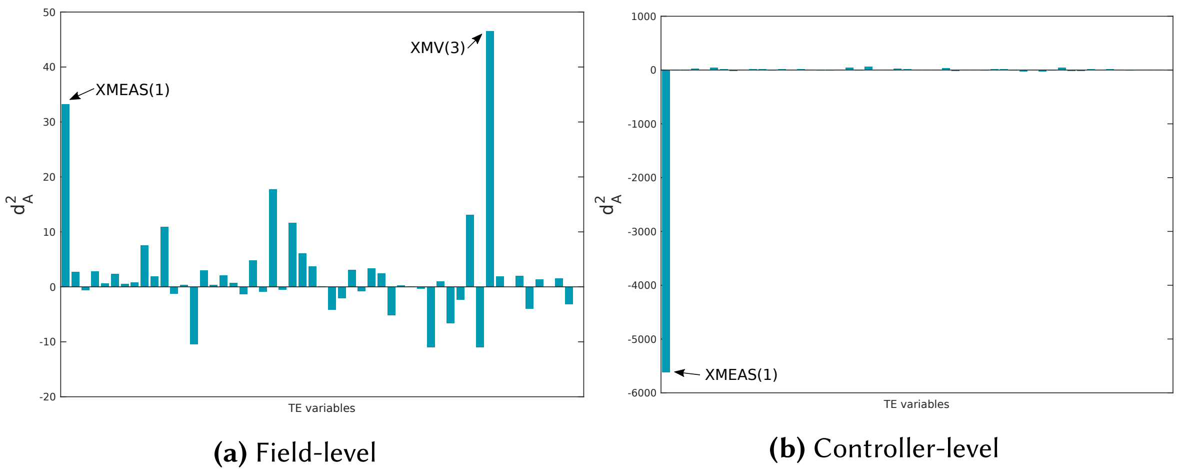


Figure 5.8: oMEDA plots of an integrity attack to XMEAS(1)

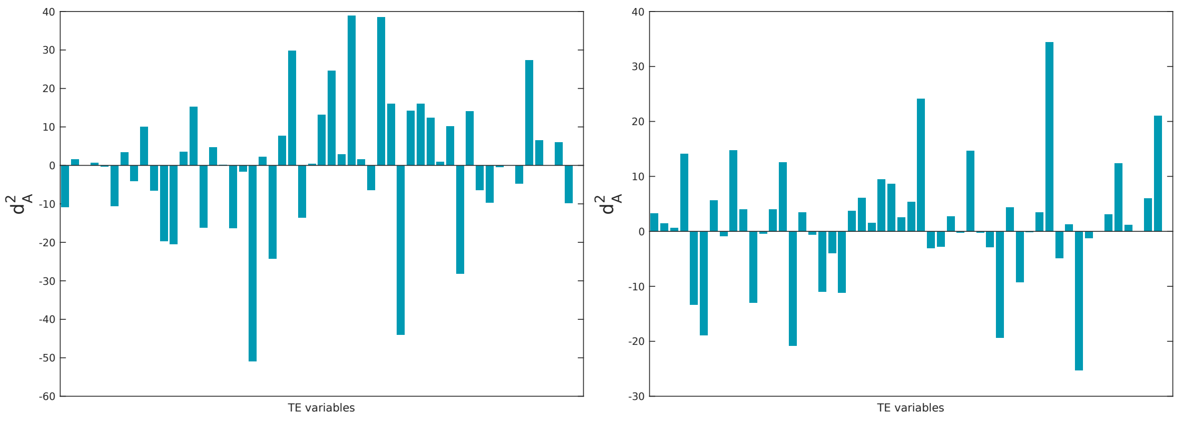
Figure 5.8 shows the oMEDA plots of an scenario where the attacker manipulates the XMEAS(1) variable and sets it to zero. Therefore, the controller receives the information that there is no flow in Feed A. That is why the XMEAS(1) value from the controller point of view is lower than usual, because the attacker has set it so. As the control algorithm tries to tackle the situation, it opens XMV(3) more, and thus allowing more reactant A to enter the process. From the process point of view, that is the reason XMV(3) and XMEAS(1) have higher values than usual.

Denial of Service attack on XMEAS(1)

oMEDA plots for a DoS attack on XMEAS(1) are shown in Figure 5.9. In this scenario, the process keeps receiving a constant value, previous to the attack. Neither of the oMEDA plots show a variable that stands out clearly among others, let alone XMEAS(1).

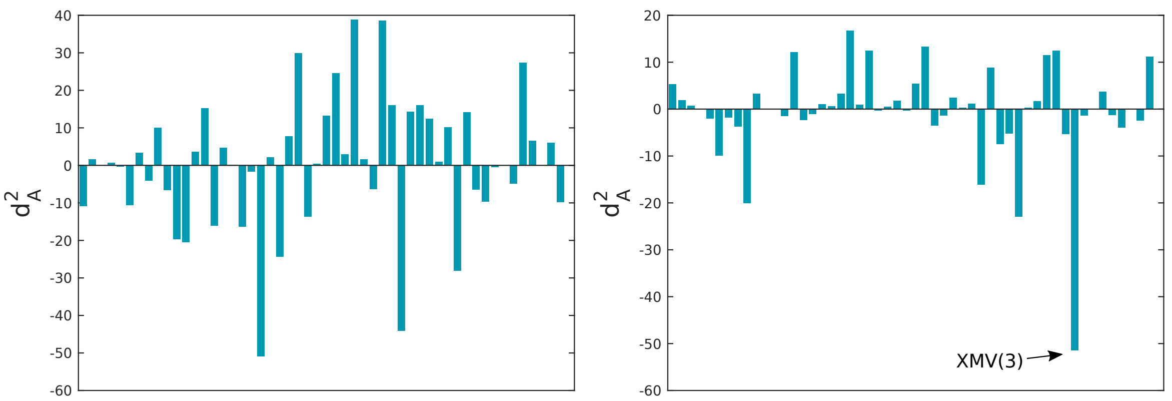
Denial of Service attack on XMV(3)

Figure 5.10 shows the scenario where a DoS attack is performed over the XMV(3) actuator signal. In this case, the attack can be diagnosed somehow from the controller point of view, as it sends a close command to the XMV(3) actuator (hence the large negative value of d_A^2 on XMV(3)), but it never reaches the process. However, as with the previous DoS case, the diagnosis of the problem is not straightforward.



(a) Field-level **(b)** Controller-level

Figure 5.9: oMEDA plots of a DoS attack to XMEAS(1)



(a) Field-level **(b)** Controller-level

Figure 5.10: oMEDA plots a DoS attack to XMV(3)

5.4.1 Discussion

Our approach successfully detects all the tested anomalous situations of disturbances and attacks. In this case, we can evaluate the performance of our approach based on two main criteria: correct diagnosis, and detection time.

For the diagnosis part, different oMEDA plots have shown that when diagnosing an anomaly, controller-based readings –on which traditional MSPC has relied on– are not enough to do so correctly. Both integrity attacks and the process disturbance are diagnosed in a very similar way, in a manner that it is not feasible to distinguish what caused the anomaly. To address this matter and be able to correctly diagnose, it is necessary to mea-

Scope	IDV(6)	XMV(3) int.	XMEAS(1) int.	XMV(3) DoS	XMEAS(1) DoS
Field-level	5.25s	5.25s	5.25s	1h 9.88m	1h 19.14m
Controller-level	5.25s	5.25s	5.25s	1h 6.73m	1h 19.28m

Table 5.1: Average Run Length (ARL) for each anomalous scenario

sure both field and controller level variables. DoS attacks have no clear diagnosis, and might require more data to correctly identify their cause.

As for the detection time, Table 5.1 shows the necessary time to detect an anomalous situation. Considering that the sampling time is 1.75s, integrity attacks and the IDV(6) are detected as soon as possible, as the three out-of-bounds readings are consecutive to the anomalous event. The system needs to verify that there are three successive out-of-bounds readings, hence the 5.25s delay. In systems where the sampling time can be increased, the ARL will decrease.

In the case of DoS attacks, the ARL is significantly longer than with integrity attacks or process disturbances and their detection takes longer than an hour. However, DoS attacks do not stop the process from working, and it can be considered a resilient against DoS. This aspect was already confirmed by Krotofil and Cárdenas [86].

5.5 Conclusions of the Chapter

We have presented a process-independent approach to detect and distinguish process disturbances from related attacks. Unlike previous approaches, it is not a process-dependent approach and it is able distinguish between disturbances and attacks.

Our methodology is based on MSPC for anomaly detection and oMEDA plots for anomaly diagnosis. We have used the popular Tennessee-Eastman process to experimentally evaluate our approach.

Distinguishing process disturbances and low level attacks at the field level is a complex task, especially if all controller's I/O are to be considered compromised.

We extended the traditional MSPC model to monitor both controller and process level variables. Often, manipulated variables are also measured by an external sensor, so this approach is feasible in these environments. This scenario, would also complicate the work of an attacker, as it would need to forge both the target manipulated variable and the associated measured one to avoid detection.

When analyzing process disturbances or integrity attacks, the oMEDA plots clearly show the implicated variables. In the case of DoS, detection time is significantly longer and the diagnosis with oMEDA might not be related to the attacked variable.

However, some limitations exist; it is not always feasible to monitor field-level data directly to improve diagnosis, and the ARL in the case of DoS attacks is still high. To overcome these limitations and improve attack traceability, it is possible to add more information to the model.

In the case of industrial settings, a promising source of additional information is the one created at the network level (packets, flows, logs etc.). This aspect is covered in the next Chapter.

Chapter 6

A Large-Scale, Field and Network-level Anomaly Detection System based on Multivariate Analysis

Contents

In this chapter we combine the techniques and tools presented in Chapters 4 and 5 among other components to a build a holistic Anomaly Detection System (ADS) for Industrial Networks (IN) that monitors both physical or field data along with data belonging to the cyber or network domain at the same time. To ensure the proposal's scalability, the ADS is implemented over Apache Spark.

6.1 Data creation and aggregation

Section 5.2 introduced the methodology of Multivariate Statistical Process Control (MSPC). There, it was described that, in order to be able to apply MSPC, it was necessary to have an $X = N \times M$ matrix comprising the recorded observations of the system. When applying this approach to field data, there is no need for pre-processing, as this type of data already has the correct format, a scalar value associated to each of the M variables at the given moment were the observation was recorded.

However, network data is not arranged in the same manner. Network data comes in form of logs, flows and packets, and it is not possible to directly insert this data into the MSPC model, as the data does not come in the form of discrete scalars. It is necessary to find the network-level equivalent of having a scalar value representing the state of a network variable.

One approach is compiling general network flow statistics and registering them in a variable. For instance, it is possible to count the number of registered flows in a given time frame, or the average number of packets per flow in the same time frame. This way, it is possible to translate some of the information given in a set of flows to a summarizing value.

Moreover, we can extend this approach by using features-as-a-counter [19, 20] which allow transforming more complex types of network data into a set of quantitative values. In this approach, network variables are rearranged into a set of counters that yield the sum of a given occurrence in a time window. For instance, if a server named *Serv1* registers two log entries containing authentication errors in a minute, an associated feature-as-a-counter variable *serv1_auth_err* that keeps track of this events in a minute long time frames would have a value of two. As network data is redefined into a new set of variables, the correct definition of feature-as-a-counter variables has an utmost importance, as the anomaly detection performance of the MSPC model is directly related to the ability of these variables to describe network behavior. Non-exhaustive or incorrect feature-as-a-counter variable definition can make anomaly detection and diagnosis more problematic.

Next, we provide a general description of the used variables for anomaly detection, both directly used from process readings and inferred from network behavior.

6.1.1 Data types

Field data

For creation of the field data, which is formed by process readings, we use the Tennessee-Eastman (TE) process, first presented in Section 3.2.5 and later used as a use case for field-level anomaly detection and diagnosis throughout Chapter 5. As in the previous case, we use the DVCP-TE version and the control approach proposed by Larsson [93].

For this model, we monitor the first 22 XMEAS readings, discarding XMEAS values from 23 to 41. This last group relates to the concentration levels of different chemicals and do not have any impact on process control, therefore, they are not useful for diagnosing anomaly causes.

The monitored 22 measurement variables are divided in the following types:

- Ten Flow Indicators (FIs).
- Five Temperature Indicators (TIs).
- Three Pressure Indicators (PIs).
- Three Level Indicators (LIs).
- One Power Indicator (JI).

The XMV manipulated variables, are classified as follows:

- Eleven valves.
- One Speed Controller (SC).

Flow statistics

We compute a set of statistics related to the network flows registered in a given time frame. These features compress a part of the information of network flows into scalar values and have been set according to the nature of Industrial Networks (INs). As INs tend to be static and stable along time, these statistics are expected not to be very volatile. These metrics will not perform well on most Information Technology (IT) networks, as they tend to have a more seasonal behavior and the values will vary significantly between readings, yielding more false positives.

- Number of registered flows.
- Number of unique source IP addresses.
- Number of unique destination IP addresses.
- Number of registered input packets.
- Number of registered input bytes.
- Number of registered output packets.
- Number of registered output bytes.
- Number of packets in the most active flow.
- Number of bytes in the most active flow.

Flow anomalies based on whitelisting

Apart from the flow summaries, we also include flow tags as an additional information source for the model. For this end, we use the whitelisting approach presented in Chapter 4. However, in this case, instead of working with the visualization systems, we process the text-based logs. In essence, we count the number of occurrences of each of the alerts the system generates during a given time frame, a case of feature-as-a-counter.

As with the flow statistics, these metrics will be most effective when used for INs. Whitelisting is a recommended security practice for INs [7] but it might prove unsuccessful for IT networks. These are the tracked variables:

- Number of anomalous flows. That is, the number of flows where every communication between the hosts that compose it is forbidden.
- Number of flows with incorrect protocol. The number of flows, that even if the communication between hosts is allowed, the used protocol is not correct. For instance, if a whitelisted Transmission Control Protocol (TCP) flow is registered using User Datagram Protocol (UDP).

- Number of flows with incorrect service port. The number of flows that link two hosts that are allowed to communicate, but using a different service. For example, a SSH connection between hosts where only Modbus/TCP traffic is whitelisted.
- The number of legitimate flows with a significantly lower or higher packet throughput when compared to its whitelisted equivalent.

6.1.2 Network and field data aggregation

We have covered the different types of data that will be added to the MSPC model. However, even if the network information is summarized in a set of scalar values, it is necessary to merge both field and network data into a unified $X = N \times M$ matrix where each row has records for M variables, where network and field-level variables are considered.

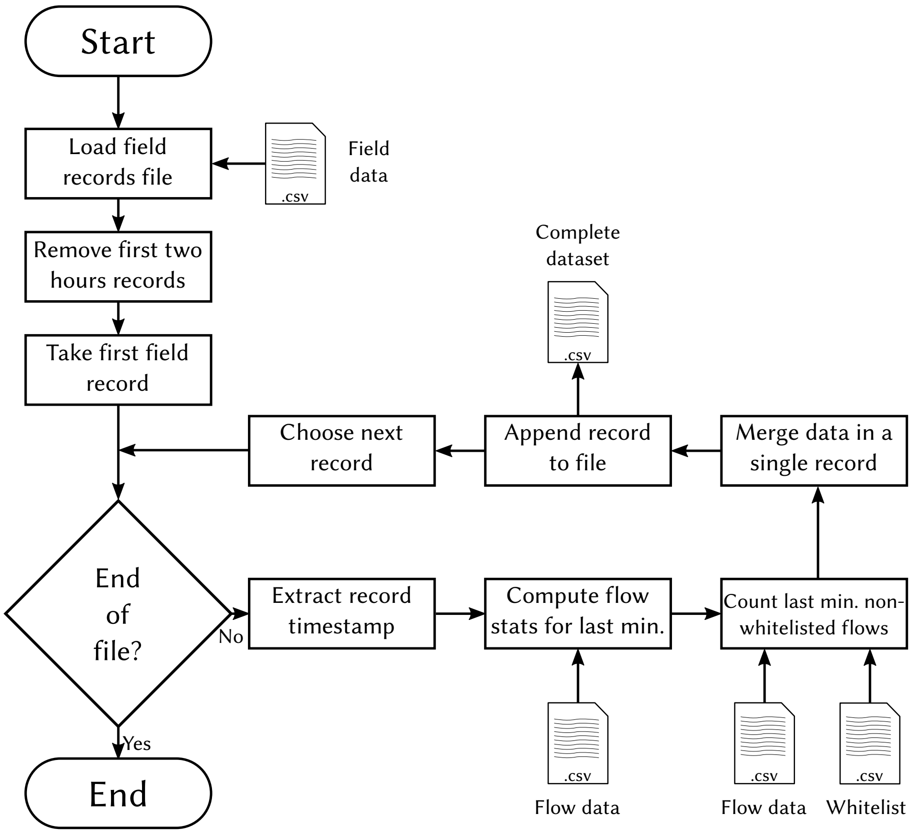


Figure 6.1: Flowchart of the data merging process

Figure 6.1 shows the process of merging field and network data in a single dataset. We consider that field data is polled at a higher frequency than network data. Most Industrial

Control Systems (ICSs) poll data multiple times every second, whereas network data is created in longer intervals, such as several seconds or minutes. For every record in the Comma-Separated Values (CSV) file containing field data, we extract the timestamp, and compute the network-level features for the events registered in the previous minute. In other words, we compute the statistics and count the non-whitelisted occurrences for the network flows that ended at most a minute from the registered field timestamp.

