DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Digital Twin-based Intrusion Detection for Industrial Control Systems

SEBA ANNA VARGHESE

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Digital Twin-based Intrusion Detection for Industrial Control Systems

Seba Anna Varghese

Master’s Programme, Communication Systems, 120 credits
Date: September 29, 2021

Supervisors: Zahra Alimadadi, Ali Balador
Examiner: Panagiotis Papadimitratos
School of Electrical Engineering and Computer Science
Host Organization: RISE Research Institutes of Sweden AB

© 2021 Seba Anna Varghese

Abstract

Digital twins for industrial control systems have gained significant interest over recent years. This attention is mainly because of the advanced capabilities offered by digital twins in the areas of simulation, optimization, and predictive maintenance. Some recent studies discuss the possibility of using digital twins for intrusion detection in industrial control systems. To this end, this thesis aims to propose a security framework for industrial control systems including its digital twin for security monitoring and a machine learning-based intrusion detection system for real-time intrusion detection. The digital twin solution used in this study is a standalone simulation of an industrial filling plant available as open-source. After thoroughly evaluating the implementation aspects of the existing knowledge-driven open-source digital twin solutions of industrial control systems, this solution is chosen. The cybersecurity analysis approach utilizes this digital twin to model and execute different realistic process-aware attack scenarios and generate a training dataset reflecting the process measurements under normal operations and attack scenarios. A total of 23 attack scenarios are modelled and executed in the digital twin and these scenarios belong to four different attack types, naming command injection, network DoS, calculated measurement injection, and naive measurement injection. Furthermore, the proposed framework also includes a machine learning-based intrusion detection system. This intrusion detection system is designed in two stages. The first stage involves an offline evaluation of the performance of eight different supervised machine learning algorithms on the labelled dataset. In the second stage, a stacked ensemble classifier model that combines the best performing supervised algorithms on different training dataset labels is modelled as the final machine learning model. This stacked ensemble model is trained offline using the labelled dataset and then used for classifying the incoming data samples from the digital twin during the live operation of the system. The results show that the designed intrusion detection system is capable of detecting and classifying intrusions in near real-time (0.1 seconds). The practicality and benefits of the proposed digital twin-based security framework are also discussed in this work.

Keywords

Digital Twin, Intrusion Detection Systems, Industry 4.0, Industrial Control Systems, Machine Learning, Stacked Ensemble Model

Sammanfattning

Digitala tvillingar för industriella styrsystem har fått ett betydande intresse under de senaste åren. Denna uppmärksamhet beror främst på de avancerade möjligheter som digitala tvillingar erbjuder inom simulering, optimering och förutsägbart underhåll. Några färska studier diskuterar möjligheten att använda digitala tvillingar för intrångsdetektering i industriella styrsystem. För detta ändamål syftar denna avhandling till att föreslå ett säkerhetsramverk för industriella styrsystem inklusive dess digitala tvilling för säkerhetsövervakning och ett maskininlärningsbaserat intrångsdetekteringssystem för intrångsdetektering i realtid. Den digitala tvillinglösningen som används i denna studie är en fristående simulering av en industriell fyllningsanläggning som finns tillgänglig som öppen källkod. Efter noggrann utvärdering av implementeringsaspekterna för de befintliga kunskapsdrivna digitala tvillinglösningarna med öppen källkod för industriella styrsystem, väljs denna lösning. Cybersäkerhetsanalysmetoden använder denna digitala tvilling för att modellera och exekvera olika realistiska processmedvetna attackscenarier och generera en utbildningsdataset som återspeglar processmätningarna under normala operationer och attackscenarier. Totalt 23 angreppsscenarier modelleras och utförs i den digitala tvillingen och dessa scenarier tillhör fyra olika angreppstyper, namnskommandoinjektion, nätverks -DoS, beräknad mätinjektion och naiv mätinjektion. Dessutom innehåller det föreslagna ramverket också ett maskininlärningsbaserat system för intrångsdetektering. Detta intrångsdetekteringssystem är utformat i två steg. Det första steget innebär en offline -utvärdering av prestanda för åtta olika algoritmer för maskininlärning övervakad på den märkta datauppsättningen. I det andra steget modelleras en staplad ensemble -klassificerarmodell som kombinerar de bäst presterande övervakade algoritmerna på olika etiketter för utbildningsdataset som den slutliga modellen för maskininlärning. Denna staplade ensemblemodell tränas offline med hjälp av den märkta datauppsättningen och används sedan för att klassificera inkommande dataprover från den digitala tvillingen under systemets levande drift. Resultaten visar att det konstruerade intrångsdetekteringssystemet kan upptäcka och klassificera intrång i nära realtid (0,1 sekunder). Det praktiska och fördelarna med den föreslagna digitala tvillingbaserade säkerhetsramen diskuteras också i detta arbete.

Nyckelord

Digital tvilling, Intrångsdetekteringssystem, Industri 4.0, Industriella styrsystem, Maskininlärning, Staplad ensemblemodell

Acknowledgments

The completion of this thesis could not have been possible without the guidance and support of many people. I would like to take this opportunity to express my sincere gratitude to each one of them.

Firstly, I would like to thank my industrial supervisor, *Ali Balador* for offering me the opportunity to do my Master Thesis in his research group at RISE. I am thankful for his overall support and insights in this field. I would also like to thank my examiner, *Prof. Panagiotis Papadimitratos* for his consultancy and comments which helped to define and refine the thesis scope better. I am highly indebted and grateful to my academic supervisor, *Zahra Alimadadi* for her motivation and helpful comments. I truly appreciate her kind support and availability whenever I needed them the most.

Thanks a ton to my family for being my constant source of strength and support. Above all, I thank God for helping me to complete this phase of my life journey.

Stockholm, September 2021
Seba Anna Varghese

Contents

List of Figures

List of Tables

Listings

List of acronyms and abbreviations

ARP	Address Resolution Protocol
CIP	Common Industrial Protocol
CPS	Cyber-Physical System
CSV	Comma Separated Values
DCS	Distributed Control System
DoS	Denial of Service
ENIP	EtherNet/Industrial Protocol
HMI	Human Machine Interface
IACS	Industrial Automation and Control System
ICS	Industrial Control System
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
MitM	Man in the Middle
ML	Machine Learning
OT	Operational Technology
PLC	Programmable Logic Controller
SCADA	Supervisory Control And Data Acquisition
SIEM	Security Information and Event Management

SYN	SYN flag
TCP	Transmission Control Protocol
VM	Virtual Machine

Chapter 1

Introduction

Industrial Control System (ICS)s are part of traditional Operational Technology (OT) infrastructure that is used in industries such as electric, oil and gas, and water, to monitor, supervise and control machines and processes. With Industry 4.0, ICSs are increasingly connected to communication networks and integrated into general-purpose Information Technology (IT) systems through Industrial Internet of Things (IIoT) [5]. This increased interconnectivity has improved the efficiency, performance, and manageability of the systems; but on the downside, these systems have become more exposed to cyberattacks than before.

Cyberattacks against ICSs aim at disrupting critical industrial processes [6] which can cause catastrophic consequences, including physical damages and casualties. Few examples of major cyberattacks that happened in recent years: the Stuxnet malware attack on the nuclear facility of Natanz, Iran, in 2010, which disrupted the nuclear program of Iran [7], a cyberattack on a Ukrainian power grid in 2015 caused a power outage affecting around 225,000 customers [8], the cyberattacks against natural gas pipeline companies in the US in 2018 [9], and the attack against Florida water treatment facility in 2021 to manipulate with water's chemical treatment levels [10].

As ICSs constitute the backbone of critical infrastructures, it is necessary to ensure the security and safety of these systems. Intrusion detection is an efficient mechanism to enhance security in ICSs. Timely detection of intrusions in ICSs alerts operators on malicious activities and enables them to apply safety countermeasures. Given the nature of industrial processes controlled by ICSs, implementing security engines inside live industrial environments can have adverse impacts on the performance and efficiency of such systems. To solve this issue, intrusion detection in ICSs can be done in the digital domain and digital twin-based intrusion detection is a novel breakthrough in this area. Digital twins can be considered as the virtual representations of physical systems that can mirror both static and dynamic characteristics of their physical counterparts [11] in near real-time.

This thesis aims to propose a security framework for an ICS that includes the digital twin of the ICS for security monitoring and a Machine Learning (ML)-based Intrusion Detection System (IDS) capable of detecting intrusions in near real-time. This framework is realized as an extension of the framework proposed by Dietz et al. [3]. Digital twin in this framework is a standalone simulation of an industrial filling plant (More details on the framework are provided in 3.2.3). The main contributions of this work are as listed below:

1. A flexible and extensible security framework for ICS including its digital twin and an ML-based IDS. We realize this framework by extending an open-source available digital twin framework with an intrusion detection module.

2. Comparison between the performances of different supervised ML algorithms in solving the intrusion detection problem and designing a stacked ensemble classifier model for implementing an ML-based IDS in the proposed security framework.

3. A proof of concept implementation of the proposed framework showing how to perform security monitoring and security analysis in ICS by modelling and executing different types of attacks in the digital twin. We also evaluate this framework in terms of its ability to detect and classify different types of attacks.

1.1 Background

The primary focus for ICSs is to ensure the availability of industrial operations. To achieve this, it is necessary to protect ICSs against cyberattacks. IDSs play an important role in enhancing security in ICSs. However, it is not a practical solution to apply security testing in live running industrial systems as this can tamper with the efficiency and availability of critical infrastructures. Another possible solution is to implement and maintain physical security test-beds or test environments. This solution is an expensive and time-consuming one, and usually leads to incomplete and out-of-date setups [12]. Digital twin-based security testing and intrusion detection in ICSs can solve the above-mentioned problems. Security analysis performed using digital twins does not interfere with the live running systems as they are completely run in the digital domain [9]. Since the digital twin covers the whole life-cycle of a physical system, it always represents an up-to-date version of the physical system [13]. Digital twin-based solutions also offer the possibility to include methods that need more computing resources for cybersecurity analysis (for example, ML and deep learning) than if deployed in the real physical systems [9].

One important question that arises is that *why systems designed to protect IT systems cannot be used for securing OT systems*. This is due to the differences in security design requirements of both these systems [14]. While IT systems focus on the confidentiality and integrity of the data, OT systems focus on the availability of the systems. Security measures deployed in ICSs must not disrupt the real-time operation of industrial processes, whereas IT systems can tolerate a moderate amount of delay [14]. For the same reason, security updates happen less frequently in OT systems compared to IT systems. As ICSs interact with the physical world and processes, attacks against these systems target the physical process and are capable of causing serious damages. Such attacks usually bypass the traditional security mechanisms and do not violate any protocol specifications as these attacks are carefully crafted after acquiring knowledge about the controlled industrial process [14].

Commercial digital twin solutions of ICSs are not easily accessible for research purposes. This is mainly because of the possible data breach of sensitive information regarding the systems used in industrial environments if they are made accessible. This calls for the availability of open-source implementations of digital twin solutions of ICSs for research purposes. However, there are not many approaches available that discuss the design and implementation aspects of digital twin using open-source tools and software. In this thesis, existing open-source digital twin solutions of ICSs are investigated and a digital twin-based security framework is proposed. This framework includes the digital twin of ICS for security monitoring and analysis and an ML-based IDS for intrusion detection. Such a framework using open-source tools and software is beneficial for researchers and students in the field of cybersecurity in ICSs. However, this thesis does not focus on the synchronization aspects between the digital twin and its physical twin; but provides some indicators in this direction.

1.2 Problem Statement

ICSs are increasingly being connected to communication networks, making them more vulnerable to cyberattacks. Due to the importance of industrial operations handled by these systems, it is quite important to ensure the security of such systems against attacks. IDSs are one of the many defense mechanisms that can ensure ICS security. If intrusion detection can be performed in the digital domain, it can offer more possibilities with respect to computing resources and zero adverse impact on the efficiency of the running systems. A digital twin is an enabler for such a system that mimics the physical system in the digital domain in near real-time. Hence digital twin-based security analysis and intrusion detection prove to be an efficient measure to secure ICSs against cyberattacks.

As part of this thesis, the following research questions will be examined:

1. How to perform cybersecurity analysis in ICS using its digital twin which is implemented using open-source tools?

2. How to carry out ML-based intrusion detection in ICS using inputs from its digital twin implementation?

1.3 Research Goals and Objectives

The main goal of this thesis is to deliver a digital twin-based security framework to perform security analysis and ML-based intrusion detection in ICSs. This goal is further divided into the following three objectives:

1. Identify an open-source digital twin solution of an ICS to perform cybersecurity analysis in ICS.

   Related tasks to achieve this objective are:

   (a) Investigate and compare different digital twin-based security use cases and solutions available for ICSs.

   (b) Based on this investigation, identify an open-source digital twin implementation of an ICS and familiarise with the open-source tools and software used for this implementation.

2. Perform security monitoring and analysis of the ICS using the selected digital twin solution. Related tasks to achieve this objective are:

   (a) Model different process-aware attack scenarios aimed at disrupting the industrial control process.

   (b) Generate labelled dataset for the modelled attack scenarios that can be used to train ML-based IDS.

3. Integrate an ML-based IDS to the digital twin-based security framework that is capable of detecting process-aware intrusions in near real-time.

   Related tasks to achieve this objective are:

   (a) Design an ML-based algorithm upon offline evaluation of different supervised ML algorithms on the generated labelled dataset.

   (b) Demonstrate the use case of ML-based intrusion detection using the proposed framework during the live operation of the system.

1.4 Research Methodology

In this thesis, we use both qualitative and quantitative research methods. The qualitative approach includes the literature review and study associated with the thesis objectives. The qualitative methods focus on modelling and executing different process-aware attacks on the digital twin, collecting and generating dataset reflecting the process measurements, and detecting intrusions using ML-based IDS trained on the generated dataset.

The research methodology used in this thesis involves four steps, as shown in Figure 1.1. The first step is the literature review which includes the study of existing digital twin-based security solutions of ICSs, modelling of attacks against ICSs, and ML-based intrusion detection approaches used in ICSs. The outcome of this step is the formulation of research questions to be addressed as part of the thesis along with the definition of research goals. As the next step, a design of the proposed solution to address the research questions is modelled. The third step of the research methodology is the proof of concept implementation of the proposed design. Finally, an evaluation is performed to measure the effectiveness of the proposed solution.

We explain in detail each of the four steps of the research methodology in the upcoming chapters. Step 1, i.e., literature review is discussed in Section 2.5. Chapter 3 discusses the design step whereas Chapter 4 provides details on the implementation step. Finally, evaluation methodology is discussed in Section 3.2.3, and evaluation results are presented in Chapter 5.

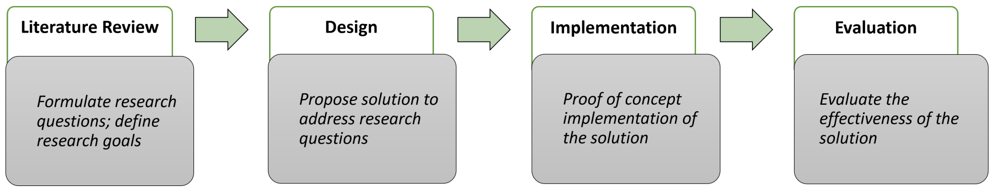

Literature Review
Formulate research questions; define research goals
Design
Propose solution to address research questions
Implementation
Proof of concept implementation of the solution
Evaluation
Evaluate the effectiveness of the solution

Figure 1.1: Research Methodology

1.5 Benefits, Ethics, and Sustainability

Digital twins can be built even long before the manufacturing and production of actual physical systems. Proper implementation and testing of these twins can be beneficial to address sustainability issues in the industry. Some of these benefits include quicker production cycles, minimal adverse environmental impacts, and reduced wastage of resources and production cost. On the other hand, digital twins developed for already existing systems can be helpful for the early detection of anomalies and intrusions, making it possible to take safety countermeasures before it is too late. Thus, such systems also address the adverse ethical impacts caused by cyberattacks against ICSs.

Digital twins need to be properly isolated and only authorized persons must be provided access. However, if the digital twin of an ICS is compromised, it can cause serious ethical and sustainability issues. Ethical issues include the disruption of industrial processes by attackers who have gained access to these systems via the compromised digital twin. This disruption can cause serious damages including casualties and wastage of resources. This further raises sustainability concerns. It is also necessary to securely store and manage data collected by digital twins from the physical environment. Otherwise, this leaked data regarding critical industrial process control parameters can be used by the attackers to acquire knowledge about the industrial control processes and devise attacks targeting specific processes.

1.6 Delimitations

The digital twin solution of ICS used in this thesis is a standalone entity; meaning there is no physical system running in parallel with the digital twin. To achieve interaction with the physical system, there needs to be a synchronization mechanism between these systems, and implementation of this mechanism is not in the scope of this thesis. It is assumed that the digital twin receives all the traffic and input parameters from the physical system in practice. The standalone digital twin solution used in this thesis simulates the physical process to operate in standalone mode. Therefore, an attacker having access to the physical system is represented as an attacker inside the digital twin network topology. However, digital twins are run in isolated and secure environments (for example, private cloud) when deployed as real commercial solutions.

The algorithms chosen to design and evaluate the ML-based IDS are purely based on the comparison of performance results of some of the most common supervised ML algorithms mentioned in the state-of-the-art research work in the area of intrusion detection in ICSs. There may be other efficient and better-performing supervised algorithms that are not considered in this work.

1.7 Structure of the Thesis

The remainder of the report is organized as follows. Chapter 2 presents necessary background information about the research topic and the related work. The proposed methodology and the research process used in the thesis are described in Chapter 3. Information regarding implementation aspects of the thesis is presented in Chapter 4. Chapter 5 presents the results as well as the analysis and discussion of the results. Chapter 6 presents the conclusions and also states the limitations of the thesis and future research directions.

Chapter 2

Background

In this chapter, Sections 2.1 to 2.4 provide necessary background information to the readers to follow this thesis. Related research works in the field of digital twin-based security solutions in ICSs, modelling of attacks against ICSs, and ML-based intrusion detection approaches in ICSs are discussed in Section 2.5. Finally, Section 2.6 introduces the metrics used to evaluate ML algorithms in this work.

2.1 Industry 4.0

Industry 4.0 is the term associated with the latest industrial revolution that marked the introduction of Cyber-Physical System (CPS)s. These systems are basically ICSs that used to be standalone entities operating in enclosed architectures and have now become highly interconnected to the Internet and hence open to remote access [15]. Industry 4.0 offers several benefits such as improved performance, increased production rate, reduced cost and waste of resources, and better manageability of systems [1].

Figure 2.1 illustrates the key enablers of the industrial revolution from its initial version 1.0 to the latest version 4.0.

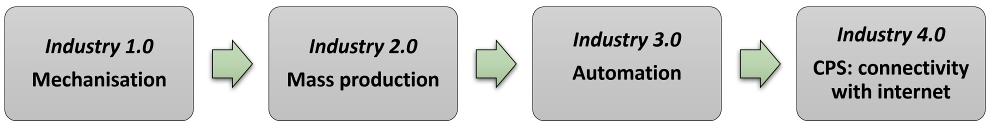


Figure 2.1: Industrial revolution from 1.0 to 4.0 (Adapted from [1])

2.2 Industrial Control Systems

ICSs are responsible for real-time system monitoring, collection and analysis of data, and automatic control and management of industrial processes [16] [17]. Some of the well-known ICSs are Supervisory Control And Data Acquisition (SCADA) systems, Industrial Automation and Control System (IACS)s and Distributed Control System (DCS)s [18]. Figure 2.2 depicts a simple ICS architecture. A short description of the roles of major components and devices that constitute an ICS as shown in Figure 2.2 is provided below. These devices are listed according to the hierarchical layer (from top to bottom) in which they operate in an ICS.

- **Human Machine Interface (HMI):** This is the interface via which operators can access and communicate with the controller hardware in the system. HMI allows operators to monitor processes, fetch the status of devices, and send control commands to field devices [19].

- **Programmable Logic Controller (PLC):** This constitutes the controller hardware of ICS. PLCs interact with field devices by sending operational control commands and also receive status updates from field devices [19]. Moreover, HMI interacts with PLCs to acquire status information of field devices.

- **Sensors and actuators:** These are field devices that reflect the state of the industrial environment (for example, the liquid level in a tank) [19]. PLCs use this information to perform control operations. Sensors are used to collect data whereas actuators are the ones that physically perform control actions [20].

With Industry 4.0, ICSs become increasingly connected to communication networks and become more intelligent and open [16]. But this has also increased the attack surface for cyberattacks against ICSs. Industrial protocols used for control and communication purposes in ICSs are different from those used in IT systems. A brief discussion on this topic is provided in Section 2.2.1. It is also important to understand the security vulnerabilities of ICSs and the nature of attacks against these systems. This information is provided in Section 2.2.2.

