UNIVERSIDADE D
COIMBRA

Ivo Micael Frazão Costeira

APPLYING MACHINE LEARNING TECHNIQUES ON THE DETECTION OF CYBER-PHYSIC ATTACKS

Dissertação no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores, na especialização de Computadores, orientada pelo Professor Doutor Tiago José dos Santos Martins da Cruz e pelo Professor Doutor Hélder de Jesus Araújo e apresentada ao Departamento de Engenharia Eletrotécnica e de Computadores.

Setembro de 2018

Faculdade de Ciências e Tecnologia da Universidade de Coimbra

APPLYING MACHINE LEARNING TECHNIQUES ON THE DETECTION OF CYBER-PHYSIC ATTACKS

Ivo Micael Frazão Costeira

Dissertação no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores, na especialização de Computadores, orientada pelo Professor Doutor Tiago José dos Santos Martins da Cruz e pelo Professor Doutor Hélder de Jesus Araújo e apresentada ao Departamento de Engenharia Eletrotécnica e de Computadores.

Setembro de 2018

UNIVERSIDADE D COIMBRA

Agradecimentos

Gostaria de começar por agradecer aos meus orientadores do Departamento de Engenharia Informática, Professor Tiago Santos e Professor Pedro Abreu, pela ajuda e aconselhamento ao longo de todo este trabalho e pelo desafio que me propuseram. Um agradecimento também ao Professor Hélder Araújo, orientador do Departamento de Engenharia Eletrotécnica e de Computadores, pelo apoio prestado.

Um grande agradecimento à minha mãe, à minha avó e ao meu tio pelo apoio incondicional que me deram desde sempre, nos bons e nos maus momentos. Posso nem sempre demonstrá-lo, mas não deixam de significar imenso para mim. Um obrigado é pouco por tudo!

A todos os meus amigos e colegas de curso, obrigado por todos os momentos que vivenciámos em conjunto e que serão sempre memórias deliciosas para recordar no futuro. Contudo, terei que destacar oito pessoas: João Fortunato, João Miguel Caniceiro, Ricardo Pereira, Miguel Antunes, João Bento, César Pereira, Rui Gouveia e Vânia Silva. De uma forma ou outra marcaram-me de uma forma especial e era impossível não vos mencionar. Não me queria repetir, mas também a vocês um obrigado é pouco por tudo!

Deixo ainda um agradecimento a todas as pessoas que tive a oportunidade de trabalhar ao longo dos quatro anos que estive no Núcleo de Estudantes de Engenharia Eletrotécnica e de Computadores, destacando a direção 2017/2018, da qual apenas não mencionei ainda, neste texto, o João Martins e o José Pedro Silva.

Um agradecimento ao Laboratório C6.5, pela forma como me receberam e pelo ambiente de trabalho descontraído que proporcionaram.

Por fim, um agradecimento a todos os que de uma forma ou outra contribuíram para quem sou hoje e que não foram mencionados até aqui.

A todos,

Muito Obrigado.

Abstract

Today's society has had an increasing reliance on Industrial (Automation and) Control Systems, especially in the control of Critical Infrastructures, such as generation, transmission and distribution of electrical energy, or water treatment, and where the uninterrupted work of the infrastructure is essential for the safety and livelihood of a modern society. A disruption of the normal operation of such infrastructures, due to a fault or an attack, has the potential to create serious consequences, whereby these systems have strategic significance, which cannot be ignored. These infrastructures were designed with the assumption that an airgap from the outside world was in place, however, their increasing reliance on Commercial, Off-The-Shelf equipment, open technologies and open protocols has exposed them to threats that previously were not a concern.

Several studies have proposed many strategies to secure these infrastructures, which mainly rely on novel algorithms or domain-adaptations of known algorithms. However, the constraints of these infrastructures prevent the existence of common datasets from which a comparison of their performances can be made. This work aims at presenting some insight on how many of the most used algorithms in the literature behave when some of the characteristics of datasets are varied, namely the size of the traffic captures and the relative size of the attacks within those captures. An insight on the effects of hyperparameter tuning of the algorithms is also studied. Finally, a generalized framework for the detection of Man-in-the-Middle attacks is presented based on the Time-To-Response of the TCP packets within the network.

KEYWORDS: Industrial (Automation and) Control System; Cyber-Physical System; Supervisory Control And Data Acquisition System

Resumo

A sociedade atual tem tido uma confiança crescente em Sistemas de Controlo Industrial (e de Automação), especialmente no controlo de Infraestruturas Críticas, tais como a geração, transmissão e distribuição de energia elétrica, ou tratamento de águas, onde o trabalho ininterrupto da infraestrutura é essencial para a segurança e manutenção de uma sociedade moderna. A disrupção das operações normais de tais infraestruturas, devido a uma falha ou ataque, tem o potencial de ter sérias consequências, pelo que estes sistemas têm um significado estratégico que não pode ser ignorado. Estas infraestruturas foram pensadas assumindo que existia uma separação do mundo exterior, contudo, o crescente uso de equipamentos comerciais, e de tecnologias e protocolos abertos tem-nas exposto a ameaças que antigamente não eram uma preocupação.

Vários estudos têm proposto várias estratégias para proteger estas infraestruturas, focando-se principalmente em algoritmos novos ou adaptação de algoritmos já conhecidos ao domínio destas infraestruturas. Contudo, as restrições das mesmas têm impedido a existência de conjuntos de dados comuns, dos quais seria possível realizar uma comparação do seu desempenho. Este trabalho procura apresentar alguma introspeção sobre como os algoritmos mais usados na literatura se comportam quando algumas das características dos conjuntos de dados variam, nomeadamente o tamanho da captura de tráfego e o tamanho relativo dos ataques nesses dados. São também estudados os efeitos da afinação de hiperparâmetros desses algoritmos. Finalmente, é também apresentada uma estrutura genérica para a deteção de ataques Man-in-the-Middle, baseada nos tempos de resposta dos pacotes TCP dentro da rede.

PALAVRAS-CHAVE: Sistema de Controlo Industrial (e de Automação); Sistema Ciber-Físico; Sistema de Supervisão e Aquisição de Dados

Contents

List of Figures

List of Tables

Acronyms

Acc.	Accuracy.
ADU	Application Data Unit.
API	Application Programming Interface.
ARP	Address Resolution Protocol.
CI	Critical Infrastructure.
COTS	Commercial, Off-The-Shelf.
CPS	Cyber-Physical System.
DCS	Distributed Control System.
DDoS	Distributed Denial-of-Service.
DoS	Denial-of-Service.
DR	Detection Rate.
DT	Decision Tree.
FN	False Negative.
FP	False Positive.
FPR	False Positive Rate.
HMI	Human-Machine Interface.
I(A)CS	Industrial (Automation and) Control System.
ICMP	Internet Control Message Protocol.
IDPS	Intrusion Detection and Prevention System.
IDS	Intrusion Detection System.
IED	Intelligent Electronic Device.
IP	Internet Protocol.

kNN k-Nearest Neighbors.

MAC Media Access Control.

MBAP MODBUS Application Protocol.

MitM Man-in-the-Middle.

NB Naïve Bayes.

NN Neural Network.

OCSVM One-Class Support Vector Machine.

PCA Principal Component Analysis.

PDU Protocol Data Unit.

PLC Programmable Logic Controller.

RF Random Forest.

RTU Remote Terminal Unit.

SCADA Supervisory Control And Data Acquisition.

SVM Support Vector Machine.

TCP Transmission Control Protocol.

TCP/IP Transmission Control Protocol/Internet Protocol.

TN True Negative.

TP True Positive.

TRP True Positive Rate.

TTR Time-To-Response.

UDP User Datagram Protocol.

Chapter 1. Introduction

Industrial (Automation and) Control Systems (I(A)CSs), often implemented by Supervisory Control And Data Acquisition (SCADA) systems, play an important role in today's society, particularly in the control of Critical Infrastructures (CIs) – such as generation, transmission, and distribution of electrical energy, or water treatment –, where the uninterrupted work of the infrastructure is essential for the safety and livelihood of a modern society. A disruption of the normal operation of such infrastructures, due to a fault or an attack, has the potential to create serious consequences, whereby these systems have strategic significance, which cannot be ignored.

These systems were originally designed for local monitoring and control activities through serial communications and with the assumption that all the entities were legitimate and operating under normal circumstances, given their isolation from the outside world, known as airgapping. As a consequence, they were not designed with any protection mechanisms, such as identity and permission validations, message encryption, or integrity checks, used for protection against deliberate attacks. However, in recent years, mainly due to cost effectiveness, these systems have been increasingly dependent on Commercial, Off-The-Shelf (COTS) equipment and open protocols and technologies, such as the Internet, to distribute the activities over large geographical areas, exposing the control systems to various threats that were not a concern with the older architectures, endangering their security [1]. As such, Intrusion Detection Systems (IDSs) are of paramount importance for these systems. Unfortunately, ”general purpose” Intrusion Detection Systems are often unsuitable for these scenarios, especially for sophisticated attacks which resort to legitimate commands – these attacks require a deeper knowledge of the protocols involved, traffic patterns, and the operational context of each system [2].

1.1 Contextualization

One of the fundamental issues in securing these networks resides in the heterogeneity of the individual components of the systems which also have long life cycles and the requirement for rigorous testing and certification of every update, which dramatically slows down the capability to protect against new types of attacks. Furthermore, given the high costs of the equipment involved alongside a mindset of avoiding to disturb systems which are working, many infrastructures still work with components which have already surpassed their support or even their end-of-life expectations.

Research on the topic of protecting these infrastructures has been relatively abundant. Many works have demonstrated the feasibility of domain-adaptations of Intrusion Detection Systems (IDSs) built for common communication infrastructures to the CPSs' domain, since some of the attacks which they now have to face originated from them. Novel techniques have also been proposed, while older ones, which were not suitable for common communications infrastructures, have also been revised with some success, given the many constraints CPSs have compared to them.

Attacks on the availability, confidentiality and integrity of these infrastructures have become concerns, with attacks such as to Stuxnet [3] or to the Ukrainian power grid [4] having exposed many of their frailties and consequences. However, research on this topic has mainly focused on novelty solutions or domain adaptations of known algorithms using proprietary or limited datasets, which has prevented a proper generalization of their results and real-world applications.

1.2 Goals

This dissertation aims to provide an insight at the frailties of machine learning algorithms in the classification process, specifically the effects that the datasets have on their performance. While performance comparisons are already available in published research, they don't provide insights about how specific dataset characteristics relate to the performance obtained.

1.3 Research Contributions

The work developed during this dissertation resulted in the following contribution:

- Ivo Frazão, Pedro Henriques Abreu, Tiago Cruz, Hélder Araújo, Paulo Simões. "Denial of Service Attacks: Detecting the frailties of machine learning algorithms in the Classification Process". Conference: 13th edition of the International Conference on Critical Information Infrastructures Security (CRITIS) (accepted, 2018).

Furthermore, a survey regarding the state-of-the-art is also being made for a journal publication.

1.4 Document Structure

The remainder of this dissertation is organized as follows: in Chapter 2 is presented the background knowledge which will be the basis of this work. Chapter 3 presents an overview of the related work done in this field. Then, Chapter 4 describes the experimental setup and Chapter 5 discusses the results obtained. Finally, Chapter 6 concludes this dissertation and presents some proposals for future work.

Chapter 2. Background Knowledge

In this chapter, an overview of the background knowledge required for the work done in this dissertation is provided.

2.1 Theoretical Definitions

In this section, a brief definition of the concepts discussed across the work done in this dissertation is presented.

2.1.1 Cyber-Physical Systems

Cyber-Physical Systems (CPSs) are defined as the intertwining of computation and physical processes and designed as a network of interacting elements with physical input and output where intelligent mechanisms coordinate them in order to control the process [5]. Smart grids, Industrial (Automation and) Control System and autonomous driving systems are examples of CPSs.

Some of these CPSs are critical to the functioning of today's modern societies which are known as Critical Infrastructures (CIs), of which electrical grids and water treatment plants are examples [6].

2.1.2 Industrial (Automation and) Control Systems

Industrial (Automation and) Control Systems (I(A)CSs) are control systems found in the industrial sector and in Critical Infrastructures (CIs), being the most common types Supervisory Control And Data Acquisition (SCADA) systems, frequently used to control geographically dispersed systems, such as an electrical grid, and Distributed Control Systems (DCSs), frequently used to control geographically confined systems, such as factories [6]. SCADA is data acquisition and event oriented, i.e., it retrieves data from the system periodically and acts when events occur, such as when a threshold is surpassed, while DCS is process and process state oriented.

I(A)CS systems are designed to accomplish a set of characteristics required to properly run their functions, namely [4]:

- **Availabilty**: ensures that the systems and information contained within them are available to authorised users. This is especially important for industrial systems and critical infrastructures where access to the data is paramount to maintain proper operations;
- **Fault-tolerance**: ensures that the systems are robust and can continue operating at a reasonable level in the event of a failure.
- **Performance**: ensures that the system is efficient and can carry out its intended tasks timely and correctly;
- **Safety**: systems must be able to detect unsafe conditions and trigger actions to reduce unsafe conditions to safe ones. In most safety-critical operations, human oversight and control of a potentially dangerous process is an essential part of the safety system;
- **Maintainability**: it is highly recommended for the system to have adequate diagnosis and control functionalities to allow correct maintainability;
- **Openness**: makes use of open standards and technologies in order to increase interoperability between devices and assets from different systems and infrastructures;
- **Security**: guarantees that the systems are protected, at least, against the most common threats that they face (such as unauthorised access or data manipulation), taking into consideration not only Availability, Integrity and Confidentiality security tenets, but also the Safety needs as they are cyber-physical devices;
- **Usability**: the ease-of-use and proper functionality of the systems and related tools and devices.

2.1.3 TCP/IP protocol stack

TCP/IP is the protocol stack for Internet-based communications, which provides end-to-end data exchange in computer networks. It specifies how the data should be packeted, addressed, transmitted, routed, and received inside the network and is functionally organised in four abstraction layers, which are, from top to bottom in the stack [7]:

- **Application layer**, which includes the protocols used by processes to perform communication, where the MODBUS over TCP/IP fits (see Section 2.1.4). These protocols rely in the transport layer to provide them the pipes where the communication occurs.
- **Transport layer**, which includes the protocols that perform end-to-end communication services, where the Transmission Control Protocol (TCP) is one of the main protocols, alongside the User Datagram Protocol (UDP).

- **Internet layer**, which includes the protocols that exchange data between network boundaries, namely the Internet Protocol (IP).
- **Link layer**, which includes the protocols that define the local networking methods and interfaces needed to transmit the information to directly connected hosts. E.g., the Ethernet protocol and the Address Resolution Protocol (ARP).