As the TE process needs stabilization when starting before reaching normal operation, the first two hours of process execution are discarded when building the dataset.

6.1.3 Detecting anomalies

Once a unified dataset has been built, the data can be used to build and apply a MSPC model over it. Figure 6.2 depicts the data processing procedure, divided in two phases: the training phase and the anomaly detection or monitoring phase.

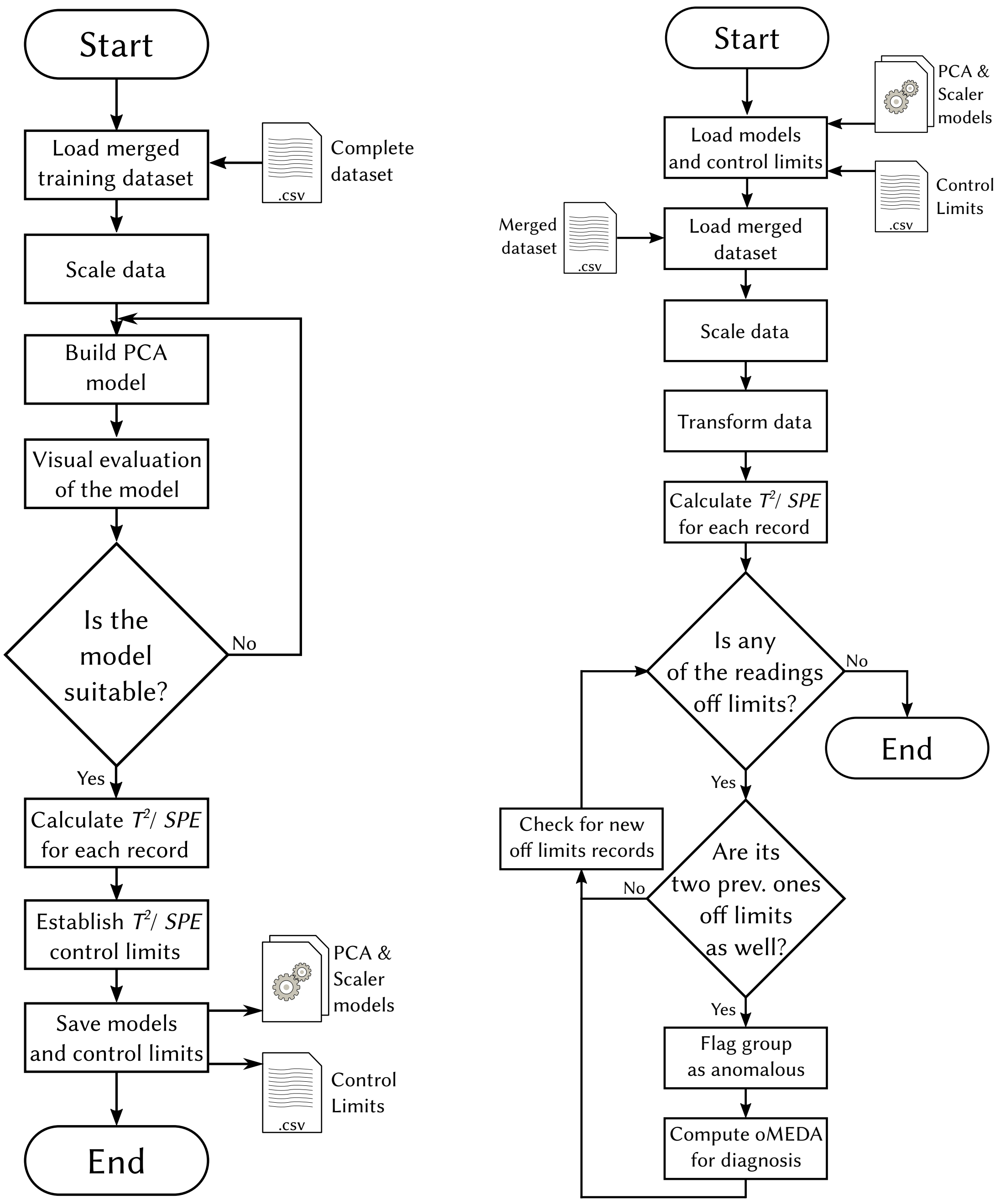
When describing MSPC in Section 5.2, we already mentioned that MSPC has a two-phase approach for anomaly detection. In the first phase, a Principal Component Analysis (PCA) model is built over scaled data (zero mean and unit variance), and the control limits for D (also known as Hotelling's T^2) and Q (or SPE) statistics are set. In the second phase, incoming data is transformed according the created models and we check whether three or more consecutive observations are out-of-bounds. If they are, we compute oMEDA plots [18] to diagnose the cause of the anomaly.

Before applying PCA for data transformation, all data is scaled to zero mean and unit standard deviation. The PCA settings and the scaling properties used for model building are stored. That way, when evaluating incoming data, it is not necessary to recompute them again.

6.2 Data processing framework

As exposed in Section 1.1, both network monitoring and field data processing can be considered a Big Data problem. We mentioned that Big Data complexity evolves from three different data qualities: volume, variety and velocity, collectively known as the three V-s. PCA is able to handle highly dimensional datasets, and as long as we are able to translate raw network data into quantitative features, MSPC is highly scalable in this sense.

To address data volume and velocity, we have implemented our approach on top of


Start
Load merged training dataset
Complete dataset
.csv
Scale data
Build PCA model
Visual evaluation of the model
Is the model suitable?
No
Yes
Calculate T^2/ *SPE* for each record
Establish T^2/ *SPE* control limits
Save models and control limits
PCA & Scaler models
Control Limits
.csv
End
Start
PCA & Scaler models
Load models and control limits
Control Limits
.csv
Merged dataset
.csv
Load merged dataset
Scale data
Transform data
Calculate T^2/ *SPE* for each record
Is any of the readings off limits?
No
End
Yes
Check for new off limits records
Are its two prev. ones off limits as well?
No
Yes
Flag group as anomalous
Compute oMEDA for diagnosis

(a) Training phase **(b)** Anomaly Detection phase

Figure 6.2: Flowcharts of the Training and Anomaly detection phase

Apache Spark [156, 157]. We introduced Apache Spark in Section 2.1 of this dissertation. Spark is a Big Data framework whose key feature are Resilient Distributed Datasets (RDDs), which allows in-memory, fault-tolerant data partitions across a set of nodes in a cluster. RDDs allow users to perform different operations on large datasets in a fast manner. Spark is a batch-oriented framework, where all operations are performed on data at rest. However, it also has a streaming interface, where incoming data is processed on micro-batches. Its core is written in Scala, although it also has programming interfaces in Scala, Python, Java and R. For this development we used the Python Application Programming Interface (API) of Spark, also called *Pyspark*. This allows the usage of native Python libraries for handling tasks not present in Apache's core, such as plotting, while also leveraging the data processing abilities of Spark. The implementation has been done on top of Spark 2.0.1.

In detail, most of the implementation has been done in top of Spark's MLlib Machine Learning library, using its pipeline API. By implementing the necessary functionalities for PCA-based MSPC we have effectively created a tool set that can also be used for multivariate data exploration. As such, it contains a subset of the features of the MEDA Toolbox [22], used for data processing in Chapter 5.

6.3 Experimental setup

In order to experimentally validate our anomaly detection approach, we design an experiment network and a set of scenarios that will allow the evaluation of our proposal.

6.3.1 Experiment network

Figure 6.3 shows the network topology of the experiment network. Out of the nine nodes of the network, six are the ones that comprise the industrial network. The rest are communication devices or network data collectors. For the industrial domain, there is a simulation node, containing the simulation of the physical process and its control, two control servers and three Human Machine Interfaces (HMIs).

Figure 6.4 depicts the main communication flows between the nodes of the industrial networks. Communication between the control servers and the simulation nodes has been carried out with Modbus/TCP. This communication has been implemented using the Pymodbus library[11], where the control servers act as masters (or clients) and the simulation node as a slave (or server). Control servers poll field data periodically from the simulation node, starting the communication, and they later store it in the servers. The HMI–control

[11] https://github.com/bashwork/pymodbus

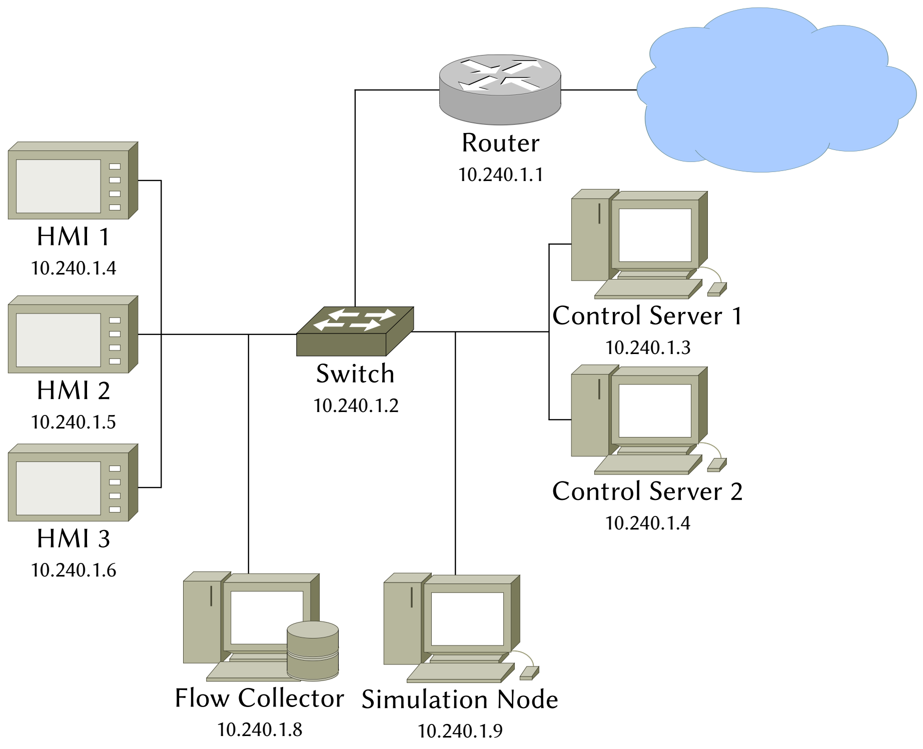


Figure 6.3: Network topology of the experiment network

server communication is implemented using the OLE for Process Control–Unified Architecture (OPC-UA) protocol, on top of the FreeOPCUA library [12]. Control servers acts as OPC-UA servers while HMIs are clients that request process data stored in the control server, to later serve it to human operators. HMI1 and HMI3 poll data from Control Server 1 and 2, respectively, while HMI2 polls data from both servers.

The simulation node is responsible for generating field-level data that is later queried by the control servers. The simulation platform used for this experiment, Simulink, does not support real-time execution on the simulation node. However, there are available tools that enable this approach. The Real-Time Pacer for Simulink [13], is one of those tools. It slows down simulation speed, down to the clock speed of the host Operating System (OS). This way, each simulation time frame is synchronized with the equivalent time frame: i.e. a simulation second lasts as long as a second in the simulation node clock.

Moreover, Simulink has basic networking abilities such as sending and receiving UDP

[12] http://freeopcua.github.io/

[13] http://es.mathworks.com/matlabcentral/fileexchange/29107-real-time-pacer-for-simulink

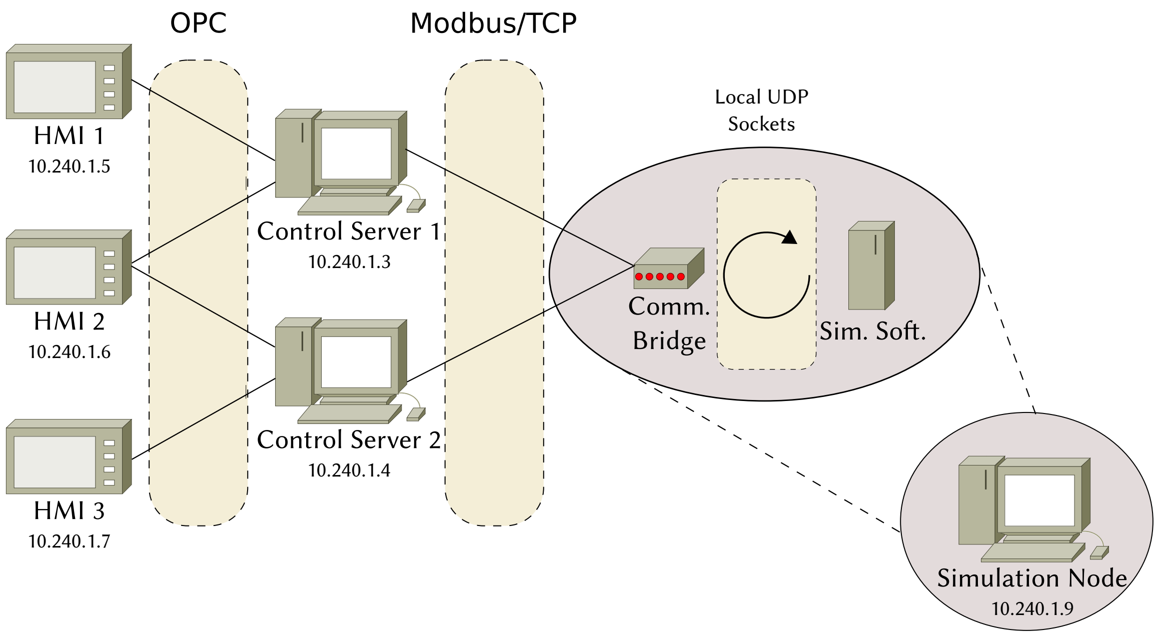


Figure 6.4: Main network flows in the experiment network

and TCP packets, but has no communication libraries that support industrial network protocols.

We modified the DVCP-TE model, (already mentioned and used in Sections 3.2.5 and 5.4) to allow variable reading and setting through local UDP sockets. Moreover, we developed a software module that acts as a bridge between Simulink and the control server. When polled by the server through Modbus/TCP, queries the current status of the TE process through local UDP sockets and returns the gathered values using Pymodbus. Similarly, if the bridge receives a Modbus/TCP command, sets the specified variable values to the obtained ones.

To prevent discrepancies between timestamps from different hosts, all nodes in the industrial network are synchronized using Network Time Protocol (NTP).

Field records are stored and timestamped in both control servers in CSV files, which they are updated each time they query the simulation node. The last entry in these files is the one that is served to the HMIs, so they always receive updated data regarding process status. As for the network data, the flow collector runs the NFDump [14] for collecting and processing of the Netflow V5-compatible flow information that the switch generates. To ensure that in a minute frame we register all possible flows including the longer ones, we set the flow timeout

[14] http://nfdump.sourceforge.net/

to one minute. Control servers poll data from the simulation node every two seconds.

6.3.2 Scenarios

After the model building phase, our approach has been evaluated in five different anomalous scenarios. All scenarios are eight hours long, and the anomalous situation is induced exactly at the start of the fifth hour.

For the model training, a 24h capture was used to set the D and Q statistics control limits. No unknown sources of variations where found in this training dataset, so no outliers where removed.

Process disturbance

In this setup, we simulate process disturbance IDV(6), which has already been explained in Section 5.3.1. It is worth mentioning IDV(6) is set at a field level, no abnormal network behavior is induced at this point.

Network Denial of Service

Denial of Service (DoS) remains one of the biggest challenges that INs face, as field controllers and other specialized ICSs have little computing power to process and answer to a high number of requests, especially if the requests are crafted to drag as much resources as possible from the target.

As in the case presented in Section 4.3.3, we consider that HMI3 has been compromised by an external attacker, or it is being used by a malicious user. As such, the attacker starts a SYN flood DoS attack against the OPC-UA port (4840) of Control Server 2.

In this attack, the attacker starts several TCP handshakes with the target, but when compared to a normal handshake, the attacker does not send the final acknowledgement (ACK) packets. The target waits for these acknowledgements and in the meantime, leaves the connections half-open. These half-open connections drag resources from the client and can cause to effectively leave the target out of service, especially in the case of devices with low computing power.

The SYN flood attack has been implemented using a Python script using the Scapy[15] library for packet manipulation.

[15] https://github.com/secdev/scapy

Port scan

Port scanning is the action of enumerating services present in a host or network by probing different ports. Though they are widely used for auditing and analyzing IT networks, port scans can have serious consequences over INs, as the usage of malformed packets in these scans, and the relatively high volume of created traffic can affect several ICSs' availability and lead to unexpected scenarios due to inconsistent ICS responses against port scans [44].

For the recreation of this scenario, we use the Nmap[16] popular port scanner, running with the default options.

Command from malicious agent

Modbus/TCP is a legacy protocol that lacks authentication measures, and therefore, any node can poll data from a Modbus slave, or send commands to it.

In the last scenario, we consider a malicious agent in control of HMI3 who sends the command to close the XMV(3) valve, eventually forcing the process to stop working. We already covered in Sections 5.3.1 and 5.4 that when auditing field data from the controller's point of view, this attack is difficult to distinguish from the IDV(6) disturbance, as both yield a lack of reactant on XMEAS(1).