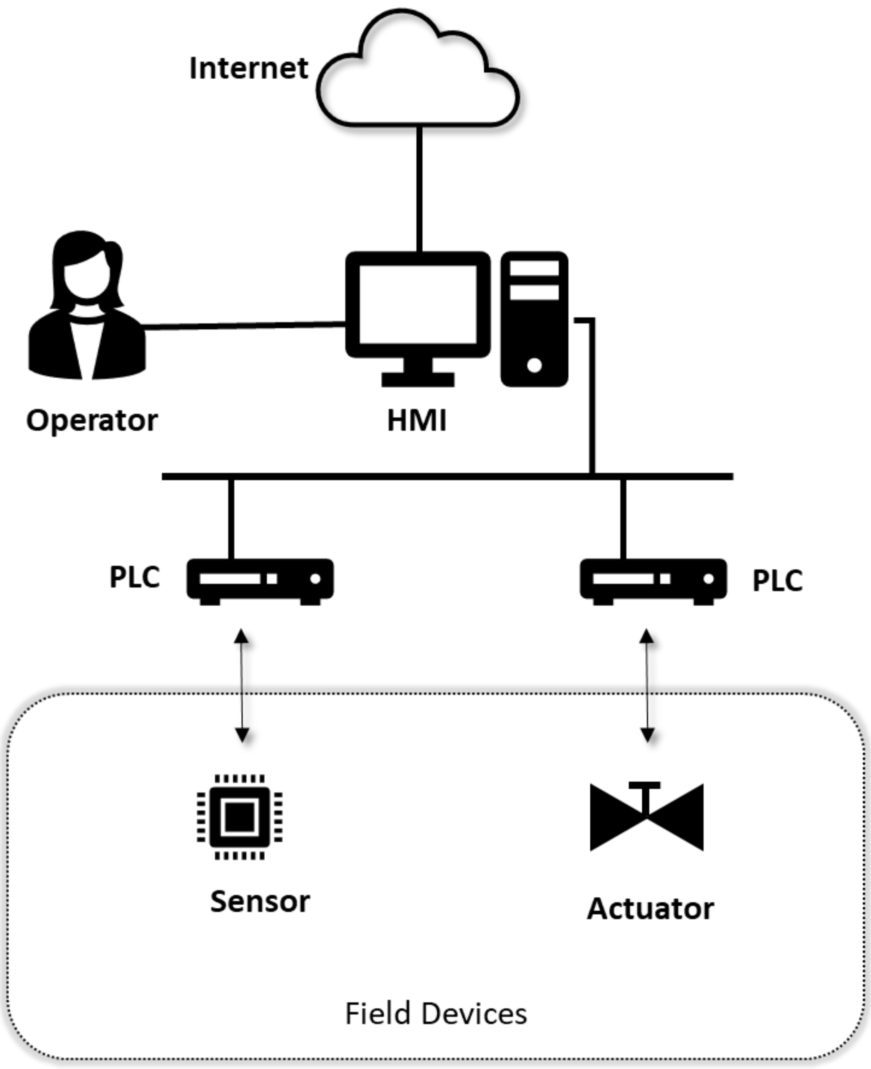


Figure 2.2: Simple ICS architecture

2.2.1 Industrial Communication Protocols

The components of an ICS use custom protocols to achieve serial communication with each other. These protocols were developed long before Industry 4.0 emergence and were not designed then foreseeing the openness to the Internet. Hence, these protocols lack security design principles such as authentication and encryption [19]. This has become one of the major vulnerabilities used for attacks against ICSs.

Table 2.1: Statistics of the number of ICS devices connected to the Internet using different industrial communication protocols as provided by Shodan search engine (Adapted from [4])

Protocols	**No. of ICS devices connected to the Internet**
BACnet	10530
DNP3	588
Ethernet/IP	3943
Modbus	13949

Table 2.1 provides the statistics of the number of ICSs using different industrial communication protocols such as BACnet, DNP3, EtherNet/Industrial Protocol (ENIP), and Modbus, connected to the Internet as provided by the Shodan search engine [4]. These statistics clearly show the impact of Industry 4.0 in increasing the connectivity of ICSs to the Internet. The digital twin simulation used in this thesis uses ENIP [21] protocol.

ENIP is one of the widely used industrial network protocols that provides the Ethernet-based implementation of Common Industrial Protocol (CIP) [22]. CIP includes messages and services for a wide range of industrial automation applications [23]. CIP uses a producer-consumer object model [23] to query readings from industrial components such as sensors and actuators, and set configurations [22]. In CIP object model, sensor readings are stored as tags [22].

2.2.2 Security Vulnerabilities and Challenges

This subsection discusses some of the major vulnerabilities and challenges that exist in ICSs which make it difficult to enhance security measures in these systems. Most of the ICSs comprise legacy sub-systems and are difficult to upgrade [16]. These legacy systems are poorly secured and hence vulnerable to security threats [24]. It is difficult to stop the operation of ICSs for fixing bugs and installing software updates [16], as this adversely affects the stability and smooth running of industrial operations handled by these systems. Moreover, the components and devices present in ICSs such as PLCs, sensors, and actuators have limited computing and storage resources [16]. Running security applications on these devices is not recommended for the same reasons. Industrial protocols developed originally to work in closed environments become vulnerable to attacks when connected to the Internet due to lack of built-in security mechanisms [16][24]. This vulnerability increases the risk of exposing critical and sensitive process information to attackers [16]. The digital twin simulation used in this thesis uses ENIP protocol which uses unencrypted messaging [25]. In our work, we use this lack of encryption vulnerability of the ENIP protocol to model measurement injection attacks. Furthermore, CIP lacks authentication control [20], and hence ENIP which adapts CIP to the Ethernet suffers from this security vulnerability. Command injection attack modelled in this work exploits this vulnerability of the ENIP protocol.

2.3 Digital Twin

In simple words, a digital twin can be considered as an up-to-date representation of a physical system in operation. Figure 2.3 depicts the concept of the digital twin. The digital twin receives input signals and data from its physical twin and mirrors the internal behavior of the physical system [9]. The digital twin can also send process control information to its physical twin if needed. For example, if a security vulnerability is detected using digital twin-based security testing, it can send control signals to the physical system to take preventive measures.

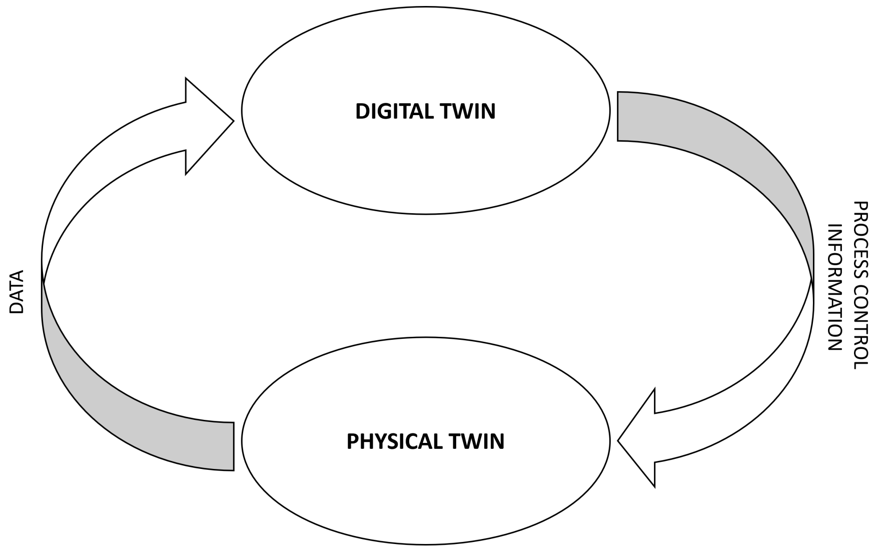


Figure 2.3: Digital twin concept (Adapted from [2])

Although the key idea behind the digital twins is to enhance the production lifecycle of systems in the manufacturing industry, some recent works discuss the application of digital twin to enhance the security of ICSs (more details are provided in Section 2.5). In this thesis, the security use case of the digital twin for intrusion detection purposes in an ICS is explored. Digital twin-based security analysis provides an efficient way to conduct security analysis outside the real infrastructure, thus avoiding disruptions and damage caused if such an analysis is performed on the actual system [26]. Moreover, using digital twin for security analysis in ICS is a better approach compared to using security testbeds. This is because of the capability of digital twins to dynamically reflect the state of physical systems in real-time while testbeds represent only a static model [27].

According to the authors of [28], there are two possible implementations for digital twins of ICSs: (i) information/knowledge-driven, and (ii) data-driven. In the first approach, information regarding the physical systems (for example, specification of systems) and details regarding the physical process controlled by the physical systems are used to model a virtual prototypical digital twin simulation model. Such an implementation does not require a physical system to be run in parallel with the digital twin. However, this implementation relies solely on the specification of the physical system, and hence it is necessary to make the correct specification details available. On the other hand, the second approach does not require the specification of the system beforehand and instead uses real-time data from devices in the physical environment as inputs to form a model of the system. Such an implementation requires synchronization between the physical system and its digital twin for real-time data acquisition.

In this thesis, existing open-source digital twin-based security solutions of ICSs are investigated in terms of their implementation aspects. Based on this investigation, a standalone digital twin solution of an ICS (solution from [3]) is identified and used in this thesis. The identified solution falls under the first category of possible implementation approaches of digital twins, i.e., knowledge-driven digital twins, as described in [28]. More details on the results of this investigation and the rationale behind choosing this particular solution are provided in Chapter 3.

2.4 Intrusion Detection

Intrusion detection is an efficient way to detect malicious activities and abnormal behaviors caused by cyberattacks in ICSs. IDS for ICS uses several data sources, such as network traffic, system logs, and process measurements, and analyses this information to detect abnormal patterns and potential attacks [14].

Based on the analysis of data collected by IDS, intrusion detection in ICSs can be categorized as (i) protocol-analysis based, (ii) traffic-analysis based, and (iii) process-analysis based [16]. The first two categories check for protocol specification violations, and abnormalities in network traffic data for detecting attacks. The third category mainly targets in detecting semantic attacks performed against ICSs; these types of attacks use information about the physical systems and the controlled processes to cause damage to ICSs [16]. This thesis focuses on detecting semantic attacks against ICSs, and hence a process-analysis based IDS is proposed.

Process-analysis based intrusion detection can be implemented either using approaches based on the dynamic model of the system or using ML-based approaches [29]. Dynamic model based process-aware IDS uses a system model and detect anomalies that deviate from the equations and control laws governing the behavior of the system [29]. This type of IDSs always require a well-defined mathematical model of the ICS and are often very complex and are specific to the ICS for which these are implemented. On the other hand, ML-based IDSs use a data-driven approach to train models using information from devices in the ICSs to detect deviations from normal operations [29]. This type of IDSs is less complex and can generalize easily to the changes made in the ICS [29]. Because of the benefits offered by the ML-based approach, this thesis focuses on the implementation of an ML-based process-aware IDS.

For successful intrusion detection using ML algorithms, it is important to train the models using high quality dataset. This calls for the need for meaningful data collection and feature extraction as prerequisites for building an intrusion model that offers high accuracy and low false positives [30]. By meaningful data collection, we mean the right set of measurements are collected for generating the dataset. This will be helpful in cases where some measurements are always constant (for example, upper and lower limit values) and do not provide any additional information to the ML model. Furthermore, it may be also required to extract only relevant features in those cases where the input feature dimension space is huge. Based on the dataset used to train the models, intrusion detection can use supervised algorithms (using labelled dataset) or unsupervised algorithms (using unlabelled dataset).

The scope of this thesis is to model and execute realistic attack scenarios belonging to different attack types as insider attacks in the digital twin implementation, and to generate labelled intrusion datasets having process measurements as attributes that can be used by supervised ML algorithms for intrusion detection. The labelled dataset contains data samples belonging to normal operation as well as different attack types.

2.5 Related Work

This section discusses three key areas of related work surrounding the research topic of the thesis: (i) digital twin-based security solutions for ICSs, (ii) modelling of attacks against ICSs, and (iii) ML-based intrusion detection in ICSs.

2.5.1 Digital twin-based Security Solutions for ICSs

Eckhart et al. [12] have proposed a framework called *CPS Twinning* that can automatically generate the digital twin of an ICS using Mininet-WiFi from the specification of ICS. This framework supports two operation modes of digital twin: (i) simulation mode where there is no need for co-existence of the physical system, and (ii) replication mode that supports synchronization with the physical system. Another work from the same authors [31] discusses a specification-based passive state replication approach to achieve synchronization between the digital twin and physical twin in the CPS Twinning framework. This work also demonstrates a proof of concept implementation of rule-based intrusion detection for a Man in the Middle (MitM) attack.

A digital twin-based security architecture for IACS is proposed by Gehrmann et al. [2]. This work mainly focuses on detailing the security requirements for different components of the proposed architecture. It also puts forward the concept of an active state replication approach using clock synchronization at regular intervals to achieve synchronization between physical twin and digital twin. Intrusion detection is mentioned as a module in the whole architecture and its implementation is left for future work.

Kamath et al. [32] present the capability of using open-source platforms to achieve digital twin capabilities, including real-time data acquisition from the Internet of Things (IoT) devices, virtual representation, data analysis, and visualization. There is no discussion on a specific type of ICS or on any security applications that can be deployed using this framework.

The digital twin framework used in this thesis is from Dietz et al. [3]. In this work, the authors demonstrate the feasibility of integrating digital twin security simulations in a security operations center. The proposed framework is realized as a microservice architecture using Docker containers. Security simulation is achieved with digital twin implementation using Mininet [33] and MiniCPS [34] and security analytics is performed with a Security Information and Event Management (SIEM) module that uses a rule-based attack detection from system logs. More details on this framework are provided in Chapter 3.

Akbarian et al. [9] have proposed the inclusion of an intrusion detection algorithm in a digital twin implementation. They have used the digital twin implementation of their earlier work [35]. The intrusion detection solution presented is useful for attack detection and classification. Attack detection is based on the comparison of the estimated output signal using a Kalman Filter [36] with the output signal from the digital twin. Further, an ML approach using a multiclass support vector machine (SVM) is deployed to classify the detected attacks into different attack categories. The physical system used in this work is a MATLAB simulation of ball and beam process [37], and this does not represent a real ICS architecture. Also, the digital twin implementation is a simulated model of this experimental physical system using system identification algorithms.

2.5.2 Modelling of Attacks against ICSs

Morris et al. [38] present a set of 17 attacks that can be performed against SCADA systems that use the Modbus communication protocol. These attacks belong to four different attack classes; reconnaissance, command injection, response and measurement injection, and Denial of Service (DoS). The attacks are executed against ICSs in a lab environment, and most of the attacks are capable of creating network traffic violations. Although this work focuses specifically on the Modbus protocol, the description of executed attacks under different attack classes is applied in this thesis to simulate complex semantic attacks.

Ahmed et al. [39] provide a detailed summary of different data injection attacks executed on sensors and actuators of a real-world water treatment testbed, along with the expected impact of these attacks on the system. The executed attacks range from causing sudden changes in sensor and actuator measurements to quite slower and stealthy changes in these measurements. However, the authors of [39] do not discuss the implementation details of the executed attacks.

Griffith et al. [40] propose a set of attacks that can be used in IDSs tests for CPSs. The attacks designed as part of this work are grouped into 3 classes, naming reconnaissance, MitM, and DoS. This work also discusses the use of the Ettercap[1] tool to execute the MitM/DoS attack which is used in this thesis.

2.5.3 Intrusion Detection in ICSs

Intrusion detection mechanisms used in ICSs can be categorized into five groups, naming signature-based, anomaly-based, statistical-based, ML-based, and specification-based [14]. In this section, we discuss the related works that use supervised ML algorithms for intrusion detection in ICSs.

Bernieri et al. [41] provide analysis of supervised and unsupervised ML-based anomaly detection using datasets containing process measurements collected from the Secure Water Treatment (SWaT)[2] testbed. In this work, process measurements chosen from the dataset include only the ones from the normal operation of the testbed and a single attack type. Support vector machine (SVM), random forest (RF), and k nearest neighbor (KNN) are the supervised ML algorithms used and the evaluation metrics used to compare these algorithms are accuracy and F1-score. According to the evaluation results, supervised algorithms show better performance compared to unsupervised algorithms on the dataset used in this work. Furthermore, RF has the best scores for accuracy and F1-score among other supervised algorithms. However, this paper does not present an evaluation of the discussed algorithms on a dataset having measurements from more than one attack type.

Another work in this area is from Zolanvari et al. [17] which performs an evaluation of 7 supervised ML algorithms for detecting intrusions on a traffic dataset collected from a water treatment testbed of a SCADA system. The algorithms used in this work are support vector machine (SVM), k nearest neighbor (KNN), naïve Bayes (NB), random forest (RF), decision tree (DT), logistic regression (LR), and artificial neural network (ANN). The evaluation metrics used in this work take into account the target class imbalance present in the dataset, and hence uses metrics such as false alarm rate, sensitivity, undetected rate, and Matthews correlation coefficient, besides accuracy score. RF classifier provides the best results among all the other algorithms. This work also provides insights on different vulnerabilities that exist in SCADA systems and ways to exploit these vulnerabilities.

Gómez et al. [42] have proposed a methodology to generate a dataset for cybersecurity analysis using ML and deep learning algorithms in an electric traction substation. The generated dataset includes network traffic features

[1]https://www.ettercap-project.org/

[2]https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/

extracted from traffic captures in the network. Supervised ML algorithms evaluated in this work are support vector machine (SVM), random forest (RF), and neural network. Metrics used to evaluate the algorithms are precision, recall, and F1-score. According to their evaluation, RF provided the best results, followed by SVM and neural network.

Tamy et al. [43] provide an extensive comparison of the performance of 4 supervised ML algorithms that can be used for network intrusion detection in a gas pipeline dataset[3]. The algorithms used in this work are support vector machine (SVM), one rule (OneR), random forest (RF), and k nearest neighbor (KNN). In addition to this, the paper also discusses the advantage of using particle swarm optimization (PSO) to optimize these algorithms. Accuracy and F-measure are the metrics used for evaluation, and RF classifier optimized by PSO gives the best results.

2.6 Metrics for Evaluation of ML Algorithms

In this section, we introduce the metrics we use in this thesis for evaluating supervised ML algorithms. The ML algorithms which are trained on the labelled dataset are evaluated in terms of their ability to correctly classify the data samples. Since the labelled dataset has more than two labels, this becomes a multiclass classification problem. Therefore, the metrics used for the evaluation of different algorithms are the common metrics used for the evaluation of classification algorithms. Listed below are the metrics used for this evaluation:

1. *Confusion matrix:* This metric gives the tabular representation of different combinations of predicted and actual values. All other metrics used in this thesis to evaluate different algorithms are based on this metric. For a better understanding of this metric, an example confusion matrix of a binary classifier for an anomaly detection problem is shown in Figure 2.4. In this example, the prediction of class as 'Anomaly' is considered as a positive case whereas prediction as 'Normal' is a negative case. The terms 'TP', 'FN', 'FP', and 'TN' represent a whole number and are defined as follows:

    (a) TN (True Negative): Number of cases in which predictions are Normal when the actual values are Normal.

[3]`https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets`

(b) FN (False Negative): Number of cases in which predictions are Normal when the actual values are Anomaly.

(c) FP (False Positive): Number of cases in which predictions are Anomaly when the actual values are Normal.

(d) TP (True Positive): Number of cases in which predictions are Anomaly when the actual values are Anomaly.

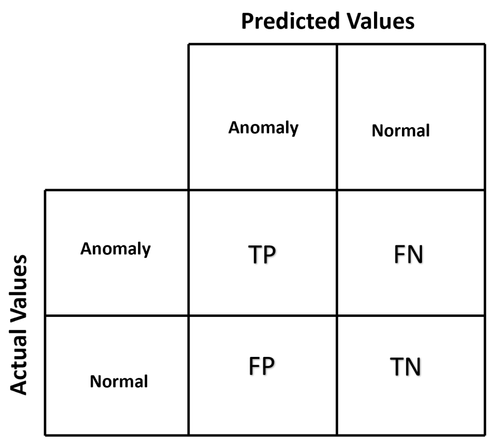


Figure 2.4: Confusion matrix example: binary classifier

The ideal scenario is when both FN and FP values are zero. However, in practical cases what needs to be minimized among these two values depends on the problem we are trying to solve.

For multiclass classification, the confusion matrix has more than two classes. In this case, there is no particular positive or negative class defined in prior. For each class, the prediction of that class is considered positive. The confusion matrix for multiclass classification with three classes is shown in Figure 2.5. The three classes defined are ’Normal’, ’Anomaly1’, and ’Anomaly2’. Each cell in the matrix is given a cell number (refer to the number highlighted at the right bottom). All the green circled cells represent the cases where predictions match the actual values, whereas the grey circled cells represent the opposite case. TP, FP, FN, and TN for each of the classes are calculated as shown in Figure 2.6. This can be further extended to the *NxN* matrix for a multiclass classifier with *N* classes.

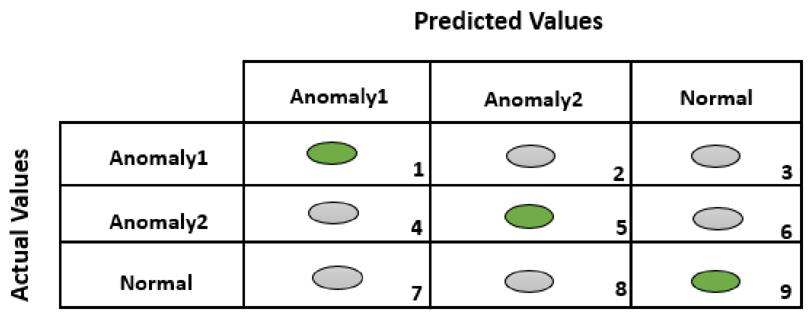
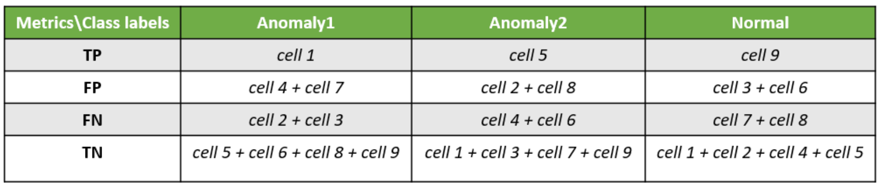


Figure 2.5: Confusion matrix example: multiclass classifier



Metrics\Class labels	Anomaly1	Anomaly2	Normal
TP	*cell 1*	*cell 5*	*cell 9*
FP	*cell 4 + cell 7*	*cell 2 + cell 8*	*cell 3 + cell 6*
FN	*cell 2 + cell 3*	*cell 4 + cell 6*	*cell 7 + cell 8*
TN	*cell 5 + cell 6 + cell 8 + cell 9*	*cell 1 + cell 3 + cell 7 + cell 9*	*cell 1 + cell 2 + cell 4 + cell 5*

Figure 2.6: Metrics calculation

2. *Accuracy:* This metric represents the number of correct predictions made by the classifier for a class over the total predictions made. This is calculated as follows:

$$Accuracy = \frac{TN + TP}{FN + FP + TN + TP}$$

For multiclass classification, macro averaged accuracy score across all classes is taken into account.