2.1.3.1 Address Resolution Protocol

The Address Resolution Protocol (ARP) is a request-response protocol that allows the discovery of link layer addresses (such as MAC addresses) associated with a given network layer address (such as an IP address) [8]. When a unknown address is required for transmission, the sender broadcasts a request to the local network it is connected to and when a host matches that address it responds to the sender which in turn stores it in a cache and is now able to communicate to it. Even when a request is not made but a response is received (known as unsolicited *is-at* messages), the address is stored in that cache or supersedes an older entry. However, due to the lack of authentication of the sender of a response, cache poisoning is a real security risk, which allows for Man-in-the-Middle (MitM) attacks [9].

Man-in-the-Middle As mentioned, the ARP protocol is susceptible to cache poisoning, allowing for Man-in-the-Middle (MitM) attacks, which have enormous potential, either as a simple source of private information or of erroneous information. In a MitM attack, the attacker gains access to the communication between two parties who believe are directly communicating with each other, as pictured in Figure 2.1. In order for a successful MitM attack to happen, the attacker should be in the same subnet of the target and it should be able to poison the Address Resolution Protocol (ARP) caches of the victims, such that it may act as the router that forwards the traffic [10].

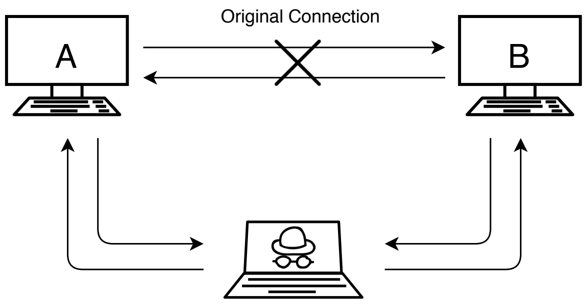


Figure 2.1: Representation of a Man-in-the-Middle (MitM) attack.

In [10] (2015), Chen et al. implemented a successful MitM attack as characterised here, using the open source tool Ettercap [11], enabling them to see the packets being exchanged within the system, relying in Wireshark [12] as a packet analyser and sniffer. After the MitM was successful, the authors were able to inject false messages to the system, generated using the *libmodbus* library [13], by emulating the MODBUS client, and cause a tripping of one of the breakers of the emulated power distribution system. The tripping of a single breaker was not catastrophic but resulted in diminished overall stability of the system and in a transient that affected the functionality of other protective relays in the system.

2.1.3.2 Internet Control Message Protocol

Internet Control Message Protocol (ICMP) is part of the TCP/IP protocol stack mainly used for diagnostic and control purposes – such as, verifying if a host is active (known an ping) or to trace the route of packets within the network – or as response messages to errors that may happen in a network – such as, being unable to reach a specific destination [14].

This protocol has some security concerns which may make it prejudicial to network safety, e.g., the mere acknowledgement that a host is active can make it a target for attacks, or ping flooding attack, which induces a Denial-of-Service (DoS) by the host as it is overwhelmed by the requests.

2.1.4 MODBUS protocol

The MODBUS protocol is an application layer messaging protocol that has been the *de facto* standard in SCADA systems due to its simplicity and robustness. The protocol uses a simple communication mode, based in a master/slave relationship between units, where the master device initiates transactions with the slave devices, which are identified with a transaction ID field [15].

It uses two basic types of communication: unicast and broadcast. In a unicast transaction the master unit starts a request that the slave unit answers (e.g., by returning the requested value, by acknowledging after performing the desired action, or by returning an error). In a broadcast transaction the master unit sends a message to all the slaves which do not reply such message (e.g., resetting the sensors and actuators) [16].

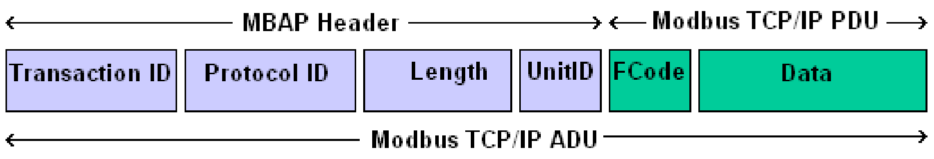
All MODBUS requests and responses are designed in such a way that the recipient can verify that a message is finished. For function codes where the MODBUS Protocol Data Unit (PDU) has a fixed length, the function code alone is sufficient. For function codes carrying a variable amount of data in the request or response, the data field includes a byte count [17].

Being an application layer protocol, MODBUS can be transmitted using multiple types of transmission protocols, being the most common the asynchronous serial transmission or the TCP/IP protocol stack.

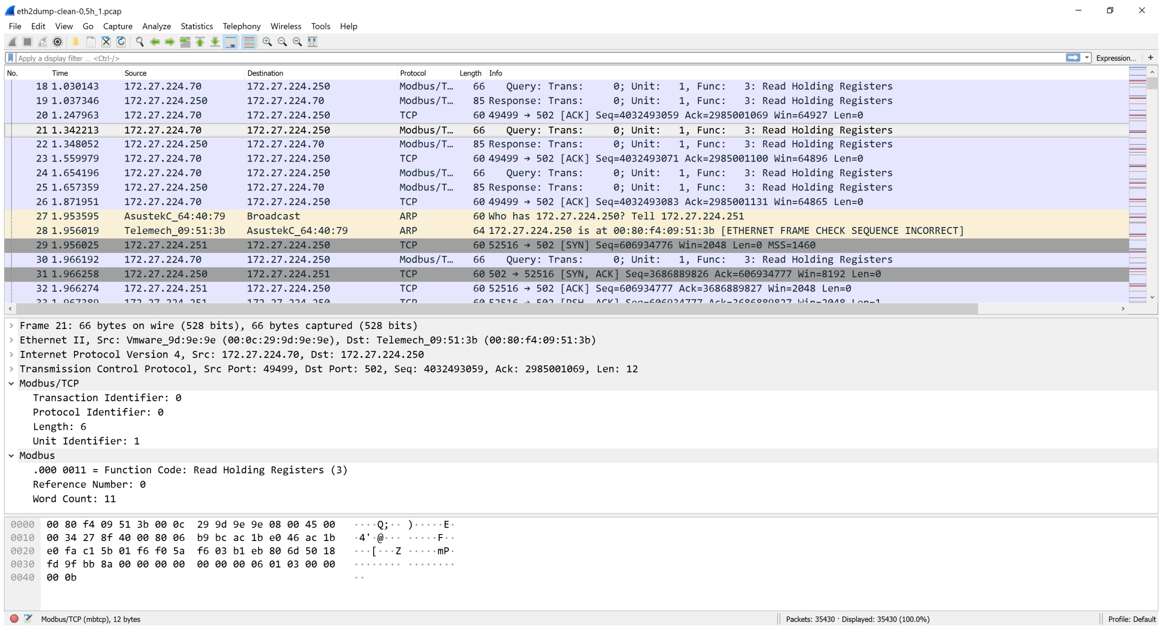
2.1.4.1 MODBUS over TCP/IP

The MODBUS over TCP/IP messaging service provides client/server communication between devices connected in a network, where a client makes a request to a server. The specification defines that the registered TCP/IP listening port number 502 is reserved for MODBUS communications [17].

The protocol specifies an Application Data Unit (ADU) – as seen in Figure 2.2 – which is encapsulated as the payload of the TCP/IP protocol stack. As can be seen, a dedicated header is used, called the MODBUS Application Protocol (MBAP) header, which precedes its Protocol Data Unit (PDU).



(a) General structure.


eth2dump-clean-0,5h_1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Time       Source              Destination         Protocol    Length Info
  18 1.030143   172.27.224.70       172.27.224.250      Modbus/T…   66    Query: Trans:     0; Unit:   1, Func:   3: Read Holding Registers
  19 1.037346   172.27.224.250      172.27.224.70       Modbus/T…   85 Response: Trans:     0; Unit:   1, Func:   3: Read Holding Registers
  20 1.247963   172.27.224.70       172.27.224.250      TCP         60 49499 → 502 [ACK] Seq=4032493059 Ack=2985001069 Win=64927 Len=0
  21 1.342213   172.27.224.70       172.27.224.250      Modbus/T…   66    Query: Trans:     0; Unit:   1, Func:   3: Read Holding Registers
  22 1.348052   172.27.224.250      172.27.224.70       Modbus/T…   85 Response: Trans:     0; Unit:   1, Func:   3: Read Holding Registers
  23 1.559979   172.27.224.70       172.27.224.250      TCP         60 49499 → 502 [ACK] Seq=4032493071 Ack=2985001100 Win=64896 Len=0
  24 1.654196   172.27.224.70       172.27.224.250      Modbus/T…   66    Query: Trans:     0; Unit:   1, Func:   3: Read Holding Registers
  25 1.657359   172.27.224.250      172.27.224.70       Modbus/T…   85 Response: Trans:     0; Unit:   1, Func:   3: Read Holding Registers
  26 1.871951   172.27.224.70       172.27.224.250      TCP         60 49499 → 502 [ACK] Seq=4032493083 Ack=2985001131 Win=64865 Len=0
  27 1.953595   AsustekC_64:40:79   Broadcast           ARP         60 Who has 172.27.224.250? Tell 172.27.224.251
  28 1.956019   Telemech_09:51:3b   AsustekC_64:40:79   ARP         64 172.27.224.250 is at 00:80:f4:09:51:3b [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
  29 1.956025   172.27.224.251      172.27.224.250      TCP         60 52516 → 502 [SYN] Seq=606934776 Win=2048 Len=0 MSS=1460
  30 1.966192   172.27.224.70       172.27.224.250      Modbus/T…   66    Query: Trans:     0; Unit:   1, Func:   3: Read Holding Registers
  31 1.966258   172.27.224.250      172.27.224.251      TCP         60 502 → 52516 [SYN, ACK] Seq=3686889826 Ack=606934777 Win=8192 Len=0
  32 1.966274   172.27.224.251      172.27.224.250      TCP         60 52516 → 502 [ACK] Seq=606934777 Ack=3686889827 Win=2048 Len=0

> Frame 21: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: Vmware_9d:9e:9e (00:0c:29:9d:9e:9e), Dst: Telemech_09:51:3b (00:80:f4:09:51:3b)
> Internet Protocol Version 4, Src: 172.27.224.70, Dst: 172.27.224.250
> Transmission Control Protocol, Src Port: 49499, Dst Port: 502, Seq: 4032493059, Ack: 2985001069, Len: 12
v Modbus/TCP
    Transaction Identifier: 0
    Protocol Identifier: 0
    Length: 6
    Unit Identifier: 1
v Modbus
    .000 0011 = Function Code: Read Holding Registers (3)
    Reference Number: 0
    Word Count: 11

0000  00 80 f4 09 51 3b 00 0c  29 9d 9e 9e 08 00 45 00   ····Q;·· )·····E·
0010  00 34 27 8f 40 00 80 06  b9 bc ac 1b e0 46 ac 1b   ·4'·@··· ·····F··
0020  e0 fa c1 5b 01 f6 f0 5a  f6 03 b1 eb 80 6d 50 18   ···[···Z ·····mP·
0030  fd 9f bb 8a 00 00 00 00  00 00 00 06 01 03 00 00   ········ ········
0040  00 0b                                              ··