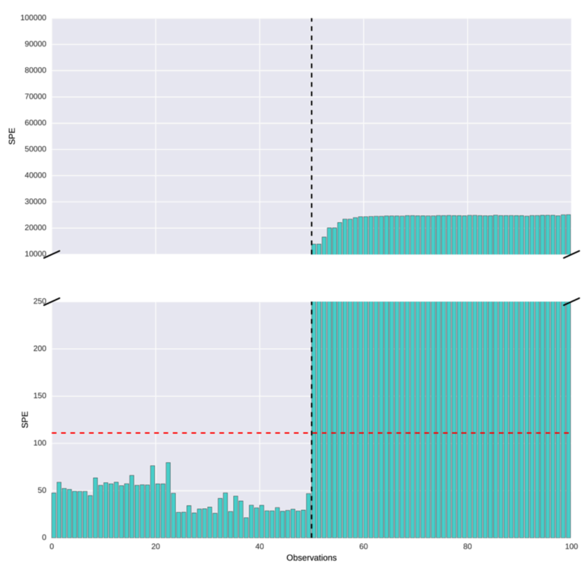
For this scenario, we develop a Modbus client on top of the previously mentioned Pymodbus that sends the closing command to the simulation node, overriding the value the control algorithm sets. The attacker compromises HMI3 and sends a control command directly to the controller in the simulation node. This would mean a direct command to the controller, without first passing through the control server, and thus, without leaving any log or trace in it.

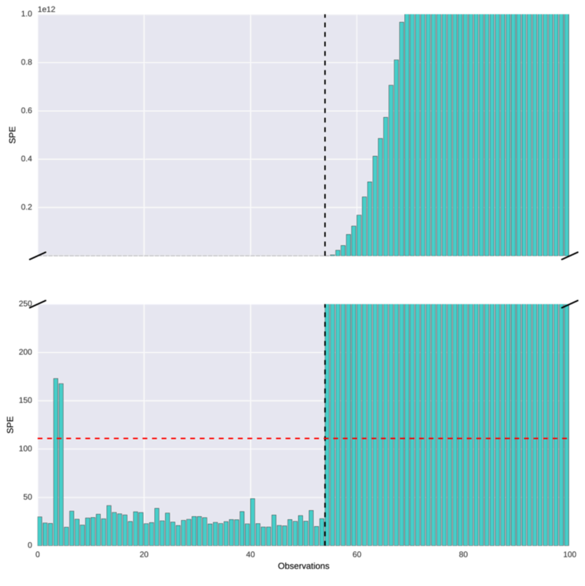
6.4 Experimental results

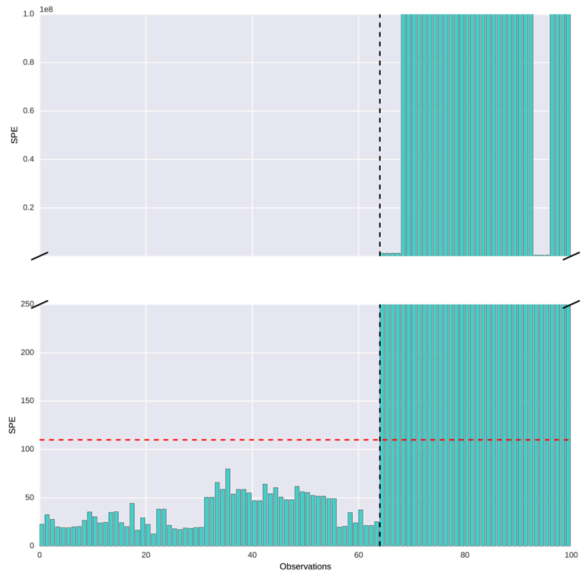
This section introduces the findings when evaluating our approach. All anomalies where detected when monitoring the Q statistic, as the D statistic did not yield any alerts.

Figure 6.5 shows the evolution of the Q statistic values, centered on the start of the anomaly. The horizontal red dashed line represents the control limit for Q, and the vertical black dashed line the moment where the anomalous situation (disturbance or intrusion) started. The vertical axes have been divided to illustrate the difference between the

[16]https://nmap.org/


(a) Process disturbance


(b) Denial of Service


(c) Port Scan

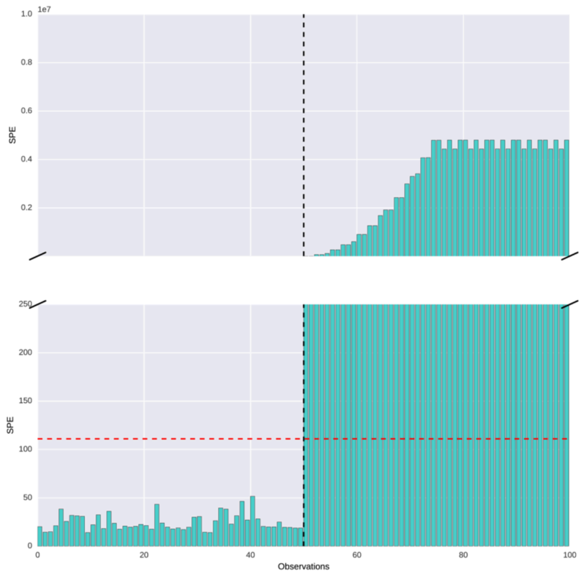

(d) Malicious Command

Figure 6.5: Evolution of the Q statistic in different scenarios, focusing on the anomalous situation. Q control limit is depicted with a horizontal red dashed line, while the starting point of the attack is shown with a vertical black dashed line

Q-statistic values before and after the start of the anomaly. As depicted, when an attack starts, Q values go immediately out-of-bounds, detecting the anomaly as soon as possible.

We now analyze the oMEDA plots of each of the anomalies. The oMEDA plots are only computed on the first out-of-bounds observation, leaving the rest of the observations out of the analysis. This way, it is possible to discern the contribution of the variables at the beginning of the anomaly.

Process disturbance

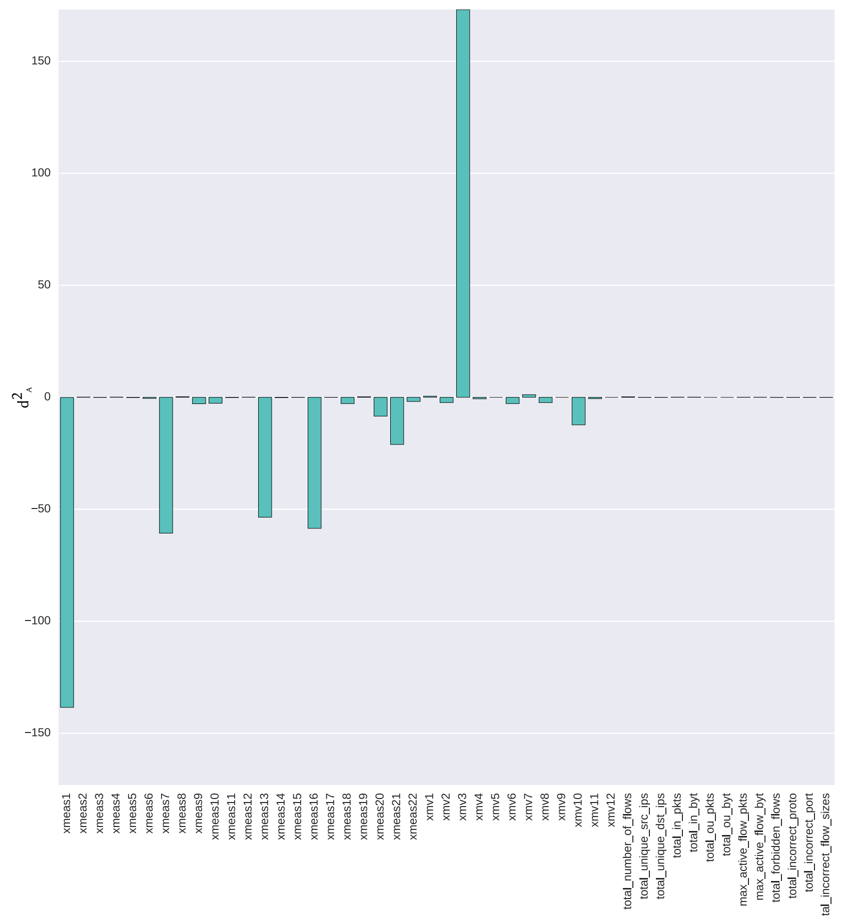


Figure 6.6: oMEDA plot of the process disturbance scenario

Figure 6.6 shows the oMEDA plot for the process disturbance. In this scenario, there is no abnormal network activity. The oMEDA shows that two variables have the most contribution for this situation: a low XMEAS(1) value and a high XMV(3) one. The low XMEAS(1) corresponds to a lack of reactant due to disturbance IDV(6). However, the control algorithm tries to counter the lack of reactant by opening the XMV(3) valve, hence its high value. As no significant contributions from the network variables is shown, it can be diagnosed that the anomaly occurred only at the field level.

Network Denial of Service

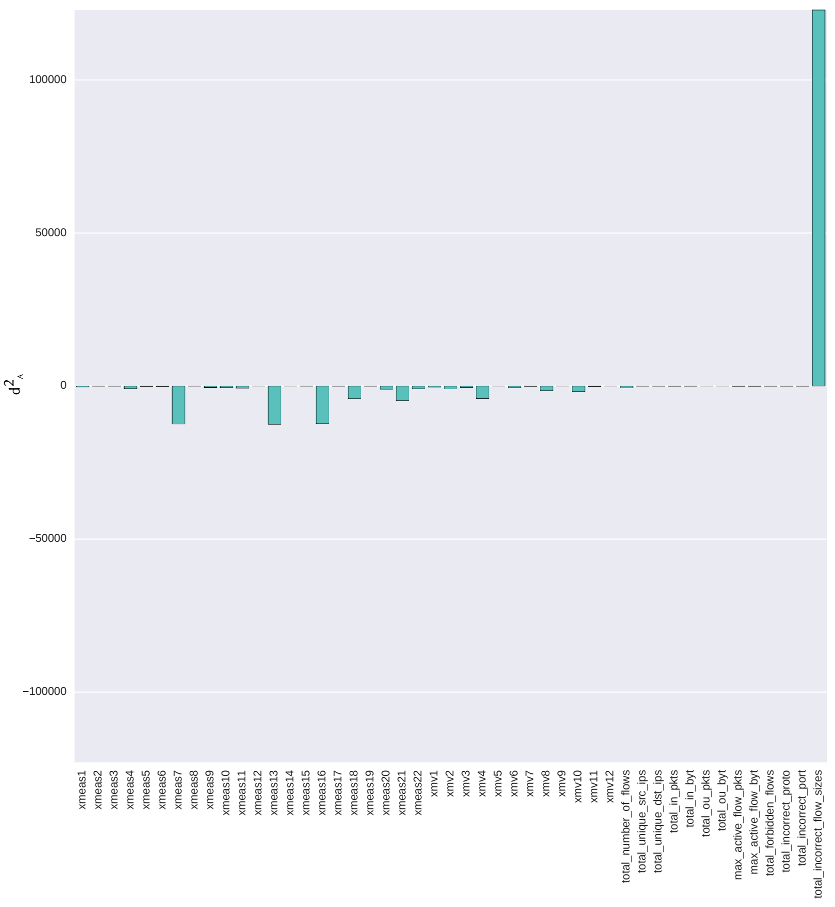


Figure 6.7: oMEDA plot of the DoS scenario

Figure 6.7 depicts the oMEDA flow where HMI3 starts a DoS attack against Control Server 2. As HMI3 regularly queries Control Server 2 to get field data, there are no alerts for forbidden flows. However, the size of the flows forming the SYN flood is different from the regular flows recorded in the whitelist between both hosts. Therefore, the oMEDA shows a large increase in incorrectly sized flows. As this intrusion only affects the network of the IN, field variables do not show an important contribution to the situation.

Port Scan

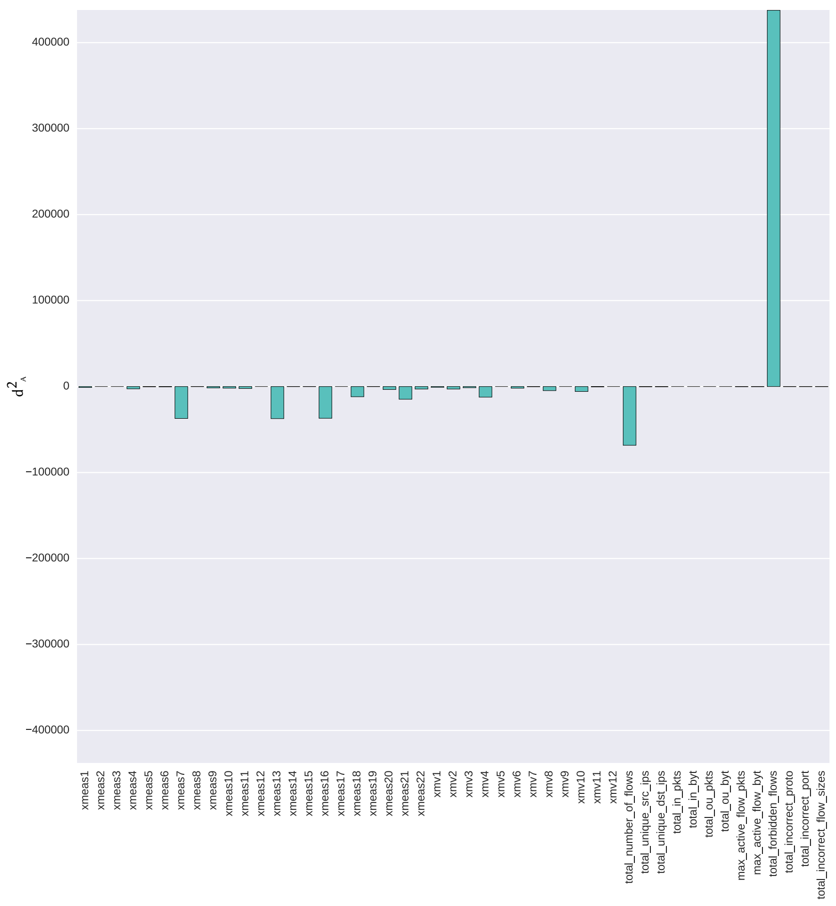


Figure 6.8: oMEDA plot of the Port Scan scenario

The oMEDA plot from Figure 6.8 depicts the effect of a port scan where HMI3 is compromised and executes Nmap. As the HMI3 has only whitelisted traffic towards Control Server 2, all other flows probing the rest of the hosts are classified as anomalous. Therefore, the variable which shows the highest impact is the number of anomalous flows. As with the previous case, field-level variables show little contribution to the anomaly.

Command from malicious agent

The diagnosis of the last scenario, where a compromised HMI3 sends a control command ordering to close XMV(3), is shown in Figure 6.9. It depicts a lower XMEAS(1) value, and as in the IDV(6) case, shown in Figure 6.6, the control algorithm tries to open the valve again, hence the high XMV(3) value. As no communication is allowed between HMI3 and

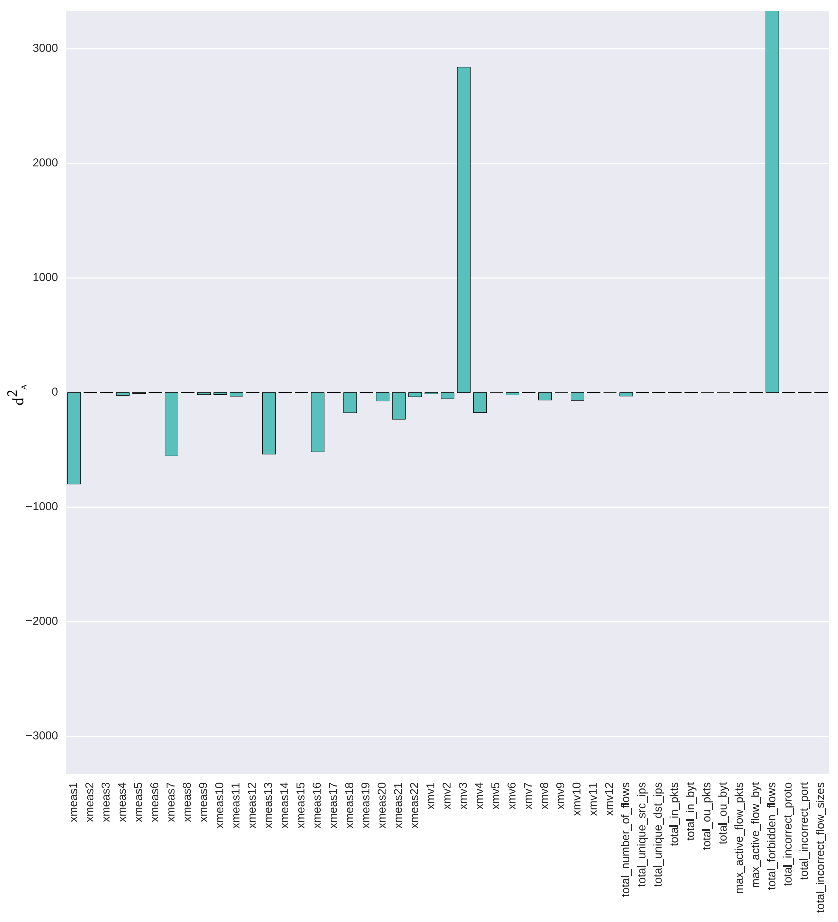


Figure 6.9: oMEDA plot of the malicious command scenario

the simulation node, the number of forbidden flows shows a great contribution, notifying process operators that the intrusion has been carried out by a command unit not allowed to communicate with the Modbus slave.