3. *Precision:* This metric gives the proportion of cases in which the model predicts a value as class x when it is actually class x. This is calculated as follows:

$$Precision = \frac{TP}{TP + FP}$$

This is a good metric to use in our case as it gives the measure of correctly detected anomalous data samples. Precision is simply the measure of how many anomalous samples the model succeeded in predicting as anomalous. Precision must be close to 100% if FP needs to be kept minimum. In multiclass classification, macro averaged precision score across all classes is considered as the comparison metric.

4. *Recall:* This metric gives the proportion of cases in which the model correctly predicts a value as class x over the number of class x cases.

$$Recall = \frac{TP}{TP + FN}$$

   This metric is also a good one as it gives the measure of detected anomalous data samples over the total anomalous samples. In simple words, recall represents how many anomalous samples the model missed to predict as anomalous. This means recall must be close to 100% if FN needs to be kept minimum. The macro average value of recall across all classes is considered in the multiclass classification case.

5. *F1-score:* This metric provides a single score that represents both precision and recall. This metric is good for cases where class distribution is imbalanced in the dataset. It is calculated as the harmonic mean of these values as shown below:

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall}$$

   Macro averaged F1-scores across all classes is considered as the metric for the multiclass classification problem.

Chapter 3

Methodology

In this chapter, we discuss the methodology used to design the proposed solution, i.e., a security framework for ICSs. Section 3.1 describes the research process used in the design of the solution whereas Section 3.2 explains the design methodology.

3.1 Research Process

Figure 3.1 depicts the high-level overview of the different steps used in the research process. Each of these steps is further divided into sub-steps to simplify the implementation phase of the thesis. All these steps contribute towards designing the proposed digital twin-based security framework for ICSs that integrates an ML-based IDS.

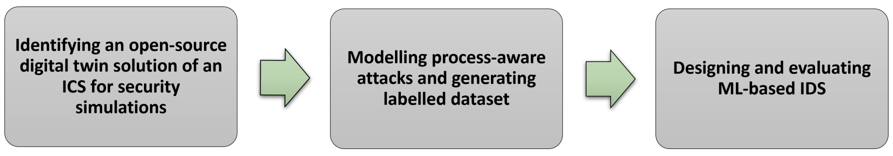

Identifying an open-source digital twin solution of an ICS for security simulations
Modelling process-aware attacks and generating labelled dataset
Designing and evaluating ML-based IDS

Figure 3.1: Research process

The first step as shown in Figure 3.1 is about identifying an open-source digital twin solution of an ICS from the reviewed solutions as part of the literature study. As the initial sub-step, we perform a detailed comparison of the identified knowledge-driven digital twin security solutions of ICSs. This comparison extends beyond the scope of the literature study and involves hands-on evaluation of these solutions in terms of their implementation aspects and suitability to perform cybersecurity analysis in ICSs. Such an evaluation is necessary to identify a solution that bridges the gap between the various digital twin concepts presented in research papers and their practical implementation aspects. Based on this comparison, the next sub-step is to choose a digital twin solution to perform cybersecurity analysis. The criteria used to make this choice are the following: open-source availability of the solution, the capability of the solution to work as a standalone solution (meaning there is no requirement to run the real physical system in parallel with the digital twin), demonstration of a real ICS use case, and the possibility to model and execute different types of attacks in the digital twin. The decision to use an existing open-source digital twin is for the sake of simplicity in reusing this solution for the thesis. Since this thesis focuses on the security applications of the digital twin, synchronization aspects with the physical twin to enable data integration into the digital twin are not evaluated.

As the second step in the research process, different realistic attack scenarios are modelled and executed in the chosen digital twin solution and a labelled dataset is generated for all the scenarios. As the first sub-step, process-aware attack scenarios are modelled as insider threats in a manner to affect the process measurements, by causing minimum abnormalities in network traffic. The main aim of modelling such attacks is not to be detected easily by network traffic monitoring alone. As the next sub-step, a labelled dataset is generated which contains process measurements collected during normal operation and attack duration.

The final step in the research process is to design and evaluate an ML-based IDS. As the first sub-step, different supervised ML algorithms are trained and evaluated on the generated dataset. Based on the results of evaluation metrics used, a stacked ensemble classifier model that combines three ML algorithms is used to implement the ML-based IDS. The second sub-step is to train this stacked ensemble model offline with the generated dataset. Finally, the incoming samples from the digital twin during the live operation of the system are classified by the IDS based on the trained model.

3.2 Design Methodology

In this section, we discuss the methodology used in this thesis to design the proposed solution. Furthermore, we also discuss the methodology to evaluate the proposed solution. The design of the proposed solution is achieved in several steps, as shown in Figure 3.1. The first step of the research process uses a qualitative method with a deductive approach to choose an implementation suitable to perform cybersecurity analysis among the different digital twin-based security solutions available for ICSs. In this work, we consider a digital twin solution as suitable to perform cybersecurity analysis when different types of attacks can be modelled and executed inside the digital twin. The remaining steps use quantitative methods to model different attack scenarios, generate a dataset consisting of process measurements, and detect intrusions using ML-based IDS. The following subsections provide a detailed description of the methodology used in different steps of the research process.

3.2.1 Identifying an Open-source Digital Twin Solution

The first sub-step in this step involves the ***evaluation of digital twin solutions*** mentioned in 2.5.1 in terms of the feasibility to reuse these solutions to perform cybersecurity analysis. However, not all of these solutions demonstrate real-world ICS use cases. Listed below is the summary of the evaluated solutions.

- Solution 1 ([12], [31]) introduces an open-source framework to generate a virtual environment automatically from the specification of the physical system using Mininet-WiFi [44]. This specification needs to be provided as an input to the framework in AutomationML[1] data exchange format. This framework supports two operation modes, namely simulation, and replication. In simulation mode, digital twin can work as a standalone simulation without the requirement of the coexistence of a physical twin. In replication mode, passive state replication is used to synchronize the physical environment and the virtual environment. Furthermore, the intrusion detection use case presented in this solution is a rule-based one.

- Solution 2 ([2]) focuses on the synchronization aspect of the physical environment and virtual environment using active state replication. This solution requires a physical twin to be run in parallel with the digital twin. The physical system used is a simple system consisting of a single PLC and a server, and hence does not represent a full-fledged real-world ICS use case. The authors state scalability issues as one of the limitations of this approach when implemented for large ICSs. This solution is not available as open-source and the security applications of the solution are left for future work.

[1] `https://www.automationml.org/`

- Solution 3 ([3]) demonstrates the use case of integration of the digital twin of an ICS with a security operations center. The ICS use case demonstrated is of an industrial filling plant. The digital twin is implemented as a standalone simulation using Mininet-based MiniCPS; meaning the physical processes are implemented as simple Python simulations. The solution offers a microservices-based architecture in which each component of the system is realized using Docker containers. Furthermore, all the components are implemented using open-source tools and the whole solution is available as open-source. However, MiniCPS-based implementations can be used to model only those CPSs which use Ethernet as the physical layer for communication (only Modbus and ENIP).

- Solution 4 ([32]) suggests the capability of open-source tools to achieve digital twin capabilities like real-time IoT data acquisition, virtualization, data analytics, and visualization. There is no specific industrial use case or physical system for which digital twin is implemented; instead, a publicly available industrial dataset collected from 100 different IoT machines in the year 2015 is used to implement 100 digital twins for evaluation purposes. Simulation of real-time scenario is achieved by a cronjob that acts as a scheduler to collect data of all these machines from the dataset every minute. However, performance testing is not performed on a real industrial use case and the proposed solution is not available as open-source.

- Solution 5 ([35], [9]) demonstrates the inclusion of intrusion detection in digital twin. The physical system used in this work is a MATLAB simulation of the ball and beam process, and there is no sufficient information provided on the implementation of the digital twin. The industrial use case demonstrated in this work is not a real-world ICS use case. IDS solution presented uses Kalman filter to estimate output signals and compares it with output signals from digital twin for attack detection; and support vector machine ML algorithm for identifying the type of detected attacks. This solution requires the coexistence of the physical twin with the digital twin and synchronization between both.

Furthermore, Table 3.1 provides a quick summary of the above-mentioned digital twin-based security solutions of ICSs.

Table 3.1: Summary of digital twin-based security solutions available for ICSs

No.	Solution	Summary	Open-source?	Standalone Solution?	Real-world ICS use case?
1	[12], [31]	Generates digital twin automatically from system specification using Mininet-WiFi.	Yes	Yes	Yes
2	[2]	Digital twins run as Virtual Machine (VM)s on isolated environments.	No	No	No
3	[3]	Digital twin simulation using Mininet and MiniCPS; physical process as simple simulation in Python.	Yes	Yes	Yes
4	[32]	Uses a benchmark industrial dataset as a cronjob for evaluation purposes.	No	Yes	No
5	[35], [9]	Digital twin implementation using system identification algorithm; MATLAB simulation of ball and beam process is deployed as the physical system.	No	No	No

As the next sub-step, we ***select a digital twin solution from the evaluated solutions***. The criteria used for this selection are already discussed in Section 3.1. Among the listed solutions in Table 3.1, only solutions 1 and 3 demonstrate real-world industrial use cases. Solution 1 demonstrates the use case of a conveyor belt in a candy factory whereas solution 3 demonstrates the use case of an industrial filling plant. Furthermore, both of these solutions are available as open-source and can work as standalone simulations. Although the idea of the digital twin is complete only with the coexistence of the physical twin, attaining real-time synchronization between both is quite complex and not in the scope of this thesis.

We carry out a hands-on feasibility study of the implementation of solutions 1 and 3 and try to bring up the non-working modules in both solutions after discussions with corresponding authors. Solution 1 poses serious limitations in terms of stability and functionality, as the open-source code is not being actively developed any longer. Attempts have been made to bring up the framework, but failures associated with instantiating the PLC node could not be resolved. This issue is raised as a bug on the author's GitHub page[2]. Solution 3 implementation is also tried out in parallel, and this solution works correctly with minor fixes. Moreover, the framework provided as part of solution 3 is implemented using Docker containers which offers the flexibility of adding IDS as a Docker container to the existing framework. Hence, solution 3 is selected as the digital twin solution for this thesis. More details on this solution are provided in Section 3.2.2.

3.2.2 Modelling Process-aware Attacks and Generating Labelled Dataset

The next step is to evaluate the digital twin solution in terms of its ability to address cybersecurity issues. This step involves modelling realistic industrial control process-aware attacks, executing these attacks in the digital twin, and generating labelled dataset for both normal operation as well as attack duration.

In this thesis, we prefer to generate the dataset by executing different process-aware attack scenarios over synthetic dataset generation. This is to utilize the capability of the digital twin in providing accurate process measurements which are the same as that of the measurements collected from the real physical system. Such an approach also eliminates the additional overheads associated with collecting measurements from the physical system. Moreover, synthetic dataset generation for different attacks may fail to consider

[2]`https://github.com/sbaresearch/cps-twinning/issues/4`

the correlation between different process measurements in a given sample and can lead to generation of incorrect dataset. Hence, it is crucial to generate the dataset by executing attacks in the digital twin. This approach also proves beneficial in the future to integrate traffic analysis-based IDS into the proposed framework.

We make an assumption here that the digital twin standalone simulation can closely reflect its physical system in near real-time and hence can reflect the consequences of attacks happening in the physical system. Therefore, it is decided to place the attacker inside the network of the digital twin to illustrate security use cases. However, in real scenarios, the digital twin is run in isolated and secure environments and is not open to unauthorized access.

To model process-aware attacks, it is important to understand the digital twin framework and the industrial use case run in this framework. A brief introduction of the framework and the network topology of the digital twin from [3] is provided below. For more details on the framework, the readers are encouraged to read [3].

Figure 3.2 depicts the network topology of the digital twin simulation realized using MiniCPS which is built on top of Mininet. MiniCPS provides a framework for emulating industrial networks and simulating industrial control devices and industrial network communication. The emulation of the ethernet-based network of the industrial filling plant using MiniCPS along with the simulations of ICS components such as PLCs and HMI is used to replicate the ICS in near real-time. Moreover, this solution also considers the interaction between components (PLCs, sensors, and actuators) in the physical layer (using ENIP industrial protocol) on top of the network communications. In ICS, the physical layer interactions between components are considered as attack targets besides the common network-based attacks. Hence, this solution proves useful in modelling and executing process-aware attacks targeting the physical layer interactions between the components of the system.

As shown in Figure 3.2, the network consists of three PLCs, one HMI, and an attacker node connected to a switch. This simulation uses ENIP as the industrial network communication protocol. The industrial use case simulated here is of an industrial filling plant, as shown on the right side of the figure. The industrial filling plant consists of a tank containing some liquid, a bottle, and a pipe through which liquid flows from the tank to the bottle. The flow of liquid through the pipe is controlled by an actuator which is a motor valve. Sensors 1, 2, and 3 read the liquid level in the tank, flow level in the pipe, and liquid level in the bottle respectively. PLC1 monitors and controls sensor 1 and the actuator; PLC2 and PLC3 are responsible for sensor 2 and sensor3, respectively. PLC1 performs the control operation of the actuator based on all three sensor values. Sensor measurements are stored as ENIP tags in corresponding PLCs, and transmitted to PLC1 periodically at every control cycle of PLC1, which is set as 0.5 seconds.

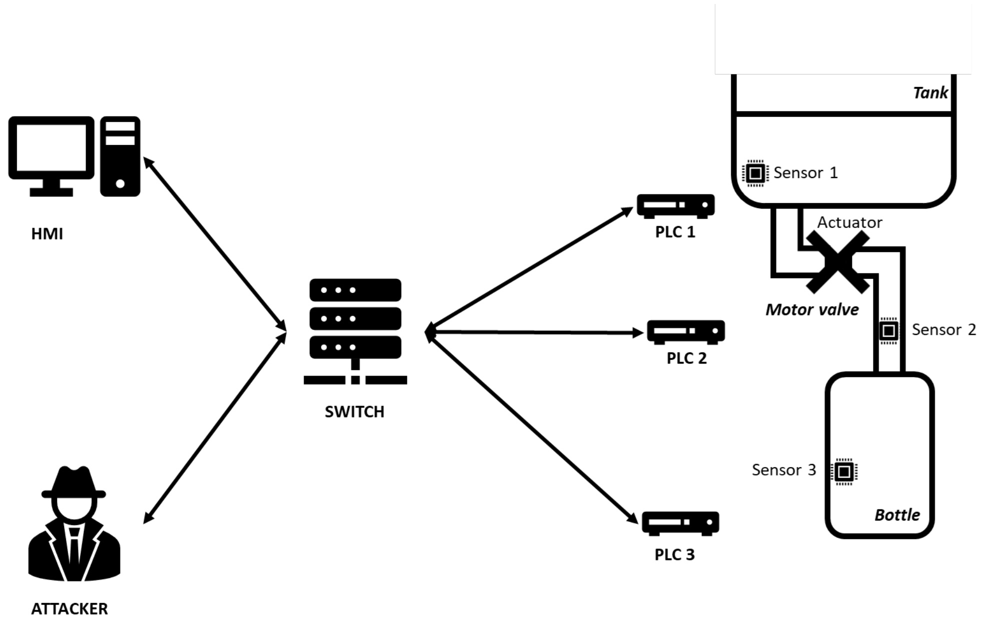


Figure 3.2: Industrial filling plant use case & digital twin network topology

The framework proposed in [3] is implemented as a microservice architecture using Docker compose, where each module runs as a separate Docker container. Different modules in this framework are: (i) *digital twin* implemented using Mininet-based MiniCPS, (ii) *filebeat* module to gather and ship log data from digital twin, (iii) *logstash* module to normalize the log data from filebeat, (iv) *Dsiem* correlation engine for rule-based incident detection from log data, (v) *elasticsearch* for storing data and executing queries, and (vi) *kibana* as the visualization tool. This framework is available at the GitHub repository of the authors of [3][3].

In this thesis, we propose an extension of the above framework as shown in Figure 3.3. Each box represents a Docker container. All grey-colored boxes are retained from the original implementation; the proposed extension is a new Docker container (yellow box) that runs an ML-based IDS. The newly added *IDS* module receives the labelled dataset generated inside the digital

[3]https://github.com/FrauThes/DigitalTwin-SIEM-integration

twin module via filebeat. This dataset is used to train the supervised ML algorithm running in the IDS module, and the trained model is further used to detect intrusions in near real-time.

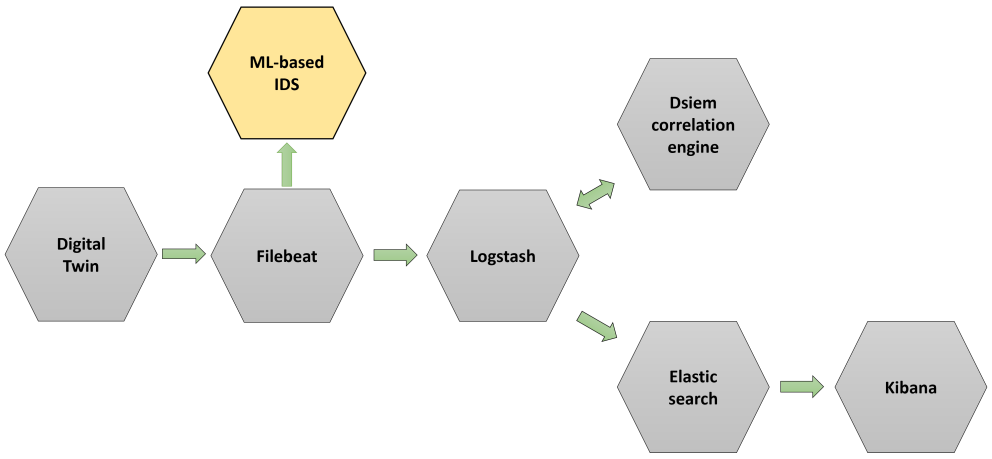


Figure 3.3: Proposed extension of framework from [3]

To model different process-aware attack scenarios inside the digital twin Docker container, it is important to familiarise with the process control variables of the system that are of interest from a security perspective. Table 3.2 describes different process control parameters of the system.

Given the details of the digital twin framework, the initial focus in this step is to ***model process-aware attacks*** aimed at disrupting the industrial control process in operation. These attacks are carefully modelled to cause minimum abnormalities in network traffic. All the attacks are executed as insider attacks; meaning attacker who is already present inside the ICS network is executing the attacks. This is based on the assumption that the attacker is successful in bypassing IT security measures and has access to the ICS.

We model attack scenarios belonging to four different attack types, naming Command Injection, Network DoS, Calculated Measurement Injection, and Naive Measurement Injection. We explain below all the four attack types and the impact of these attacks on the system. Furthermore, the implementation details of the modelled attack scenarios are provided in Chapter 4.

1. ***Command Injection Attack***: This type of attack aims at exploiting an interface to remotely inject malicious commands. In our case, PLC1 that is responsible for the control operation of the motor valve in the system is the node under attack. PLC1 does this control operation by sending

Table 3.2: Process control parameters and their description

No.	Parameters	Description
1	**motor_status**	integer value representing the motor status; 0 for OFF and 1 for ON
2	**bottle_liquidlevel**	float value from sensor 3 indicating the liquid level in the bottle in m^3
3	bottle_lowerbound	lower bound for the liquid level in the bottle; constant value set to 0.0
4	bottle_upperbound	upper bound for the liquid level in the bottle; constant value set to 0.9
5	**flowlevel**	float value from sensor 2 indicating the flow level of the pipe
6	sensor2_thresh	threshold value of the flow level of the pipe; constant value set to 3.0
7	**tank_liquidlevel**	float value from sensor 1 indicating the liquid level in the tank in m^3
8	tank_lowerbound	lower bound for the liquid level in the tank; constant value set to 0.3
9	tank_upperbound	upper bound for the liquid level in the tank; constant value set to 5.81

control commands to switch ON/OFF the motor valve. To tamper with this control logic, the attacker disguised as HMI reads the actuator value from PLC1 and sends a command to PLC1 to set a toggled value before the actual control operation is performed. This attack is performed in such a way to inject the toggled value before the actual value is set, and is not a MitM attack. This attack exploits the vulnerability of no authentication control present in the ENIP protocol. Here, the attacker intends to disrupt the outflow of the tank which in turn disrupts the rate of filling of the bottle.

2. ***Network DoS attack***: We model network DoS attacks in our work using two approaches. In the first approach, the attacker places himself in between PLC1 and PLC2 (or PLC3) and then uses Address Resolution Protocol (ARP) poisoning to sniff remote connections coming towards PLC1. In addition to sniffing, the attack scenario is designed in such a way that the attacker node does not forward the intercepted communication to the target node. Such an approach enables the attacker to perform selective erasure of messages. For example, the attacker can selectively choose to deny PLC1 of the measurements

coming from PLC2, while allowing measurements from PLC3 to reach PLC1. This approach is called MitM/DoS in the literature [40].