Modbus/TCP (mbtcp), 12 bytes        Packets: 35430 · Displayed: 35430 (100.0%)        Profile: Default
```

(b) Capture of a frame using Wireshark.

Figure 2.2: MODBUS over TCP/IP frames.

The fields of the frame are the following:

- ’Transaction ID’, which identifies the current transaction between the devices and is initialized by the client;

- ’Protocol ID’, which is always zero for the MODBUS protocol;
- ’Length’, which indicates the number of bytes following the field;
- ’Unit ID’, which is used to communicate via devices such as bridges, routers and gateways that use a single IP address to support multiple independent MODBUS end units;
- ’Function Code’, which indicates the desired action requested of the server by the client;
- ’Data’, which contains the actual information being transmitted.

2.1.4.2 Security in MODBUS

The MODBUS protocol was not designed with cybersecurity in mind, since it was meant to be used within an airgapped network, i.e., isolated from the outside world [18]. Consequentially, it lacks some basic security features, such as data encryption, integrity checks, and mutual authentication. Furthermore, this protocol was designed for serial connections, namely RS-485 serial buses, including their data models and APIs, but it has since been ported for use in the TCP/IP protocol stack, without truly adapting to the new paradigm. Although new technologies and protocols have emerged, such as DNP3, that solve many of the aforementioned problems, MODBUS is still a widely used protocol and, as such, was the main focus of this work.

2.1.5 Supervisory Control And Data Acquisition

Supervisory Control And Data Acquisition (SCADA) is a control system architecture with a significant role in CPSs, consisting of hardware, software and communication components used to monitor and control physical processes [19]. A generic configuration of a SCADA system is visible in Figure 2.3.

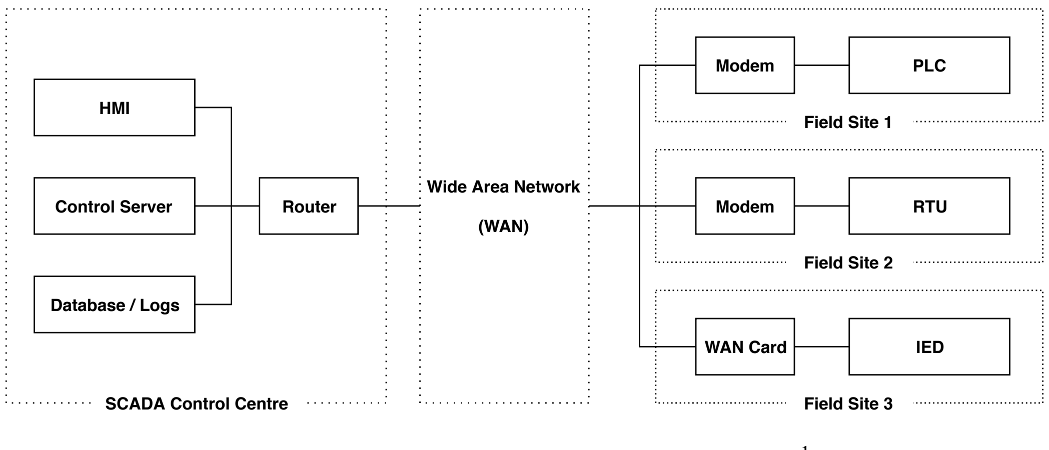


Figure 2.3: Generic architecture of a SCADA system[1] [19].

[1]It does not include any redundancies normally employed in these systems for clarity.

The generic SCADA system represented is comprised of a SCADA Control Centre, which performs centralized monitoring and control of the entire system, based on the information received, especially from the field sites. Those sites contain local devices that control local operations, collect data from sensors, and monitor local environments [19]. The control centre contains:

- **Control Server**, which hosts the supervisory control software responsible for communication with the lower level control devices;
- **Human-Machine Interface (HMI)**, which allows human operators to monitor the state of the system, change its settings, and manually override automatic control procedures;
- **Database/Logs**, which stores information used to support different types of analysis.

The field sites, besides the communication component, can contain a wide array of devices, depending on the activity needed in each site [19]:

- **Remote Terminal Units (RTUs)**, which are special purpose data acquisition and control units;
- **Programmable Logic Controllers (PLCs)**, which are small industrial computers, designed to perform logic functions executed by electrical hardware, such as relays, switches, and mechanical timers/counters, that have evolved to control complex processes and even substitute RTUs since they are more economical, versatile, flexible and configurable devices;
- **Intelligent Electronic Devices (IEDs)**, which are sensors/actuators capable of acquiring and communicating data, as well as perform local processing and control.

The communication architecture of SCADA systems can be grouped into three main generations: monolithic, distributed, and networked [20]. In the first generation, these systems were physically and logically independent of other systems (i.e., isolated from the outside world) therefore security was not a concern. However, as these systems evolved, they became less isolated from other systems and dependent on connectivity provided by modern information technologies, which has made these systems vulnerable to several threats [21, 19, 6].

SCADA systems are an integral part of the Purdue Reference Model [22], a reference architecture which models enterprises in multiple layers and stages for computer integrated manufacturing. This model defined the concepts and tasks involved in enterprise integration, using a phased approach, while also providing informational models to improve understanding and monitorization. It defined 5 levels of integration:

- **Level 0**, corresponding to the physical process;

- **Level 1**, corresponding to intelligent devices, such as sensors and actuators;
- **Level 2**, corresponding to control systems, which supervise, monitor and control the physical processes, where SCADA systems can play an integral role;
- **Level 3**, corresponding to manufacturing operations systems, where production work flow is defined;
- **Level 4**, corresponding to business logistic systems, where schedules, materials, shipments and inventory is controlled.

2.1.5.1 Vulnerabilities

Many SCADA systems were not designed with security in mind, which introduced many vulnerabilities to them which can be exploited. Some of the most common vulnerabilities of these systems are [4]:

- **Non-existent monitoring process**: Without active network monitoring, it is very difficult to detect suspicious activity, identify potential threats, and quickly react to cyber-attacks, and Intrusion Detection Systems (IDSs) are not as common as in common IT networks. Furthermore, even if they are in-place they may not be able to fully understand I(A)CS protocols. This can be partially addressed by implementing anomaly detection systems. Firewalls and antivirus are more common, but it is not a universal solution and does not cover all the risks;
- **Deficient traffic content understanding**: Managers need to know what type of traffic is going through their networks in order to be able to make informed decisions on how to respond to potential threats and on which kinds of traffic to allow and which to filter. This also helps to establish proper segregation and network segmentation;
- **Staff inexperienced in cyber-security related topics**: SCADA system staff and operators are familiar with keeping control systems running. The normal goals of reliability and availability can initially feel in conflict with security efforts. With a bent for engineering and technical solutions to problems, the important role of developing security policies can be a foreign concept to typical SCADA staff. Furthermore, SCADA staff may not be receptive to IT staff recommendations;
- **Operating System vulnerabilities**: The whole host of normal IT operating system vulnerabilities are present in SCADA systems. The difference from an IT system is that patching may be performed less rigorously. It is usual for a SCADA system operator to have a running system that is expected to perform without interruptions;

- **Slow / lack of updates**: Maintaining I(A)CS/SCADA firmware and software up-to-date is not easy, and it can be very complex for critical infrastructure systems, as an update error could cause severe issues on the whole system. Cyber fragility results from applying a change to the system without having tested it beforehand and having foreseen its effects;
- **Remote Processor operations**: Certain classes of remote processors have known security vulnerabilities. In this case the difficulty is two-fold: First the computation power and memory resources of the processors are modest and not suitable for security upgrades. Second, once they are installed they typically stay in place for ten years or more. The result is vulnerable equipment that stays vulnerable for a long time;
- **SCADA Software features**: SCADA applications and software usually provides basic and modest security features, however these are not always enabled by default, and could act as additional weaknesses if operators are unaware of the need of enabling these features;
- **Inappropriate applications installed on critical SCADA host computers**: Because very few security measures are used in SCADA host computers, this leads to an operator or administrator inadvertently installing an inappropriate application on a critical system network device;
- **Lack of knowledge regarding the devices**: Since most SCADA systems have been developed gradually over time, it's not uncommon to see technology that's a couple of years old working alongside an industrial network environment. Knowledge transfer regarding functioning and maintenance of ageing devices should be ensured;
- **Authentication weaknesses**: Authentication solutions are designed to keep unauthorized people away from accessing the SCADA systems. However, this can easily be defeated if the solution is not properly implemented (e.g. allowing weak passwords, hard-wired passwords, user credential sharing, no user logging, etc.), or in the case of older devices, which make use of weaker, more primitive authentication methods. In some cases moving to two-factor authentication is limited by work conditions that may impede iris scans or fingerprint scans because of dirty hands or the wearing of safety goggles. Confidentiality and authentication is often compromised by the use of clear text transmissions. This weakness eliminates authentication and accountability validity;
- **Unauthenticated PLC/RTU network connections**: Older SCADA systems lack basic security features, so it is imperative for organizations that own such systems to insist on vendors to provide security measures in the form of product patches and upgrades which can protect the system from unauthenticated PLC/RTU network connections;

- **Remote access supervision**: Because of economics for staffing control centres around the clock, it is not uncommon for SCADA systems to be configured with remote access. This can include dial-up access or VPN access over the Internet. These scenarios should be controlled and monitored, and should include, at least, the same security measures as internal connections;
- **Interconnection management**: The more connections, the more exposure a SCADA system has. Economic and enterprise pressures often result in the existence of internal connections between the SCADA network and the business network in order to allow remote access, control and/or maintenance;
- **Wireless connections**: SCADA systems often use microwave, data radios and cellular packet services for communications. Depending on the implementation, these forms of communication can be vulnerable to certain types of attacks;
- **Available public information**: In the past it was not unusual for SCADA system owners to publish information on the design of their systems, as security was not a top concern and most devices where not interconnected, required physical access. It is also fairly common for consultants/contractors to advertise past experience including information regarding the systems they have worked on. Both these scenarios can result in the exposure of system vulnerabilities which is more serious as these systems have now become interconnected;
- **The wrong belief that SCADA systems have the benefit of security through obscurity**: The use of closed-source proprietary protocols does not provide security, and it can be counterproductive. Security by obscurity is not a good practice, and usually gives users a false sense of security, while in fact they are at greater risk;
- **The wrong belief that SCADA systems are isolated**: Just because a SCADA network is not connected to the internet does not make it secure. Physical access is still possible, and regardless of the network size, all connection points should be always controlled and monitored. For maintenance and support reasons devices belonging to segmented networks are sometimes exposed to Internet through Virtual Private Networks (VPNs) or remote access connections. These connections should be controlled, and be enabled on-demand only when required;
- **Physical security**: SCADA systems are usually distributed over large distances with multiple unstaffed locations. The physical protection of SCADA devices becomes important in these cases. But, as pin tumbler locks, master keys and cylinder locks all have reported weaknesses it is important to be realistic about the level of protection they provide and take it into account when protecting physical locations.

2.1.5.2 Examples of Cyberattacks

Given the security vulnerabilities previously mentioned, its possible to describe real attacks against I(A)CS. As an example, Estonia was targeted by a series of cyberattacks that began 27 April 2007 and flooded websites of Estonian organizations (parliament, banks, ministries, and media) due to the country's disagreement with Russia about the relocation of a Soviet-era memorial [23]. In 2010, Stuxnet – a trojan designed to attack Siemens Step7 HMI software and S7 PLCs – temporarily set back Iran's nuclear program. It almost ruined one-fifth of the Iranian nuclear centrifuges by spinning them out of control while simultaneously replaying recorded system values to fake normal system behaviour during the attack [3]. More recently, in 2015, a cyberattack on the Ukrainian power grid left over 230 thousand people without electricity for more than six hours by taking more than 30 substations and 2 power supply centres offline. The attack had replaced the firmware of multiple SCADA components, which disabled remote control and required manual control, while also compromising uninterruptible power supplies of the control centres, effectively making them unaware of the problems [4].

Industry's reaction Considering the increasing threat of cyberattacks against these systems, the industry has reacted by publishing a series of recommendations and standards regarding I(A)CS cyber-security, namely:

- IEC 62443 series of standards for industrial automation and control systems security (superseeding ISA99);
- NIST SP 800-82;
- NERC CIP standards;
- IEEE std 1686-2013, IEEE Standard for Intelligent Electronic Devices Cyber Security Capabilities;
- ENISA guidelines for ICS security.

2.2 Attacks

There are several types of attacks that affect SCADA systems which can be grouped into three main categories [16]:

- **Attacks on availability** (also known as a Denial-of-Service (DoS) attacks) where the perpetrator seeks to make a resource unavailable to its intended users by indefinitely disrupting services of a host connected to its users. It may be accomplished by flooding the targeted machine or resource with superfluous requests preventing legitimate ones from being fulfilled or even by a malformed packet that causes a crash on a PLC. When the origin of those superfluous requests originates from many different sources, making it impossible to stop the attack by blocking a single source, it is called a Distributed Denial-of-Service (DDoS) attack;

- **Attacks on confidentiality**, where the attacker seeks to obtain information about the actors and the system it is attacking, such as its topology and protocols involved, or the information that system is communicating, such as the sensor information of a SCADA field device. It is usually a precursor to more serious attacks, since the attacker can gather enough information to avoid detection;