6.4.1 Discussion

The addition of network variables by extending traditional MSPC models allows the detection of other types of anomalies such as port scans, network DoSs and the case where an anomaly

	IDV(6)	**Proc. dist.**	**Network DoS**	**Port scan**	**Malicious command**
Run length	6s	6s	6s	6s	6s

Table 6.1: Run Length for anomaly detection using network variables

at the process level is caused by disrupting network security policies.

Furthermore, this extended MSPC model eases anomaly detection diagnosis, at least distinguishing the affected domains in case of an attack or a disturbance and the violated policies, if that is the case. It is to be expected that the development of new network-level variables will enhance anomaly diagnosis properties.

Table 6.1 shows the run lengths for the cases presented. As data is polled every two seconds and three consecutive out-of-bounds observations are needed to flag an anomaly, our approach detects anomalies in the minimum 6s time in all five cases.

6.5 Conclusions of the Chapter

This chapter presents a network and field-level Anomaly Detection System (ADS) that deals with data complexity. Scalability is achieved by implementing the data processing solution over Apache Spark, thus migrating some of the tools present in the MEDA toolbox to this framework. The ability to process heterogeneous data is achieved by using PCA and features-as-a-counter and some general statistics of a given time. High data creation rates as the incoming can be streamed, especially for the detection.

We have experimentally validated our approach in five different scenarios concerning network-level and field-level anomalies, along with a scenario where both levels were implicated. We did so by adding communication abilities to the DVCP-TE simulation model and building a test IN with real industrial traffic. The system has been implemented on top of Apache Spark to ensure its scalability to address larger datasets. The presented system has detected all anomalies promptly and has aided in the process of diagnosing them by showing the contribution of each of the variables to its cause.

Chapter 7

Conclusions, and Future Work

Contents

This chapter presents the final remarks and also identifies some future research lines for further development of the area.

7.1 Conclusions

This dissertation has been centered on the development of novel data-driven ADSs for INs.

We first performed a comprehensive literature review where we analyzed the field of large-scale, heterogeneous ADSs and their applicability to INs. We identified some gaps in current research and some room for future improvements.

Due to the necessity of conducting IN security research in a safe, faithful and reproducible environment, we presented a testbed that dynamically emulates network behavior and also simulates a physical process. Together, the testbed is able to emulate existing IN functionalities, based on the software Emulab and the Tennessee-Eastman (TE) process. Additonally, to support future research in the area, the testbed supports Sofware-Defined Networking (SDN).

Later, we introduced a visual flow monitoring system based on whitelisting and chord diagrams. Network flows are depicted in a chord diagram and highlighted according to its legitimate or anomalous nature. For this end, we develop time-based whitelists, where the traffic nature is registered in a given time-frame. When compared to traditional whitelists, with time-based whitelists, it is possible to detect anomalous network flow sizes. Historical data is stored in a scalable search server. To the best of our knowledge, this is the first security-oriented visualization for INs.

Next, we analyzed the performance of MSPC for anomaly detection at a field level. We concluded that MSPC is an effective methodology for anomaly detection, but process diagnosis can be a complex task in cases where the real process status is hidden from the operator. Moreover, detection times are sometimes long. As a data-driven approach, one of its main advantages is that no prior knowledge about the process is needed.

Finally, we extended the MSPC model to include network-level variables. This was achieved by summarizing flow data in a set of quantitative features, and using features-as-a-counter to include anomalous flow information. The approach has been validated by building an experiment network with real traffic and a simulated TE process which has been modified to include network communication abilities. This extended ADS is able to detect network and field level anomalies, along with intrusions that disturb both levels. Furthermore, by using oMEDA plots, it is possible to analyze the contribution of each of the monitored variables to the anomaly, allowing the diagnosis of the attack by inspecting the prominent variables.

7.2 Future Work

In this section we outline future research directions that can lead to additional contributions in the field of IN anomaly detection. This dissertation, though limited in scope, offers development opportunities that deserve the attention of the scientific community for further advances in the field. We now list the aforementioned opportunities arranged by topic. These opportunities can be seen as a direct continuation of the work presented in this dissertation, and therefore this future work outlines can be complemented with the research gaps identified in Section 2.5 of the Literature Review.

Testbed evaluation Once a testbed has been designed and implemented, the next step is the evaluation of its fidelity when compared to real installations. There are several evaluation metrics that fulfill this purpose. Siaterlis et al. [127] use the simulated model's execution time as a metric, when compared with its latency requirements. Reaves and Morris [122] identify metrics related to Modbus packet payloads. Finally, other proposals gathered by Holm et al. [61] compare their testbeds to the guidelines and standards published by prominent institutions [132]. However, it is difficult to compare different proposals, as the used evaluation metrics are not the same. Therefore, the development of a unified evaluation framework that will allow a common ground for testbed comparison would be a valuable contribution that would improve future IN testbed development.

Anomaly Detection in INs Anomaly detection in INs is an active field, and recently, the field-level, physics-based ADSs are gaining importance. In this sense, MSPC, as a data-driven, scalable methodology, offers a novel approach for anomaly detection. The identification of new quantitative variables can improve the efficiency of MSPC-based approaches, by adding new information to the model. Examples of possible information sources for anomaly detection include control server alerts, protocol parsers and ICS logs. The quantification of these variables by using the feature-as-a-counter approach can lead to a new research field where complex data can be transformed into simpler features for anomaly detection. Moreover, in a similar case to the previously mentioned testbed evaluation, it is necessary to provide a common ground for ADSs validation. The existence of *de facto* standards such as the TE process helps in this matter, but it is necessary to go a step further and provide a public set of attacks against INs and ICSs to help ADS assessment. In this sense, the development of stealthy and advanced attacks against a simulated implementation of a real process, such as the TE, is vital for the evaluation performance of ADSs.

Large-scale multivariate tool set When developing the large-scale ADS on top of Apache Spark, we implemented the necessary functionalities for MSPC. These functionalities can also be used for exploring and analyzing large multivariate datasets. Further development of this tool can lead to a fully fledged piece of software that would allow the exploration of large and heterogeneous datasets, in a similar manner to the MEDA toolbox [22] but for larger volumes.

Anomaly response in INs So far, this dissertation has been centered in the detection and diagnosis of anomalies in INs. The topic of anomaly response has not been covered. Nevertheless, the fragility of INs due to the low computing abilities of some of the nodes in the network and the high availability constraints they have, correct anomaly response would be a desirable feature for INs. For instance, to lessen the effects of a port scan that could eventually disrupt controller communication. In this direction, SDN is a promising technology that can help in lessening the effect of intrusions and anomalies in INs. However, the particularities of these networks, where availability is the primary concern, have to be taken into account when designing response solutions.

Appendix A

Tesiaren laburpena (Basque Summary)

Contents

Datuek gidaturiko anomalien detekzioa sare industrialetan –Laburpena–

1960ko hamarkadan lehen Kontrolatzaile Logiko Programagarriak (PLC) sortu zirenetik, Industri Kontrol Sistemek (IKS) garapen handia izan dute. Hasiera bateko instalazio isolatuetatik abiatuta, IKSak gero eta elkarkonektatuago egotera pasatu dira, gaur egun Sare Industrial modura (SI) ezagutzen ditugun ingurune saretu konplexuak osatzeraino. IKSak hainbat prozesu fisikoren kontrolaren ardura duten heinean, eta baita Azpiegitura Kritikoak (AK) osatzen dituzten prozesuen kontrolarenean ere, SIak babesteak berebiziko garrantzia du gizarte modernoen ongizaterako. Arlo horretan eginiko aurrerapenen artean, Anomalien Detekzio Sistemek (ADS) toki nabaria dute. Sistemok SI edota IKSen jokabidea aztertzen dute ohiz kanpoko gertakariak detektatzeko, ezagunak zein ezezagunak izan. Dena den, SIak konplexuago bihurtu ahala, haien segimendua egitea Big Data arazoa bilakatu da. Beste era batera esanda, SIetan sortzen diren datu-sortak konplexuegiak bihurtu dira ohiko bitartekoen bidez prozesatzeko, duten eskala, aniztasun eta sorrera-abiadura handiak direla-eta. Gauzak honela, SIetarako diseinatutako ADSek ez dute abiadura beretsuko garapenik izan, eta esparru honetan eginiko ekarpen berriak ez dira gai datuen konplexutasun honi aurre egiteko, ez baitira eskalagarriak edo ez baitituzte sortutako datuen gehiengoa baliatzen anomalien detekziorako.

Tesi honek hutsune hori betetzeko asmoa dauka, bi ekarpen nagusi eginez: (i) sare-fluxuen monitorizazio-sistema bisual bat eta (ii) aldagai-anizkoitzeko ADS bat, eskala handian lan egin eta datuen heterogeneotasunari aurre egiteko gai dena. Fluxuen monitorizaziorako sistema bat proposatzen dugu, zeinak, momentuko sare-fluxuen datuez baliatuta, segurtasun-bisualizazioak sortzen baititu, non unean aktibo diren fluxuak irudikatzen dituen eta anomaloak direnak nabarmentzen diren. Aldagai anitzeko ADSerako, aldiz, lehenbizi aztertzen dugu Aldagai Anitzeko Prozesuen Kontrol Estatistikoak (AAPKE) anomalien detekziorako eta diagnosirako duen eraginkortasuna, ondoren Big Data-rekin lan egiteko gai den eta AAPKE-n oinarrituriko ADS bat aurkezteko. ADS honek sare eta prozesu mailako aldagaien segimendua egiten du anomaliak detektatzerako orduan. Bi ekarpen hauek esperimentalki balidatzen dira, probetarako inguruneetan SIak eraikiz eta sortutako datuak aztertuz. SIen segurtasunerako ikerketa eremu erreproduzigarri eta zehatzetan egiteko behar honi helduz, helburu hori betetzen duen banku-proba baten diseinua ere aurkezten dugu.

Eranskin honetan, doktoretza-tesian zehar eginiko lanaren ikuspegi orokorra aurkezten da, euskaraz. Lehenik eta behin, sare industrialetan datuek gidaturiko anomalien detekzioa gai modura aukeratzearen zergatiak zerrendatzen dira. Ondoren, tesi honetako helburuak, hipotesiak eta ekarpen nagusiak azaltzen dira, erabilitako ikerketa-metodologiarekin batera. Ostean, dokumentuaren egitura zein den aipatzen da. Azkenik, tesiaren ondorio nagusiak ematen dira ezagutzera, etorkizunerako zenbait lan-ildo ere identifikatuz.

A.1 Motibazioa

Sare Industrial (SI) deritze kolektiboki Industri Kontrol Sistema (IKS) modura ezagunak diren gailu espezializatu, heterogeneo eta elkarkonektatuek osatzen duten ingurune saretuei, zeinetan IKSek prozesu fisikoak automatizatzen, kontrolatzen eta gainbegiratzen dituzten. Hala, IKSak hainbat eta hainbat prozesu fisikoren arduradunak dira, bai industri sektore desberdinetan, baita Azpiegitura Kritikoetan (AK) ere [132]. Europako Kontseiluak [46] honela definitzen du AKa: "aktibo bat sistema bat edo haietariko baten zatia (...) zeina ezinbestekoa den gizarte-funtzioen eta pertsonen osasuna, segurtasuna eta ongizate ekonomiko eta soziala mantentzeko, eta zeinaren eteteak edo suntsitzeak ondorio larriak izango lituzkeen (...) funtzio horiek mantentzeko lanean porrot egitearren". IKSek kontrolaturiko AKen adibide modura, energia-sorrera eta garraioa, ur-hornikuntza eta garraio-sistemak har daitezke.

Beraz, AK-en funtzionamendu egokiak berebiziko garrantzitsua du gizarte modernoen ongizaterako. Miller eta Rowe-k [107] AK-ekin zerikusia izandako segurtasun gertakariak zerrendatu zituzten. Gaur egun, bi dira SIen aurkako erasoek sortzen dituzten kezka nagusiak:

1. IKSek kontrolatzen duten prozesu fisikoen gainean eragina duten SIen aurkako erasoak, prozesuaren eta inguruaren segurtasuna arriskuan jartzen delarik. Gisa honetako erasoen adibide dira Aurora [158], Stuxnet [92], Maroochy ur-sistemako erasoa [129], Georgia-Pacific-eko gertakaria [117] eta Alemaniako altzairutegian jazotako erasoa [17].

2. Espiotzarako diseinaturiko software kaltegarri edo *malware*aren ugaritzea. Software berezitu hauen jomuga informazioa biltzea da, kontrolaturiko prozesuari buruzkoa edota haren jabe den enpresari buruzkoa. Helburua bitarikoa izan daiteke: prozesuari buruzko informazio sekretua lapurtzea (adib., produktu baten fabrikazio-errezeta) edo hirugarren erakunde baten aurkako erasoa burutu ahal izateko informazioa jasotzea.

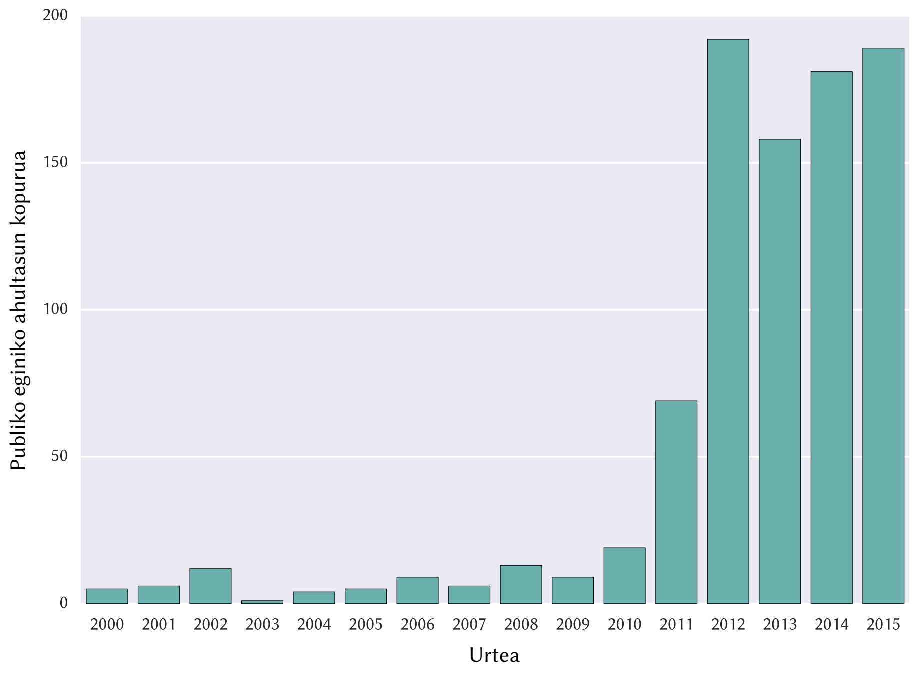
Gisa honetako software kaltegarrien adibide dira Duqu [8] eta Dragonfly [135].

Usadioz, SI eta IKSak bi printzipio nagusitan oinarritu dira euren babeserako, askotan informazio-teknologietan (IT) ohikoak diren segurtasun-neurri gehigarrien erabilera arbuiatuz (adibidez sarbide-kontrola, autentifikazioa edo zifraketa). Alde batetik, iluntasun bidezko segurtasuna, non IKS fabrikatzaileek eurek sortutako sistemen sekretutasuna (software eta hardware jabeduna eta sare-protokolo pribatuak) bultzatzen zuten balizko erasotzaileek euren produktuetan segurtasun-ahultasunak aurkitzea ekiditeko. Joera hau ez-eraginkorra dela demostratu da jada [114]. Sekretutasunak ahultasunak aurkitzea zailtzen duen arren, zailagoa egiten baitu sistema baten barne-funtzionamendua ulertzea, ez ditu eskura sistema ageri batek dituen egiaztatze eta balidazio-prozesuak. ?? irudiak erakusten du IKSekin zerikusia daukaten eta publiko eginiko aurkituriko software-ahultasunen bilakaera denboran zehar. IKS inplementazio asko oraindik jabedun eta sekretuak diren arren, aurkitutako ahultasun-kopuruak gora egin du hamarkada honen hasieratik hona. Ahultasun horiez baliatuz erasoak edota segurtasun-ikuskaritzak egiteko moduluak ere badaude eskuragarri software espezializatuetan, Metasploit plataforman[17] adibidez. Hots, iluntasun lortu nahi hori jada ez da aplikagarria. Sistema bateko zehaztasunak sekretuak izaten jarraitu dezaketen arren, bere ahultasunak eta haien probetxu (txarra) ateratzen duten tresna asko publikoak dira jada. Gainera, segurtasun-ahultasun hauek konpontzeko beharrezkoak diren eguneraketak egitea askotan ez da bideragarria, askotan eguneraketak aplikatzeak prozesu industriala geldiaraztea ekartzen baitu. Hau ez da onargarria prozesu kritiko gehienetan, eskuragarritasun handia behar izaten baitute [32]. Hortaz, maiz, gisa honetako prozesuak eguneratu gabe eta beraz, ahul, uzten dira.