The second approach uses Transmission Control Protocol (TCP)/SYN flag (SYN) flooding to launch DoS attacks. The attacker disguises his Internet Protocol (IP) address as that of a valid IP address and floods the network towards PLC1 with TCP SYN packets. This attack targets the specific TCP port used by ENIP protocol, i.e., 44818. This flooding attack congests the network, and PLC1 does not receive any measurements from PLC2 and PLC3.

For the normal operation of the system, PLC1 must receive sensor measurements from other PLCs at regular intervals. But network DoS attacks make it impossible for PLC1 to get these values. Here, the attacker intends to disrupt the rate of flow of liquid from the tank to the bottle. The operation continues with previously-stored process control values causing undesirable impacts.

3. ***Calculated Measurement Injection Attack***: This attack type is modelled as MitM attack to alter the sensor measurements sent to PLC1 from other nodes on the fly. This attack type exploits the vulnerability of the lack of encryption of the ENIP protocol used in this simulation. Here, the attacker places himself between PLC1 and PLC2 (or PLC3) and alters the sensor measurements sent towards PLC1 by a calculated value (using a positive or negative scaling factor). This scaling is performed gradually and carefully to disrupt the operation of the system without the risk of being detected sooner. This type of attack disrupts the normal operation of the system at a slow pace.

4. ***Naive Measurement Injection Attack***: This attack type is also modelled as MitM attack and exploits the same vulnerability as that of the previous type. Here, the attacker places himself in between PLC1 and PLC2 (or PLC3) and uses a naive approach to alter the sensor measurement sent towards PLC1 to a constant/random value without taking into account the actual measurement at that time. This can cause abrupt impacts on the system disrupting the normal operation of the system and hence may be detected sooner compared to the previous attack type.

The next sub-step is to ***generate labelled datasets***. All the process control parameters which are highlighted in Table 3.2 along with the timestamp data are stored in Comma Separated Values (CSV) format. The non-highlighted measurements are the threshold values that do not vary over time and are therefore neglected. New rows are added to this CSV file at every control cycle of PLC1. The time for which data needs to be collected is configured via HMI.

Once this dataset is generated, the next step is to label the dataset which can be further used to train the ML-based IDS in the framework. These labels are added as a new column named *class* in the dataset. We use a comprehensive labelling approach to label the process measurements collected during normal operation as 'Normal' and during attack duration with the corresponding attack type. Such an approach to include different labels can help to easily isolate and resolve the detected attacks in the future. Labelling of data samples collected during attack duration is done either based on threshold condition checks (for command injection attack) or according to the time duration in which attacks are performed. An excerpt of the labelled dataset collected during command injection attack is provided in Table 5.6.

3.2.3 Designing and Evaluating ML-based IDS

This step involves choosing an ML algorithm that works best on the generated dataset to implement the IDS. Due to time constraints, only supervised ML algorithms are considered here. The choice of ML algorithm is based on the evaluation of different supervised algorithms used in the state-of-the-art research works done in this field.

Evaluation of ML Algorithms

Based on the literature study on ML-based intrusion detection in ICSs, identified supervised algorithms for evaluating the labelled dataset are support vector machine (SVM), k nearest neighbor (KNN), naive bayes (NB), random forest (RF), logistic regression (LR), artificial neural network (ANN), gradient boosting (GB), and decision tree classifier (DTC).

The metrics used to evaluate these algorithms are already explained in Section 2.6. Accuracy is a good metric when target classes in data are balanced. But for those cases where target classes are not balanced, the model may not be good in accurately predicting the minority class even when the overall accuracy is high. In our case, the distribution of target classes is not balanced since attacks are executed usually for short duration. Therefore, it is not advised to use accuracy alone as the metric to choose the final classifier model.

Furthermore, it is necessary to correctly classify anomalous samples in the case of intrusion detection use case. This means all the diagonal elements of the confusion matrix (positive classes) for a classifier model are at the highest and FNs for the anomaly classes need to be kept at a minimum. As discussed in Section 2.6, high recall value is considered good in our evaluation to keep FN values to minimum. In case of intrusion detection, we can only tolerate very few FPs. Otherwise, this can result in high number of false alarms. To take this factor into account, we also consider F1-score as another metric as this represents both precision and recall values.

In addition to the evaluation of these individual algorithms, an ensemble approach called stacking is also evaluated. This approach makes it possible to combine the predictions of individual classifiers to make the final predictions. The reason to use a stacking classifier in our case is to check if the classification results improve upon using this approach. Figure 3.4 shows the concept of stacking classifier with two levels of classification: Level 0 and Level 1. Level 0 has three individual classifiers and Level 1 is the final classifier. The choice of Level 0 classifiers is based on the evaluation results of the eight individual classifiers on distinct class labels. This is because the classifier which gives the best overall scores across all class labels may not be the best one for each of the labels. Predictions from each of the classifiers in Level 0 are represented by P1, P2, and P3. Level 1 classifier is the final classifier that combines the predictions of Level 0 classifiers and makes the final predictions. Here, Level 1 classifier is trained using the the cross-validated predictions from Level 0 classifiers [45].

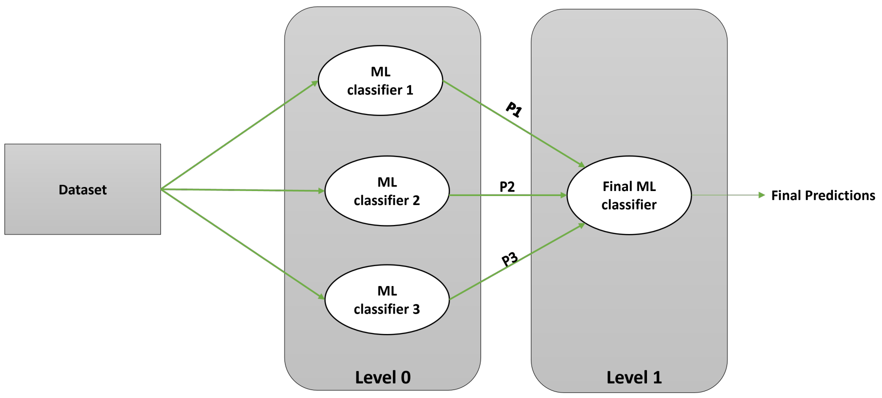


Figure 3.4: Stacked ensemble classifier model concept

IDS Design and Implementation

Based on the comparison of the performance of the individual ML classifier algorithms using the above-listed metrics, the final classifier used in the implementation of IDS is a stacked ensemble classifier model that combines the predictions from three individual classifiers. More details on the dataset as well as comparison results of different individual algorithms and the stacked ensemble model are provided in Chapter 5.

Figure 3.5 depicts the security framework consisting of an ML-based IDS implemented as part of the thesis. During the offline training phase, the ML model with a stacked ensemble classifier model is trained in the Docker container running IDS using the labelled dataset collected from the digital twin. During the operation of ICS, this model classifies the incoming data samples (unlabelled test dataset) from the digital twin Docker container based on the trained model. Results from IDS are shown as visualizations using a dashboard in the kibana container.

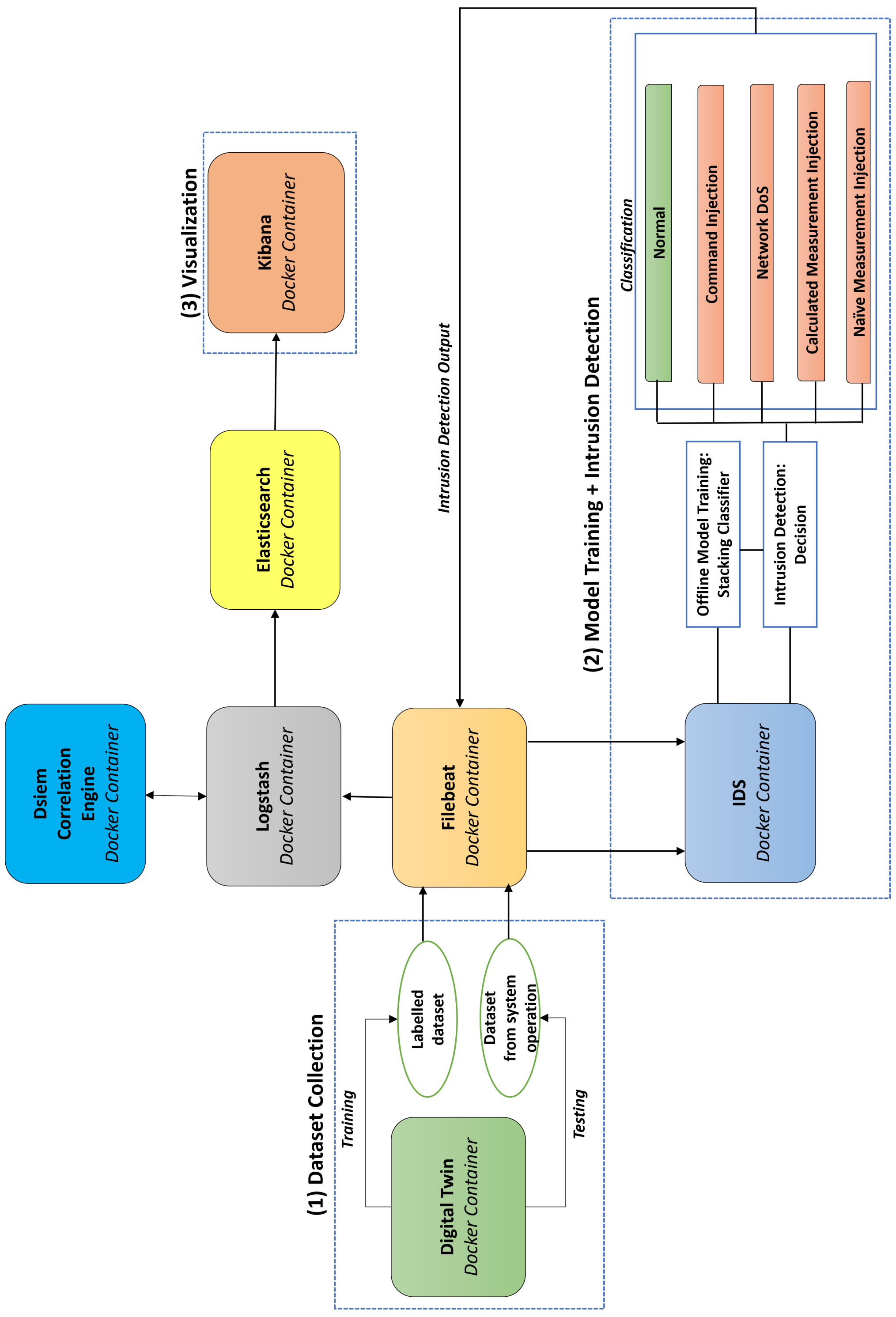

(1) Dataset Collection
Digital Twin Docker Container
Training
Testing
Labelled dataset
Dataset from system operation
Filebeat Docker Container
Logstash Docker Container
Dsiem Correlation Engine Docker Container
Elasticsearch Docker Container
(3) Visualization
Kibana Docker Container
Intrusion Detection Output
(2) Model Training + Intrusion Detection
IDS Docker Container
Offline Model Training: Stacking Classifier
Intrusion Detection: Decision
Classification
Normal
Command Injection
Network DoS
Calculated Measurement Injection
Naïve Measurement Injection

Figure 3.5: Security framework including ML-based IDS

Chapter 4

Implementation

This chapter discusses the implementation details that are relevant to carry out this thesis. The tools and frameworks used in this thesis are also explained in this chapter.

4.1 Digital Twin Framework

As discussed in Section 3.2.1, the selected digital twin framework is from [3]. This framework is available as an open-source implementation as a GitHub repository[1]. This repository also provides a README file that lists all the steps to be followed to make this framework up and running. Additionally, the below-listed prerequisites need to be followed to use this framework.

1. Use Ubuntu version 18.04 to run the framework on a VM.

2. ENIP tags used in the digital twin simulation are supported on older versions of cpppo package (4.1.x to 4.3.x). The default cpppo-package version installed along with the Python package ’minicps’ is the latest version 4.4.2 and does not support ENIP tags. Therefore, this version needs to be downgraded manually. Listing 4.1 provides the command to downgrade the cpppo version to 4.1.0.

```
pip install cpppo==4.1.0
```

Listing 4.1: Command to downgrade cpppo version to 4.1.0

[1] https://github.com/FrauThes/DigitalTwin-SIEM-integration

4.2 Modelling Process-aware Attacks

This section explains the implementation details regarding the modelling and execution of different process-aware attacks inside the digital twin. The tools and libraries used for this implementation are also explained here.

4.2.1 Command Injection Attack

This attack is implemented using a simple Python script that is run on the attacker node for a certain time duration. In this attack, a naive approach is used to inject commands to PLC1 to set wrong actuator values causing wrong control operation of the motor valve. The corresponding process measurement reflecting this actuator value is the *motor_status*.

The script reads the actuator values which are stored as ENIP tags in PLC1, toggles this value, and sends this toggled value to PLC1. This action is performed every x seconds, where x is a value that is less than the interval in which PLC1 performs the control operation of the motor valve. If PLC1 receives the command to set the toggled value before it performs the control operation, then the actuator value is set to this toggled value, and the attack execution is considered successful. This attack is executed carefully to avoid causing flooding in the network, thus making it difficult to detect this type of attack by network traffic monitoring.

4.2.2 Network DoS Attack

We explain the implementation of two approaches used to model and execute network DoS attacks in this work.

1. *MitM/DoS*: This attack uses the Ettercap tool to simulate a MitM attack. Using this tool, the attacker node is placed between PLC1 and other PLCs. An ARP poisoning MitM attack is performed to sniff remote connections towards PLC1. In this attack, the intercepted packets reaching the attacker node are not forwarded to PLC1 resulting in a DoS attack. A total of three attack scenarios are performed in this category in which the attacker is placed between (i) PLC1 and PLC2, (ii) PLC1 and PLC3, and (iii) PLC1 and any other hosts.

```
ettercap -T -i attacker-eth0 -M ARP
    /10.0.0.1// /10.0.0.2//
```

Listing 4.2: Ettercap command to launch MitM/ARP poisoning

Listing 4.2 shows the command to launch MitM/DoS between PLC1 and PLC2 using Ettercap. Here, attacker-eth0 is the interface on the attacker node that launches the attack, 10.0.0.1 is the IP address of PLC1, and 10.0.0.2 is the IP address of PLC2. On execution of this command, the attacker node is placed between PLC1 and PLC2 and can sniff all connections coming towards PLC1 from PLC2.

2. *TCP SYN flooding*: hping3[2] network tool is used to execute this attack. This tool allows sending custom TCP/IP packets to hosts. In this attack, the attacker node disguises its source IP address as a different but valid address and floods the network with TCP SYN packets targeting TCP port 44818 on PLC1. PLC1 sends back SYN/ACK packets to the source known to it; however, these packets go unacknowledged exhausting the resources reserved on PLC1 for handling this communication. Thus, PLC1 cannot receive new packets from PLC2 and PLC3 resulting in a DoS attack.

```
hping3 -S -a 10.0.0.4 --flood -V -p 44818
    10.0.0.1
```

Listing 4.3: hping3 command for TCP/SYN flooding DoS attack

Listing 4.3 shows the hping3 command run on the attacker node to perform this attack. Here, 10.0.0.1 is the IP address of the target which is PLC1 whereas 10.0.0.4 is the IP address of HMI disguised by the attacker.

4.2.3 Calculated Measurement Injection Attack

These types of attacks are implemented as custom Python scripts that use a third-party library called *scapy*[3] to decode and alter packets sent on the network. Here, the attacker node is placed between PLC1 and other PLCs, and the payload of ENIP packets sent towards PLC1 is modified on the fly. These payload values contain the process measurements (*flowlevel* and

[2] https://linux.die.net/man/8/hping3
[3] https://scapy.net/

bottom_liquidlevel) and this attack alters these measurements to a scaled value. This scaling is done by a small factor to make this attack stealthy. Scaling of process measurements is done as follows:

$$modified\_value = (1 \pm scaling\_factor) * decoded\_payload\_value$$

The scaling factor used can be either a constant value or a random value from a uniform distribution. A total of twelve attack scenarios are performed under this attack type. Table 5.5 provides a summary of these attack scenarios.

To launch this attack, a MitM ARP poisoning is executed at first using the command provided in Listing 4.2. IP forwarding is enabled using the command provided in Listing 4.4. This ensures that the manipulated packet using the custom Python script reaches the destination node. Further, custom scripts are run to manipulate packets on the fly.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Listing 4.4: Enable IP forwarding

4.2.4 Naive Measurement Injection Attack

These types of attacks also use the scapy library to modify process measurements. Here, the packet payload is modified to either a fixed constant value or a random value within the predefined limits of process measurements. There are six attack scenarios performed under this attack type and details of these attack scenarios are provided in Table 5.4. This attack also uses the commands provided in Listings 4.2 and 4.4.

4.3 Generating Labelled Dataset

In this step, process measurements are collected from the PLC1 node. This is because PLC1 is responsible for the control operation of the whole system running as the digital twin, and the modelled process-aware attack scenarios target the measurements reaching PLC1 to disturb the normal control operation of the system.

To train the ML model, data is collected for both normal operation and attack duration. In real operation scenarios, the majority of the data samples belong to the normal operation since attacks are performed only for short durations. To retain this realistic scenario, each of the modelled attack scenarios is executed for short durations (3 to 5 minutes). Balancing of classes in the generated dataset is intentionally not considered to retain the distribution of samples across different classes as that of a real scenario. This approach to not rebalance the dataset is used by the authors of [42] and [17]. Instead of using data sampling methods to circumvent the imbalance in classes, these works emphasize considering the choice of proper evaluation metrics for the classification of imbalanced datasets.

Once the data samples with process measurements are collected, this dataset is labelled with a class label. Data samples collected during normal operation are labelled as 'Normal'. Labels for different attack scenarios represent the attack types to which these belong, and these are 'Command Injection', 'Network DoS', 'Calculated Measurement Injection', and 'Naive Measurement Injection'. Labelling of the dataset is automated based on threshold condition checks and the duration of attacks.

4.4 ML-based IDS

To evaluate ML algorithms, we split the labelled dataset into two sets: 70% as training set and 30% as testing set. This split is done in such a way that both the training set and test set have a similar distribution of target class labels. All the algorithms used for this evaluation are implemented using the *scikit-learn*[4] library for Python. Anaconda Jupyter notebook (Python 3)[5] is used as the programming environment to implement these algorithms. Table 4.1 lists the technical specifications of the system setup used to run these algorithms. Table 4.2 lists the scikit-learn APIs along with the parameters that are used for this evaluation. Hyperparameter tuning for different models is not taken into account here.

Table 4.1: Specifications of system setup used to run ML algorithms

System Type	x64-based PC
OS Name	Microsoft Windows 10 Home
OS Version	10.0.19043 Build 19043
Processor	Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz, 4 Core(s) , 4 Logical Processor(s)
RAM	16GB

[4]https://scikit-learn.org/stable/
[5]https://anaconda.org/anaconda/jupyter

Table 4.2: Scikit-learn APIs used

Scikit-learn API	**Description**	**Parameters**
sklearn.preprocessing.StandardScaler	Data preprocessing	
sklearn.model_selection.train_test_split	To split labelled data into training and test set	train_size:0.7 random_state:1 stratify:<y_train>
sklearn.svm.SVC	To implement support vector machine classifier	kernel:linear
sklearn.ensemble.RandomForestClassifier	To implement random forest classifier	random_state:1 n_estimators:100 max_depth:None
sklearn.neighbors.KNeighborsClassifier	To implement KNN classifier	n_jobs:-1 n_neighbors:5
sklearn.linear_model.LogisticRegression	To implement logistic regression classifier	n_jobs:-1 random_state:0
sklearn.tree.DecisionTreeClassifier	To implement decision tree classifier	criterion:'gini' max_depth:None
sklearn.naive_bayes.GaussianNB	To implement naive bayes algorithm	var_smoothing:1e-09
sklearn.neural_network.MLPClassifier	To implement ANN using multi-layer perceptron classifier	solver:'lbfgs' hidden_layer_sizes:100 activation:relu
sklearn.ensemble.GradientBoostingClassifier	To implement gradient boosting classifier	learning_rate:0.1 n_estimators:100 max_depth:3
sklearn.multiclass.OneVsRestClassifier	To fit one classifier per class in multiclass classification; for all classifiers except SVM	
sklearn.multiclass.OneVsOneClassifier	To fit one classifier per class pair in multiclass classification; used for SVM	
sklearn.ensemble.StackingClassifier	To implement ensemble stacking algorithm	cv:10
sklearn.metrics	Classification metrics to evaluate models	

Evaluation of different algorithms is done on the labelled dataset. Based on the evaluation results, the stacking classifier is the one chosen to implement IDS. IDS implementation is also done in Python using Anaconda Jupyter notebook. The stacking classifier which is already trained using the labelled dataset is further used to classify the incoming data samples collected during the live operation of the system. Results of this detection are written in real-time into a CSV file which is shipped to logstash via filebeat. Logstash processes this file and stores the data in elasticsearch. This elasticsearch data that is updated in real-time is fed as an ingest pipeline to kibana for visualization purposes.

Chapter 5

Results and Analysis

In this chapter, we present the results and provide a detailed analysis of these results. Section 5.1 discusses 23 process-aware attack scenarios modelled as part of this work. An excerpt of the generated labelled dataset along with the distribution of classes in this dataset is provided in Section 5.2. Section 5.3 provides the evaluation results of different supervised ML algorithms in classifying data samples of the generated dataset. Finally, the results of IDS are presented in Section 5.4.