- **Attacks on integrity**, where the attacker may inject unreliable data in the traffic, by either providing erroneous measurements to the monitoring system, which might react, potentially harming the system, or by providing malicious commands to the control system [24]. These attacks usually require that the attacker has sufficient knowledge of the system in order to avoid suspicions.

SCADA systems have numerous types of threats/attacks, whereby a taxonomy is of great importance to better understand the risks inherent to these systems and how to prevent them. The following taxonomy focuses in the MODBUS over TCP/IP protocol and is mainly based in the works made by Huitsing et al. in [16] (2008), and by Drias, Serhrouchni and Vogel in [25] (2015).

2.2.1 Attacks on availability

MODBUS is particularly susceptible to this type of attack since the protocol does not include any mechanisms for authentication or integrity validation, whereby the servers cannot identify erroneous or malicious messages and comply with the requests made.

Some examples of attacks on availability generic to Cyber-Physical Systems:

- **Baseline Response Replay**, where the attacker replays genuine traffic between a master and a field device back to the master, interrupting the actual communication between units and effectively blinding the master to any changes in the system [25, 16];

- **Response Delay**, where the attacker delays the response messages from the field devices to the master device so that it receives out-of-date information, which might result in an overreaction by the automated mechanisms or by the operator [25, 16].

Some examples of attacks on availability specific to the MODBUS protocol are:

- **MODBUS Query Flooding**, where the attacker floods a device with MODBUS queries which saturates trying to comply with them and ignores the actual control messages from the master;
- **Remote Restart Command Flooding**, where the attacker floods a field device with the MODBUS Remote Restart Command[2] which causes the server device to restart and execute its power-up test, rendering the device inoperable [16];
- **Broadcast Message Flooding**, where the attacker sends multiple fake broadcast messages to slave devices, which can become saturated with those messages. It's particularly hard to detect if only the communications at the master device are monitored since no response messages are returned to it [25, 16].

Some examples of attacks on availability specific to the TCP/IP protocol stack are:

- **TCP SYN Flooding**, where the attacker sends multiple TCP SYN requests (connection establishment requests) to the target's system, which will open a new connection for each request until it cannot handle any more TCP connections and starts to drop new requests, even from legitimate sources [26, 16];
- **TCP FIN/RST Flooding**, where the attacker sends spoofed TCP packets with the FIN/RST flag set in order to close legitimate TCP connections [16];
- **Ping Flooding**, where the attacker sends multiple ICMP packets in order to overload the victim.

Some examples of attacks on availability specific to the MODBUS over TCP/IP are:

- **Irregular TCP Framing**, where the attacker injects improperly framed messages or modifies legitimate ones in the TCP frame which can cause the closure of that connection (e.g., a single TCP frame should only carry a single MODBUS message and a unit can be programmed to close connections that carry improperly framed messages) [16];
- **MODBUS TCP Pool Exhaustion**, where the attacker floods the two connection pools available in the devices, which starts to refuse new connections [16].

Several attacks on availability have been made against these systems, of which some examples are the attacks made in [10] (2015), Chen et al. implemented a TCP SYN Flood attack directed to a MODBUS Master device, using the Hping tool [27] to produce the TCP SYN requests at a rate of

[2]A well-defined command in the specification of the protocol, with function code 08 and sub-function code 01.

120 spoofed packets per second. To assess the impact of the attack, the authors measured the delay of the response message to a tripping command during and after the attack was happening. During the attack, the time difference was approximately 3 seconds, ten times higher than the delay without the attack. In [26] (2009), Queiroz et al. built a simulated water plant's SCADA system where they conducted a DDoS attack through a TCP SYN flood to the RTU of the system in order to affect its functionalities. With no detection and/or prevention system active, the authors were able to disrupt the system during the entire time of the attack and partially disrupt afterwards, since the legitimate connection were only able to be accepted after the connections already established began to close. In [28] (2016), Miciolino et al. performed a MODBUS flooding attack, using the nping tool [29], against a PLC which received 100 thousand fake MODBUS queries with a small delay between them and tried to reply to every request until saturation. Even after the attack had ended, the PLC replies presented slight delays compared to the base scenario. In [30] (2016), Cazorla, Alcaraz and Lopez performed a ping flooding attack on a testbed simulating a water system of a small city. The first objective was to measure the effect of the size of the packets used for the ping flooding on the network usage, finding that for 8 kB the network[3] had 100% usage, disrupting the normal operation of the system. Even when the usage was lower, it was measurable some delays in the exchange of information between the system.

2.2.2 Attacks on confidentiality

MODBUS is also particularly susceptible to this type of attack since it has no encryption built-in whereby the information is transmitted in plain text across the communication network and has no mutual authentication feature, allowing for spoofed requests of information.

Examples of attacks on confidentiality against MODBUS are:

- **MODBUS Network Scanning**, where the attacker sends benign messages to all possible addresses on a MODBUS network to obtain information about the devices [16];
- **Passive Reconnaissance**, where the attacker passively reads the traffic of the system in order to obtain knowledge of said system [25, 16];
- **MODBUS Slave Reconnaissance**, where the attacker sends the Return Status Information[4] to a slave device which complies, divulging its information [16].

Rosa et al. defined, in [9], how to perform a reconnaissance attack to a network using MODBUS over TCP/IP. First, a TCP scan for hosts should be made, using available strategies such as half-open

[3]The target machine was connected to a switch port limited to 10 Mbit/s
[4]A well-defined command in the specification of the protocol, with function code 17.

SYN or FIN flag scanning. At last, an enumeration of all MODBUS devices should be made using malformed requests to every ’unitID’ possible and wait for replies from the devices which matched those IDs. This second and final step was necessary because of possible field devices ”hidden” behind gateways in the system.

2.2.3 Attacks on integrity

Given the lack of security mechanisms, namely authentication, MODBUS is unable to distinguish correct from erroneous commands, being particularly susceptible to this attack.

Example of attacks on integrity:

- **Baseline Response Replay**, where the attacker replays genuine traffic between units recorded previously [25, 16];
- **MODBUS Diagnostic Register Reset Command**, where the attacker sends the command[5] and the device resets all the counters and diagnostic registers, changing the configuration of the device and impacting diagnostic operations [16];
- **Direct Slave Control**, where the attacker blocks the control of a master unit over one or more slaves, controlling them directly by spoofing its identity with the master’s identity causing the slave to comply with the commands sent by the attacker [25, 16];
- **Data Modification**, where the attacker manipulates the data being exchanged within a network, which requires access to the traffic to be manipulated and an already successful MitM attack.

In [28] (2016), Miciolino et al. performed a data modification attack, which modified the content of specific values within the packets destined to field devices. As a result, the commands sent by the Human-Machine Interface (HMI) of the system intended to deactivate a pump were modified as to close a valve, effectively contradicting the desired result, overflowing the tanks being controlled.

Table 2.1: Taxonomy of attacks on MODBUS.

Attack	**Availability**	**Confidentiality**	**Integrity**
Baseline Response Replay	X [16]		X [25, 16]
Broadcast Message Flooding	X [16]	X [25]	X [25, 16]
Data Modification			X

Continued on next page

[5]A well-defined command in the specification of the protocol, with function code 08 and sub-function code 0A.

Table 2.1 – continued from previous page

Attack	Availability	Confidentiality	Integrity
Direct Slave Control	X [25, 16]		X [25, 16]
Irregular TCP Framing	X [16]		
MODBUS Diagnostic Register Reset Command			X [16]
MODBUS Network Scanning		X [25, 16]	
MODBUS Query Flooding	X	X	X
MODBUS Slave Reconnaissance		X [16]	
MODBUS TCP Pool Exhaustion	X [16]		
Passive Reconnaissance		X [25, 16]	
Ping Flooding	X		
Remote Restart Command Flooding	X [16]		X [16]
Response Delay	X [25, 16]		X [25, 16]
TCP FIN/RST Flooding	X [16]		
TCP SYN Flooding	X [16]		

2.3 Intrusion Detection and Prevention Systems

Intrusion consists of wrongfully entering a system, either maliciously or not, with the intention to disrupt its operation by the actions performed during such intrusion. By monitoring and analysing the events within a system, an Intrusion Detection System (IDS) may detect signs of those intrusions (e.g., by comparing with a number of patterns – such as signatures – of how the system and their actors should behave). If it is also able to attempt to stop such possible incidents in real time, the system is called an Intrusion Detection and Prevention System (IDPS) [31].

Many methodologies can be used by IDPS, but, according to the work in [31], there are four widely used:

- **Signature-based detection**, in which a signature - a recognizable pattern of an event - is compared to a database of known attacks and threats, identifying possible intrusions. It is rather simple to implement and has little overhead since it does not inspect the activity itself, it only matches the signature. Is effective against known attacks but it is not flexible to adapt and find new or adapted attacks until the database is updated, which requires extensive work.
- **Anomaly-based detection**, in which a deviation from a defined profile - also known as nor-

mal or expected behaviour and defined by a set of attributes monitored over a period of time - is sought. The profile may be either static - where the profile does not change once established - or dynamic - where the IDPS updates the profile as the system evolves.

A dynamic profile adds extra overhead to the system, given the continuous update of the profile, but it may be required if the process is not stationary. Furthermore, if a threat is temporally spaced, it can become itself a part of the profile and thus avoid detection by the system.

- **Stateful Protocol Analysis**, in which the observed behaviour is compared to the expected behaviour of the protocols involved, requiring a deep understanding of how the protocols and its actors should interact/work, placing a big overhead on the system. Although the deeper understanding of the actions, attacks can evade detection by maintaining themselves within the acceptable/expected behaviour of the protocols.
- **Hybrid-based methodology**, in which a combination of two or more methodologies are used to take advantage of the strengths of each methodology.

According to Zhu in [32], IDS approaches can be grouped into several categories, as depicted in Table 2.2.

Table 2.2: Taxonomy of IDS approaches [32].

Approach	**Detection basis**
Signature	Known attacks are listed and detected when their signature appears
Anomaly	Models of normal communication of the system are learned and anomalous events are detected
Probabilistic	Models of communication are constructed and when patterns of misuse are detected the system calculates the probability of being an intrusion
Specification	Models of the system are constructed with knowledge of specifications of the system and protocol involved and a violation of such model raises an alarm
Behavioural	A behavioural pattern of attacks is captured and are detected

2.4 Machine Learning Techniques

Machine learning techniques provide to the systems the capacity to learn and improve upon some task from experience without being explicitly programmed to adjust. They can be generally divided in the following three categories [33]:

- **Single machine learning techniques**, of which Neural Networks, genetic algorithms, and Support Vector Machines are good examples;
- **Hybrid techniques**, where several single machine learning techniques are combined into a system, typically by cascading different techniques, i.e., where the outputs of a technique are analysed as the input of other techniques;
- **Ensemble techniques**, where multiple machine learning techniques are used independently in a system and the result of each technique is combined (e.g., majority vote of all the results) to achieve better prediction accuracy, e.g., Random Forests.

2.4.1 Techniques Implemented

Numerous techniques are available in the field of machine learning and they can be categorized within five major schools of thought [34]:

- **Symbolists**: focused on the premise of inverse deduction. Instead of the classical model of starting with a premise and looking for the conclusions, inverse deduction starts with a set of premises and conclusions and works backward to fill in the gaps;
- **Connectionists**: one of the most well-know, focused on reengineering the brain. Generally, it is based on connecting artificial neurons in a neural network;
- **Evolutionaries**: focused on applying the idea of genomes and DNA in the evolutionary process to data processing. In essence, evolutionary algorithms will constantly evolve and adapt to unknown conditions and processes;
- **Bayesians**: focused on handling uncertainty using techniques like probabilistic inference. Typically, Bayesian models will take a hypothesis and apply a type of *a priori* thinking, believing that there will be some outcomes that are more likely;
- **Analogizers**: focused on matching bits of data to each other.

The techniques used in this work represent the ones most used in research of IDS and are succinctly reviewed in the following sections.

2.4.1.1 Support Vector Machine

An algorithm where an optimum hyperplane, which can maximize the separation between data from different classes, is sought [35]. The margin of separation between data defines the decision frontier, i.e., the optimum hyperplane, and the patterns nearest to that frontier are the support vectors. It is currently a state-of-the-art technique in pattern recognition given its ability to work in highly complex classification problems. Its usage of kernel functions (e.g., Gaussian, Linear, and Polynomial) – capable of operating in high dimensions which map the data into a higher dimension – allows for the generalization of non-linear frontiers, facilitating pattern classification.

2.4.1.2 k-Nearest Neighbours