IKSen fabrikatzaileek eta erabiltzaileek haien segurtasunerako euren sareen babeserako erabilitako beste printzipioa sareen isolamenduarena da. Tradizionalki SIek ez dute kanpokonexiorik izan eta beraz, erasotzaileek sarbide fisikoa behar izan dute SIetan erasoak burutu ahal izateko. Dena den, 1990eko hamarkadatik aurrera, sare-baliabideen sarbide non-nahikoaren beharrak bultzatuta, SIak kanpoko sareekin gero eta elkarkonektatuago izatera igaro ziren, hala nola, enpresetako IT sarea edota Internet [32, 69]. SHINE (SHodan Intelligence Extraction) proiektuak 2.186.971 IKS gailu identifikatu zituen Internetera zuzenki konektatuta 2012–2014 urteen artean [138]. Era berean, Shodan bilatzaileak badauka *ICS Radar*[18] izeneko zerbitzu publikoa, zeinak Internet aztertzen duen IKSetara sarbide zuzena eskaintzen duten sare-protokoloen bila. IKS mota desberdinak gordetzen dituen kategoria

[17]`https://scadahacker.com/resources/msf-scada.html`

[18]`https://ics-radar.shodan.io/`



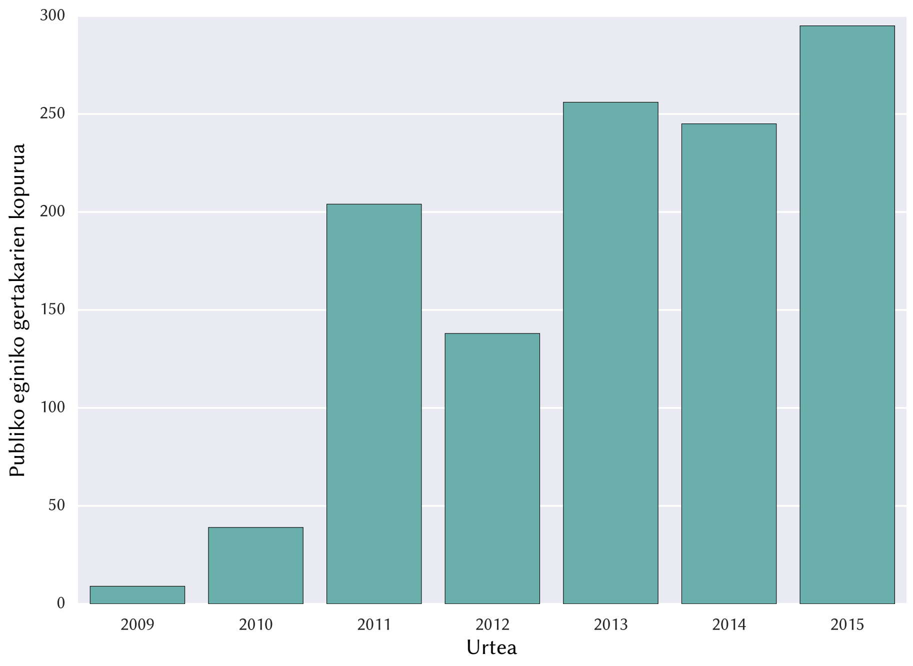
A.1 Irudia: Jakinarazitako segurtasun-ahultasunak denboran zehar [3]

bat ere badu, erabiltzaileari sarbide zuzena emanez gailuetara[19]. Publikoki eskuragarriak diren IKSak izateak erakusten du hainbat SI ez daudela behar den modura bananduta Internetetik, eta beraz balizko isolamendu hori ez dela existitzen.

Zuzenki Internetera konektaturik ez dauden SIen kasuan, ez da egongo sarbide publikorik haietara, baina kanpo-sareetan eginiko erasoek sarbidea lor dezakete SIetako barne-sarera. Hau izan zen Alemaniako altzairutegiko erasoaren [17] kasua, non erasotzaileek IT sarean sarbidea lortu zuten e-posta kaltegarri bat baliatuz, eta hortik jauzi egin zuten produkzioko SIra, erasoan labe garaietako bat kaltetuz. Eta SIak kanporanzko inolako sare-konexiorik ez duen kasuetarako, baliabide handiak dituzten erasotzaileek bitartekoak aurkitu ditzateke muga hauek gainditzeko, USB memoriak erabiliz adibidez (hala gertatu zen Stuxnet-en [92] kasuan, adibidez).

Ameriketako Estatu Batuetako ICS-CERT erakundeak kontrol-sistemak ziber-erasoetatik babesteko bitartekoak ematen ditu [64]. Helburu hau betetzeko, bere betebehar nagusietako bat kontrol-sistemekin zerikusia izan duten segurtasun-gertakariei erantzutea da, bereziki tartean AKak daudenean. Urtero kaleratzen dituzten txostenetan [64–68], ICS-CERTek az-

[19] https://www.shodan.io/explore/category/industrial-control-systems

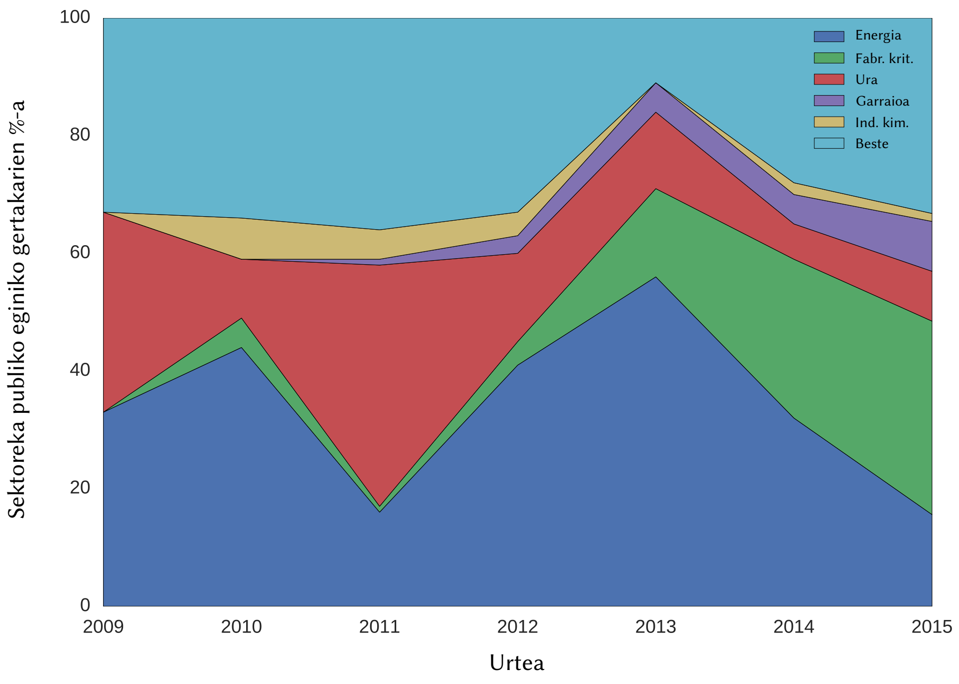


A.2 Irudia: ICS-CERTi jakinarazitako segurtasun-gertakarien kopurua [64–68]

pimarratzen du gorakada izan dela gisa honetako segurtasun gertakarietan, erregistroak hasi zirenetik. A.2 irudiak erakusten du gertakari hauek izan duten bilakaera, 2009an 9 gertakari jakinarazi zizkioten ICS-CERT-eri eta 295 gertakari 2015ean.

A.3 irudiak erakusten du gertakari hauek izan duten bilakaera AK sektore bakoitzeko. Hasieran ur eta energia-sektoreak ziren gertakari gehien batzen zituzten sektoreak, % 60tik gora. Denborarekin, bi sektore hauen garrantziak behera egin du eta beste batzuek hartu diete aurre, bereziki fabrikazio kritikoak.

AKek daukaten izaera kritikoa ikusita, eta ohiko bideen bidez hauek babesteko dagoen ezintasuna dela eta, SIen babesa ikerketa esparru aktiboa da. Hala, SIen babesak arreta handia jaso du industria eta komunitate zientifikoaren eskutik. SIen babeserako ikerketa-esparru desberdinen artean, intrusioen detekzio-sistemek (IDSek) eta bereziki anomalien detekzio-sistemek (ADS) rol garrantzitsua jokatzen dute eta hainbat dira esparru honetan eginiko ekarpenak [51, 108, 164]. Ekarpen horietako gehienek sareko trafikoa aztertuz [51, 164] edota prozesuko magnitude fisikoen jarraipena eginez [82, 88] detektatzen dituzte anomaliak. ADS batzuk, bereziki magnitude fisikoei begira daudenak, ereduetan oinarrituetakoak dira [88, 105, 134]. ADS hauek beharrezkoa dute aztertzen duten prozesuaren eredu ma-



A.3 Irudia: AKetako segurtasun-gertakarien ehunekoaren bilakaera, denboran zehar [64–68]

tematiko bat izatea. Haatik, prozesu fisiko baten eredua eraikitzea lan zaila izan daiteke, baita ezinezkoa ere zenbait kasutan, prozesua bera oso konplexua denean. Bestalde, datuek gidaturiko metodologiek ez dute inolako eredurik behar anomaliak detektatzeko, datuetan oinarriturik bakarrik hartzen baitituzte erabakiak. Gaitasun honek datuek gidaturiko AD-Sen erabilera sustatzen du SI konplexuetan, eta baita prozesu mota desberdinetan ADS berak erabiltzea ere, ez baita beharrezkoa ereduak berraztertzea edota birsortzea aplikazio desberdinetarako.

Bestalde, MapReduce [37] moduko konputazio banatuko egiturak eta HDFS [12] gisako fitxategi-sistema banatuak sortu zirenetik, konputazio-paradigma berria jaio da, Big Data analisia (BDA) izenarekin ezaguna.

Big Data (*Datu-multzo handia*) delakoak ohiko bitartekoen bidez, denbora tarte onargarrian, prozesatzeko konplexuegia den datu-multzoari egiten dio erreferentzia [33]. Nahiz eta kontsentsurik ez den existitzen, datuen konplexutasun hau hiru ezaugarrik definitzen dute: datu-kopurua, datuen sorrera eta transmisio abiadura eta datu-moten aniztasuna, bai egituratua edo egituratu gabea [91]. Berriki, laugarren ezaugarri bat gehitu zaie aurreko hirurei: Big Data datu-multzo baten barruan informazio baliodun aurkitzeko gaitasuna. Dena den, Big Data terminoak datu-multzoaren izana gainditu eta berau prozesatzeko gai diren tresna eta teknologiak izendatzeko ere erabiltzen da egun. Big Data analisiak (BDA) helburu du jakintza balioduna erauztea Big Data datu-multzoetatik, era eskalagarrian analizatuz.

BDAk dituen hainbat aplikazioren artean, Cárdenas et al.-ek [26] eta Everett-ek [47], BDA-ren eraginkortasuna aztertu zuten intrusio-detekziorako, sare konplexuetan berez sortzen diren datuak (erregistorak, fluxuak, paketeak...) Big Data kontsideratu daitezkeelako. Beraz, datu-multzo handi hauetan anomaliak aurkitzea ez da egingarria ohiko mekanismoen bidez. Hala, ondorioztatzen dute BDA erabiltzeak IDS eraginkorragoak garatzea ahalbidetu dezakeela. Dena den, euren analisia IT ordenagailu-sare ohikoetan dago oinarrituta, SIetan teknologia hauek dituzten aukerak alde batera utziz. Era beretsuan, beste autore batzuek BDAk industria-aplikaziotarako dituen aukerak aztertu dituzte. SIetan hainbat datu oso desberdin sortzen dira era oso azkarrean, adibidez, prozesuaren azterketa egiteko erabiltzen diren magnitude fisikoen balioak erregistratuz (tenperaturak, presioak...) edota edozein komunikazio saretan sortzen diren datuak (paketeak, fluxuak eta erregistroak). Segurtasunerako aplikazioak alde batera utzita, badira hainbat lan BDAren erakargarritasuna azpimarratzen duten prozesuen analisirako [80, 116, 125, 151, 165].

Beraz, naturala da bi mundu hauek lotzea, eta BDA erabiltzea SIetan anomaliak detektatzeko ingurune heterogeneo hauetan, zeintzuetan datuek oso forma desberdinak izaten duten. Tesi honek esparru hori betetzeko helburua izan du, BDA aplikatuz SIetan anomaliak detektatzeko.

A.2 Helburuak, hipotesiak eta ekarpen nagusiak

Tesi honen helburu nagusia hau da: *SIetarako datuek gidaturiko anomalien-detekzio sistemak garatzea* eskala handiko datu-multzo heterogeneoak prozesatzea ahalbidetuko dutenak segurtasun-gertakariak detektatzeko prozesuaren eredu bat eraiki behar gabe. Xede honetarako, bi irizpide desberdin jorratzen ditugu: sareko fluxuen jarraipen bisuala eta SIetan sortzen diren datu mota desberdinak analizatzen dituen hurbiltze holistiko bat, sarearen esparru bakar batean fokatzea behar ez duena.

A.2.1 Hipotesiak

Hau da frogatzen saiatuko garen hipotesien zerrenda:

- SIek duten sare-fluxuen izaera estatiko eta errepikakorrari esker, segurtasunbistaratzeek SI operadoreei lagun diezaiekete fuxuekin zerikusia daukaten anomaliak detektatzeko. Kordoi-diagramak hautagai egokiak dira xede honetarako.

- Aldagai Anitzeko Prozesuen Kontrol Estatistikoa (AAPKE) erabiliz, gai izan gaitezke SIetan anomaliak detektatzeko eta haien diagnosia egiteko, intrusioak eta perturbazioak desberdintzea ahalbidetuz prozesuaren eredurik erabili gabe.

- AAPKE eredu tradizionalak hedatuz eta sareko aldagaiak bertan txertatuz, ADS bat sortzea posible da zeina SIetan sortzen diren datu-mota guztiak analizatzeko gai den era eskalagarrian.

A.2.2 Ekarpenak

Doktore-tesi honetako ekarpen nagusiak hurrengo puntuetan laburbildu daitezke:

- Eskala handiko ADS heterogeneoen esparruaren literatura-aztertze sakona, bereziki euren SIetarako aplikagarritasunean fokatuz eta oraindik garatzeke dauden ikerketa-esparruak identifikatuz [73].

- SIetan segurtasun-ikerketa egiteko banku-proba baten diseinua. Banku-proba hau Emulab softwarea eta prozesu fisikoen simulaizoaren inguran dago egituratua, ingurunearen fideltasuna bermatzeko. Software bidezko sareentzako euskarria dauka, baita datu-analisirako modulua ere, etorkizuneko ikerketarako teknologia hauek erabiltzea ahalbidetuz [76].

- SIetarako segimendu sistema bisuala, sare-fluxuak eta hauekin erlazionatutako anomaliak bistaratzen dituena. Kordoi-diagrametan dago oinarrituta eta bertan, fluxuak zilegi ala anomalo modura adierazten dira, aurretik zenbait zerrenda zuritan deskribatutako sareko fluxuen arabera [74, 75].

- AAPKE erabilita IKSetan intrusioak eta prozesuko perturbazioak desberdintzeko dagoen bideragarritasunari buruzko ikerketa bat. Bertan, ondorioztatzen da AAPKE gai dela IKSetan anomaliak detektatzeko, baina haien kausa identifikatzerako orduan, beharrezkoa dela prozesuko benetako datuen analisia egitea. Egoera hau ezkutatzen bada operadorearengadik, lagungarria da datu gehigarriak (adib., sareko trafikoa) erabiltzea anomalia era zuzenean diagnostikatzeko [71, 72].

- Eskala handiko datu-multzo heterogeneoekin lan egiteko gai den ADS bat, detekziorako sare eta prozesu-mailako datuak analizatzen dituena. Honetarako, ohiko AAPKE eredua hedatu egiten dugu sareko datuak txertatzeko ereduan, prozesuko datuekin batera. Inplementazioa Apache Spark Big Data softwarearen gainean egiten da proposamenaren eskalagarritasuna bermatzeko. Aurkeztutako ADSa gai da sare eta prozesu mailan gertaturiko anomaliak detektatzeko eta haien diagnosia egiteko, baita bi mailetan gertatzen direnak ere.

- Osagai Nagusien Analisia (ONA) erabiliz aldagai anitzeko datu-multzo handien esplorazio eta analisia egitea ahalbidetzen duen Big Data tresna multzoa. Aurretik aurkezturiko ADSa eraikitzeko erabilitako tresnak eta inplementazioak daude bertan.

Argitalpenak

Doktoretza-tesi honetan aurkezturiko zenbait atal dagoeneko argitaratu dira edo errebisiorako bidali dira pare bidezko azterketa duten aldizkarietara eta nazioarteko zein estatu mailako konferentzietara. Segidan, doktore-tesi honetako lanarekin zuzenki erlazionaturik dauden lanen zerrenda aurkezten dugu, argitaratuta zein azterketa-prozesuan dauden lanekin:

Aldizkarietako argitalpenak

- Mikel Iturbe, Iñaki Garitano, Urko Zurutuza, eta Roberto Uribeetxeberria. Towards Large-Scale, Heterogeneous Anomaly Detection Systems for Industrial Networks. *Argitaratzeko bidalia IEEE Transactions on Industrial Informatics aldizkarira*, 2017.

Kongresuetako argitalpenak

- Mikel Iturbe, Iñaki Garitano, Urko Zurutuza, eta Roberto Uribeetxeberria. Visualizing Network Flows and Related Anomalies in Industrial Networks using Chord Diagrams and Whitelisting. Non: *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016)*, 2. liburukia, 99–106 orr., Erroma, Italia, 2016ko Otsaila.