5.1 Modelled Process-aware Attack Scenarios

In our work, 23 attack scenarios belonging to four different attack types are modelled and executed in the digital twin simulation. Table 5.1 provides the mapping of these attack scenarios (summarized in tables 5.2 to 5.5) to attack types.

Table 5.1: Mapping of modelled attack scenarios to different attack types

Attack Scenario No	**Attack Type**
1	Command Injection
2-5	Network DoS
6-11	Naive Measurement Injection
12-23	Calculated Measurement Injection

Tables 5.2 to 5.5 provide a summary of 23 process-aware attack scenarios belonging to four different attack types which are modelled and executed in the digital twin simulation. In each table, *Attack Point* refers to the targeted

node(s), and *Affected Process Measurement(s)* lists the process measurements affected by the attack. The *Attack Description* column provides the description of attack scenarios. Finally, the intent of the attacker and the impact of attacks on the system is explained in the last column, namely *Attack Impact*.

Table 5.2: Modelled attack scenario under command injection attack type

Attack Scenario No.	**Attack Point**	**Affected Process Measurement(s)**	**Attack Description**	**Attack Impact**
1	Actuator value	motor_status	Toggle actuator value every 0.5 seconds	Closes motor valve when it must be open and vice-versa; leads to tank overflow or bottle overflow

Table 5.3: Modelled attack scenarios under network DoS attack type

Attack Scenario No	**Attack Point**	**Affected Process Measurement(s)**	**Attack Description**	**Attack Impact**
2	PLC2	flowlevel	MitM attack to intercept and drop packets sent towards PLC1 from PLC2	Prevents sensor2 measurements from reaching PLC1; disturbs the control operation of the motor valve as PLC1 takes control decision based on the last received sensor2 value
3	PLC3	bottle_liquidlevel	MitM attack to intercept and drop packets sent towards PLC1 from PLC3	Prevents sensor3 measurements from reaching PLC1; disturbs the control operation of the motor valve as PLC1 takes control decision based on the last received sensor3 value
4	PLC2, PLC3	flowlevel, bottle_liquidlevel	Attack scenario 2 + Attack scenario 3	Causes the combined impact of attack scenarios 2 & 3
5	PLC2, PLC3	flowlevel, bottle_liquidlevel	TCP SYN flood attack targeting ENIP port 44818 of PLC1	Causes the combined impact of attack scenarios 2 & 3

Table 5.4: Modelled attack scenarios under naive measurement injection attack type

Attack Scenario No	**Attack Point**	**Affected Process Measurement(s)**	**Attack Description**	**Attack Impact**
6	PLC2	flowlevel	Modify measurements sent by PLC2 towards PLC1 to a constant value	PLC1 upon receiving false flowlevel measurements closes the valve sooner or later and can result in incomplete filling of the bottle or overflow of the bottle
7	PLC3	bottle_liquidlevel	Modify measurements sent by PLC3 towards PLC1 to a constant value	PLC1 upon receiving false bottle_liquidlevel measurements fails to open or close the valve at the right time when the bottle is empty or full
8	PLC2, PLC3	flowlevel, bottle_liquidlevel	Attack scenario 6 + Attack scenario 7	Causes the combined impact of attack scenarios 6 & 7
9	PLC2	flowlevel	Modify measurements sent by PLC2 towards PLC1 to a random value within the allowed limits for flowlevel	Similar impact as that of attack scenario 6
10	PLC3	bottle_liquidlevel	Modify measurements sent by PLC3 towards PLC1 to a random value within the allowed limits for bottle capacity	Similar impact as that of attack scenario 7
11	PLC2, PLC3	flowlevel, bottle_liquidlevel	Attack scenario 9 + Attack scenario 10	Similar impact as that of attack scenario 8

Table 5.5: Modelled attack scenarios under calculated measurement injection attack type

Attack Scenario No	**Attack Point**	**Affected Process Measurement(s)**	**Attack Description**	**Attack Impact**
12	PLC2	flowlevel	Positive scaling of measurements sent by PLC2 towards PLC1 by a fixed scaling factor	Increased flowlevel in the pipe increases the rate of filling of the bottle; can lead to bottle overflow
13	PLC3	bottle_liquidlevel	Positive scaling of measurements sent by PLC3 towards PLC1 by a fixed scaling factor	PLC1 upon receiving wrong measurements regarding bottle capacity decreases the rate of flow of liquid from the tank to the bottle
14	PLC2, PLC3	flowlevel, bottle_liquidlevel	Attack scenario 12 + Attack scenario 13	Causes the combined impact of attack scenarios 12 & 13
15	PLC2	flowlevel	Scale up measurements from PLC2 by a random value from a uniform distribution	Similar impact as that of attack scenario 12
16	PLC3	bottle_liquidlevel	Scale up measurements from PLC3 by a random value from a uniform distribution	Similar impact as that of attack scenario 13
17	PLC2, PLC3	flowlevel, bottle_liquidlevel	Attack scenario 15 + Attack scenario 16	Similar impact as that of attack scenario 14
18	PLC2	flowlevel	Negative scaling of measurements sent by PLC2 towards PLC1 by a fixed scaling factor	Decreased flowlevel in pipe decreases the rate of filling of the bottle; can slow down the normal operations of the plant
19	PLC3	bottle_liquidlevel	Negative scaling of measurements sent by PLC3 towards PLC1 by a fixed scaling factor	PLC1 upon receiving wrong measurements regarding bottle capacity increases the rate of flow of liquid from the tank to the bottle
20	PLC2, PLC3	flowlevel, bottle_liquidlevel	Attack scenario 18 + Attack scenario 19	Causes the combined impact of attack scenarios 18 & 19
21	PLC2	flowlevel	Scale down measurements from PLC2 by a random value from a uniform distribution	Similar impact as that of attack scenario 18
22	PLC3	bottle_liquidlevel	Scale down measurements from PLC3 by a random value from a uniform distribution	Similar impact as that of attack scenario 19
23	PLC2, PLC3	flowlevel, bottle_liquidlevel	Attack scenario 21 + Attack scenario 22	Similar impact as that of attack scenario 20

5.2 Generated Dataset

Table 5.6 shows an excerpt of the generated labelled dataset containing process measurements during normal operation of the system and attack durations. This dataset is in CSV format and has 2705 data samples collected for 3 hours. Each data sample has 6 attributes as shown in Table 5.6. *Timestamp*[1] (format is YYYY-MM-DD HH24:MI:SS.FF7) reflects the time at which measurements are collected from PLC1 node, next 4 attributes reflect various process measurement values collected, and *class* represents the class label given to the data sample reflecting whether the sample is collected during normal operation or a specific attack.

Figure 5.1 shows the distribution of normal and anomalous data samples in the generated dataset. This dataset is further used to train the ML-based IDS. There are 1920 normal samples and 785 anomalous samples. Anomalous samples are collected for different types of attacks executed in the digital twin. Out of the 785 anomalous samples, 434 belong to calculated measurement injection attacks, 227 are from naive measurement injection attacks, 88 from network DoS attacks, and 36 from command injection attacks.

[1]https://docs.python.org/3/library/datetime.html#datetime.datetime.now

Table 5.6: Sample dataset (labelled)

timestamp	tank_liquidlevel	flowlevel	bottle_liquidlevel	motor_status	class
32:02.7	2.419907	0	0	1	Normal
32:06.1	2.374537	2.45	0.158796	1	Normal
32:09.6	2.238426	2.45	0.499074	1	Normal
32:12.7	2.102315	2.45	0.748611	1	Normal
32:15.6	1.977546	2.45	0.907407	0	Normal
32:18.8	1.886806	0	0	1	Normal
32:21.8	1.852778	2.45	0.090741	1	Normal
32:26.5	1.728009	2.45	0.362963	1	Normal
32:31.1	1.614583	2.45	0.521759	1	Normal
32:36.7	1.535185	0	0.748611	1	Normal
32:41.7	1.387731	2.45	0.907407	0	Normal
32:46.9	1.319676	0	0.181481	1	Normal
32:51.9	1.194907	2.45	0.431019	1	Normal
32:57.2	1.070139	2.45	0.6125	0	Command Injection
33:02.5	0.979398	0	0.793981	1	Normal
33:07.6	0.831944	2.45	0	0	Command Injection
33:13.1	0.741204	0	0.158796	0	Command Injection
33:18.4	0.63912	0	0.431019	1	Normal
33:23.5	0.514352	2.45	0.635185	1	Normal
33:28.2	0.412269	0	0.907407	0	Normal
33:32.9	0.33287	0	0.136111	1	Normal
33:37.5	5.765972	0	0.249537	0	Command Injection
33:42.2	5.675231	0	0.453704	1	Normal
33:47.9	5.561806	2.45	0.703241	0	Command Injection
33:52.9	5.437037	2.45	0.884722	1	Normal

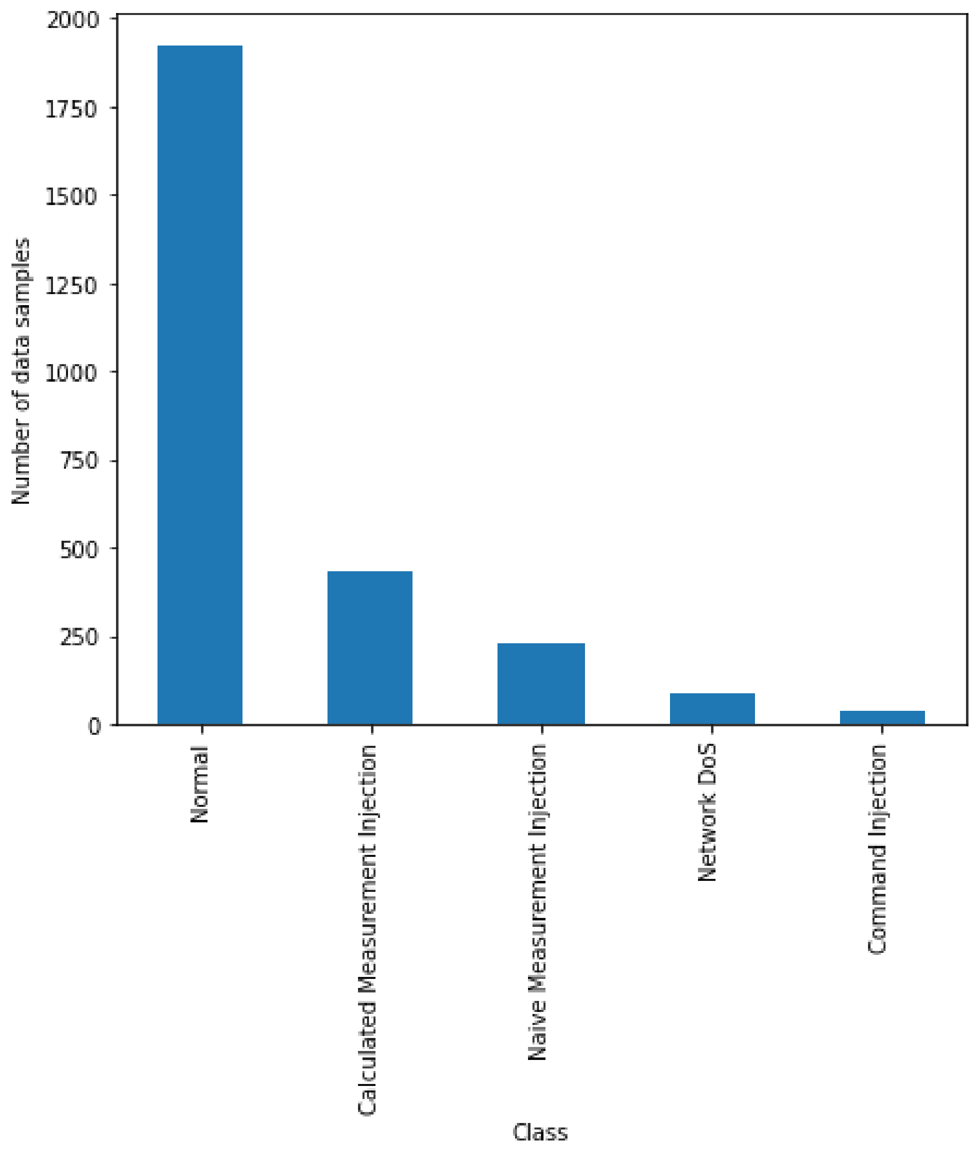
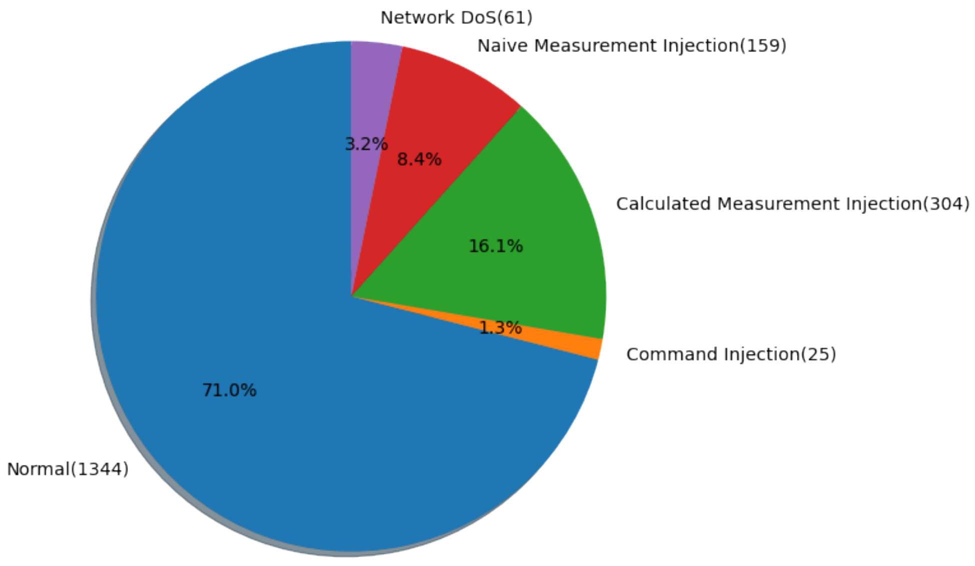


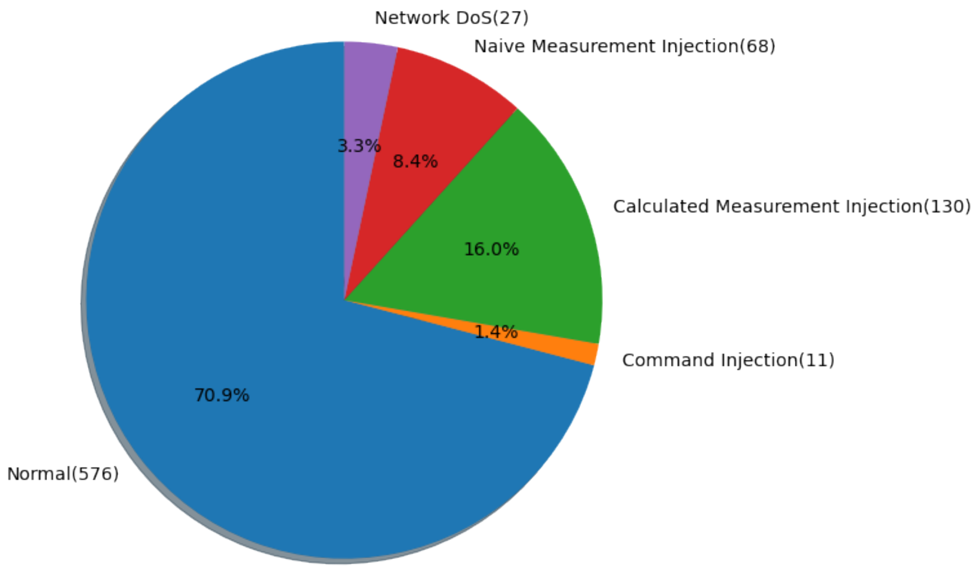
Figure 5.1: Distribution of data samples across different classes in the dataset

The size of the dataset used in ML-based intrusion detection on process data varies across different related work. Related works that use datasets generated from real-time simulations and testbeds of ICSs ([46], [42]) usually use training data containing 1000 to 10 000 data samples. Since this thesis uses generated dataset from the digital twin simulation, the size of the dataset used here is on par with the approaches that uses simulations and testbeds.

Furthermore, the distribution of target class labels in the training set and test set is shown in Figure 5.2. The labels in pie charts indicate the class labels along with the number of data samples. Additionally, the distribution percentage is marked on the slices of the pie chart. The distribution of class labels are similar across these two sets.



(a) Training set



(b) Test set

Figure 5.2: Distribution of target class labels in training set and test set

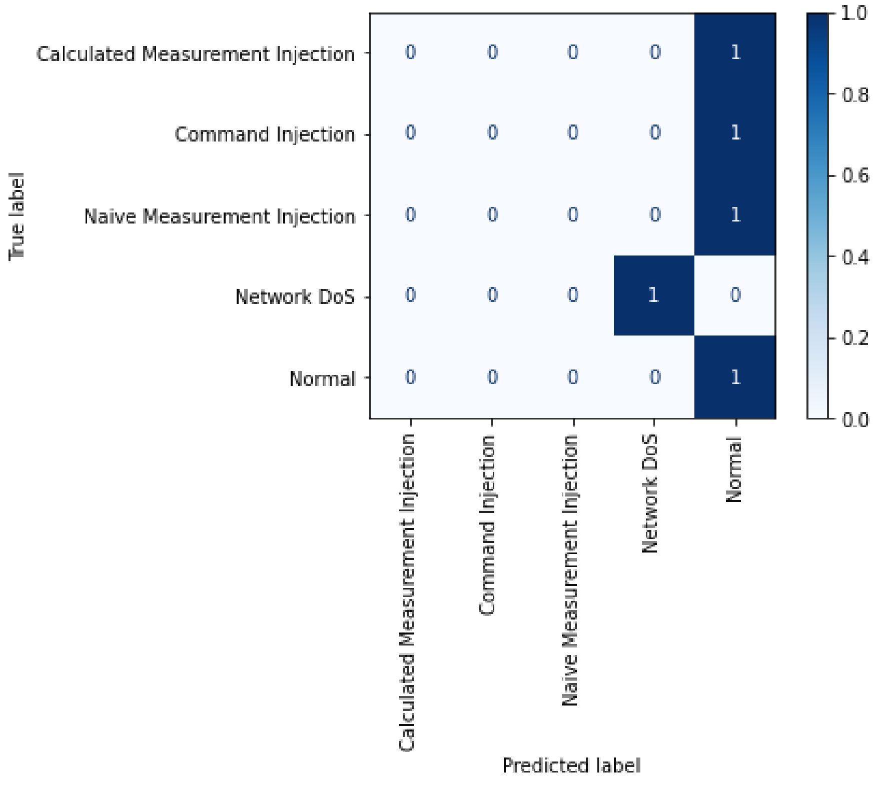
5.3 Evaluation of ML Algorithms

Figures 5.3 to 5.6 show the confusion matrices with normalization by the number of elements in each class for the eight supervised ML algorithms evaluated on the test set of the labelled dataset. The confusion matrix is a useful metric to evaluate the efficiency of an algorithm to correctly classify the unseen data samples based on a training dataset. An algorithm is considered good if the normalized matrix has all diagonal elements with values equal to 1 or closer to 1. The color scheme used represents the values varying from 0 to 1 as blue color ascending from a lighter shade to a darker shade. Thus, if the diagonal elements of a matrix are all dark in color, it constitutes a good classifier.

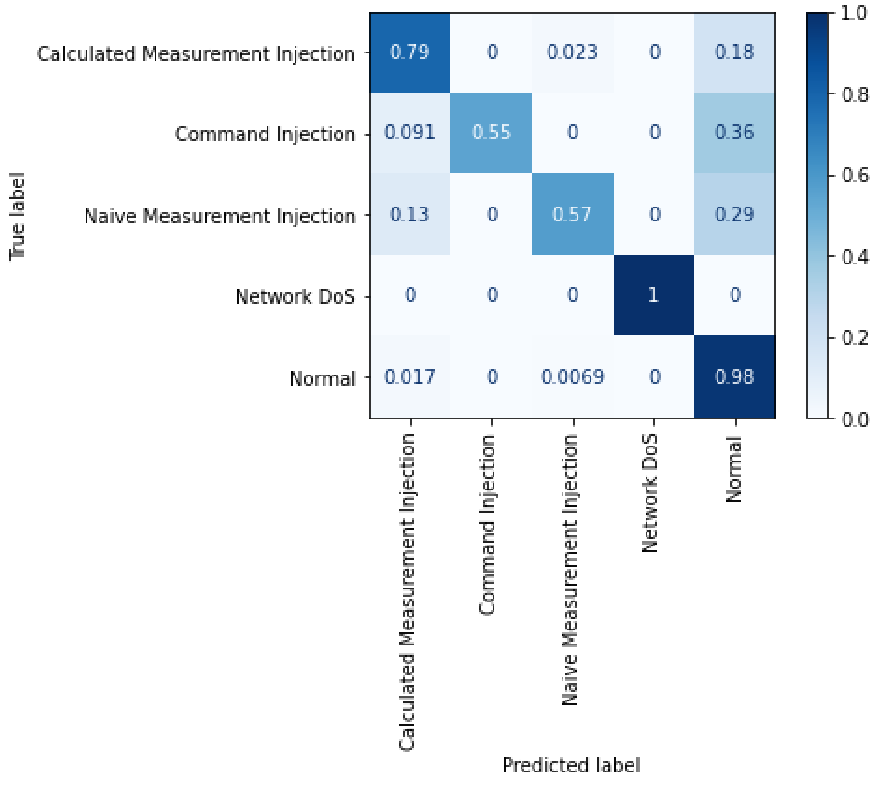
It is clear from Figures 5.3 to 5.6 that all the models are successful in classifying correctly the samples belonging to network DoS attacks. However, SVM, KNN, LR, MLP, and NB have zero values in some of the diagonal elements. This means that these models are not able to classify the data samples belonging to that particular class for which the values are zero. For example, in the case of SVM, the model is not able to classify correctly the samples from naive measurement injection, command injection, and calculated measurement injection attacks.