A classification algorithm where the k nearest neighbours to an instance are chosen to perform its classification, resorting to the minimization of a distance metric (e.g., Euclidean, Manhattan or Minkowski distance) [36]. To classify an instance its distance to the training instances is calculated, allowing the determination of its neighbours, from which the k nearest ones are chosen to classify. The class of that instance is then determined by a majority vote of its neighbours, i.e., the class most represented is chosen to classify the new instance, or by the class that has the most weight, when a weighing procedure is applied and smaller distances have bigger weights in the classification. The biggest drawback of this algorithm is the distance calculation required for each new instance to the entire dataset, which can be a computational heavy task. This algorithm also requires the determination of the best k and the distance metric that is most appropriate.

2.4.1.3 Naïve Bayes

An algorithm which considers the distribution of probability of the data in each class to perform a decision on the classification process, assuming that exists a probabilistic relationship between the features of the instances and its class and that they are independent between them [37]. As such, the algorithm determines the probability that a given instance **x** belongs to class c_i, called the *a posteriori* probability, represented as $P(c_i|\mathbf{x})$, using Bayes formula, as seen in Equation (2.1), where $P(c_i)$ represents the *a priori* probability of occurring that class, $P(\mathbf{x}|c_i)$ the likelihood of **x** in a given class, and $P(\mathbf{x})$ is the probability of **x**.

$$P(c_i|\mathbf{x}) = P(\mathbf{x}|c_i)\frac{P(c_i)}{P(\mathbf{x})} \tag{2.1}$$

In a binary classification problem, where the decision is between classes c_1 and c_2, the algorithm determines two *a posteriori* probabilities, $P(c_1|\mathbf{x})$ and $P(c_2|\mathbf{x})$, and decides according to the class with higher probability, or randomly when the probabilities are equal.

2.4.1.4 Decision Tree

A supervised learning algorithm applied in data mining, which recursively divides the data in nodes of a tree until a leaf is reached. Each node corresponds to a test done to a feature of the instances and each branch corresponds to a value (or set of values) that feature has. The leafs of the tree are associated with a class and correspond to the decision being made for a given instance. The tree is built from a training set of data and the classification process on new instances is made using the resulting tree [38].

2.4.1.5 Random Forest

An ensemble learning algorithm, used in classification, which combines multiple Decision Trees predictors independently constructed and classifies new instances as the mode of the predictions made. It tends to improve the Decision Tree's generalization error as the number of trees in the forest increases, although it depends on the strength of the individual trees in the forest and the correlation between them.

2.4.1.6 Neural Network

A learning algorithm used in classification which mimics the neurons of the human brain [39]. It is based on a collection of connected units called neurons that process signals given to them and whose results signal additional neurons connected to it. The training process is done by tuning the weights of the connections between neurons until a satisfactory result is achieved, i.e., the minimization of the error of misclassification – usually done by relying in the backpropagation learning algorithm.

2.4.2 Cross-Validation

Cross-validation is a common strategy for performance analysis of a given technique by giving insight on how their results will generalize to an independent dataset [40]. Several strategies may be implemented, such as k-fold or holdout cross-validation, of which the latter was the one employed in this work, given it requires less computational resources. In holdout cross-validation, a dataset is randomly divided into two sets, usually called training and test sets, whose sizes are arbitrarily defined, where the former is used during the training procedure of the technique being evaluated while the latter is used to test the resulting model for performance. Given that the test data was not used in the training procedure, the performance obtained in that set can give insight to the generalization capabilities or overfitting of the model.

2.4.3 Evaluation Metrics

The performance of a machine learning technique must be measured using empirical metrics [41]. The metrics presented can be related to the confusion matrix, a bi-dimensional table which represents the number of real and predicted cases in each class within a dataset. Although it can be used for multi-class classification problems, the confusion matrix can become hard to interpret, while in the binary classification problem its a powerful tool. A binary classification problem's confusion matrix can be seen in Table 2.3.

		True Condition	
		Positive	**Negative**
Predicted Condition	**Positive**	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Table 2.3: Confusion matrix for a binary classification problem.

From the confusion matrix it is possible to define the following terms: True Positive (TP) and True Negative (TN), when the predictions made correspond to the reality; and False Positive (FP) and False Negative (FN), otherwise. We can also define the following evaluation metrics [41]:

- **Detection Rate (DR)**, which represents the ration between the correctly identified positive states and the overall number of positive states existent. It is defined as

$$\text{DR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

- **False Positive Rate (FPR)**, which represents the ratio between false states misidentified as positive and the overall number of existent negative states. It is defined as

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.3}$$

- **Accuracy**, which represents the ratio of correct predictions over the number of total predictions made. It is defined as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.4}$$

- **Precision**, which represents the ratio between the correctly identified positive states and the overall number of positives states that were identified (correctly or incorrectly). It is defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.5}$$

- **Recall**, which represents the ratio between the correctly identified positive states and the overall number of positives states that should have been identified. It is defined as

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.6}$$

- **F1 score**, which represents the harmonic average of the precision and recall measures of a test. It is defined as

$$\text{F}_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FN} + \text{FP}} \tag{2.7}$$

Chapter 3. Literature Review

Cyber-Physical Systems' security has had a lot of research, where multiple types of techniques have been explored and Intrusion Detection Systems from simple to complex have been proposed. In this section, a review of that research is presented. A comparison table of the results of the works reviewed is also available in Table 3.1.

One of the fields of research has been the development or domain-adaptation of techniques to security of SCADA systems. A good example of this kind of research was made by Shitharth and Prince Winston, in [42] (2017), where they proposed an Intrusion Weighted Particle-based Cuckoo Search Optimization with Hierarchical Neuron Interface based Neural Network (IWP-CSO with HNA-Neural Network) technique in order to detect and classify intrusions in SCADA networks. The technique starts by organizing the input data into clusters where the attributes available are arranged and the optimal attributes are selected by the IWP-CSO. The intermediate output is then given to the HNA-NN algorithm which classifies the results. The combination of both algorithms intends to reduce the dimensionality of features while improving the overall accuracy. The approach was tested against four datasets – Single-hop Indoor Real Data (SIRD), Single-hop Outdoor Real Data (SORD), Multi-hop Indoor Real Data (MIRD), and Multi-hop Outdoor Real Data (MORD) – to detect external attacks (DoS and spoofing, specifically). The approach presented a detection rate ranging from 94.28% to 100% with a false positive rate ranging from 0% to 1.14%. The approach was also tested using the ADFA-LD dataset and compared to SVM, HNA-NN, and IWP-CSO with SVM presenting better results in every metric measured against those techniques.

Another example was made by Lil, Wanl and Zengl, in [43] (2015), were they proposed an anomaly detection system based on an improved One-Class Support Vector Machine (OCSVM) algorithm, reliant in data preprocessing and normalization, an appropriate kernel function and the calculation of a decision function, which was done with the Particle Swarm Optimization (PSO) algorithm, since I(A)CS systems tend to be more well-behaved than traditional networks. The approach was tested in a SCADA testbed using the MODBUS protocol. The PSO optimization with the OCSVM detection proved successful, with accuracies of 96% and 100% for the test and training sets, respectively, while having better efficiency than traditional grid-search OCSVM, requiring 10 times less computational power, which should allow the construction of more concise models, with stronger generalization and fewer support vectors.

However, more research has been made in anomaly detection, which have the prospect of detecting abnormal behaviour of the system. Different type of techniques have been proposed, such as the Ant Colony Clustering Model (ACCM) within a multi-agent IDS architecture by Chi-Ho Tsang and Sam Kwong, in [44] (2005). Different agents perform certain functions in order to provide the detection capabilities, such as traffic monitorization and pre-processing, classification (where the ACCM algorithm is embedded), coordination, counter-attack action, user-interaction, and registration. The ACCM algorithm specifically is based on positive and negative feedbacks for self-organization in order to create larger clusters and destroy smaller ones, using a heuristically approach. Against the KDD-Cup99 dataset[1], the algorithm presented better results and more resilience when using a Principal Component Analysis and the number of principal components increased. It presented around 90% detection rate with a FPR of 1%. It was also compared against other approaches (K-Means, E-M, Ant-based clustering, multiple classifier, and KDD-Cup99 winner) and it outperformed them in almost every case in its recall rates.

Yang, Usynin and Hines proposed, in [45] (2005), an anomaly-based IDS composed of an Auto-Associative Kernel Regression (AAKR) and Sequential Probability Ratio Test (SPRT) for SCADA systems. The AAKR is an empirical modelling technique which makes predictions based in historical, exemplar observation of the traffic used for distance calculation, similarity quantification, and output estimation, while the SPRT technique tests the likelihood of an observation to be abnormal. The technique was tested against a simulation of a SCADA system, where an attack was introduced to simulate a malicious inside action. Such attack was correctly identified by their IDS. However, the results lack consistent testing against multiple types of attacks and variable states of the SCADA system, as stated by the authors themselves. This work also lacks, unfortunately, quantitative results to assess the performance claimed.

Two other anomaly detection techniques have been proposed by Valdes and Cheung, in [46] (2009): pattern-based host-to-host communication anomaly detection, and flow-based detection. In the pattern-based technique, the approach relies in an adaptive library of known patterns – initially empty – that can record both normal and abnormal activity (which does not require attack-free training data) and is periodically updated and pruned (similar patterns are consolidated and rare ones are discarded). The inputs are compared to the library of patterns and, if the probability of the input matching one of the recorded pattern is below a certain threshold, an alarm is raised. In the flow-based technique, a database of active and historical flow records is maintained and an alarm is raised when anomalies of the current network flow does not match the expected behaviour – e.g., a

[1]Available at `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

significant change of the rate of flow, an absence of an expected flow, or the emergence of a new flow.

Erez and Wool proposed, in [47] (2015), a domain-aware anomaly detection system to detect control register values with irregular changes. An automatic classifier of the registers was developed using a single-window classification algorithm in order to distinguish different types of registers (namely, sensor, counter, and constant registers) given their respective characteristics in runtime of the PLCs, such as their variance and monotony. Using the dataset provided by the Tel Aviv University (used in [48]), the authors tested the classifier by varying several aspects of the classification parameters, such as, the window size and thresholds. They found that after a certain size, the classification window did not yield better results. Upon further analysis, the authors found that certain characteristics of registers were too slow for the algorithm to notice, misclassifying them, but, as mentioned, increasing the size of the window would not yield better results. Adapting the approach, a long time-frame was divided in several parts where the single-window algorithm would be applied and the register classification would be incremental, affected by the classification in the current window and in the previous windows, using a finite-state machine. This new approach improved the performance with a True Positive Rate (TRP) ranging from 91% to 100% and a FPR lower than 3% in each type of register. Then, each type of register was modelled according to their class. Testing against the dataset found the system achieved a 93% classification accuracy and a 0.86% false alarm rate. However, the technique requires further testing, such as against subtle anomalies or attacks to the system, to assess its full potential.

SCADA systems tend to be highly static, which present an unique advantage against attacks: by modelling the expected behaviour of a system, any deviation from normal operation should raise a flag, even if it is not an attack, and quickly dealt with. In this field of research, Cheung and Skinner proposed, in [49] (2006), an IDS wichh explored three different techniques: a protocol-level modelling for MODBUS over TCP/IP, a communication pattern modelling of the network, and a service availability modelling of the network. The protocol-level intrusion detection was implemented using Snort rules derived from the specifications of the protocol, allowing the detection of relatively generic threats against these types of networks. However, given the limitations of this approach, detecting system-specific threats are cumbersome and error-prone, and the authors aim to use the model as a detector itself, but the approach is still in an embryonic state. The communication pattern-based detection allows the definition of a set of rules that the communication in these networks should follow, such as the communication of a server unit to another unit which should not happen in MODBUS. The service availability-based detection uses a heuristic approach of learning the models of availability of servers and services in the network. A service discovery component maintains a list of active services and periodically detects their state. When any of them become unavailable for no apparent reason or new services appear, an alarm is raised. The IDS was tested in a SCADA testbed against an attack scenario where the system was compromised by a zone of the system connected to the Internet that successively gained access to the entire system. The authors claim that their approach proved effective for monitoring SCADA networks, where different sensors detected different aspects of their multistep attack scenario, however no quantitative results were made available to assess such claim.