- Mikel Iturbe, José Camacho, Iñaki Garitano, Urko Zurutuza, eta Roberto Uribeetxeberria. On the Feasibility of Distinguishing Between Process Disturbances and Intrusions in Process Control Systems Using Multivariate Statistical Process Control. Non: *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 155–160 orr., Tolosa, Frantzia, 2016ko ekaina. IEEE.

- Mikel Iturbe, Iñaki Garitano, Urko Zurutuza, eta Roberto Uribeetxeberria. Sistema visual de monitorización de seguridad de flujos de red industriales. Non: *Proceedings of I Jornadas Nacionales de Investigación en Ciberseguridad (JNIC 2015)*, 59–65 orr., León, Espainia, 2015ko iraila. Universidad de León.

- Mikel Iturbe, Unai Izagirre, Inaki Garitano, Ignacio Arenaza-Nuno, Urko Zurutuza, eta Roberto Uribeetxeberria. Diseno de un banco de pruebas híbrido para la investigación de seguridad y resiliencia en redes industriales. Non: *Proceedings of II Jornadas Nacionales de Investigación en Ciberseguridad (JNIC 2016)*, 3–10 orr., Granada, Espainia, 2016ko ekaina. Universidad de Granada.

- Mikel Iturbe, José Camacho, Iñaki Garitano, Urko Zurutuza, eta Roberto Uribeetxeberria. Distinguiendo entre perturbaciones de proceso e intrusiones en sistemas de control: caso de estudio con el proceso Tennessee-Eastman. Non: *Proceedings of the XIV Spanish Meeting on Cryptology and Information Security (RECSI 2016)*, 117–122 orr., Maó, Espainia, 2016ko urria. Universitat de les Illes Balears.

A.3 Metodologia

Tesi honetan erabilitako ikerketa-metodologia *Diseinu eta sorkuntza* da. Oates-en [115] arabera, metodologia hau "IT produktu berritzaileak –*artefaktuak*– garatzen zentratzen dena da". March eta Smith-en [102, 115] arabera, artefaktu hauek lau kategoria nagusitan bana daitezke:

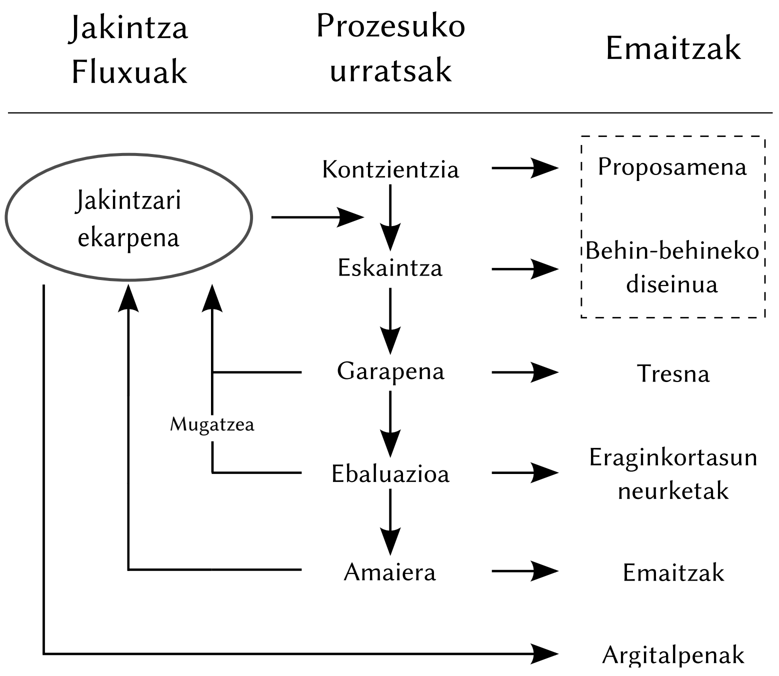
Konstruktuak Kontzeptuekin daude erlazionaturik, hala nola entitate, objektu edota data fluxuen nozioekin.

Ereduak Konstruktuz osaturiko egoera baten irudikapenak dira, arazoen ulermena eta irtenbideen garapena erraztuko dutena. Adibidez, fluxu-diagrama bat edo erabilera-kasu konkretu baten zehaztapena.

Metodoak Sortuko diren ereduen gaineko eta arazoa konpontzeko jarraituko den prozesuaren pausuen gida. Adibide nagusiak algoritmoak dira.

Instantziazioak Martxan jar daitekeen sistema bat, konstruktuak, ereduak, metodoak, ideiak edota teoriak ordenagailu-sistema batean inplementatu daitezkeela demostratzen duena.

Zenbaitetan ikerketaren emaitza lau emaitza hauen arteko konbinazioa izan daiteke. Diseinu eta sorkuntza prozesua eta ohiko software-garapenaren prozesua nahastu daitezke. Dena den, ikerketa-lan hau IT produktu bat garatzea baino harago doa. Ikerketa-lan honetan analisiaren azalpena, argumentuak, justifikazioa eta emaitzen ebaluazio kritikoa aurkitu daitezke. Hortaz, jakintza berriaren sorkuntza du helburu, eta honen emaitzen zabalkundea argitalpen zientifikoen bidez, horretarako artefaktu baten garapenaz baliatuz.



A.4 Irudia: Tesi honetan erabilitako diseinu eta sortze-prozesua, emaitza desberdinekin batera [145]

Vaishnavi and Kuechler-ek [145] bost pausu iteratibotan banatzen dituzte diseinu eta sorkuntza prozesua:

Kontzientzia Esparru jakin batean ikerketa ahalbidetzen duen arazo baten identifikazioa eta artikulazioa. Fase honen emaitza proposamen bat da, formala zein informala.

Eskaintza Fase honetan, funtzionalitate berria identifikatzen da, exisitzen diren elementuekin edo existitzen diren eta berriak diren elementuekin konfigurazio berritzaile batean oinarrituz. Funtzionalitate berri honen helburua kontzientzia-fasean identifikaturiko arazoari irtenbidea aurkitzea da. Fase honetan behin-behineko diseinu bat garatzen da, non aurreko fasean lorturiko proposamena landu eta artefaktuari buruzko detaile gehiago ematen diren (eredu posibleak, prototipoa...).

Garapena Fase honetan, behin-behineko diseinua are gehiago garatzen da eta artefaktu batean inplementatu. Espero bezala, inplementazio honen forma, sortzen den artefaktuaren araberakoa da (softwarea,algoritmoak...).

Ebaluazioa Artefaktuaren inplementazioa proposamenean era inplizituan zein esplizituan zehazturiko irizpideen arabera aztertzen da.

Amaiera Fase hau ikerketa-zikloaren amaiera da, non diseinu eta sorkuntza-prozesuko emaitzek aurreko faseetan definitutako funtzionalitateak betetzen dituzten. Emaitzak sendotu egiten dira eta jasotako jakintza berria zehaztu eta zabaldu egiten da.

A.4 irudian ikus daitekeen bezala, prozesu nagusia iteratiboa da, artefaktuaren inguruko erreakzioek hura birmoldatzea ahalbidetzen baitute emaitzak hobetzeko. Dena den, badago ataza bat ez dagoena erlazionatuta prozesu iteratiboarekin: emaitzen zabalkundea argitalpen zientifikoen bidez. Lan honekiko jarrera jarraitua izan da, hots, argitalpenak ebaluatzeko bidali dira emaitzak nahikoa helduak zirenean, komunitate zientifikoak aurrerapenok ahalik eta arinen balia ditzan. Hala ere, azken emaitzak, ikerketa-prozesuaren zehaztasun guztiekin batera, doktoretza-tesi honetan daude aurkezturik.

A.4 Dokumentuaren egitura

Doktoretza-tesi hau zazpi kapitulutan dago banaturik. Atal honetan, zazpi kapituluen edukiari buruzko laburpen txikia jasotzen dugu.

1. kapitulua sarrera da. Eranskin honetako lehen lau ataletan euskaraz paratutako edukiaren baliokidea da.

2. kapituluak eskala-handiko ADS heterogeneoen eremuan eginiko ekarpenen azterketa egiten du. Are gehiago, oraindik konpondu gabe dauden zenbait arazo identifikatzen ditu, esparru horretan ikerketa-ildo berrietara eraman dezakeena.

3. kapituluak SIetako segurtasunean ikertzeko baliagarria den banku-proba baten diseinua aurkezten du. Banku-proba honek Emulab softwarea darabil, prozesu fisikoaren simulazioarekin batera gisa honetako ikerketa burutzeko eremu fidagarria eta erreproduzigarria sortuz.

4. kapituluak SIetako sare-fluxuen jarraipena egiteko sistema bisual bat proposatzen du. Kordoi-diagrametan eta zerrenda zurietan oinarrituz, sistemak fluxuen egoera erakusten du, bisualki nabarmenduz anomaloak diren horiek, hots, zerrenda zuriko arauak betetzen ez dituztenak.

5. kapituluak AAPKEren kontzeptua azaltzen du, baita bere aplikagarritasuna aztertu IKSetan anomaliak detektatzeko eta haien diagnosia egiteko. Ondorioztatzen du AAPKE metodologia baliagarria dela anomaliak detektatzeko kontestu honetan, dituen mugak aipatuz anomalien diagnosia egiterako orduan.

6. kapituluak AAPKEren eskala handiko hedapen baten garapena erakusten du SIetan anomaliak detektatzeko. AAPKE eredua hedatu egiten da bertan, prozesuko eta sareko aldagaien jarraipena egiteko era eskalagarrian. Emaitzek erakusten dute proposaturiko sistema egokia dela anomaliak detektatzeko eta eskalagarria dela, bai jarraituriko aldagai kopuruari dagokionez, baita aztertu beharreko datu-multzoen bolumenari dagokionez ere.

7. kapituluak amaiera ematen dio tesiari, ondorioen eta ekarpen nagusien laburpena eginez. Era berean, etorkizunerako zenbait ildo ere zehazten ditu. Kapitulu honen edukiak euskaraz paratu dira eranskin honetako hurrengo ataletan.

A.5 Ondorioak

Doktore-tesi hau datuek gidaturiko ADS berritzaileen garapenean egon da zentratuta.

Lehenik eta behin, literatura-azterketa sakona egin dugu, non eskala-handiko ADS heterogeneoen esparrua aztertu dugun, baita SIetara duten aplikagarritasuna ere. Bertan, zenbait hutsune identifikatu ditugu gaur eguneko ikerketa-proposamenetan, baita zenbait lan-ildo etorkizunerako ere.

SIetako segurtasun-ikerketa eremu seguru, fidel eta erreproduzigarrian aurrera eramateko beharrari erantzunez, sareko jokabidea era dinamikoan emulatzen duen eta prozesu fisikoa simulatzen duen banku-proba aurkeztu dugu. Honekin, banku-proba gai da SI errealen funtzionalitateak emulatzeko, Emulab softwarean eta Tennessee-Eastman prozesuan oinarriturik. Etorkizunean ikerketa-lan berrietan erabili ahal izateko, banku-probak software bidezko sareetako euskarria ere badu.

Ondoren, sare-fluxuen jarraipena egiteko sistema bisuala aurkeztu dugu, kordoi-diagrama eta zerrenda zurietan oinarriturikoa. Sare fluxuak kordoi-diagrama batean irudikatzen dira, eta daukaten zilegitasunaren arabera nabarmentzen dira. Hau ahalbidetzeko, denbora aintzat hartzen duten zerrenda zuriak garatzen ditugu, non trafikoaren izaera erregistratzen den denbora tarte jakin batean. Ohiko zerrenda zuriekin alderatutakoan, posiblea da tamaina ezohikoa duten fluxuak atzematea. Fluxuen erregistro historikoak bilaketa-zerbitzari eskalagarrian gordetzen dira. Eginiko azterketen arabera, SIetarako espreski diseinatutako gisa honetako lehen bistaratze-tresna da hau.

Gero, AAPKEren eraginkortasuna aztertu dugu anomaliak detektatzerako orduan prozesu mailan. Ondorioztatu dugu AAPKE metodologia erabilgarria dela anomaliak detektatzeko, baina haien diagnosia ataza konplexua izan daitekeela, prozesuaren benetako egoera operadorearengandik ezkutatzen denean. Gainera, batzuetan anomaliak detektatzerako orduan dauden denbora-tarteak luzeak izaten dira. Datuek gidaturiko metodologia izaki, duen abantaila nagusienetako bat da jarraipena egiten dion prozesuari buruzko aurretiko ezagupenik izatea ez dela beharrezkoa.

Azkenik AAPKE eredua hedatu dugu sare-mailako aldagaiak ere kontuan hartzeko. Hau lortzeko sare fluxuetako informazioa zenbait aldagai kuantitatibotan laburtu dugu, baita kontagarriak diren zenbait ezaugarri definitu ere, fluxu horiek anomalo modura etiketatuak izan diren edo ez AAPKE barruan sartzeko. Proposamen hau balidatzeko, SI esperimental bat eraiki dugu, sare-trafiko industrial errealarekin eta simulatutako Tennessee-Eastman prozesuarekin, zeina aldatua izan den sare bidez komunikatu ahal izateko. ADS hedatu hau gai da sare eta prozesu mailako anomaliak detektatzeko, baita bi mailetan eragina duten intrusioak detektatzeko ere. Gainera, oMEDA grafikoak erabilita, posible da aldagai bakoitzak egoera anomaloan izan duen eragina aztertzea, arazoaren diagnosia erraztuz.

A.6 Etorkizunerako ildoak

Atal honetan zenbait etorkizunerako ildo zehazten ditugu, SIetako anomalien detekzioaren esparruan ekarpen berrietara eraman dezaketena. Doktore-tesi honek, nahiz eta irismen mugatukoa izan, eskaintzen ditu zenbait aukera garapen gehiagorako. Orain aipatutako aukera horiek zerrendatzen ditugu, gaika ordenatuta. Aukera hauek tesi honetan aurkeztutako lanaren jarraipen berehalako modura har daitezke, eta beraz hemen identifikatutako lerroak aurretik 2.5 atalean aipatutakoei gehitu diezaizkiete.

Banku-probaren ebaluazioa Banku-proba bat diseinatu eta inplementatu eta gero, hurrengo pausua bere fideltasuna neurtzea da, instalazio errealekin alderatuz ahalik eta berdintsuenak izateko. Funtzio hau betetzen duten ebaluazio-metrika desberdinak daude. Siaterlis et al.-ek [127] aurkezturiko lanean, simulaturiko prozesuaren exekuzio-denbora erabiltzen dute metrika modura, prozesu horrek dituen latentzia-eskakizunak betetzen dituen edo ez alderatuz. Reaves eta Morris-ek [122] Modbus sare-protokoloko paketeen kargarekin zerikusia duten metrika batzuk proposatzen dituzte. Azkenik Holm et al.-ek [61] batutako proposamenetako batzuek euren banku-probak instituzio esanguratsuek argitaraturiko estandarei eta gidei jarraipena nola egiten dien aztertuz ebaluatzen dituzte. Gauzak honela, oso zaila da banku-proba desberdinen arteko konparazio orekatua egiteko, erabilitako metrikak desberdinak baitira. Beraz, banku-probak ebaluatzea eta konparatzea ahalbidetuko duen metrika multzo bateratu bat ekarpen garrantzitsua litzateke banku-proben ikerketa aurrera eramateko.

Anomalien detekzioa SIetan Aurretik aipatu modura, SIetan anomaliak detektatzea ikerketa esparru aktiboa da, eta azkenaldian asko ugaritu dira proposamenak, batez ere prozesu mailako jarraipena egiten dutenak. Ildo honetan, AAPKE, datuek gidaturiko metodologia eskalagarri gisa, anomalien detekziorako era berritzailea da. AAPKE barnean erabili ahal izango diren aldagai kuantitatibo berriak garatzea metodologiaren hobekuntza ekar dezake, informazio gehigarria izango bailuke analisirako. SIen esparruan, aldagai berri hauen informazio-iturri interesgarriak izan daitezke alertak, sare-protokoloen analizatzaileak eta IKSetako erregistroak. Aldagai konplexu hauen kuantifikazioak, ikerketa-esparru berria ireki dezake, non hasierako datuak sinplifikatu daitezkeen era desberdinetako ADSak erabili ahal izateko. Gainera, aurretik banku-proben ebaluazioarekin aipatu modura, beharrezkoa da plataforma komun bat garatzea ADSen ebaluaziorako. Tennessee-Eastman prozesuaren moduko *de facto* estandarrak egoteak laguntzen du afera honetan, baina beharrezkoa da pausu bat harago joan eta SIak eta IKSak jomugan dituzten eraso-multzo publikoak sortzea. Era honetan, ADSen detekzio-ahalmenak era errazagoan konparatu ahal izango lirateke proposamen desberdinen artean.

Eskala handiko aldagai anitzeko tresna-multzoa Apache Spark-en gainean eskalahandiko ADSa garatzerako orduan, AAPKE metodologia erabili ahal izateko beharrezko metodoak inplementatu ditugu. Funtzionalitate hauek erabiltzea posiblea da, halaber, aldagai anitzeko datu-multzo handiak esploratu eta aztertzeko. Tresna hau garatzen jarraitzeak, software-aplikazio oso batera eraman dezake, aldagai anitzeko datu-multzoen analisirako lagunduko duena, gaur egun MEDA tresna-kutxak [22] egiten duen modura, baina datu-multzo handiagoetarako.