From Figure 5.5, DTC shows better results for naive measurement injection attacks compared to RF and GB classifiers. The GB classifier outperforms RF and DTC in terms of normal samples. GB has same results as that of the RF classifier for calculated measurement injection attacks.

Based on the evaluation of confusion matrices of the above 8 classifiers, it can be concluded that the GB classifier is the best algorithm when it comes to predicting samples belonging to normal, network DoS, and calculated measurement injection. DTC outperforms the GB algorithm in the case of predicting naive measurement injection attacks. For command injection, NB gives the best score compared to all the other algorithms. Therefore, GB, DTC, and NB are chosen as the Level 0 classifiers for the stacked ensemble classifier approach. The Level 1 classifier used is a neural network using MLP classifier. This is done to utilize the benefit of the learning process used in neural network model compared to other algorithms. The normalized confusion matrix for the stacking classifier is shown in Figure 5.7. It is clear from this figure that stacking shows better results compared to individual classifiers.

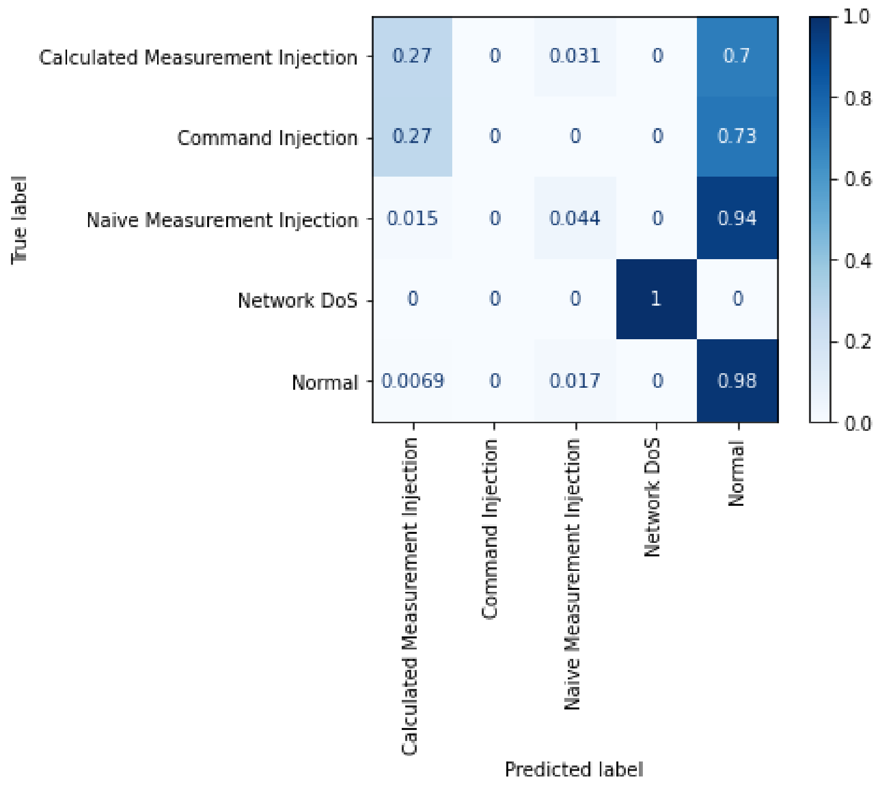


(a) Support Vector Machine(SVM)

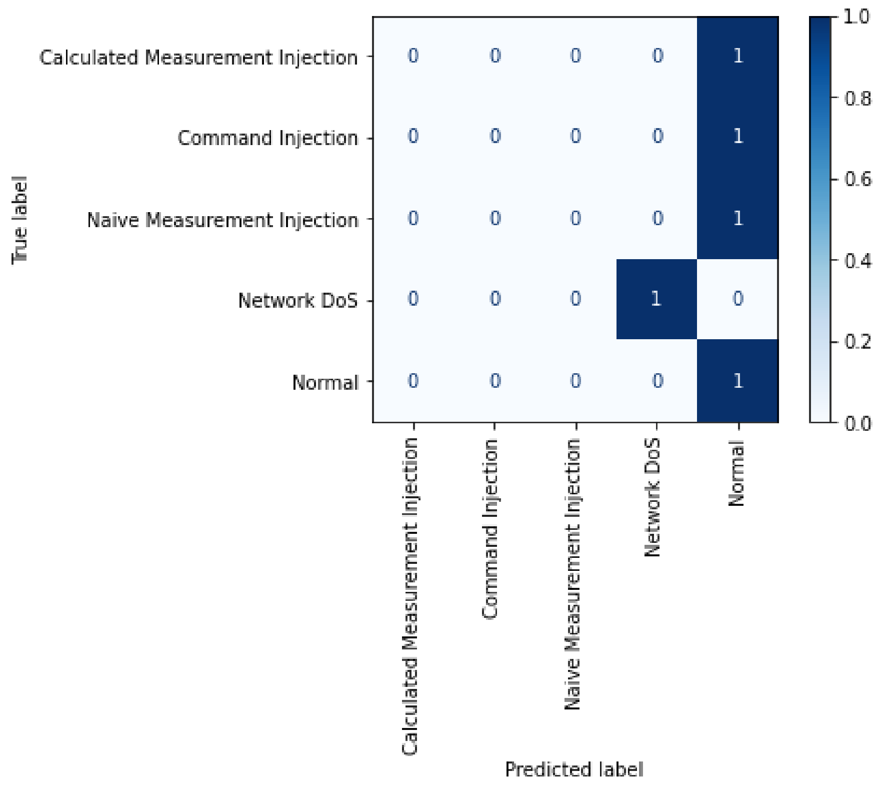


(b) Random Forest(RF)

Figure 5.3: Normalized confusion matrices for SVM and RF

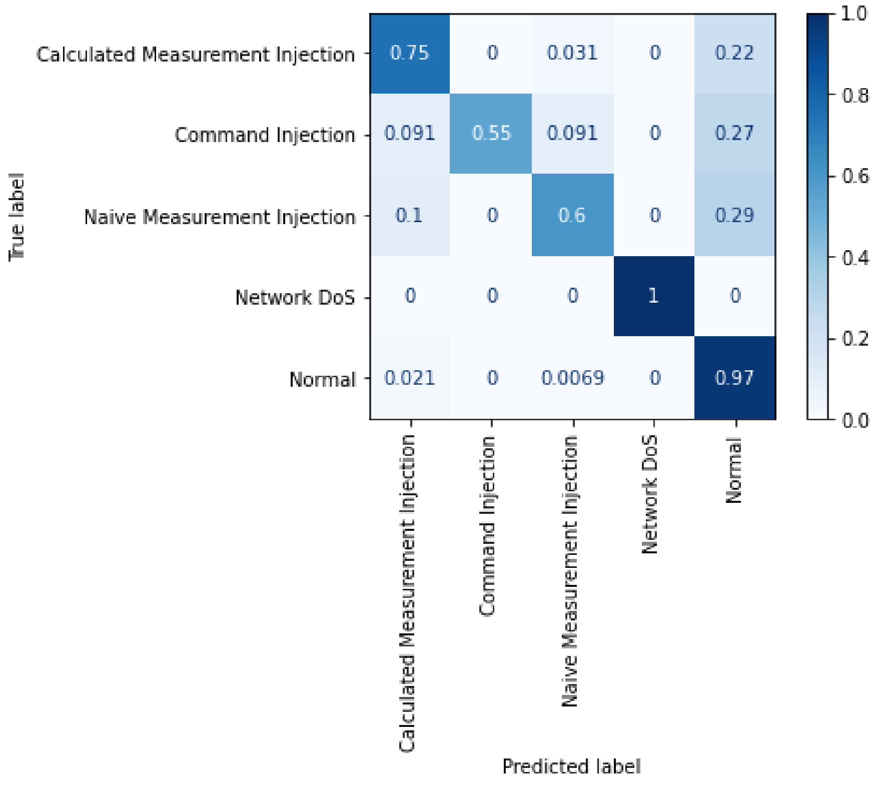


(a) K Nearest Neighbor(KNN)

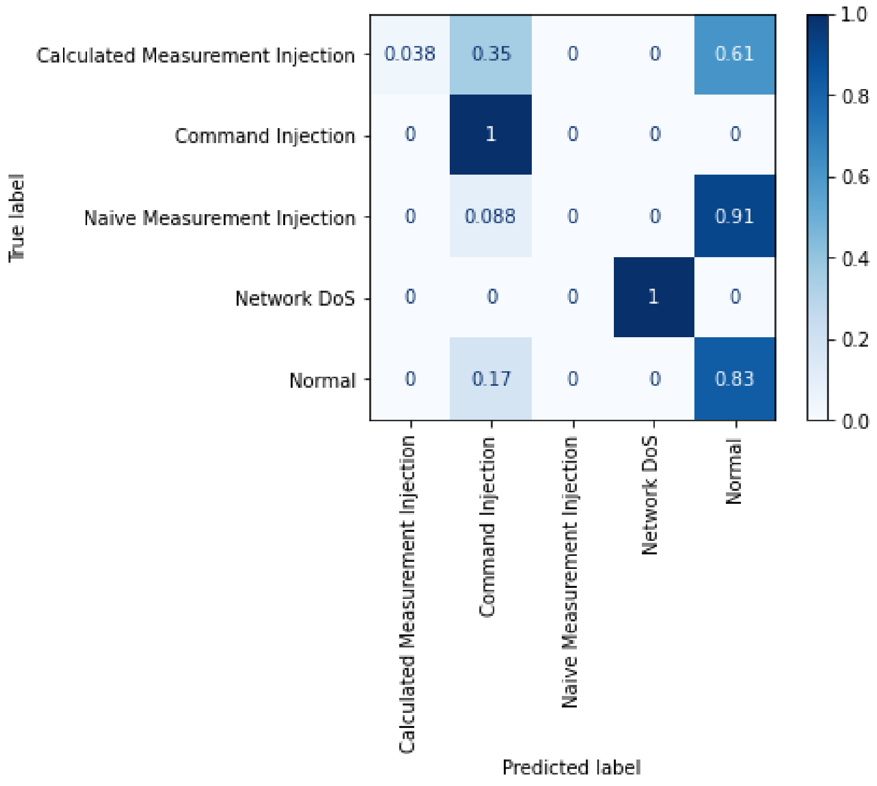


(b) Logistic Regression(LR)

Figure 5.4: Normalized confusion matrices for KNN and LR

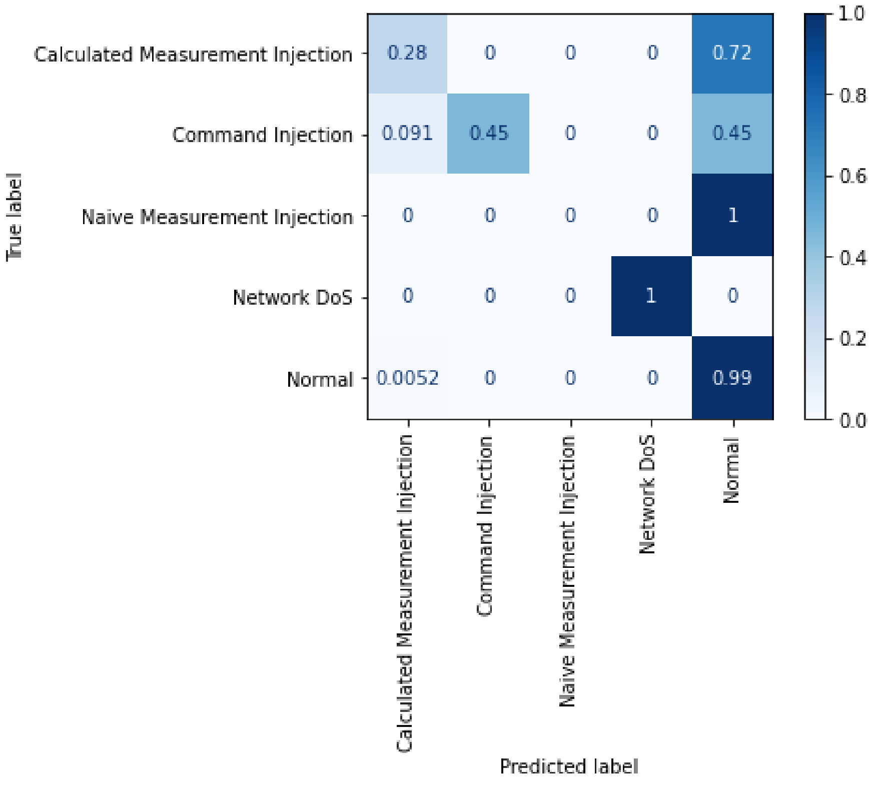


(a) Decision Tree Classifier(DTC)

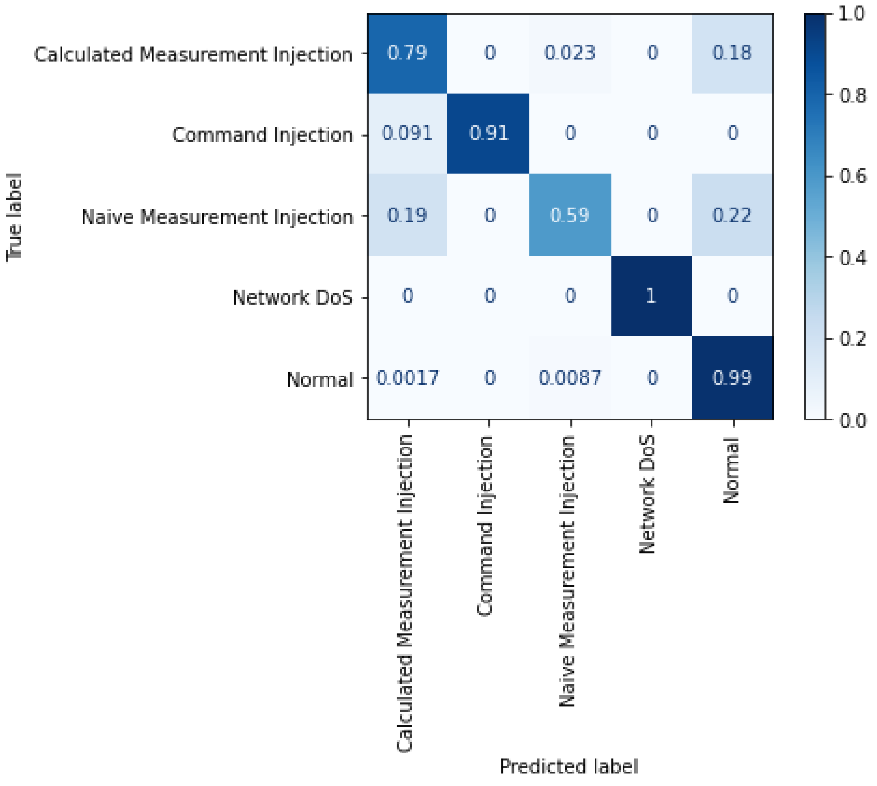


(b) Naive Bayes(NB)

Figure 5.5: Normalized confusion matrices for DTC and NB



(a) Multi-Layer Perceptron(MLP)



(b) Gradient Boosting(GB)

Figure 5.6: Normalized confusion matrices for MLP and GB

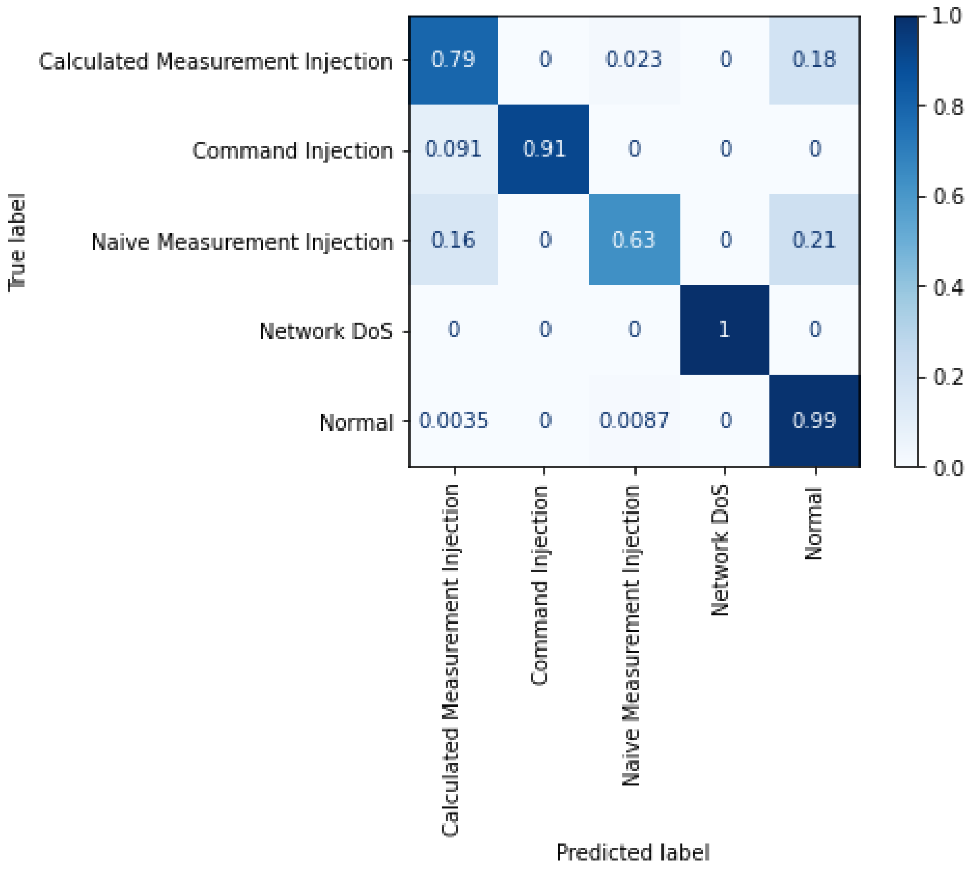


Figure 5.7: Normalized confusion matrix for stacking classifier

Figures 5.8 to 5.11 provide the plots of different ML algorithms vs the classification metric score. The X-axis for all the plots in these figures represents the eight algorithms along with the stacking classifier evaluated using the labelled dataset. Y-axis represents the overall score for classification metrics used.

As shown in Figure 5.8, the best accuracy score is 0.926 given by the stacking classifier model, followed by GB with a score of 0.924. NB model gives the worst score of 0.64. From Figure 5.9, the precision score is the highest for the stacking classifier model and is 0.932, followed by the GB classifier with a score of 0.928. SVM and LR models give the worst precision score of 0.347. In terms of recall score (refer to Figure 5.10), the stacking classifier has the highest score of 0.864 followed by GB with a score of 0.856. SVM and LR give the worst recall score of 0.4. As shown in Figure 5.11, F1-score is the highest for the stacking classifier and that is 0.884. This is not surprising as F1-score is the highest when precision and recall scores are high, which is true for the stacking classifier model. F1-score is the worst for SVM and LR, and the value is 0.37.

For all the plots shown in Figures 5.8 to 5.11, the stacking classifier has the best score (close to 1) for all of the classification metrics. For ML-based classification algorithms trained using an imbalanced dataset, the most important metrics are recall and F1-score. Since the stacking classifier model outperforms all other algorithms in all metrics in terms of recall and F1-score, this model is considered a good model for implementing ML-based IDS.