Valdes et al., in [1] (2009), proposed a multilayer security architecture that uses model-based monitoring, complemented by a conventional signature-based detection, and a hierarchical security event management framework to correlate events. The model-based monitorization implemented included specification-based, change, and statistical anomaly detection, which varied in the level of automation used and the abstraction layers used for the model. A heuristic approach was also used to model availability of servers and services used in MODBUS over TCP/IP to generate reports of ”interesting” events, such as the detection of new units or the use of function codes unused previously. These model-based sensors were incorporated in an IDS containing a multi-algorithm and multilevel analysis, a stateful packet reassembly and protocol analysis engine, a Bayesian protocol anomaly detection engine, a signature-based detection system (Snort), and a probabilistic alert aggregation engine. Using an exemplary testbed [49], the multi-algorithm intrusion detection was experimentally validated against a multistep attack scenario, which included reconnaissance and direct attacks to system hosts. The algorithms were able to detect different aspects of the attack with various degrees of success and removing any of the algorithms resulted in diminished detection of the system.

A SCADA IDS architecture based on system knowledge and system state analysis has been proposed by Fovino et al., in [50] (2010). It maintained a virtual image of the various components, information flows, critical states, and known vulnerabilities (known vulnerabilities are mentioned in Section 2.1.5.1). Alongside a signature-based detection system, every command was interpreted in terms of the consequences to the state of a virtual image of the system – relying in behavioural profiles of the components –, raising alerts when critical states were detected. The architecture was tested in three phases – single packet signature detection, critical state detection, and performance analysis – using a reproduction of a power plant’s environment and a limited set of rules, both for signature-based and critical state detections, where the proposed architecture proved successful in the detection of the attacks. In the third test, the system was tested against various amounts of traffic and it was found that over 180 kbit/s the number of alerts raised by the IDS did not match the number of expected alerts. However, the authors considered it a satisfactory result given the number of rules inspected and the average low bandwidth typical in industrial networks.

Goldenberg and Wool proposed, in [48] (2013), an IDS based on the modelling of SCADA traffic with the assumption of rigid periodicity of communications between elements of the systems, which allowed simple models to be made with high predictive capabilities. The approach relied in the modelling of each communication channel with a Deterministic Finite Automata (DFA). The communication patterns are described by the set of states and the transition between them, where four types of transitions are defined, specifically, normal – corresponding to the pattern of communication –, retransmission – usually due to congestion in the TCP traffic –, miss – when a pattern is out of order, which can also happen due packet loss in the communication process –, and unknown – un unknown pattern occurred, which can signal a human operation, an unmodelled pattern, or an attack. In order to generate the models, the authors presented an automatic model generator which did not require labelling of the data, requiring only a capture of normal traffic longer than the pattern to model. To validate the technique, the authors captured two MODBUS datasets from the system of the Tel Aviv University used to monitor the campus power-grid several months apart from each other. The DFA-based technique proved successful with the first dataset, where the unknown transitions in the system were localized in time (a suspicious occurrence) and correlated to maintenance of the system. However, in the second dataset the same approach was not successful, as the unknown rate was alarmingly high with no localization in time. After analysing the system, the authors found various patterns with different frequencies corresponding to normal operation of the system. As a consequence, the approach was adapted (since a single DFA would require an enormous time interval of traffic to construct the model with every pattern occurring once which would result in a DFA with millions of states) and a multi-level DFA was devised. A first-level DFA, corresponding to the fast periodic sequence, evaluates the stream and, when an unknown event occurs, passes the unknown input to the second-level DFA, corresponding to the slow periodic sequence, which evaluates as normal or unknown – this last unknown raises a signal to the operator. This adapted approach proved successful, diminishing the unknown rate by two orders of magnitude. However, the validity of this approach was not fully assessed as the datasets did not contain any kind of attacks and a single SCADA system was evaluated, requiring further investigation, specifically to more dynamic systems.

A passive SCADA IDS based on unsupervised anomaly detection focused on integrity attacks has been proposed by Almalawi et al., in [51] (2014). The approach was based in two different techniques: a scoring of inconsistency of unlabelled data, which used the density factor of the kNN of each observation, and the extraction of proximity-based rules allowing for a small set of rules to define the system as a model. The authors tested the approach using three different datasets, namely, a real dataset (available at [52]) and two datasets obtained from a simulation of a water distribution SCADA system using the MODBUS protocol suffering a MitM attack. The approach presented overall good results identifying consistent from inconsistent states in all of the three datasets, reaching detection rates of around 98% for the simulated ones and 86% for real one.

Yoon and Ciocarlie proposed, in [53] (2014), a method for modelling normal I(A)CS behaviour based on sequencing of requests and replies in order to determine the state of the devices in a network using a Probabilistic Suffix Tree (PST) which in turn uses variable-order Markov chains. An incremental PST algorithm was introduced for online-learning capabilities, where the tree is incrementally updated to include recently-learned sequences. A missing element approach is also included in order to avoid the learning of erroneous sequences and cause multiple false alarms by the system. The approach was tested in a testbed configured with a MODBUS network, from where a clean dataset was obtained. The authors compared the PST approach with batch learning, the incremental learning introduced, and the incremental learning with a missing element detector. The batch learner produced a better False Positive Rate (FPR) in every connection tested than the incremental learner (1.429% against 1.819%), an expected result since the incremental learner only knows the history of communication up to the point is testing whilst the batch learner had already known the entire history. However, the incremental learner with missing element detector presented the better average FPR of the three (0.848%), although presenting the less robust results of with the highest standard deviation (0.011 against 0.007 of the batch learner and 0.010 of the incremental learner). When introducing a synthetic anomaly in the dataset, namely, a set of random sequences is introduced with a repeating pattern that randomly drops some packets in order to randomize the sequence further. With a low missing probability of packets, as in the previous test, the incremental learner with missing element detector outperformed the other two learners, presenting a better capacity to distinguish attacks from noise. However, increasing the missing packet probability, the incremental learner with missing element detector has a poorer performance, which indicates that this learner is appropriate only for communications with a dominant pattern present.

Caselli, Zambon and Kargl, in [54] (2015), proposed a sequence-aware IDS for Industrial (Automation and) Control System that analyses sequences of allowed operations within the system that pose significant threats to the systems, using Discrete-Time Markov Chain (DTMC), where a model of regular behaviour of the system is built and anomalies are detected when a DTMC built during the detection phase does not match the one created during the training stage. Distances between the expected and current states are measured and raise an alarm when a predefined threshold is surpassed. The approach was tested against a trace of a water treatment plant using the MODBUS protocol. In a first test, the detector ran across a clean trace using a small threshold for alerts, raising a total of 211 alerts (false positives, since it was a clean trace), of which 96% were due to delays on ”read” operations. In a second test, two sequence attacks were introduced to the same trace and the detector raised the same 211 false positives as before plus 8 alarms corresponding to the introduced attacks. In order to diminish the number of false positives of the system, the authors introduced an event importance monitor which employed various different approaches, such as categorization of the events – events that control the system have higher risk than events that only serve to monitor the system –, proximity to other important events, and transition timing variability. With this new setup, the clean trace only raised 9 alarms (false positives) and the trace containing the attacks raised 17 alerts on the attacks, nine more than the previous setup, improving the detection capability of the system.

Finally, Faisal, Cardenas and Wool, in [55] (2016), proposed a specification-based IDS complimentary to the DFA (presented in [48]) and DTMC (presented in [54]) models employed for MODBUS over TCP/IP. The specification-based approach is built as a set of rules defining allowed behaviours within the network constructed using *a priori* knowledge about the protocols involved and the system specifications. The approaches were tested against two datasets: a one-day trace from a real-world operational large-scale water facility in the USA, and a trace of a university power grid monitoring process. The authors found that the DFA approach could generate models for all the channels in the second dataset but was unable for 68% of the channels analysed in the first dataset. For the DTMC approach, the algorithm was able to generate values for all the channels analysed, where the size of training data did not affect significantly the size of the model but increased the number of transitions used within those models. Their specification-based approach was not thoroughly tested since the authors did not possess the a priori design and configurations of the systems evaluated, requiring inferred information from analysis of the datasets, stating only that it raised 15 alarms on one of the channels of the first dataset for delayed responses.

Combined approaches are also relevant research in the field of intrusion detection. The work made by Samdarshi, Sinha and Tripathi, in [56] (2016), where they proposed a triple layer IDS for SCADA systems is a good example. The first layer ensured the protection of the communication network, namely, the TCP/IP network. This layer was tested using the J48 and the Random Forest algorithms available at WEKA, which presented 99.96% and 99.98% accuracy and 0.998 and 0.999 Kappa coefficient, respectively. However, the Random Forest took 447.54 seconds against the more reasonable 54 seconds of the J48 algorithm for the same number of instances. The second layer ensured command and state authentication by detecting tampering of data by an intruder with a *What-if* module, which fed data from the field to an existing model of the system, checking the authenticity of the commands, whose result is simulated and warns the operator of insecure states of the system. The third layer ensured the authentication of the data entering the control centre and classified the state of the system coming from the second layer as a disturbance, an attack or a normal state. This layer was tested with one of the ORNL-MSU datasets using Naïve Bayes, J48, Random Forest, and Adaboost with JRipper algorithms, where the accuracies were, respectively, 33.88%, 70.90%, 75.68%, and 94.01%.

A significant work of comparing the effectiveness of multiple machine learning algorithms in SCADA systems has been made in [57] (2014), by Borges Hink et al. Using Weka as the framework and the open-source power system data provided by The Mississippi State University divided in 15 datasets, randomly sampled at 1%, the authors used three testing schemes, using multiclass, three-class, and binary class classification of events. They tested several algorithms (OneR, NNge, Random Forests, Naïve Bayes, SVM, JRipper, and Adaboost) in terms of accuracy within each of the 15 sets and of average precision, recall, and F-measure. The first noteworthy aspect found was the consistency of the results of each learner regardless of the dataset used. Secondly, there could be no definitive conclusion of using different classification schemes, since each learner has a different response to each scheme (the multiclass scheme was clearly better for the Random Forest learner, while the binary scheme was better for Naïve Bayes, and the three-class scheme was better for JRipper). Thirdly, the Adaboost+JRipper learner had the best performance of all the learners.

Table 3.1: Comparison of the works reviewed.

Pub.	Scope	Dataset	Technique[2]	DR	FPR	Acc
[44] (2005)	I(A)CS	KDD-Cup99	K-Means & FastICA	89.17%	4.29%	-
			E-M & FastICA	90.94%	4.24%	-
			ACCM & FastICA	92.23%	1.53%	-
[53] (2014)	MOD-BUS	Testbed trace	Batch PST trees	-	1.429%	-
			Incremental PST trees	-	1.819%	-

Continued on next page

[2]In bold, is(are) the technique(s) introduced in the paper, while the others are comparisons made in the same paper.

Table 3.1 – continued from previous page

Pub.	Scope	Dataset	Technique	DR	FPR	Acc
			Incremental PST trees with element missing inference	-	0.848%	-
[57] (2014)	SCADA	Open-source simulated power system data by Mississippi State University	OneR, Nearest Neigbor, Random Forests, Naïve Bayes, SVM, JRip, Adaboost+JRip	-	-	-
[47] (2015)	MOD-BUS	2 traces of Tel Aviv University's power grid monitor	Domain-aware classification algorithm	-	0.86%	93%
[43] (2015)	MOD-BUS	Testbed trace	**PSO-OCSVM**	-	-	96%
			Grid-OCSVM	-	-	87.5%