Anomaliei erantzuna SIetan Aurretik esan bezala, doktoretza-tesi hau anomalien detekzioan eta diagnosian zentratu da. Anomalia bat detektatu eta honen kausaren diagnosia egin eta gero ez da aztertu zer erantzun egin daitekeen anomalia horren eragina mugatzeko. Adibidez, portu-eskaner baten eragina deusestatzeko. Dena den, SIek duten eskuragarritasun behar handiei lotuta, beharrezkoa da erantzun mota guztiak aurretik ondo aztertzea, sarearen funtzionamendu egokiari ez eragiteko. Erantzuteko esparru honetan, software bidezko sareek lagun dezakete era dinamikoan sarearen egoera desberdinei aurre egiten, betiere, SIen berezitasunak kontuan hartzen badira soluzio hauek garatzeko orduan.

Bibliography

[1] Carlos F Alcala and S Joe Qin. Analysis and generalization of fault diagnosis methods for process monitoring. *Journal of Process Control*, 21(3):322–330, 2011.

[2] Edward J Amoroso. Fundamentals of Computer Security. *PTR Prentice Hall. Englewood Cliffs, NJ*, 1994.

[3] Oxana Andreeva, Sergey Gordeychik, Gleb Gritsai, Olga Kochetova, Evgeniya Potseluevskaya, Sergey I. Sidorov, and Alexander A. Timorin. Industrial Control Systems Vulnerabilities Statistics. Technical report, Kaspersky Lab, Jul. 2016.

[4] Luc Anselin. Local indicators of spatial association–LISA. *Geographical analysis*, 27(2):93–115, 1995.

[5] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Technical report, Chalmers University of Technology, 2000.

[6] Michael Baker, David Turnbull, and Gerald Kaszuba. Finding Data Needles in Haystacks (the Size of Countries). In *Prooceedings of Blackhat Europe*, 2012.

[7] Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras. Flow Whitelisting in SCADA Networks. *International Journal of Critical Infrastructure Protection*, 6(3):150–158, 2013.

[8] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Márk Félegyházi. Duqu: Analysis, detection, and lessons learned. In *ACM European Workshop on System Security (EuroSec)*, 2012.

[9] Andrzej Bialecki, Michael Cafarella, Doug Cutting, and Owen O'Malley. Hadoop: a framework for running applications on large clusters built of commodity hardware. *Wiki at http://hadoop.apache.org*, 2005.

[10] Timo Bingmann, Michael Axtmann, Emanuel Jöbstl, Sebastian Lamm, Huyen Chau Nguyen, Alexander Noe, Sebastian Schlag, Matthias Stumpp, Tobias Sturm, and Peter Sanders. Thrill: High-Performance Algorithmic Distributed Batch Data Processing with C++. In *Proceedings of the 2016 IEEE International Conference on Big Data (Big Data)*, pages 172–183, Dec. 2016.

[11] Ettore Bompard, Paolo Cuccia, Marcelo Masera, and Igor Nai Fovino. Cyber vulnerability in power systems operation and control. In *Critical Infrastructure Protection*, pages 197–234. Springer, 2012.

[12] Dhruba Borthakur. The hadoop distributed file system: Architecture and design. 2007.

[13] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.

[14] Paul G Brown. Overview of scidb: large scale array storage, processing and analysis. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 963–968. ACM, 2010.

[15] Anna Buczak and Erhan Guven. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection.

[16] Vernon K.C. Bumgardner and Victor W. Marek. Scalable Hybrid Stream and Hadoop Network Analysis System. In *Proceedings of the 5th ACM/SPEC international conference on Performance engineering (ICPE '14)*, pages 219–224, Dublin, Ireland, Mar. 2014. Association for Computing Machinery.

[17] Bundesamt für Sicherheit in der Informationstechnik. Die Lage der IT-Sicherheit in Deutschland 2014 (Technical Report), Dec. 2014.

[18] José Camacho. Observation-based missing data methods for exploratory data analysis to unveil the connection between observations and variables in latent subspace models. *Journal of Chemometrics*, 25(11):592–600, 2011.

[19] José Camacho, Gabriel Maciá Fernández, Jesús Díaz Verdejo, and Pedro García Teodoro. Tackling the Big Data 4 vs for anomaly detection. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 500–505, April 2014.

[20] José Camacho, Roberto Magán-Carrión, Pedro García-Teodoro, and James J Treinen. Networkmetrics: multivariate big data analysis in the context of the internet. *Journal of Chemometrics*, 30(9):488–505, 2016.

[21] José Camacho, Alejandro Pérez Villegas, Pedro García Teodoro, and Gabriel Maciá Fernández. PCA-based multivariate statistical network monitoring for anomaly detection. *Computers & Security*, 59:118–137, 2016.

[22] José Camacho, Alejandro Pérez Villegas, Rafael A Rodríguez Gómez, and Elena Jiménez Mañas. Multivariate Exploratory Data Analysis (MEDA) Toolbox for Matlab. *Chemometrics and Intelligent Laboratory Systems*, 143:49–57, 2015.

[23] Richard Candell, Timothy Zimmerman, and Keith Stouffer. An Industrial Control System Cybersecurity Performance Testbed. Technical Report NISTIR 8089, National Institute of Standards and Technology, Nov 2015.

[24] Paris Carbone, Stephan Ewen, Seif Haridi, Asterios Katsifodimos, Volker Markl, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *Data Engineering*, page 28, 2015.

[25] Alvaro A Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM symposium on information, computer and communications security*, pages 355–366. ACM, 2011.

[26] Alvaro A. Cárdenas, Pratyusa K. Manadhata, and Sreeranga P. Rajan. Big Data Analytics for Security. *Security & Privacy, IEEE*, 11(6):74–76, 2013.

[27] Godwin Caruana, Maozhen Li, and Hao Qi. SpamCloud: A MapReduce based anti-spam architecture. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*, 2010.

[28] Godwin Caruana, Maozhen Li, and Man Qi. A MapReduce based parallel SVM for large scale spam filtering. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, 2011.

[29] Pedro Casas, Johan Mazel, and Philippe Owezarski. Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. *Computer Communications*, 35(7):772–783, 2012.

[30] Subha Chakraborti. Run Length, Average Run Length and False Alarm Rate of Shewhart X-bar Chart: Exact Derivations by Conditioning. *Communications in Statistics-Simulation and Computation*, 29(1):61–81, 2000.

[31] Duen Horng Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. Polonium: Tera-Scale Graph Mining and Inference for Malware Detection. In *Proceedings of SIAM International Conference on Data Mining (SDM) 2011*, Mesa, AZ, USA, Apr 2011.

[32] Manuel Cheminod, Luca Durante, and Adriano Valenzano. Review of Security Issues in Industrial Networks. *IEEE Transactions on Industrial Informatics*, 9(1):277–293, 2013.

[33] Min Chen, Shiwen Mao, and Yunhao Liu. Big Data: A Survey. *Mobile Networks and Applications*, 19(2):171–209, 2014.

[34] Siming Chen, Cong Guo, Xiaoru Yuan, Fabian Merkle, Hanna Schaefer, and Thomas Ertl. OCEANS: online collaborative explorative analysis on network security. In *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, pages 1–8. ACM, 2014.

[35] Tao-Wei Chiou, Shi-Chun Tsai, and Yi-Bing Lin. Network security management with traffic pattern clustering. *Soft Computing*, 18(9):1757–1770, Sept. 2014.

[36] Junho Choi, Chang Choi, Byeongkyu Ko, Dongjin Choi, and Pankoo Kim. Detecting web based DDoS attack using MapReduce operations in cloud computing environment. *Journal of Internet Services and Information Security*, 3(3/4):28–37, Nov 2013.

[37] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[38] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, 1999.

[39] Antonio Di Pietro and Stefano Panzieri. Taxonomy of SCADA systems security testbeds. *International Journal of Critical Infrastructures*, 10(3):288–306, 2014.

[40] Djellel Eddine Difallah, Philippe Cudre Mauroux, and Sean A McKenna. Scalable anomaly detection for smart city infrastructure networks. *Internet Computing, IEEE*, 17(6):39–47, 2013.

[41] Xinshu Dong, Hui Lin, Rui Tan, Ravishankar K Iyer, and Zbigniew Kalbarczyk. Software-defined networking for smart grid resilience: Opportunities and challenges. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, pages 61–68. ACM, 2015.

[42] James J Downs and Ernest F Vogel. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3):245–255, 1993.

[43] Juliette Dromard, Gilles Roudière, and Philippe Owezarski. Unsupervised Network Anomaly Detection in Real-Time on Big Data. In Tadeusz Morzy, Patrick Valduriez, and Ladjel Bellatreche, editors, *New Trends in Databases and Information Systems*, volume 539 of *Communications in Computer and Information Science*, pages 197–206. Springer International Publishing, Aug 2015.

[44] David Duggan, Michael Berg, John Dillinger, and Jason Stamp. Penetration testing of industrial control systems. Technical Report SAND2005-2846P, Sandia National Laboratories, March 2005.

[45] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[46] European Council. Council Directive 2008/114/EC. Technical report, Official Journal of the European Union, December 2008.

[47] Cath Everett. Big data–the future of cyber-security or its latest threat? *Computer Fraud & Security*, 2015(9):14–17, Sep 2015.

[48] Romain Fontugne, Johan Mazel, and Kensuke Fukuda. Hashdoop: a MapReduce framework for network anomaly detection. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 494–499, 2014.

[49] Jerome François, Shaonan Wang, Walter Bronzi, R State, and Thomas Engel. BotCloud: detecting botnets using MapReduce. In *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, pages 1–6. IEEE, 2011.

[50] Brendan Galloway and Gerhard Hancke. Introduction to Industrial Control Networks. *IEEE Communications Surveys & Tutorials*, 15(2):860–880, 2012.

[51] Iñaki Garitano, Roberto Uribeetxeberria, and Urko Zurutuza. A review of SCADA anomaly detection systems. In *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011*, pages 357–366. Springer, 2011.

[52] Béla Genge, Christos Siaterlis, and Marc Hohenadel. Impact of network infrastructure parameters to the effectiveness of cyber attacks against industrial control systems. *International Journal of Computers Communications & Control*, 7(4):674–687, 2012.

[53] Béla Genge, Christos Siaterlis, and Georgios Karopoulos. Data fusion-base anomaly detection in networked critical infrastructures. In *Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on*, pages 1–8. IEEE, 2013.

[54] Paul Giura and Wei Wang. Using large scale distributed computing to unveil advanced persistent threats. *SCIENCE*, 1(3):93–105, 2012.

[55] Daniel Gonçalves, João Bota, and Miguel Correia. Big data analytics for detecting host misbehavior in large logs. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 1, pages 238–245. IEEE, 2015.

[56] Marcin Gorawski, Anna Gorawska, and Krzysztof Pasterak. A survey of data stream processing tools. In *Information Sciences and Systems 2014*, pages 295–303. Springer, 2014.

[57] Govind P Gupta and Manish Kulariya. A framework for fast and efficient cyber security network intrusion detection using apache spark. *Procedia Computer Science*, 93:824–831, 2016.

[58] Dina Hadžiosmanović, Damiano Bolzoni, and Pieter H. Hartel. A log mining approach for process monitoring in SCADA. *International Journal of Information Security*, 11(4):231–251, 2012.

[59] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[60] Zachary Hanif, Calhoun Telvis, and Jason Trost. BinaryPig: Scalable Static Binary Analysis Over Hadoop. In *Proceedings of Blackhat USA*, Las Vegas, NV, USA, 2013.

[61] Hannes Holm, Martin Karresand, Arne Vidström, and Erik Westring. A Survey of Industrial Control System Testbeds. In Sonja Buchegger and Mads Dam, editors, *Secure IT Systems*, volume 9417 of *Lecture Notes in Computer Science*, pages 11–26. Springer International Publishing, 2015.

[62] Harold Hotelling. Multivariate quality control. *Techniques of Statistical Analysis*, 1947.

[63] Wolfgang Hurst, Madjid Merabti, and Paul Fergus. Big Data Analysis Techniques for Cyber-threat Detection in Critical Infrastructures. In *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, pages 916–921. IEEE, 2014.

[64] ICS-CERT. ICS-CERT Incident Response Summary Report (2009-2011). Technical report, Industrial Control Systems Cyber Emergency Response Team, 2012.

[65] ICS-CERT. Monthly Monitor. October - December. Technical report, 2012.

[66] ICS-CERT. ICS-CERT Year in Review (2013). Technical report, Homeland Security, 2013.

[67] ICS-CERT. ICS-CERT Year in Review. 2014. Technical report, Industrial Control Systems Cyber Emergency Response Team, 2014.

[68] ICS-CERT. NCCIC/ICS-CERT Year in Review. FY2015. Technical report, Industrial Control Systems Cyber Emergency Response Team, 2015.

[69] Vinay M Igure, Sean A Laughter, and Ronald D Williams. Security issues in SCADA networks. *Computers & Security*, 25(7):498–506, 2006.

[70] Luca Invernizzi, Sung-Ju Lee, Stanislav Miskovic, Marco Mellia, Ruben Torres, Christopher Kruegel, Sabyasachi Saha, and Giovanni Vigna. Nazca: Detecting Malware Distribution in Large-Scale Networks. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS 2014)*, 2014.

[71] Mikel Iturbe, José Camacho, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. Distinguiendo entre perturbaciones de proceso e intrusiones en sistemas de control: caso de estudio con el proceso Tennessee-Eastman. In *Proceedings of the XIV Spanish Meeting on Cryptology and Information Security (RECSI 2016)*, pages 117–122, Maó, Spain, Oct. 2016. Universidad de les Illes Balears.

[72] Mikel Iturbe, José Camacho, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. On the feasibility of distinguishing between process disturbances and intrusions in process control systems using multivariate statistical process control. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 155–160, Toulouse, France, Jun. 2016. IEEE.

[73] Mikel Iturbe, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. Towards Large-Scale, Heterogeneous Anomaly Detection Systems for Industrial Networks. *Submitted for publication to the IEEE Transactions on Industrial Informatics*, 2017.

[74] Mikel Iturbe, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. Sistema visual de monitorización de seguridad de flujos de red industriales. In *Proceedings of Jornadas Nacionales de Investigación en Ciberseguridad (JNIC 2015)*, pages 59–65, León, Spain, Sep. 2015.

[75] Mikel Iturbe, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. Visualizing Network Flows and Related Anomalies in Industrial Networks using Chord Diagrams and Whitelisting. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 2, pages 99–106, Feb. 2016.

[76] Mikel Iturbe, Unai Izagirre, Iñaki Garitano, Ignacio Arenaza-Nuño, Urko Zurutuza, and Roberto Uribeetxeberria. Diseño de un banco de pruebas híbrido para la investigación de seguridad y resiliencia en redes industriales. In Pedro García-Teodoro, Rafael A. Rodrıíguez-Gómez, and Fco. Javier López Muñoz, editors, *Proceedings of II Jornadas Nacionales de Investigación en Ciberseguridad (JNIC 2016)*, pages 3–10, Granada, Spain, Jun. 2016. University of Granada.

[77] J Edward Jackson and Govind S Mudholkar. Control procedures for residuals associated with principal component analysis. *Technometrics*, 21(3):341–349, 1979.

[78] Jiyong Jang, David Brumley, and Shobha Venkataraman. Bitshred: Fast, scalable malware triage. *Cylab, Carnegie Mellon University, Pittsburgh, PA, Technical Report CMU-Cylab-10*, 22, 2010.

[79] William Jardine, Sylvain Frey, Benjamin Green, and Awais Rashid. SENAMI: Selective non-invasive active monitoring for ICS intrusion detection. In *Proceedings of the Second ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, Vienna, Austria, October 2016. ACM.

[80] Mladen Kezunovic, Le Xie, and Santiago Grijalva. The role of big data in improving power system operation and protection. In *Bulk Power System Dynamics and Control-IX Optimization, Security and Control of the Emerging Power Grid (IREP), 2013 IREP Symposium*, pages 1–9. IEEE, 2013.

[81] Hyojoon Kim and Nick Feamster. Improving network management with software defined networking. *Communications Magazine, IEEE*, 51(2):114–119, 2013.

[82] Istvan Kiss, Bela Genge, and Piroska Haller. A clustering-based approach to detect cyber attacks in process control systems. In *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*, pages 142–148, July 2015.

[83] István Kiss, Béla Genge, Piroska Haller, and Gheorghe Sebestyén. Data clustering-based anomaly detection in industrial control systems. In *Proceedings of the 2014 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 275–281, Cluj Napoca, Romania, Sep. 2014. IEE.

[84] Theodora Kourti. Process analysis and abnormal situation detection: from theory to practice. *Control Systems, IEEE*, 22(5):10–25, 2002.

[85] Marina Krotofil, Alvaro Cardenas, and Kishore Angrishi. Timing of cyber-physical attacks on process control systems. In *International Conference on Critical Infrastructure Protection*, pages 29–45. Springer, 2014.

[86] Marina Krotofil and Alvaro A Cárdenas. Resilience of process control systems to cyber-physical attacks. In *Nordic Conference on Secure IT Systems*, pages 166–182. Springer, 2013.

[87] Marina Krotofil and Jason Larsen. Rocking the pocket book: Hacking chemical plants for competition and extortion. *DEF CON 23*, 2015.

[88] Marina Krotofil, Jason Larson, and Dieter Gollmann. The Process Matters: Ensuring Data Veracity in Cyber-Physical Systems. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, pages 133–144. ACM, 2015.