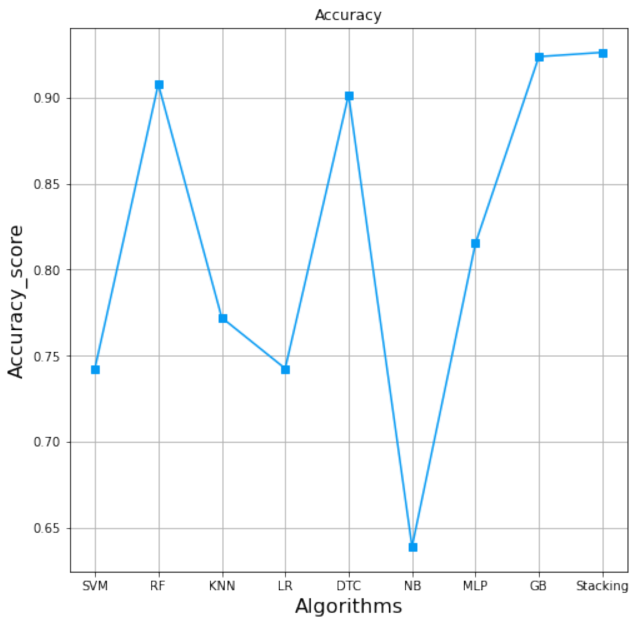


Figure 5.8: Accuracy scores for ML algorithms

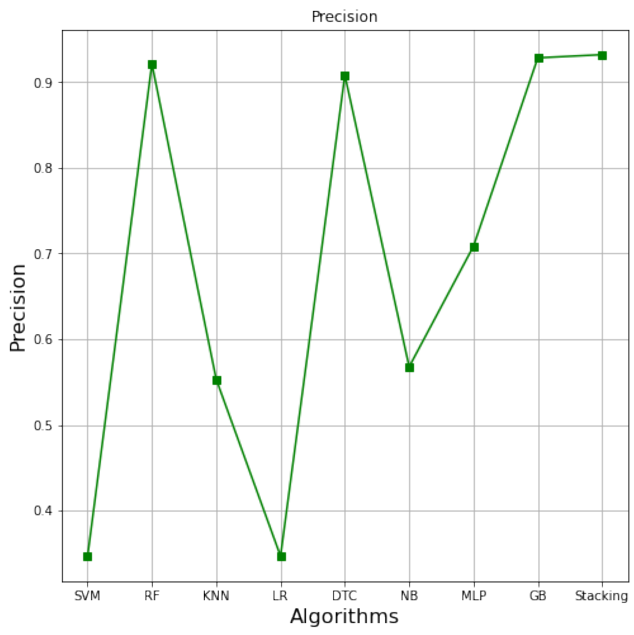


Figure 5.9: Precision scores for ML algorithms

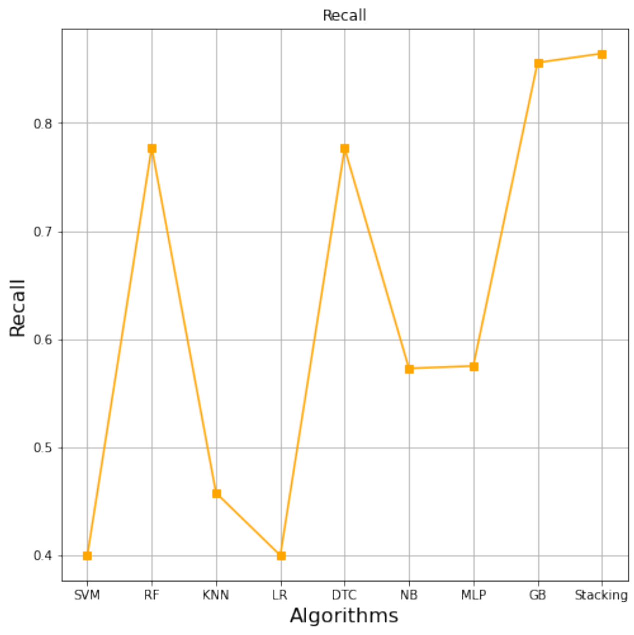


Figure 5.10: Recall scores for ML algorithms

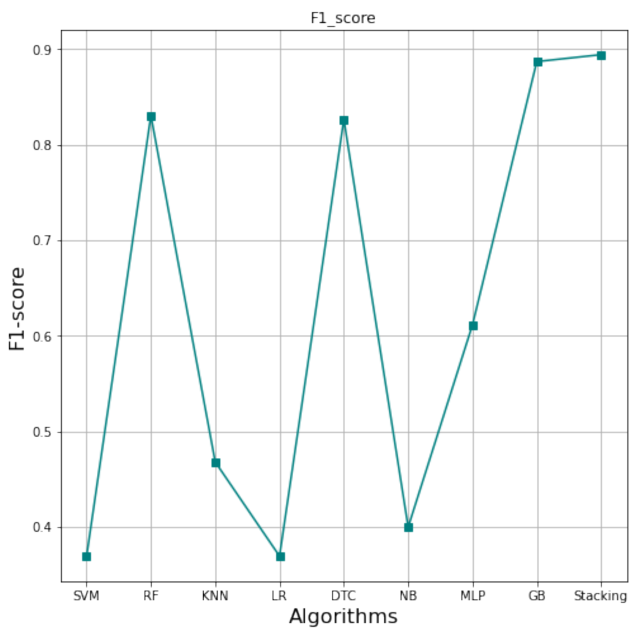


Figure 5.11: F1-score values for ML algorithms

Table 5.7 lists the scores of different classification metrics used to evaluate the ML algorithms on the labelled dataset. The scores are rounded to three decimal places. The best score for each metric is highlighted in green color, whereas the worst score is highlighted in orange color.

Table 5.7: Classification metric score values on evaluating different ML algorithms

Algorithm\Classifcation Metric	**Accuracy**	**Precision**	**Recall**	**F1-score**
SVM	0.743	0.347	0.4	0.37
RF	0.908	0.921	0.777	0.83
KNN	0.772	0.553	0.458	0.468
LR	0.743	0.347	0.4	0.37
DTC	0.901	0.913	0.775	0.827
NB	0.64	0.568	0.573	0.4
MLP	0.789	0.735	0.545	0.584
GB	0.924	0.928	0.856	0.887
Stacking classifier	0.926	0.932	0.864	0.894

Furthermore, we perform an analysis of the samples misclassified by the stacking classifier model. Initially, we compute the prediction probabilities for different class labels as predicted by the stacking classifier. This step helps to check if the misclassification of a sample happened because of a case where the model chooses a label based on its alphabetical name when two or more labels (including the correct label) are predicted with the same probability. However, no such instances are found in the final predictions given by stacking classifier. We do not eliminate the possibility of occurence of this case in the prediction outputs of Level 0 classifiers. Using optimized Level 0 classifiers after hyper parameter tuning can be beneficial in this case. As discussed already, this optimization is left for future work.

We also perform an analysis of the process measurement samples that are misclassified by the stacking classifier model. Samples belonging to normal, naive measurement injection, and calculated measurement injection are the ones that get misclassified. This is because the current implementation of ML models learn only the relationship between different measurements in a single sample. There is no learning of correlation among process measurements across samples. Such a learning using time-series based algorithms can be useful in decreasing the misclassification rate in this case. We leave this study and implementation for future work.

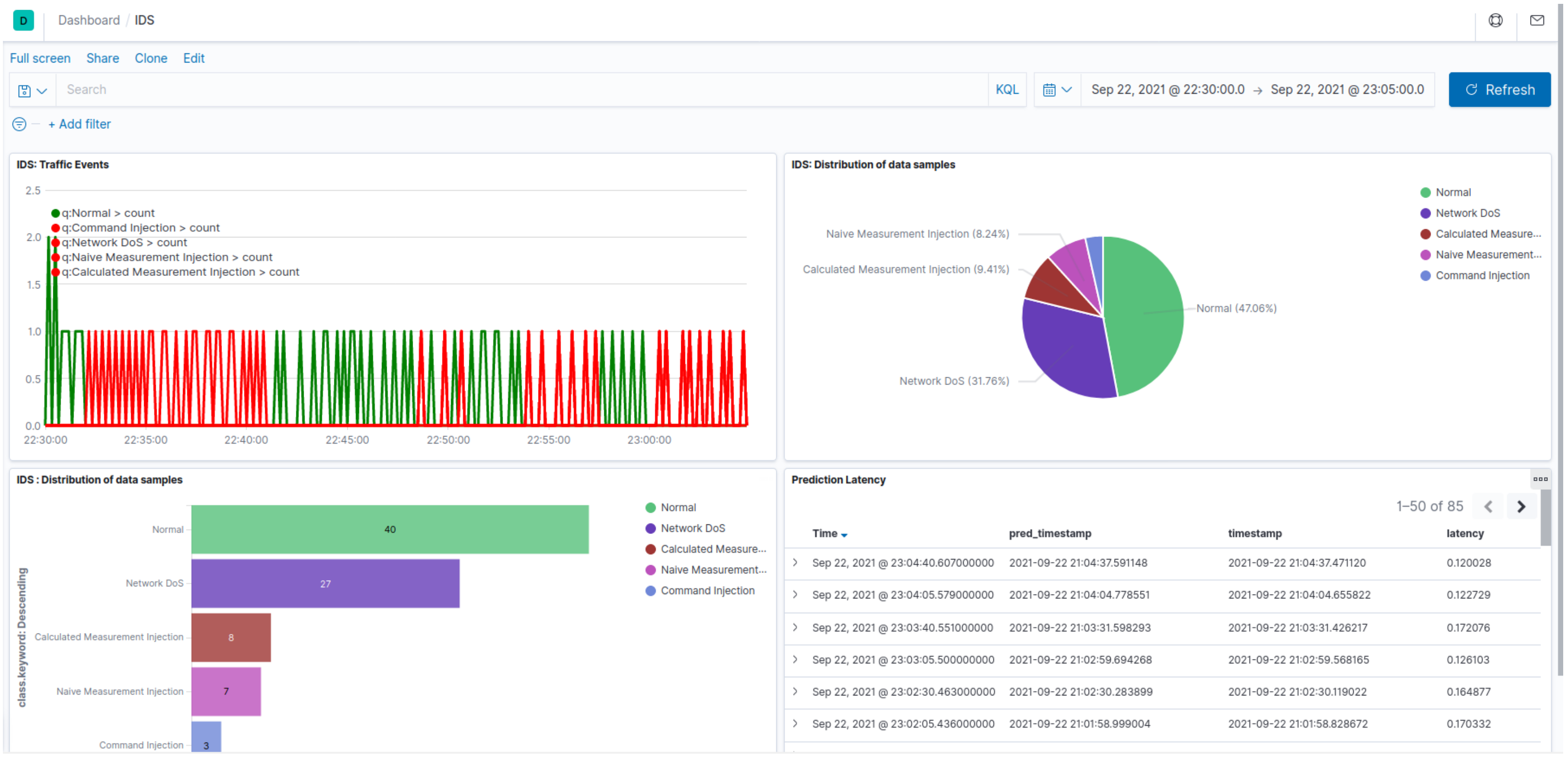
Table 5.8 lists the computation time taken to execute all of the above mentioned algorithms. *Training time* indicates the time taken to train a model on the training set in seconds whereas *Prediction time* indicates the time taken by the model to make predictions on the test set in seconds. MLP classifier takes the highest time to train, whereas NB classifer takes the least time. All the algorithms take less time for prediction compared to training. Instead of comparing all the algorithms in terms of computational complexity, we discuss and compare only those algorithms which are identified best for implementing IDS. GB, DTC, and NB took less training time and prediction time compared to the stacking classifier model. This clearly shows that there is always a computational overhead when a stacked ensemble model is used. The trade-off between classification performance and computational performance also needs to be evaluated thoroughly before using stacked ensemble models. However, such an evaluation is not in the scope of this work. Also, training of IDS in this work is done offline and at once. Hence, we do not consider this as a huge overhead. However, we mention that the training time can become an overhead if the training dataset is large.

Table 5.8: Computation time for ML algorithms

Algorithm	**Training time (seconds)**	**Prediction time (seconds)**
SVM	0.091	0.045
RF	1.058	0.088
KNN	0.025	0.071
LR	0.071	0.006
DTC	0.024	0.008
NB	0.017	0.009
MLP	80.345	0.016
GB	0.852	0.013
Stacking classifier	14.98	0.041

5.4 IDS Results

ML-based IDS running in the IDS Docker container is trained offline at first using the generated dataset. Further, this IDS classifies the incoming data samples from the digital twin container during the live operation of the system. The results of this classification are stored as a CSV file. These results are displayed as visualizations in a dashboard created in kibana. Figure 5.12 depicts a screenshot of the dashboard in kibana with the IDS results for a 30-minutes duration. There are four visualizations displayed in this dashboard. The zoomed versions of each of these visualizations are provided as Figures 5.13, 5.14, 5.15, and 5.16.



Time	pred_timestamp	timestamp	latency
Sep 22, 2021 @ 23:04:40.607000000	2021-09-22 21:04:37.591148	2021-09-22 21:04:37.471120	0.120028
Sep 22, 2021 @ 23:04:05.579000000	2021-09-22 21:04:04.778551	2021-09-22 21:04:04.655822	0.122729
Sep 22, 2021 @ 23:03:40.551000000	2021-09-22 21:03:31.598293	2021-09-22 21:03:31.426217	0.172076
Sep 22, 2021 @ 23:03:05.500000000	2021-09-22 21:02:59.694268	2021-09-22 21:02:59.568165	0.126103
Sep 22, 2021 @ 23:02:30.463000000	2021-09-22 21:02:30.283899	2021-09-22 21:02:30.119022	0.164877
Sep 22, 2021 @ 23:02:05.436000000	2021-09-22 21:01:58.999004	2021-09-22 21:01:58.828672	0.170332

Figure 5.12: Screenshot of IDS dashboard

Figure 5.13 is a time-series visualization of data samples using Timelion[2]. The X-axis of this visualization represents the timestamp from incoming data samples. Y-axis represents the number of samples at a given point of time. All data samples classified as normal by the IDS are shown in green color while all the anomalies are shown in red color.

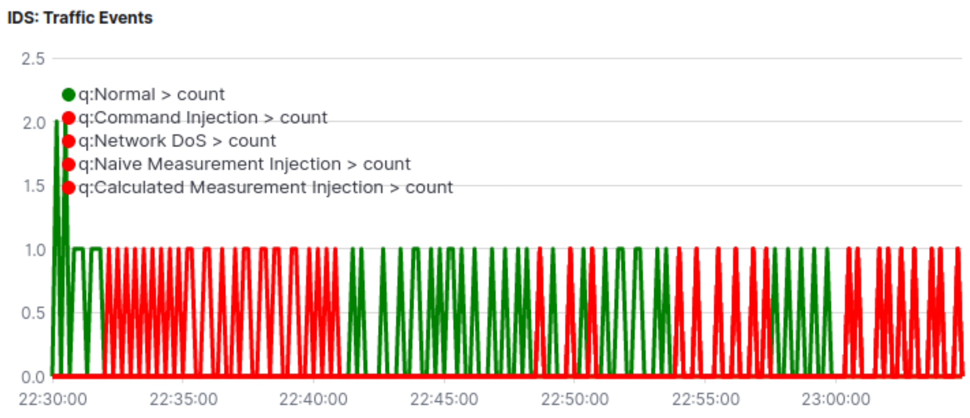


Figure 5.13: IDS Events

Figure 5.13 shows the IDS classification results for 4 attack scenarios executed in the 30-minutes duration. These scenarios belong to four different attack types and are executed for 3-minutes duration each. The attack scenarios executed are in the following order: attack scenario 1 from command injection type, attack scenario 4 from network DoS, attack scenario 6 from naive measurement injection, and attack scenario 14 from calculated measurement injection. The results look promising as we observed only two misclassified data samples (1 command injection sample misclassified as normal and 1 calculated measurement injection attack sample misclassified as naive measurement injection) out of the total 85 samples.

Figure 5.14 is a pie-chart visualization of the distribution of data samples across different class labels as classified by the IDS. As seen in the figure, the percentage of samples belonging to different classes is represented by different colors.

[2]https://www.elastic.co/guide/en/kibana/current/timelion.html

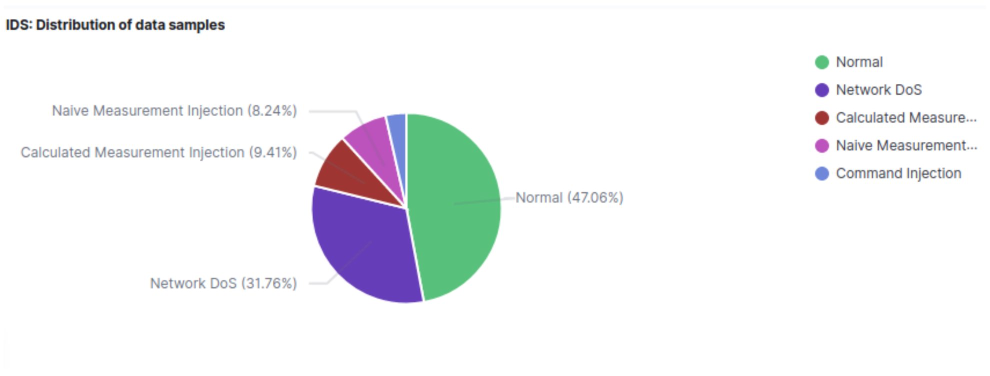


Figure 5.14: Pie-chart representation of classification of data samples by IDS

Figure 5.15 is a bar chart visualization of the *class* label predicted on the incoming data samples by the IDS. Different classes are represented in different colors.

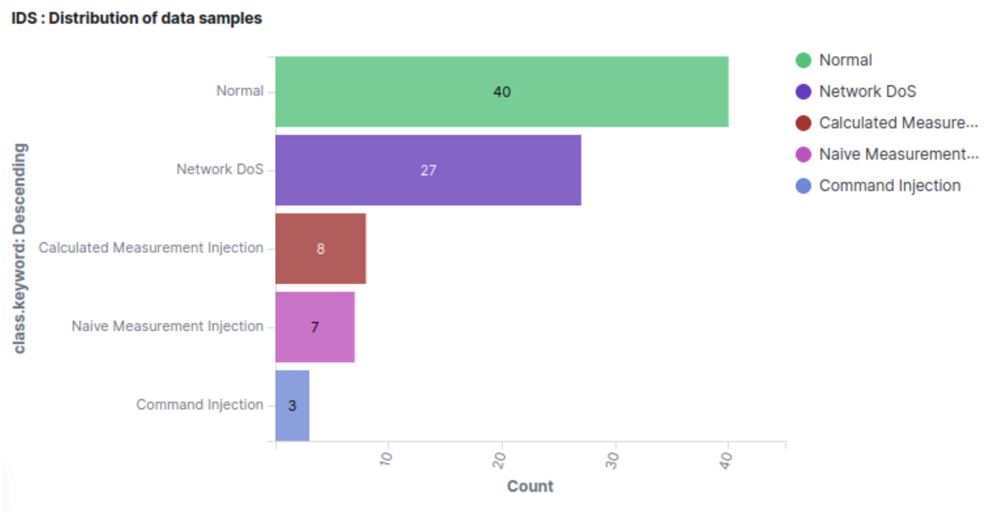
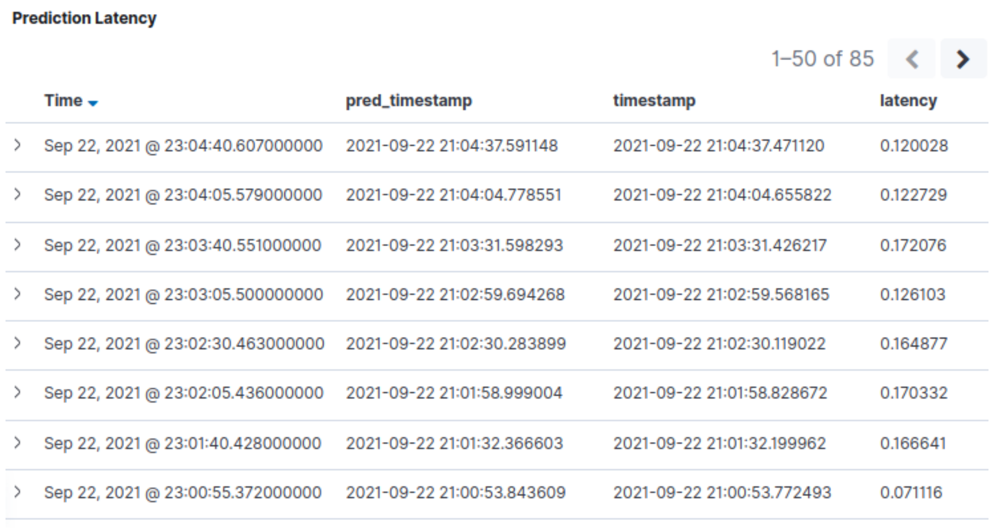


Figure 5.15: Bar chart representation of classification of data samples by IDS

Figure 5.16 shows the latency in classifying the samples by the IDS. ’timestamp’ indicates the timestamp from the incoming data samples where as ’pred_timestamp’ indicates the time of predictions. Another field named ’latency’ is also provided which gives the latency in predictions by IDS in seconds. For the timeframe shown in this screenshot, the average latency is 0.1 seconds. Hence we can say that the predictions happen in near real-time.