[42] (2017)	SCADA	SIRD	EDDC	100%	0%	-
			IWP-CSO & HNA-NN	100%	0%	-
		SORD	EDDC	96.88%	1.94%	-
			IWP-CSO & HNA-NN	97.57%	1.14%	-
		MIRD	EDDC	100%	0.31%	-
			IWP-CSO & HNA-NN	100%	0.18%	-
		MORD	EDDC	92.98%	0.04%	-
			IWP-CSO & HNA-NN	94.28	0%	-
		ADFA-LD	SVM	43.53%	13.02%	-
			HNA-NN	61.51%	9.99%	-
			IWP-CSO & SVM	93.01%	8.85%	-
			IWP-CSO & HNA-NN	100%	8.35%	-

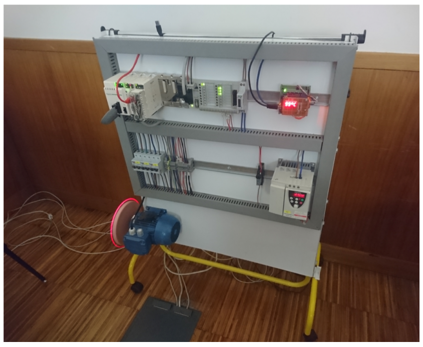
Although many approaches, with varying degrees of success, have been presented over the years, the lack of comparable datasets with comparable attack procedures prevents the proper generalization of these techniques and real-world application. This work aims to present the frailties of machine learning algorithms when datasets with different characteristics are fed to them and serve a stepping stone to future works to proper generalize them.

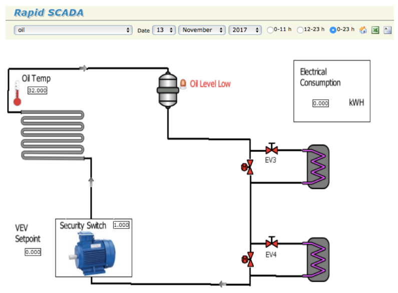
Chapter 4. Proposed Architecture

The availability of datasets or network traces containing normal SCADA operations, as well as attacks aimed at those systems, is very limited. To overcome this limitation, which would not allow a proper evaluation of IDS based on machine learned classifiers, a testbed (described in Section 4.1) was used to generate those datasets. The trace collection procedure also included attacks, which were performed as described in Section 4.2, which were then processed by the algorithms as described in Section 4.3.

4.1 Simulation Environment

The testbed emulates a CPS process controlled by a SCADA system using the MODBUS protocol [58]. It consists of a liquid pump simulated by an electric motor controlled by a Variable Frequency Drive (VFD) – allowing for multiple rotor speeds –, which in its turn is controlled by a Programmable Logic Controller (PLC). The motor speed is determined by a set of predefined liquid temperature thresholds, whose measurement is provided by a MODBUS Remote Terminal Unit (RTU) device providing a temperature gauge, which is simulated by a potentiometer connected to an Arduino. The PLC communicates horizontally with the RTU, providing insightful knowledge of how this type of communications may have an effect on the overall system. The PLC also communicates with the Human-Machine Interface (HMI) controlling the system, which was built using RapidSCADA [59]. The testbed, including the HMI used, can be seen in Figure 4.1, while a schematic representation can be seen in Figure 4.2.


(a) The installation.



(b) The HMI.

Figure 4.1: The testbed used for the simulation environment.

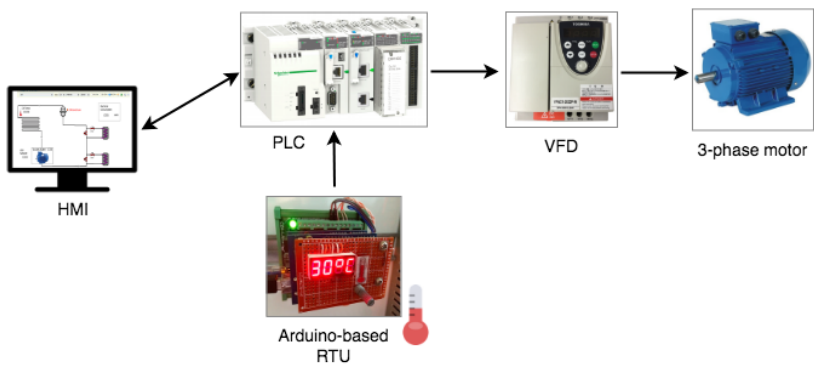


Figure 4.2: Schematic representation of the testbed.

4.2 Attack Implementation

There are several types of attacks that are effective against SCADA-based systems using the MODBUS over TCP/IP protocol implementation. For analysis purposes, a subset of the attacks mentioned in the introductory section was implemented in the testbed, namely:

1. Ping flooding;
2. TCP SYN flooding;
3. MODBUS Query Flooding - Read Holding Registers;
4. Passive Reconnaissance;
5. Data Modification.

The first three attacks targeted the PLC, while the last two attacks targeted the HMI. The first two attacks attempt to overwhelm the capacity of the network or the networking subsystem in the target device with requests (operating mostly at OSI layers 2 to 4), and were implemented using the *hping3* tool [27], which is capable of generating conventional DoS attacks and simulated DDoS by spoofing the packet's IP address (specifically, by setting the source with a random valid IP address). This last configuration was the one used in this work.

The third attack works at the SCADA protocol layer, flooding the device with read request operations which may lead to side effects such as device resource exhaustion, scan cycle latency deviations or loss of connectivity and was implemented using an adaptation of the *SMOD* tool [60].

The final two attacks present a MitM with two different objectives. The first one aims at discovering which type of information is being presented to the human operators of the system and allows for the definition of a baseline of the system, which can be exploited in future attacks. The second one modifies the data sent to the HMI, specifically the temperature reading went from 30ºC to 39ºC, which could trigger an intervention by human operators.

As mentioned, the aim of this work is to present the frailties of machine learning algorithms when different characteristics of datasets are fed to them. As such, an analysis of the impact of the dataset size and the relative magnitude of the attack traces within such datasets on the behaviour of machine-learned classifiers was made. For this purpose, the following experiments were made: varying the time of the capture (30 minutes, 1 hour, and 6 hours of capture); and varying the time of the attack (1 minutes, 5 minutes, 15 minutes, and 30 minutes of attack within each capture[1]).

The network captures were made using the *tshark* network analyzer tool [61], which was used to acquire the network traces on the testbed. For this purpose, the network switch that was used to interconnect the equipment was configured with a mirror port, providing access to a replica of all the network traffic on the testbed.

The captures were then processed for feature extraction within MATLAB [62], where a total of 68 features were extracted (packet timestamps, inter-packet arrival times, binary features defining which protocols were involved, and every field of the Ethernet, ARP, IP, ICMP, UDP, TCP and MODBUS over TCP/IP headers). However, to generalize the captures for normal use of the system, the timestamps were removed from the datasets before the analysis, maintaining only the inter-packet arrival times.

In the case of the MitM attacks, the attacks were only detectable in the trace by the presence of unsolicited *is-at* messages of the ARP protocol, which would not allow a proper generalization of the classification procedure of an attack. As such, the captures were reduced to the TCP packets only and their features were also reduced, but now contained the timespan of the TCP flows. Alongside a clean trace of the system, which allowed for the calculation of the mean time (and its standard deviation) of packet travel between each host, the classification procedure was then implemented by analysing those timespans.

4.3 Classifier Implementation

Six of the most used classifiers in literature (as reviewed in Chapter 3) were implemented in this analysis study, namely: k-Nearest Neighbors (kNN), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), Neural Network (NN) and Naïve Bayes (NB). Two experiments were devised: a default MATLAB implementation of each of the algorithms; and a tuned implementa-

[1]The 30 minutes capture did not have a 30 minute attack, since it would represent the entire trace.

tion of the algorithms, i.e., an hyperparameter selection was introduced to the algorithms. However, given time constraints and the prolonged nature of the learning procedure of the algorithms, the tuning could only be made to the kNN and SVM algorithms and were only tested against the first two attacks enumerated. Specifically, the kNN was tested with k being 1, 3, 5, 7 or 9; the distances being *cityblock*, *chebychev*, *correlation*, *cosine*, *euclidean*, *hamming*, and *jaccard*; and the distance weight parameter set as *equal*, *inverse*, and *squaredinverse*. The SVM was tested with kernel functions as *gaussian*, *linear*, and *polynomial* (the polynomial order was only tested as 2, due to the aforementioned constraints); the solvers as *ISDA*, *L1QP*, and *SMO*; and the ν parameter set as 0.1, 0.3, 0.5, 0.7, and 0.9. All the algorithms belong to the Statistics and Machine Learning Toolbox [63], except the Neural Network, which belongs to the Deep Learning Toolbox [64].

In order to validate the models created, a cross-validation procedure was performed with a 70%/30% ratio for training and validation sets.

Chapter 5. Experimental Results

In this chapter a report of the results obtained from the experiments described in Chapter 4 is made. This chapter is subdivided in two main sections: an analysis of the Denial-of-Service (DoS) attacks is made in Section 5.1, which includes the tuning analysis referenced; and an analysis of the Man-in-the-Middle (MitM) attacks is made in Section 5.2.

5.1 Denial-of-Service Attacks

This part of the experiment proposed has the goal of visualizing the effects of different characteristics of datasets, specifically, the effects of dataset size and the relative magnitude of the attack within such datasets, on the classification procedure with a broader goal of defining a baseline when comparing different works on the security of Cyber-Physical Systems.

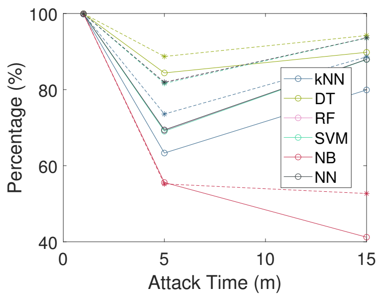
As mentioned in Chapter 4, the following experiments were made: varying the time of the capture (30 minutes, 1 hour, and 6 hours of capture); and varying the time of the attack (1 minutes, 5 minutes, 15 minutes, and 30 minutes of attack within each capture[1]). Two different analysis were performed: a direct application of the default configuration of the algorithms in MATLAB, covered in Section 5.1.1; and a hyperparameter tuning of the algorithms, covered in Section 5.1.2.

5.1.1 Default Configurations

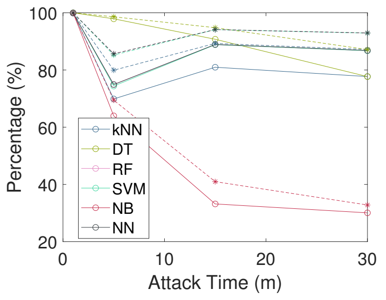
Three different traces were performed to allow generalization of the results. Figures 5.1 to 5.3 present the average results[2] obtained in the three captures of the accuracy and F1-score of the classification procedure of the six classifiers implemented in their default MATLAB configurations.

On a first approach, the analysis of the results reveals a trend for consistent high accuracy results for the smallest attack timespan. However, this is a misleading result, given the high imbalance in the data present in these situations, associated with the small variance of the attacks in such a small timespan. When the attacks increase in size, a decay of the performance can be observed, explained by the decrease of the aforementioned effects. The increasing attack timespan overcomes the imbalance within the dataset, allowing the algorithms to systematically learn more differences between

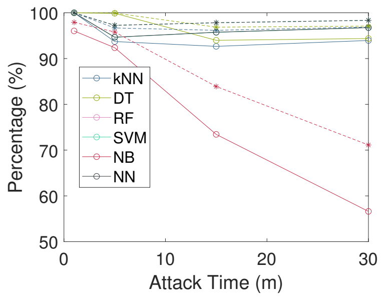
[1]The 30 minutes capture did not have a 30 minute attack, since it would represent the entire trace.
[2]The three captures followed the same trends.



(a) 0.5h of capture.

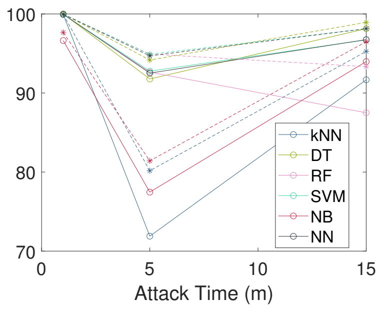


(b) 1h of capture.

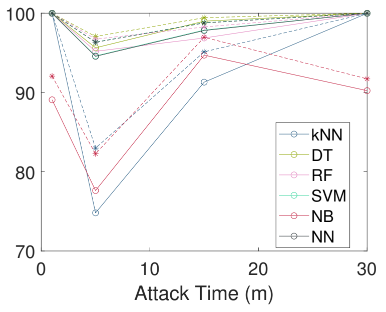


(c) 6h of capture.

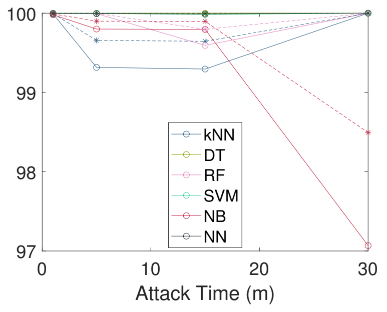
Figure 5.1: Average accuracy (full line) and F1-score (dashed line) of the implemented classifiers for the first attack.



(a) 0.5h of capture.



(b) 1h of capture.



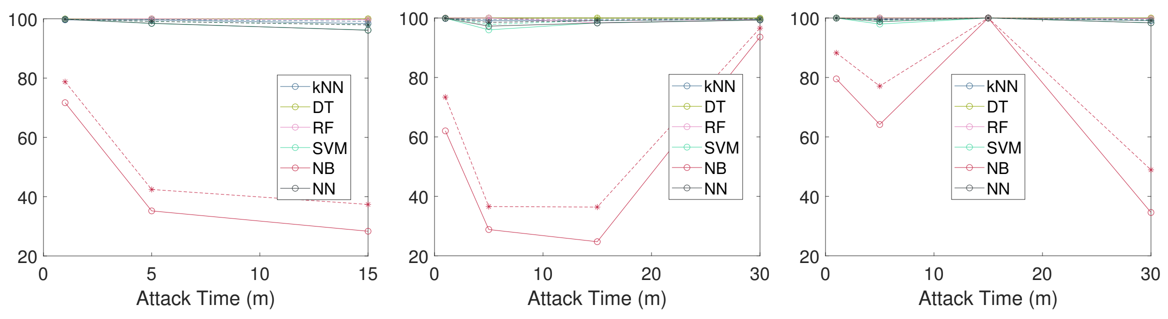
(c) 6h of capture.

Figure 5.2: Average accuracy (full line) and F1-score (dashed line) of the implemented classifiers for the second attack.

normal and anomalous traces and improve the results – a similar effect is also accomplished with an increase in the size of the capture. Although the increase of the size of the capture increases the imbalance of the dataset, it also increases the number of packets available to differentiate the traces, improving the performance of the classifiers, as can be seen by the reduced decay in performance when the time of capture increases. The improvement of the results as a consequence of the increase in the relative size of the attack traffic is also reduced when the size of the capture increases.

The Decision Tree classifier presents the best results of all the studied classifiers. By analysing the trees obtained, some of the attacks were classified by detection of the reduced inter-packet arrival times (a good metric for flooding attacks), however, when the inter-packet arrival times were not sufficient, the algorithm tended to overfit the data and, consequentially, exhibited higher performances but lacked generality. The Random Forest algorithm aims to prevent these overfitting issues and, consequentially, showcases worse accuracy.

The Naïve Bayes classifier tends to generally decrease in performance with increasing attack timespans for the ping flooding attack, while in the other attacks it presents the same trend as the other classifiers. Given its more statistical nature and assumption of statistical independence between features, which clearly is not the case, since some of the values are dependent of others, its perform-



(a) 0.5h of capture. (b) 1h of capture. (c) 6h of capture.

Figure 5.3: Average accuracy (full line) and F1-score (dashed line) of the implemented classifiers for the third attack.

ance may be penalized because of that.

All the classifiers presented unusually good results for the third attack (the MODBUS query flood), prompting a deeper analysis of the dataset obtained. This allowed for the detection of a field with little variance during the attack, which allowed the algorithms to detect the attack with ease. Consequentially, inferring upon these results may be overreaching, requiring further analysis and an adaptation of the implemented attack in future works.

In conclusion, the effects of data imbalance on the classification process constitute a real problem that can lead to unintentional misleading when analysing classifier performance. Moreover, this situation can be hard to detect since the datasets used for CPS security research are frequently restricted (and consequentially, difficult to analyse and characterize).

5.1.2 Hyperparameter Tuning

In the previous experiment, a default configuration of the algorithms implemented was used. However, those algorithms have some parameters which can be tuned in order to present better results, such as the distance metric used in the kNN algorithm.

Table 5.1 presents the performance of the various parameters tuned in the k-Nearest Neighbors algorithm, alongside their variation from the default configuration – one neighbour, euclidean distance and equal distance weight. It is possible to observe that simple parameter differences can have some impact on the performance of the algorithm, such as what happened when the distance metric was changed from ’euclidean’ to ’hamming’, which introduced a variation of about 5% in accuracy and almost 19% in F1-score, while other variations have almost no effect on the performance. This part of the experiment does not hope to define the best parametrization of the algorithm for detecting DoS attacks, but to demonstrate that a search of the best parametrization should be considered when implementing these algorithms.

Table 5.1: Performance of the kNN algorithm with hyperparametrization tuning for the ping flooding attack.

Number of Neighbours	Distance	Distance Weight	Acc. (%)	Acc. Variation (%)	F1 (%)	F1 Variation (%)
1	chebychev	equal	63.10	-0.36	73.43	-0.20
1	chebychev	inverse	63.10	-0.36	73.43	-0.20
1	chebychev	squaredinverse	63.10	-0.36	73.43	-0.20
1	cityblock	equal	63.82	0.77	73.99	0.56
1	cityblock	inverse	63.82	0.77	73.99	0.56
1	cityblock	squaredinverse	63.82	0.77	73.99	0.56
1	correlation	equal	63.62	0.46	73.82	0.33
1	correlation	inverse	63.62	0.46	73.82	0.33
1	correlation	squaredinverse	63.62	0.46	73.82	0.33
1	cosine	equal	63.77	0.69	73.95	0.51
1	cosine	inverse	63.77	0.69	73.95	0.51
1	cosine	squaredinverse	63.77	0.69	73.95	0.51
1	**euclidean**	**equal**	**63.33**	**0**	**73.58**	**0**
1	euclidean	inverse	63.33	0	73.58	0
1	euclidean	squaredinverse	63.33	0	73.58	0