[89] Martin Krzywinski, Jacqueline Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J Jones, and Marco A Marra. Circos: an information aesthetic for comparative genomics. *Genome Research*, 19(9):1639–1645, 2009.

[90] M. Kumar and M. Hanumanthappa. Scalable intrusion detection systems log analysis using cloud computing infrastructure. In *Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on*, pages 1–4, Dec 2013.

[91] Doug Laney. 3D data management: Controlling data volume, velocity and variety. *META Group Research Note*, 6, 2001.

[92] Ralph Langner. Stuxnet: Dissecting a Cyberwarfare Weapon. *Security Privacy, IEEE*, 9(3):49–51, May 2011.

[93] Truls Larsson, Kristin Hestetun, Espen Hovland, and Sigurd Skogestad. Self-Optimizing Control of a Large-Scale Plant: The Tennessee Eastman Process. *Industrial & Engineering Chemistry Research*, 40(22):4889–4901, 2001.

[94] Pedro HB Las-Casas, Vinicius Santos Dias, Wagner Meira, and Dorgival Guedes. A big data architecture for security data and its application to phishing characterization. In *Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on*, pages 36–41. IEEE, 2016.

[95] Robert Layton, Paul Watters, and Richard Dazeley. Unsupervised authorship analysis of phishing webpages. In *Communications and Information Technologies (ISCIT), 2012 International Symposium on*, pages 1104–1109. IEEE, 2012.

[96] Yeonhee Lee, Wonchul Kang, and Youngseok Lee. An Internet traffic analysis method with MapReduce. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 357–361, April 2010.

[97] Yeonhee Lee, Wonchul Kang, and Youngseok Lee. A Hadoop-Based Packet Trace Processing Tool. In Jordi Domingo-Pascual, Yuval Shavitt, and Steve Uhlig, editors, *Traffic Monitoring and Analysis*, volume 6613 of *Lecture Notes in Computer Science*, pages 51–63. Springer Berlin Heidelberg, 2011.

[98] Yeonhee Lee and Youngseok Lee. Detecting DDoS Attacks with Hadoop. In *Proceedings of The ACM CoNEXT Student Workshop*, page 7. ACM, 2011.

[99] Shun-Te Liu and Yi-Ming Chen. Retrospective detection of malware attacks by cloud computing. *International Journal of Information Technology, Communications and Convergence*, 1(3):280–296, 2011.

[100] Zhiqiang Luo, Jun Shen, Huamin Jin, and Dongxin Liu. Research of Botnet Situation Awareness Based on Big Data. In *Web Technologies and Applications*, pages 71–78. Springer, 2015.

[101] John F MacGregor and Theodora Kourti. Statistical process control of multivariate processes. *Control Engineering Practice*, 3(3):403–414, 1995.

[102] Salvatore T March and Gerald F Smith. Design and natural science research on information technology. *Decision support systems*, 15(4):251–266, 1995.

[103] Samuel Marchal, Xiuyan Jiang, Radu State, and Thomas Engel. A Big Data Architecture for Large Scale Security Monitoring. In *2014 IEEE International Congress on Big Data (BigData Congress)*, pages 56–63, Anchorage, AK, USA, 2014. IEEE Computer Society.

[104] Johan Mazel, Romain Fontugne, and Kensuke Fukuda. Visual comparison of network anomaly detectors with chord diagrams. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 473–480. ACM, 2014.

[105] Thomas McEvoy and Stephen Wolthusen. A plant-wide industrial process control security problem. In *Critical Infrastructure Protection V*, pages 47–56. Springer, 2011.

[106] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation

in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[107] Bill Miller and Dale Rowe. A survey of SCADA and Critical Infrastructure incidents. In *Proceedings of the 1st Annual conference on Research in information technology*, pages 51–56. ACM, 2012.

[108] Robert Mitchell and Ingray Chen. A Survey of Intrusion Detection Techniques for Cyber Physical Systems. *ACM Computing Surveys*, 46(4), April 2014.

[109] Masataka Mizukoshi and Masaharu Munetomo. Distributed denial of services attack protection system with genetic algorithms on Hadoop cluster computing framework. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 1575 – 1580, Sendai, Japan, May 2015. IEEE.

[110] Elias Molina, Eduardo Jacob, Jon Matias, Naiara Moreira, and Armando Astarloa. Using software defined networking to manage and control IEC 61850-based systems. *Computers & Electrical Engineering*, 43:142–154, 2015.

[111] Thomas Morris and Wei Gao. Industrial Control System Traffic Data Sets for Intrusion Detection Research. In Jonathan Butts and Sujeet Shenoi, editors, *Critical Infrastructure Protection VIII*, volume 441 of *IFIP Advances in Information and Communication Technology*, pages 65–78. Springer Berlin Heidelberg, 2014.

[112] Thomas Morris, Anurag Srivastava, Bradley Reaves, Wei Gao, Kalyan Pavurapu, and Ram Reddi. A control system testbed to validate critical infrastructure protection concepts. *International Journal of Critical Infrastructure Protection*, 4(2):88–103, 2011.

[113] Prashanth Mundkur, Ville Tuulos, and Jared Flatow. Disco: a computing platform for large-scale data analytics. In *Proceedings of the 10th ACM SIGPLAN workshop on Erlang*, pages 84–89. ACM, 2011.

[114] A. Nicholson, S. Webber, S. Dyer, T. Patel, and H. Janicke. SCADA security in the light of Cyber-Warfare. *Computers & Security*, 31(4):418 – 436, 2012.

[115] Briony J Oates. *Researching Information Systems and Computing*. Sage, 2006.

[116] Marek Obitko, Václav Jirkovský, and Jan Bezdíček. Big Data Challenges in Industrial Automation. In Vladimír Mařík, JoseL.Martinez Lastra, and Petr Skobelev, editors,

Industrial Applications of Holonic and Multi-Agent Systems, volume 8062 of *Lecture Notes in Computer Science*, pages 305–316. Springer Berlin Heidelberg, 2013.

[117] Department of Justice, U.S. Attorney's Office, Middle District of Louisiana. Former systems administrator sentenced to prison for hacking into industrial facility computer system, Feb. 2017. [Online]. Available: `https://www.justice.gov/usao-mdla/pr/former-systems-administrator-sentenced-prison-hacking-industrial-facility-computer`. (Retrieved: 2017-02-27).

[118] Chhabi Rani Panigrahi, Mayank Tiwari, Bibudhendu Pati, and Rajendra Prasath. Malware Detection in Big Data Using Fast Pattern Matching: A Hadoop Based Comparison on GPU. In *Mining Intelligence and Knowledge Exploration*, pages 407–416. Springer, 2014.

[119] Henk-Jan Ramaker, Eric NM Van Sprang, Johan A Westerhuis, Stephen P Gurden, Age K Smilde, and Frank H Van Der Meulen. Performance assessment and improvement of control charts for statistical batch process monitoring. *Statistica Neerlandica*, 60(3):339–360, 2006.

[120] M Mazhar Rathore, Awais Ahmad, and Anand Paul. Real time intrusion detection system for ultra-high-speed big data environments. *The Journal of Supercomputing*, pages 1–22, 2016.

[121] Alan S. Ratner and Phillip Kelly. Anomalies in network traffic. In *Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on*, pages 206–208. IEEE, 2013.

[122] Bradley Reaves and Thomas Morris. An open virtual testbed for industrial control system security research. *International Journal of Information Security*, 11(4):215–229, 2012.

[123] N Lawrence Ricker. Decentralized control of the Tennessee Eastman challenge process. *Journal of Process Control*, 6(4):205–221, 1996.

[124] Bruce Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[125] Weiwei Shi, Yongxin Zhu, Tian Huang, Gehao Sheng, Yong Lian, Guoxing Wang, and Yufeng Chen. An integrated data preprocessing framework based on apache spark for

fault diagnosis of power grid equipment. *Journal of Signal Processing Systems*, pages 1–16, 2016.

[126] Christos Siaterlis, Alejandra P Garcia, and Béla Genge. On the use of Emulab testbeds for scientifically rigorous experiments. *Communications Surveys & Tutorials, IEEE*, 15(2):929–942, 2013.

[127] Christos Siaterlis, Bela Genge, and Marc Hohenadel. EPIC: a testbed for scientifically rigorous cyber-physical security experimentation. *Emerging Topics in Computing, IEEE Transactions on*, 1(2):319–330, 2013.

[128] Kamaldeep Singh, Sharath Chandra Guntuku, Abhishek Thakur, and Chittaranjan Hota. Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests. *Information Sciences*, 278:488–497, Sep. 2014.

[129] Jill Slay and Michael Miller. *Lessons learned from the Maroochy Water Breach*. Springer, 2007.

[130] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[131] Ljiljana Stojanovic, Marko Dinic, Nenad Stojanovic, and Aleksandar Stojadinovic. Big-data-driven anomaly detection in industry (4.0): An approach and a case study. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 1647–1652. IEEE, 2016.

[132] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to Industrial Control Systems (ICS) Security, Special publication 800-82. Technical report, National Institute of Standards and Technology, June 2011.

[133] Zachary G Stoumbos, Marion R Reynolds Jr, Thomas P Ryan, and William H Woodall. The state of statistical process control as we proceed into the 21st century. *Journal of the American Statistical Association*, 95(451):992–998, 2000.

[134] Nils Svendsen and Stephen Wolthusen. Using physical models for anomaly detection in control systems. In *Critical Infrastructure Protection III*, pages 139–149. Springer, 2009.

[135] Symantec Incident Response. Dragonfly: Cyberespionage attacks against energy suppliers. Technical report, Symantec, July 2014.

[136] Tim Tack, Alexander Maier, and Oliver Niggemann. On Visual Analytics in Plant Monitoring. In *Informatics in Control, Automation and Robotics*, pages 19–33. Springer, 2014.

[137] Hajime Tazaki, Kazuya Okada, Yuji Sekiya, and Youki Kadobayashi. MATATABI: Multi-layer Threat Analysis Platform with Hadoop. In *3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, 2014.

[138] SHINE team. Project SHINE (SHodan INtelligence Extraction) findings report. Technical report, Infracritical, Oct 2014.

[139] Jakrarin Therdphapiyanak and Krerk Piromsopa. An analysis of suitable parameters for efficiently applying K-means clustering to large TCPdump data set using Hadoop framework. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2013 10th International Conference on*, pages 1–6, May 2013.

[140] Jakrarin Therdphapiyanak and Krerk Piromsopa. Applying Hadoop for log analysis toward distributed IDS. In *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, page 3. ACM, 2013.

[141] Mark Thomas, Leigh Metcalf, Jonathan Spring, Paul Krystosek, and Katherine Prevost. SiLK: A Tool Suite for Unsampled Network Flow Analysis at Scale. In *Big Data (BigData Congress), 2014 IEEE International Congress on*, pages 184–191, Anchorage, AK, USA, Jun 2014. IEEE.

[142] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[143] Geng Tian, Zhiliang Wang, Xia Yin, Zimu Li, Xingang Shi, Ziyi Lu, Chao Zhou, Yang Yu, and Dan Wu. TADOOP: Mining Network Traffic Anomalies with Hadoop. In Bhavani Thuraisingham, XiaoFeng Wang, and Vinod Yegneswaran, editors, *Security and Privacy in Communication Networks*, volume 164 of *Lecture Notes of the Institute*

for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 175–192. Springer, 2015.

[144] Shweta Tripathi, Brij Gupta, Ammar Almomani, Anupama Mishra, and Veluru Suresh. Hadoop Based Defense Solution to Handle Distributed Denial of Service (DDoS) Attacks. *Journal of Information Security*, 4(3):150–164, 2013.

[145] Vijay Vaishnavi and William Kuechler. Design research in information systems. 2004. [Online]. Available: http://desrist.org/design-research-in-information-systems/. (Retrieved: 2017-02-27).

[146] Emmanouil Vasilomanolakis, Shreyas Srinivasa, Carlos Garcia Cordero, and Max Mühlhäuser. Multi-stage attack detection and signature generation with ics honeypots. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 1227–1232, April 2016.

[147] Scott Wallace, Xinghui Zhao, Duc Nguyen, and Kuei-Ti Lu. Big data analytics on smart grid: Mining pmu data for event and anomaly detection. In Rajkumar Buyya, Rodrigo N. Calheiros, and Amir Vahid Dastjerdi, editors, *Big Data: Principles and Paradigms*, chapter 17, pages 417–429. Morgan Kaufmann, 2016.

[148] Ziyu Wang, Jiahai Yang, and Fuliang Li. An on-line anomaly detection method based on lms algorithm. In *Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific*, pages 1–6. IEEE, 2014.

[149] Ziyu Wang, Jiahai Yang, Hui Zhang, Chenxi Li, Shize Zhang, and Hui Wang. Towards online anomaly detection by combining multiple detection methods and storm. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pages 804–807. IEEE, 2016.

[150] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. *ACM SIGOPS Operating Systems Review*, 36(SI):255–270, 2002.

[151] Stefan Windmann, Alexander Maier, Oliver Niggermann, Christian Frey, Ansgar Bernardi, Ying Gu, Holger Pfrommer, Thilo Steckel, Michael Krüger, and Robert Kraus.

Big Data Analysis of Manufacturing Processes. *Journal of Physics: Conference Series*, 659(1):012055, 2015.

[152] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I. Jordan. Detecting Large-scale System Problems by Mining Console Logs. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 117–132, New York, NY, USA, 2009. ACM.

[153] Shun-Fa Yang, I.Wei-Yu Chen, and Yao-Tsung Wang. ICAS: An inter-VM IDS Log Cloud Analysis System. In *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pages 285–289, Sept 2011.

[154] Ting-Fang Yen, Alina Oprea, Kaan Onarlioglu, Todd Leetham, William Robertson, Ari Juels, and Engin Kirda. Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 199–208. ACM, 2013.

[155] Shen Yin, Xin Gao, Hamid Reza Karimi, and Xiangping Zhu. Study on support vector machine-based fault detection in tennessee eastman process. In *Abstract and Applied Analysis*, volume 2014. Hindawi Publishing Corporation, 2014.

[156] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pages 10–10, 2010.

[157] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[158] Mark Zeller. Myth or reality–Does the Aurora vulnerability pose a risk to my generator? In *Protective Relay Engineers, 2011 64th Annual Conference for*, pages 130–136. IEEE, 2011.

[159] Wei Zeng, Chi-Wing Fu, Stefan Müller Arisona, and Huamin Qu. Visualizing interchange patterns in massive movement data. In *Eurographics Conference on Visualization (EuroVis)*, volume 32, pages 271–280, 2013.

[160] Jianchao Zhang, Boon-Chong Seet, Tek-Tjing Lie, and Chuan Heng Foh. Opportunities for software-defined networking in smart grid. In *Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on*, pages 1–5. IEEE, 2013.

[161] Jingling Zhao, Shilei Chen, Mengchen Cao, and Baojiang Cui. Malware algorithm classification method based on big data analysis. *International Journal of Web and Grid Services*, 13(1):112–130, 2017.

[162] Teng Zhao, Dan Chia-Tien Lo, and Kai Qian. A Neural-Network Based DDoS Detection System Using Hadoop and HBase. In *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*, pages 1326–1331. IEEE, 2015.

[163] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. A taxonomy of cyber attacks on SCADA systems. In *Internet of things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing*, pages 380–388. IEEE, 2011.

[164] Bonnie Zhu and Shankar Sastry. SCADA-specific intrusion detection/prevention systems: a survey and taxonomy. In *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*, 2010.

[165] H.P. Zhu, Y. Xu, Q. Liu, and Y.Q. Rao. Cloud service platform for big data of manufacturing. *Applied Mechanics and Materials*, 456:178–183, 2014. cited By (since 1996)0.

[166] Artur Ziviani, Antonio Tadeu A Gomes, Marcelo L Monsores, and Paulo SS Rodrigues. Network anomaly detection using nonextensive entropy. *IEEE Communications Letters*, 11(12):1034–1036, 2007.

[167] Richard Zuech, Taghi M Khoshgoftaar, and Randall Wald. Intrusion detection and big heterogeneous data: A survey. *Journal of Big Data*, 2(1):1–41, 2015.

Colophon

This document was typeset by the author using the LaTeX 2ε typesetting system, originally developed by Leslie Lamport, based on Donald Knuth's TeX; and the BibTeX reference manager, created by Leslie Lamport and Oren Patashnik. The typesetting engine was Jonathan Kew's XeTeX. Source files were edited with Bram Molenaar's Vim text editor and its Vimtex plugin (created by Karl Yngve Lervåg) after careful planning and organization with Doc*ear*, first presented by Joeran Beel, Bela Gipp, Stefan Langer, and Marcel Genzmehr. The text fonts are Linux Libertine for serif, and Linux Biolinum for sans serif figures, captions and tables. Both fonts were initially developed by Philipp H. Poll. Figures were created using the following free and open-source applications and libraries: Inkscape, Dia, Seaborn, Matplotlib, D3.js and the MEDA toolbox.

The picture of the cover depicts blast furnace number five of the former Thyssen Meredich Ironworks steel mill in Duisburg, North Rhine-Westphalia, Germany. Founded in 1902 and closed in 1985, the factory produced 37 million tons of pig iron. Since 1991, there is a park on site, the Landschaftspark Duisburg-Nord. The author of the original photo is user *Tuxyso* from Wikimedia Commons, and was published under the Creative Commons BY-SA 3.0 license on said site.

* * *