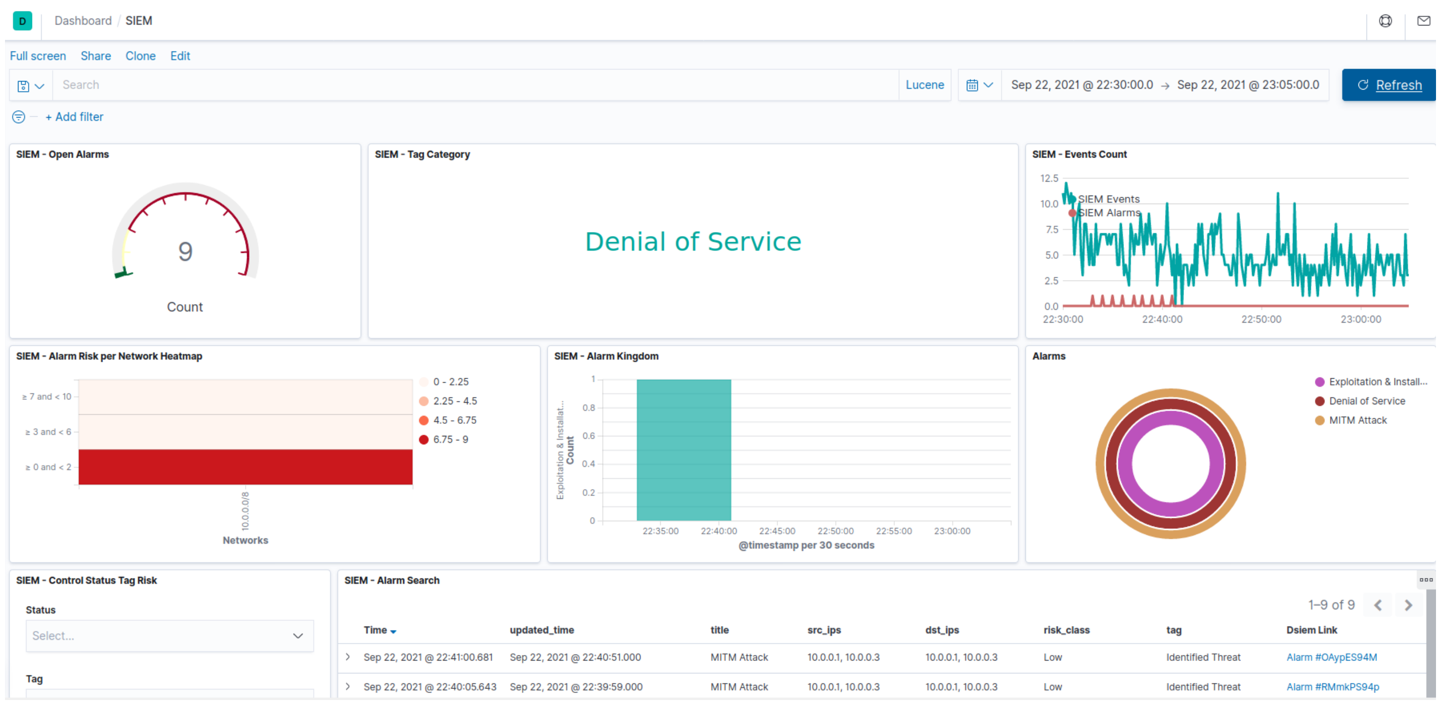

Prediction Latency

1–50 of 85

Time	pred_timestamp	timestamp	latency
Sep 22, 2021 @ 23:04:40.607000000	2021-09-22 21:04:37.591148	2021-09-22 21:04:37.471120	0.120028
Sep 22, 2021 @ 23:04:05.579000000	2021-09-22 21:04:04.778551	2021-09-22 21:04:04.655822	0.122729
Sep 22, 2021 @ 23:03:40.551000000	2021-09-22 21:03:31.598293	2021-09-22 21:03:31.426217	0.172076
Sep 22, 2021 @ 23:03:05.500000000	2021-09-22 21:02:59.694268	2021-09-22 21:02:59.568165	0.126103
Sep 22, 2021 @ 23:02:30.463000000	2021-09-22 21:02:30.283899	2021-09-22 21:02:30.119022	0.164877
Sep 22, 2021 @ 23:02:05.436000000	2021-09-22 21:01:58.999004	2021-09-22 21:01:58.828672	0.170332
Sep 22, 2021 @ 23:01:40.428000000	2021-09-22 21:01:32.366603	2021-09-22 21:01:32.199962	0.166641
Sep 22, 2021 @ 23:00:55.372000000	2021-09-22 21:00:53.843609	2021-09-22 21:00:53.772493	0.071116

Figure 5.16: Latency in predictions by IDS

A screenshot of the SIEM dashboard which is already available as part of the original framework in [3] is shown in Figure 5.17. This dashboard displays the results of the correlation engine that uses system logs to identify incidents. This screenshot is captured for the same timeframe as that of the IDS dashboard shown in Figure 5.12. It is quite clear that the SIEM dashboard does not report any alarm for the process-aware attacks executed, except for network DoS. This is because the executed attacks except for network DoS attacks do not tamper with the system logs and hence the correlation engine which is based on a set of rules checking the severity of system logs is unable to detect such attacks. In the case of DoS attacks, system logs are reported with a *WARNING* severity and therefore get detected by the correlation engine. This clearly shows the advantage of adding ML-based IDS to the original framework.



Time	updated_time	title	src_ips	dst_ips	risk_class	tag	Dsiem Link
Sep 22, 2021 @ 22:41:00.681	Sep 22, 2021 @ 22:40:51.000	MITM Attack	10.0.0.1, 10.0.0.3	10.0.0.1, 10.0.0.3	Low	Identified Threat	Alarm #OAypES94M
Sep 22, 2021 @ 22:40:05.643	Sep 22, 2021 @ 22:39:59.000	MITM Attack	10.0.0.1, 10.0.0.3	10.0.0.1, 10.0.0.3	Low	Identified Threat	Alarm #RMmkPS94p

Figure 5.17: Screenshot of SIEM dashboard

Chapter 6

Conclusions and Future work

The conclusions of this thesis are stated in Section 6.1. This is followed by the limitations of the work which are stated in Section 6.2. Finally, Section 6.3 lists the future work that can be done on top of the current implementation of the thesis.

6.1 Conclusions

The key goal of this thesis is to deliver a security framework for ICSs which includes the digital twin of the ICS for security monitoring and an ML-based IDS for detecting intrusions in near real-time. Such a framework can be deployed in production environments along with the real physical system and can be used for cybersecurity analysis of the system without interfering with the operation of the physical system.

The first step towards achieving this goal is to identify an open-source implementation of the digital twin of an ICS. Although many research works discuss the concept of digital twin for ICSs, there are not many works that dive deep into its implementation aspects using open-source tools. To this end, a thorough literature study followed by an extensive evaluation in terms of implementation aspects is performed on the available open-source knowledge-driven digital twin solutions of ICSs. At the end of this step, an open-source solution is selected among the evaluated solutions based on several aspects such as ease of implementation of the solution, ability of the solution to run as a standalone simulation, and representation of a real industrial use case. This work intentionally omits the synchronization aspects of the digital twin with the physical system.

The selected digital twin solution is of an industrial filling plant which is implemented using Mininet-based MiniCPS. Furthermore, this thesis uses a microservices-based architecture realized using Docker compose as the digital twin framework. This offers great flexibility to add or remove components to this framework. Additionally, this framework offers the benefit of easy integration of the solution in production environments.

As the next step, different realistic process-aware attack scenarios are modelled and executed in the digital twin. These attacks target to disturb the normal operation of the industrial control process running in the system. The modelled attack scenarios belong to four different attack types, naming network DoS, calculated measurement injection, naive measurement injection, and command injection. A total of 23 attack scenarios are modelled and executed as part of this work. At first, a training dataset to train ML-based IDS is generated which contains process measurements from the digital twin during normal operation as well as each of the modelled attack scenarios. This dataset is further labelled with the target class labels for different attack types. Further, the process measurements collected during the live operation of the system are provided to the IDS to classify them based on this pre-trained model.

The final step is to design and evaluate an ML-based IDS which can be integrated into the proposed security framework. This IDS is implemented as a Docker container which is built as an extension of the digital twin framework. The choice of ML algorithm to design IDS is done in two stages. The first stage involves the evaluation of eight supervised ML classifier models on the labelled dataset. As the next stage, a stacked ensemble model that combines those individual classifiers which perform best for a particular target class label is designed and evaluated on the labelled dataset. This evaluation uses 70% of the labelled dataset as the training set and the remaining 30% as the test set. Based on this evaluation, the stacking classifier approach gives the best results compared to individual classifiers and is hence chosen as the final classifier model to implement IDS.

ML-based IDS is trained offline at first using the whole labelled dataset and is further used to classify the new incoming data samples based on this trained model. This classification task is done in near real-time, showcasing the effectiveness of ML-based intrusion detection.

In conclusion, this thesis has successfully proposed a security framework for an ICS using its digital twin for security monitoring. This framework also includes an ML-based IDS capable of detecting and classifying process-aware intrusions in the system. The proposed framework is extensible and flexible and is implemented using open-source tools. To the best of our knowledge, there is no other work that introduces such a holistic security framework for ICSs that includes both the digital twin and ML-based IDS, and demonstrates the whole use case of security monitoring and intrusion detection in ICSs.

6.2 Limitations

The identified limitations of this thesis are as listed below:

1. MiniCPS-based implementation of digital twin supports only Modbus and ENIP protocols. Therefore, emulation of systems using other protocols in MiniCPS is not possible. However, the microservices-based framework used in this thesis offers the flexibility for easy replacement of digital twin with other implementations.

2. Evaluation of ML algorithms does not take into account hyper-parameter tuning to make better predictions. The default parameters used to evaluate models may not be the best parameters for that model because of this.

3. ML algorithms used in this work do not consider correlation between process measurements across data samples. Analysis of the misclassified data samples by the stacked ensemble model shows that considering this correlation may improve the classification results. This can be achieved by using time-series based algorithms which are not considered in this thesis.

6.3 Future work

Following the limitations listed in Section 6.2, the immediate work that can be done on top of the current implementation of the thesis is to reevaluate the algorithms with hyper-parameter tuning.

The current implementation of ML-based IDS uses supervised learning. Evaluation of unsupervised and semi-supervised learning algorithms to detect intrusions can be taken up as an improvement in the future. Such an approach can help detect zero-day attacks and can also avoid the need for the labelling of the dataset.

Another improvement that can be used to reduce the misclassification rate is to consider including time series-based algorithms to implement IDS. Such algorithms learn the correlation between process measurements across data instances and take into account how these measurements change over time.

Improving the fidelity of the digital twin by integrating data and system states from its physical twin can be also considered for future work. This can be done either using data acquisition from live systems in real-time (maybe at regular clock intervals) or by using passive state replication approaches. Furthermore, this synchronization ensures that the effects of attacks happening in physical twin will be reflected in the digital twin and security analysis using digital twin ensures the smooth running of the real physical process.

Additionally, this framework can be used to develop and test intrusion prevention measures against different attack types without the need for a physical system. Such a testing platform to conduct a what-if analysis is useful before the final integration of preventive measures in the physical environment.

References

[1] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020. doi: 10.1109/ACCESS.2020.2998358

[2] C. Gehrmann and M. Gunnarsson, "A digital twin based industrial automation and control system security architecture," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 669–680, 2020. doi: 10.1109/TII.2019.2938885

[3] M. Dietz, M. Vielberth, and G. Pernul, "Integrating digital twin security simulations in the security operations center," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ser. ARES '20. New York, NY, USA: Association for Computing Machinery, 2020. doi: 10.1145/3407023.3407039. ISBN 9781450388337. [Online]. Available: https://doi.org/10.1145/3407023.3407039

[4] Shodan. [Online]. Available: https://ics-radar.shodan.io/

[5] H.-K. Shin, W. Lee, J.-H. Yun, and H. Kim, "HAI 1.0: Hil-based augmented ICS security dataset," in *13th USENIX Workshop on Cyber Security Experimentation and Test (CSET 20)*. USENIX Association, Aug. 2020. [Online]. Available: https://www.usenix.org/conference/cset20/presentation/shin

[6] A. Bécue, E. Maia, L. Feeken, P. Borchers, and I. Praça, "A new concept of digital twin supporting optimization and resilience of factories of the future," *Applied Sciences*, vol. 10, no. 13, 2020. doi: 10.3390/app10134482. [Online]. Available: https://www.mdpi.com/2076-3417/10/13/4482

[7] R. Langner, “Stuxnet: Dissecting a cyberwarfare weapon,” *IEEE Security Privacy*, vol. 9, no. 3, pp. 49–51, 2011. doi: 10.1109/MSP.2011.67

[8] T. Alladi, V. Chamola, and S. Zeadally, “Industrial control systems: Cyberattack trends and countermeasures,” *Computer Communications*, vol. 155, pp. 1–8, 2020. doi: https://doi.org/10.1016/j.comcom.2020.03.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366419319991

[9] F. Akbarian, E. Fitzgerald, and M. Kihl, “Intrusion detection in digital twins for industrial control systems,” 09 2020. doi: 10.23919/SoftCOM50211.2020.9238162 pp. 1–6.

[10] L. Dhirani, E. Armstrong, and T. Newe, “Industrial iot, cyber threats, and standards landscape: Evaluation and roadmap,” *Sensors*, vol. 21, 06 2021. doi: 10.3390/s21113901

[11] B. A. Talkhestani, T. Jung, B. Lindemann, N. Sahlab, N. Jazdi, W. Schlögl, and M. Weyrich, “An architecture of an intelligent digital twin in a cyber-physical production system,” *at - Automatisierungstechnik*, vol. 67, pp. 762 – 782, 2019.

[12] M. Eckhart and A. Ekelhart, “Towards security-aware virtual environments for digital twins,” in *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*, ser. CPSS ’18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3198458.3198464. ISBN 9781450357555 p. 61–72. [Online]. Available: https://doi.org/10.1145/3198458.3198464

[13] M. Dietz and G. Pernul, “Unleashing the digital twin’s potential for ics security,” *IEEE Security Privacy*, vol. 18, no. 4, pp. 20–27, 2020. doi: 10.1109/MSEC.2019.2961650

[14] M. Kaouk, J.-M. Flaus, M.-L. Potet, and R. Groz, “A review of intrusion detection systems for industrial control systems,” in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2019. doi: 10.1109/CoDIT.2019.8820602 pp. 1699–1704.

[15] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, ser. CPSS '15. New York, NY, USA: Association for Computing Machinery, 2015. doi: 10.1145/2732198.2732200. ISBN 9781450334488 p. 13–24. [Online]. Available: https://doi.org/10.1145/2732198.2732200

[16] Y. Hu, A. Yang, H. Li, Y. Sun, and L. Sun, "A survey of intrusion detection on industrial control systems," *International Journal of Distributed Sensor Networks*, vol. 14, p. 155014771879461, 08 2018. doi: 10.1177/1550147718794615

[17] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6822–6834, 2019. doi: 10.1109/JIOT.2019.2912022

[18] M. R. Asghar, Q. Hu, and S. Zeadally, "Cybersecurity in industrial control systems: Issues, technologies, and challenges," *Computer Networks*, vol. 165, p. 106946, 10 2019. doi: 10.1016/j.comnet.2019.106946

[19] R. Bitton, T. Gluck, O. Stan, M. Inokuchi, Y. Ohta, Y. Yamada, T. Yagyu, Y. Elovici, and A. Shabtai, "Deriving a cost-effective digital twin of an ics to facilitate security evaluation," in *Computer Security*, J. Lopez, J. Zhou, and M. Soriano, Eds. Cham: Springer International Publishing, 2018. ISBN 978-3-319-99073-6 pp. 533–554.

[20] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A survey on scada systems: Secure protocols, incidents, threats and tactics," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1942–1976, 2020. doi: 10.1109/COMST.2020.2987688

[21] Ethernet/ip. [Online]. Available: https://www.odva.org/technology-standards/key-technologies/ethernet-ip/

[22] D. Antonioli and N. O. Tippenhauer, "Minicps: A toolkit for security research on cps networks," in *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, ser. CPS-SPC '15. New York, NY, USA: Association for Computing Machinery, 2015. doi: 10.1145/2808705.2808715. ISBN 9781450338271 p. 91–100. [Online]. Available: https://doi.org/10.1145/2808705.2808715

[23] Common industrial protocol(cip). [Online]. Available: https://www.odva.org/technology-standards/key-technologies/common-industrial-protocol-cip/

[24] M. R. I. F. D. A. R. M. T. Sinil Mubarak, Mohamed Hadi Habaebi, "Anomaly detection in ics datasets with machine learning algorithms," *Computer Systems Science and Engineering*, vol. 37, no. 1, pp. 33–46, 2021. doi: 10.32604/csse.2021.014384. [Online]. Available: http://www.techscience.com/csse/v37n1/41436

[25] Ethernet/ip security. [Online]. Available: https://www.odva.org/wp-content/uploads/2020/05/PUB00269R1.1_ODVA-Securing-EtherNetIP-Networks.pdf

[26] M. Atalay and P. Angin, "A digital twins approach to smart grid security testing and standardization," in *2020 IEEE International Workshop on Metrology for Industry 4.0 IoT*, 2020. doi: 10.1109/MetroInd4.0IoT48571.2020.9138264 pp. 435–440.

[27] D. Shangguan, L. Chen, and J. Ding, "A hierarchical digital twin model framework for dynamic cyber-physical system design," in *Proceedings of the 5th International Conference on Mechatronics and Robotics Engineering*, ser. ICMRE'19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3314493.3314504. ISBN 9781450360951 p. 123–129. [Online]. Available: https://doi.org/10.1145/3314493.3314504

[28] A. Ayodeji, Y. kuo Liu, N. Chao, and L. qun Yang, "A new perspective towards the development of robust data-driven intrusion detection for industrial control systems," *Nuclear Engineering and Technology*, vol. 52, no. 12, pp. 2687–2698, 2020. doi: https://doi.org/10.1016/j.net.2020.05.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1738573320300590

[29] A. Keliris, H. Salehghaffari, B. Cairl, P. Krishnamurthy, M. Maniatakos, and F. Khorrami, "Machine learning-based defense against process-aware attacks on industrial control systems," in *2016 IEEE International Test Conference (ITC)*, 2016. doi: 10.1109/TEST.2016.7805855 pp. 1–10.

[30] A. Chavez, C. Lai, N. Jacobs, S. Hossain-McKenzie, C. B. Jones, J. Johnson, and A. Summers, "Hybrid intrusion detection system design

for distributed energy resource systems," in *2019 IEEE CyberPELS (CyberPELS)*, 2019. doi: 10.1109/CyberPELS.2019.8925064 pp. 1–6.

[31] M. Eckhart and A. Ekelhart, "A specification-based state replication approach for digital twins," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, ser. CPS-SPC '18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3264888.3264892. ISBN 9781450359924 p. 36–47. [Online]. Available: https://doi.org/10.1145/3264888.3264892

[32] V. Kamath, J. Morgan, and M. I. Ali, "Industrial iot and digital twins for a smart factory : An open source toolkit for application design and benchmarking," in *2020 Global Internet of Things Summit (GIoTS)*, 2020. doi: 10.1109/GIOTS49054.2020.9119497 pp. 1–6.

[33] Mininet. [Online]. Available: http://mininet.org/

[34] Minicps. [Online]. Available: https://github.com/scy-phy/minicps

[35] F. Akbarian, E. Fitzgerald, and M. Kihl, "Synchronization in digital twins for industrial control systems," 2020, 16th Swedish National Computer Networking Workshop (SNCNW 2020) ; Conference date: 26-05-2020 Through 27-05-2020. [Online]. Available: https://arxiv.org/pdf/2006.03447.pdf

[36] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*, 01 2006.

[37] Ball and beam process. [Online]. Available: https://ctms.engin.umich.edu/CTMS/index.php?example=BallBeam&section=SystemModeling

[38] T. Morris and W. Gao, "Industrial control system cyber attacks," 09 2013. doi: 10.14236/ewic/ICSCSR2013.3

[39] C. M. Ahmed, J. Prakash, R. Qadeer, A. Agrawal, and J. Zhou, "Process skew: Fingerprinting the process for anomaly detection in industrial control systems," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '20. New York, NY, USA: Association for Computing Machinery, 2020. doi: 10.1145/3395351.3399364. ISBN 9781450380065 p. 219–230. [Online]. Available: https://doi.org/10.1145/3395351.3399364

[40] S. Griffith and T. H. Morris, "Using modeled cyber-physical systems for independent review of intrusion detection systems," in *National Cyber Summit (NCS) Research Track*, K.-K. R. Choo, T. H. Morris, and G. L. Peterson, Eds. Cham: Springer International Publishing, 2020. ISBN 978-3-030-31239-8 pp. 116–125.

[41] G. Bernieri, M. Conti, and F. Turrin, "Evaluation of machine learning algorithms for anomaly detection in industrial networks," in *2019 IEEE International Symposium on Measurements Networking (M N)*, 2019. doi: 10.1109/IWMN.2019.8805036 pp. 1–6.

[42] L. Perales Gómez, L. Fernández Maimó, A. Huertas Celdrán, F. J. García Clemente, C. Cadenas Sarmiento, C. J. Del Canto Masa, and R. Méndez Nistal, "On the generation of anomaly detection datasets in industrial control systems," *IEEE Access*, vol. 7, pp. 177 460–177 473, 2019. doi: 10.1109/ACCESS.2019.2958284

[43] "Cyber security based machine learning algorithms applied to industry 4.0 application case: Development of network intrusion detection system using hybrid method," 2020.

[44] Mininet-wifi. [Online]. Available: https://github.com/intrig-unicamp/mininet-wifi

[45] Understanding stacked ensemble model. [Online]. Available: https://towardsdatascience.com/ensemble-learning-stacking-blending-voting-b37737c4f483

[46] F. Zhang, H. A. D. E. Kodituwakku, J. W. Hines, and J. Coble, "Multilayer data-driven cyber-attack detection system for industrial control systems based on network, system, and process data," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4362–4369, 2019. doi: 10.1109/TII.2019.2891261

Appendix A

Source Code

All code implementations used in this thesis are available on the GitHub repository: `https://github.com/sebavarghese/MasterThesis`

For DIVA

```
{
"Author1": {
              "Last name": "Varghese",
              "First name": "Seba Anna",
              "Local User Id": "u1eqmlcn",
              "E-mail": "vargh@kth.se",
              "organisation": {"L1": "School of Electrical Engineering and Computer Science ",
                                   }
             },
"Degree": {"Educational program": "Master’s Programme, Communication Systems, 120 credits"},
"Title": {
              "Main title": "Digital Twin-based Intrusion Detection for Industrial Control Systems",
              "Language": "eng" },
"Alternative title": {
              "Main title": "",
              "Language": "swe"
},
"Supervisor1": {
                     "Last name": "Alimadadi",
                     "First name": "Zahra",
                     "Local User Id": "u10hmcar",
                     "E-mail": "alimadad@kth.se",
                     "organisation": {"L1": "School of Electrical Engineering and Computer Science ",
                                          "L2": "Computer Science" }
                 },
"Supervisor2": {
                     "Last name": "Balador",
                     "First name": "Ali",
                     "E-mail": "ali.balador@ri.se",
                     "Other organisation": "RISE Research Institutes of Sweden, Department: Digitala plattformar"}
                 },
"Examiner1": {
                     "Last name": "Papadimitratos",
                     "First name": "Panagiotis",
                     "Local User Id": "u1ysvc1i",
                     "E-mail": "papadim@kth.se",
                     "organisation": {"L1": "School of Electrical Engineering and Computer Science ",
                                          "L2": "Computer Science" }
                 },
"Other information": {
"Year": "2021", "Number of pages": "xvi,81"}
}
```

TRITA-EECS-EX-2021:903

www.kth.se