1	hamming	equal	60.30	-4.79	59.87	-18.63
1	hamming	inverse	60.30	-4.79	59.87	-18.63
1	hamming	squaredinverse	60.30	-4.79	59.87	-18.63
1	jaccard	equal	60.30	-4.79	59.87	-18.63
1	jaccard	inverse	60.30	-4.79	59.87	-18.63
1	jaccard	squaredinverse	60.30	-4.79	59.87	-18.63
3	chebychev	equal	63.77	0.69	74.55	1.32
3	chebychev	inverse	63.70	0.58	74.47	1.21
3	chebychev	squaredinverse	63.70	0.58	74.33	1.02
3	cityblock	equal	64.34	1.59	75.07	2.03
3	cityblock	inverse	64.41	1.70	75.09	2.05
3	cityblock	squaredinverse	64.31	1.55	74.88	1.77
3	correlation	equal	64.19	1.360	74.95	1.86
3	correlation	inverse	64.17	1.320	74.78	1.63
3	correlation	squaredinverse	64.08	1.18	74.52	1.28
3	cosine	equal	64.08	1.18	74.86	1.74
3	cosine	inverse	63.99	1.04	74.64	1.44

Continued on next page

Table 5.1 – continued from previous page

Number of Neighbours	Distance	Distance Weight	Acc. (%)	Acc. Variation (%)	F1 (%)	F1 Variation (%)
3	cosine	squaredinverse	64	1.06	74.46	1.20
3	euclidean	equal	64.07	1.170	74.86	1.74
3	euclidean	inverse	64.05	1.14	74.82	1.69
3	euclidean	squaredinverse	63.98	1.02	74.65	1.46
3	hamming	equal	65	2.64	66.21	-10.01
3	hamming	inverse	65	2.64	66.21	-10.01
3	hamming	squaredinverse	65	2.64	66.21	-10.01
3	jaccard	equal	65	2.64	66.21	-10.01
3	jaccard	inverse	65	2.64	66.21	-10.01
3	jaccard	squaredinverse	65	2.64	66.21	-10.01
5	chebychev	equal	64.34	1.59	75.32	2.37
5	chebychev	inverse	64.28	1.50	75.24	2.26
5	chebychev	squaredinverse	64.12	1.250	74.97	1.89
5	cityblock	equal	64.88	2.450	75.89	3.14
5	cityblock	inverse	64.82	2.35	75.78	2.99
5	cityblock	squaredinverse	64.72	2.19	75.54	2.67
5	correlation	equal	64.89	2.46	75.84	3.07
5	correlation	inverse	64.66	2.1	75.49	2.6
5	correlation	squaredinverse	64.24	1.44	74.85	1.73
5	cosine	equal	64.97	2.59	75.87	3.12
5	cosine	inverse	64.61	2.02	75.43	2.52
5	cosine	squaredinverse	64.40	1.69	74.96	1.88
5	euclidean	equal	64.64	2.070	75.68	2.86
5	euclidean	inverse	64.68	2.130	75.65	2.82
5	euclidean	squaredinverse	64.48	1.810	75.32	2.37
5	hamming	equal	68.72	8.510	70.89	-3.65
5	hamming	inverse	68.72	8.510	70.89	-3.65
5	hamming	squaredinverse	68.74	8.540	70.9	-3.64
5	jaccard	equal	68.72	8.510	70.89	-3.65
5	jaccard	inverse	68.72	8.510	70.89	-3.65
5	jaccard	squaredinverse	68.74	8.540	70.9	-3.64
7	chebychev	equal	64.91	2.49	76.06	3.37
7	chebychev	inverse	64.77	2.270	75.9	3.16
7	chebychev	squaredinverse	64.41	1.7	75.44	2.53

Continued on next page

Table 5.1 – continued from previous page

Number of Neighbours	Distance	Distance Weight	Acc. (%)	Acc. Variation (%)	F1 (%)	F1 Variation (%)
7	cityblock	equal	64.96	2.570	76.27	3.66
7	cityblock	inverse	64.97	2.59	76.20	3.56
7	cityblock	squaredinverse	65.07	2.750	76.04	3.35
7	correlation	equal	64.9	2.480	76.23	3.6
7	correlation	inverse	64.78	2.29	75.87	3.12
7	correlation	squaredinverse	64.53	1.89	75.24	2.26
7	cosine	equal	64.89	2.46	76.20	3.56
7	cosine	inverse	64.68	2.130	75.78	2.99
7	cosine	squaredinverse	64.56	1.940	75.27	2.3
7	euclidean	equal	65.12	2.82	76.34	3.75
7	euclidean	inverse	64.97	2.59	76.16	3.51
7	euclidean	squaredinverse	64.94	2.540	75.89	3.14
7	hamming	equal	71.15	12.35	73.82	0.33
7	hamming	inverse	71.15	12.35	73.82	0.33
7	hamming	squaredinverse	71.19	12.41	73.84	0.36
7	jaccard	equal	71.15	12.35	73.82	0.33
7	jaccard	inverse	71.15	12.35	73.82	0.33
7	jaccard	squaredinverse	71.19	12.41	73.84	0.36
9	chebychev	equal	64.70	2.160	76.13	3.47
9	chebychev	inverse	64.66	2.1	76.04	3.35
9	chebychev	squaredinverse	64.65	2.080	75.82	3.05
9	cityblock	equal	65.60	3.580	76.99	4.64
9	cityblock	inverse	65.66	3.680	76.94	4.57
9	cityblock	squaredinverse	65.51	3.44	76.60	4.11
9	correlation	equal	65.35	3.19	76.84	4.43
9	correlation	inverse	65.17	2.9	76.39	3.82
9	correlation	squaredinverse	64.58	1.970	75.42	2.5
9	cosine	equal	65.42	3.3	76.85	4.45
9	cosine	inverse	65.13	2.840	76.32	3.73
9	cosine	squaredinverse	64.55	1.920	75.41	2.49
9	euclidean	equal	65.20	2.950	76.68	4.22
9	euclidean	inverse	65.24	3.010	76.63	4.15
9	euclidean	squaredinverse	65	2.64	76.20	3.56
9	hamming	equal	72.57	14.59	75.60	2.75

Continued on next page

Table 5.1 – continued from previous page

Number of Neighbours	Distance	Distance Weight	Acc. (%)	Acc. Variation (%)	F1 (%)	F1 Variation (%)
9	hamming	inverse	72.57	14.59	75.60	2.75
9	hamming	squaredinverse	72.81	14.97	75.76	2.97
9	jaccard	equal	72.57	14.59	75.60	2.75
9	jaccard	inverse	72.57	14.59	75.60	2.75
9	jaccard	squaredinverse	72.81	14.97	75.76	2.97

5.2 Man-in-the-Middle Attacks

This part of the experiment has the goal of exploring the classification process on Man-in-the-Middle attacks. These attacks were implemented using ARP cache poisoning, which, if correctly implemented, would mean that single ARP messages for each cache to poison would appear in the capture, which by themselves do not mean an attack is being implemented, although being a good indicator. Furthermore, a MitM attack would not present other variations in the features extracted in this experiment, unless a deep inspection of the values were compared to a previous built model of the system and a reckless change was perpetrated. However, such models are not easily constructed, scalable and generalized for widespread use in protecting these systems.

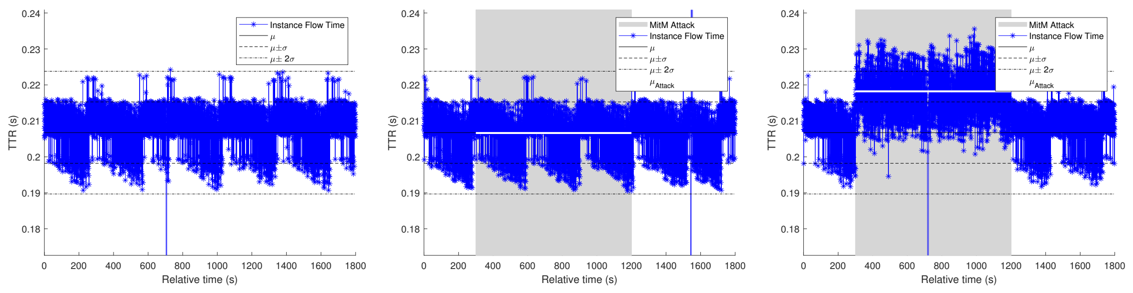
In a MitM attack, the traffic between two hosts is unknowingly redirected through a third party. As such, that third party, even if it is only immediately forwarding that traffic, should introduce a delay on the time of travel of the packets through the network. To verify such effect, the Time-To-Response (TTR) of the TCP packets in the captures of the two MitM attack mentioned in Chapter 4, was measured and compared to the results of a clean capture. Table 5.2 presents the mean and standard deviation values measured in different clean captures.

Table 5.2: Time-To-Response (TTR) of clean captures.

Capture	Length of Capture	TTR	
		μ (s)	σ (s)
1	0.5 h	0.21	0.01
2	1 h	0.21	0.01
3	6 h	0.21	0.05
Mean values		**0.21**	**0.02**

Figure 5.4 presents a comparison of the TTR of the clean capture and the attacks implemented. The mean (μ) and standard deviation (σ) of the values of the clean capture are marked on the figure

for comparability, as well as the mean TTR during the attack (marked with a white bar).



(a) Clean capture. (b) Passive reconnaissance attack. (c) Data modification attack.

Figure 5.4: Time-To-Response of TCP packets in a half hour capture with attacks of 15 minutes.

It is notorious that the passive reconnaissance attack introduces an almost negligible effect on the TTR, where the mean value varied by 0.05%, a variation which can easily be attributed to normal noise on the measurements or fluctuations on the network, whereas the data modification attack introduced a higher and measurable variation (5.55%), which pushed the mean value well beyond the first σ of normal variation. This bigger variation is to be expected, given that data modification implies dissecting a packet, changing its values – which in this case was one of the values of the innermost protocol of the entire protocol stack involved –, and the recalculation of the headers of the outer protocols. Tables 5.3 and 5.4 present the mean value of TTR and its variation by varying the size of the datasets[3] and the timespan of the passive reconnaissance and data modification attacks, respectively.

Table 5.3: TTR of the passive reconnaissance attack.

		TTR				
Length of Capture	**Attack timespan**	μ **(s)**	**Mean Variation (%)**	μ_{Attack} **(s)**	μ_{Normal} **(s)**	**Normal / Attack Variation (%)**
0.5 h	1 min	0.2078	0.5124	0.2031	0.2079	-2.3088
0.5 h	5 min	0.2074	0.3189	0.2069	0.2075	-0.2892
0.5 h	15 min	0.2074	0.3189	0.2066	0.2082	-0.7685
1 h	1 min	0.2078	0.5124	0.2068	0.2078	-0.4812
1 h	5 min	0.2076	0.4157	0.2071	0.2077	-0.2889
1 h	15 min	0.2079	0.5608	0.2069	0.2084	-0.7198
1 h	30 min	0.2069	0.0771	0.2071	0.2067	0.1935

Continued on next page

[3]The 6 hours captures were not analysed due to time constraints.

Table 5.3 – continued from previous page

		TTR				
Length of Capture	**Attack timespan**	μ (s)	**Mean Variation (%)**	μ_{Attack} (s)	μ_{Normal} (s)	**Normal / Attack Variation (%)**
Mean Value		**0.2075**	**0.3880**	**0.2064**	**0.2077**	**-0.6661**
Standard Deviation		**0.0003**	**0.1672**	**0.0014**	**0.0006**	**0.7926**

Table 5.4: TTR of the data modification attack.

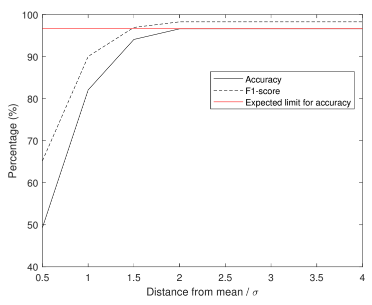
		TTR				
Length of Capture	**Attack timespan**	μ (s)	**Mean Variation (%)**	μ_{Attack} (s)	μ_{Normal} (s)	**Normal / Attack Variation (%)**
0.5 h	1 min	0.2094	1.2863	0.2172	0.2091	3.8737
0.5 h	5 min	0.2104	1.7700	0.2171	0.2091	3.8259
0.5 h	15 min	0.2132	3.1244	0.2182	0.2083	4.7528
1 h	1 min	0.2092	1.1896	0.2164	0.2091	3.4912
1 h	5 min	0.2097	1.4314	0.2181	0.2089	4.4040
1 h	15 min	0.2111	2.1086	0.2186	0.2087	4.7437
1 h	30 min	0.2132	3.1244	0.2175	0.2089	4.1168
Mean Value		**0.2109**	**2.0049**	**0.2176**	**0.2089**	**4.1726**
Standard Deviation		**0.0017**	**0.8249**	**0.0008**	**0.0003**	**0.4819**

In the passive reconnaissance attack, a negative variation is consistently observed, i.e., the mean time for response during an attack is lower than the for a normal state, which does not comply with the assumption that a delay in the response is always introduced by a MitM attack. This negative variation could be the result of the path taken by the packets through the middle man being faster or less congested than the direct path.

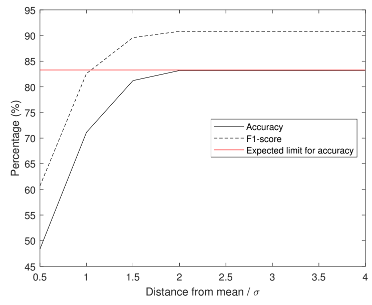
In the data modification attack, a positive and significant variation is observed, which was expected. As a result, this type of attacks my be detected using a technique involving the Time-To-

Response. Three approaches are possible for the classification procedure regarding this attack: a direct classification of all the packets that surpass a given distance from the expected mean value; a classification of the datasets using machine-learning algorithms; and a sliding-window analysis. In all the analysis made, the reduced dataset mentioned in Chapter 4 was used. Due to time constraints, the analysis was only made for the first of the three captures made.

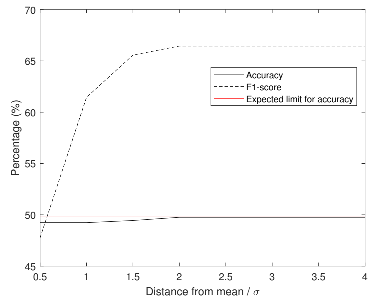
Regarding the direct classification procedure, various distances to the expected mean value of the TTR. In Figures 5.5 and 5.6, the accuracy and F1-score of the procedure are shown when varying that distance. When the distance is big enough, all of the packets will be marked as safe, which means the accuracy of the procedure will tend to the percentage of safe packets within a dataset, which is marked in the figures.



(a) 1 minute of attack.

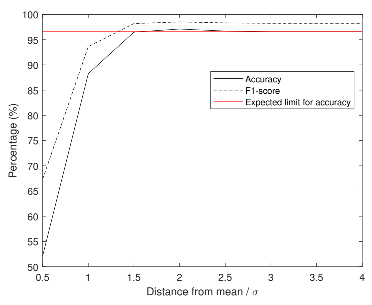


(b) 5 minute of attack.

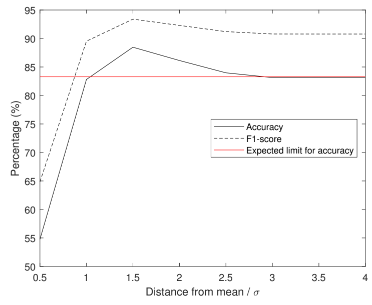


(c) 15 minute of attack.

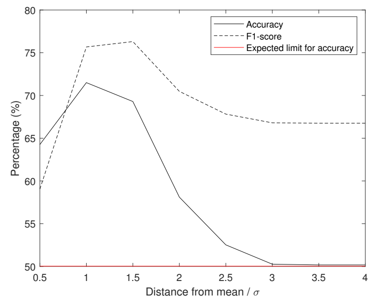
Figure 5.5: Accuracy (full line) and F1-score (dashed line) of the direct classification of the passive reconnaissance attack in a half hour capture.



(a) 1 minute of attack.



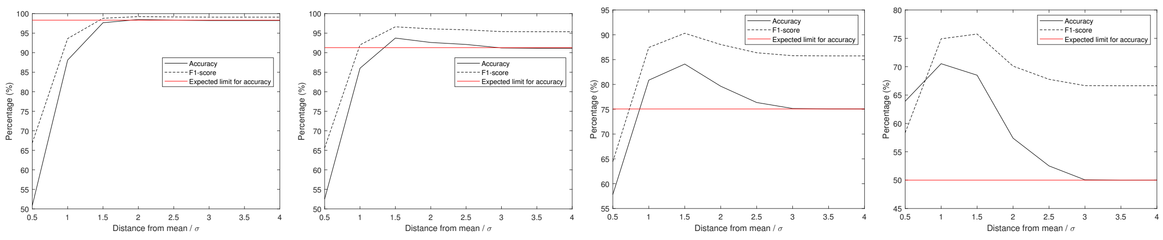
(b) 5 minutes of attack.



(c) 15 minutes of attack.

Figure 5.6: Accuracy (full line) and F1-score (dashed line) of the direct classification of the data modification attack in a half hour capture.

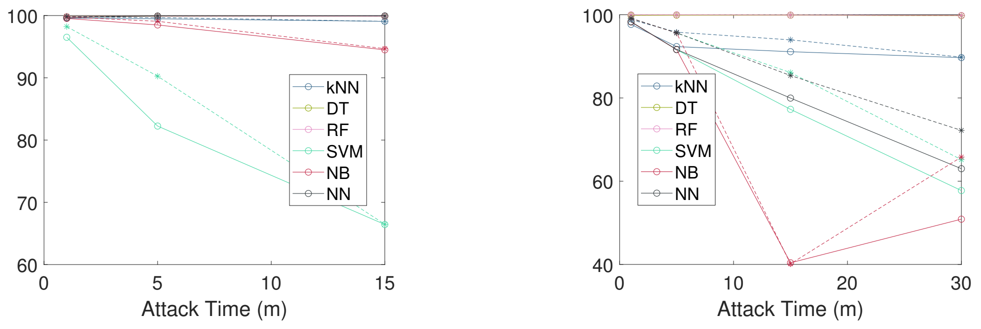
As expected, the performance of the direct classification procedure for the passive reconnaissance attack never surpasses the proportion of safe vs unsafe packets in the capture, but in the data modification attack the performance indicates that the attack is being correctly identified. Although the performance degrades as the proportion of the attack increases, the performance difference to the weight of safe/unsafe packets increases significantly. Another trend that is visible is that the optimal distance is between 1 and 2 σ, while distances bigger than 2.5σ present performances almost near the expected limit. When visualizing Figure 5.7, the same trends are visible.



(a) 1 minutes of attack. (b) 5 minutes of attack. (c) 15 minutes of attack. (d) 30 minutes of attack.

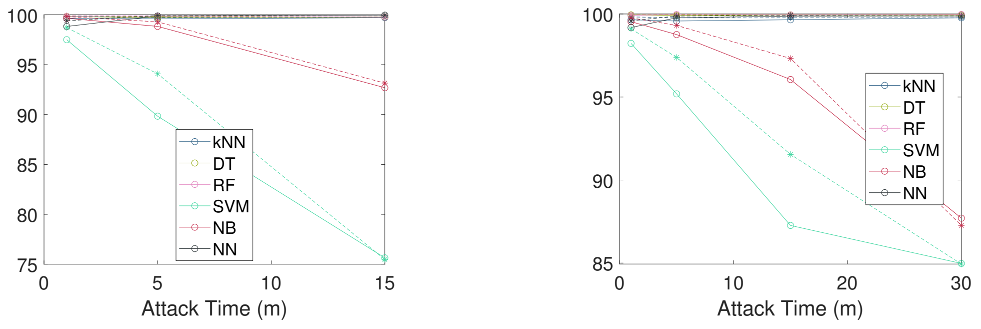
Figure 5.7: Accuracy (full line) and F1-score (dashed line) of the direct classification of the data modification attack in a one hour capture.

Regarding the application of machine learning algorithms on the classification, the default configuration of the algorithms used in the first experience was used. The datasets were greatly reduced, containing no information about the headers of the protocol stack, but containing the TTR information of the packets. In Figures 5.8 and 5.9, the performance of the classifiers is compared for the passive reconnaissance and data modification attacks, respectively,



(a) 0.5h of capture. (b) 1h of capture.

Figure 5.8: Accuracy (full line) and F1-score (dashed line) of the implemented classifiers for the passive reconnaissance attack.

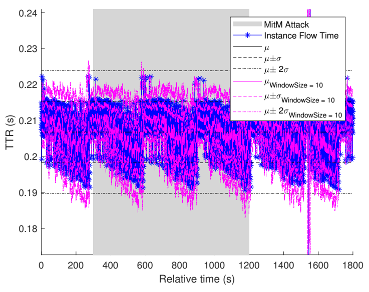


(a) 0.5h of capture. (b) 1h of capture.

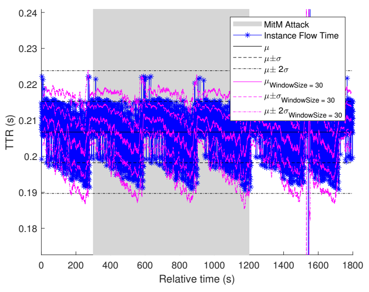
Figure 5.9: Accuracy (full line) and F1-score (dashed line) of the implemented classifiers for the data modification attack.

Analysing the results for the passive reconnaissance attack, the performance tends to degrade with the increase in the proportion of the attack timespan within the capture and the increase in the capture length accelerates that degradation. However, due to time constraints, a deeper analysis was not possible to confirm those trends. Nevertheless, since differences in TTRs was not discernible for this attack, these results are not able to be considered as reliable. The results obtained for the data modification attack, show performance degradations for the SVM and NB algorithms, while an increase in performance is discernible for the remainder of the algorithms. Again, given the a deeper analysis was not possible, these results still require further analysis.

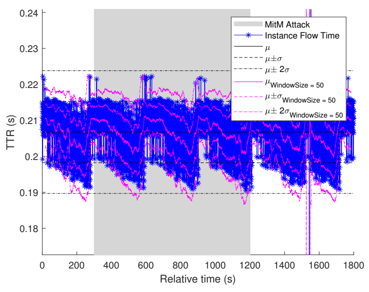
Regarding the analysis with a sliding-window, the first step taken regarded the effect of the window's size in the TTR measured. Figures 5.10 and 5.11 present the values for the Time-To-Response (TTR) of a capture made of the passive reconnaissance and data modification attacks, respectively.



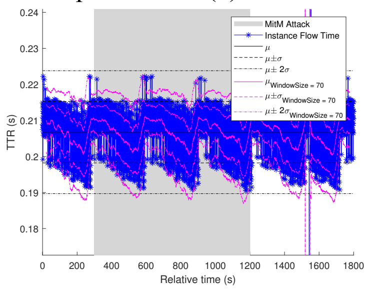
(a) Window Size of 10 packets.



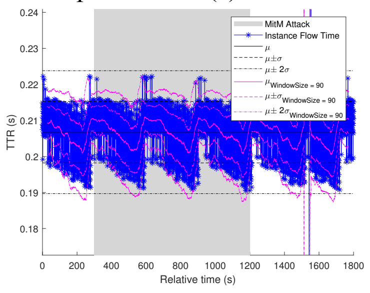
(b) Window Size of 30 packets.



(c) Window Size of 50 packets.



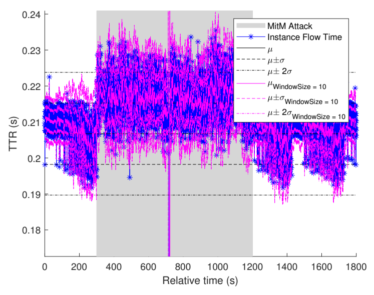
(d) Window Size of 70 packets.



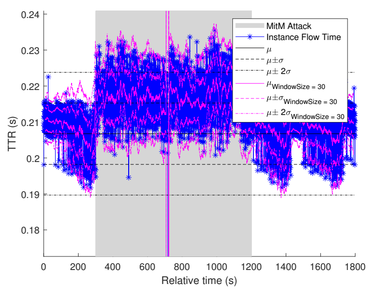
(e) Window Size of 90 packets.

Figure 5.10: TTR measured by sliding windows of various sizes for the passive reconnaissance attack.

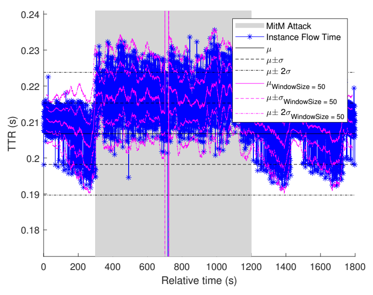
As had happened previously, no discernible variation happens to the TTR on the passive reconnaissance attack. Nevertheless, it is possible to visualize the effect that varying the size of the sliding window has on the values measured, specifically that bigger window's sizes smoothen the crisp variations of the measurements, comparable to the effects of a low-pass filter. This effect is also discernible in the data modification attack. This technique allows for the adaptability of the detection capabilities, since small long-term variations can become part of the expected values of the TTR and avoid the recalculation of the baseline when the system organically changes, while maintaining its capability of detecting attacks, since they would introduce a sudden change to the values.



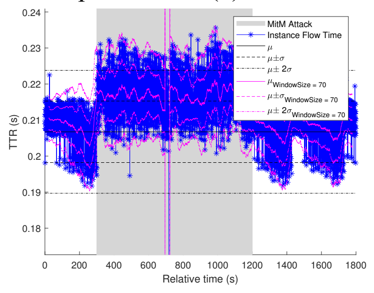
(a) Window Size of 10 packets.



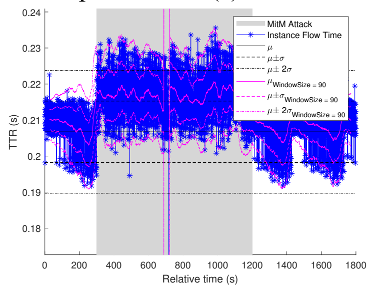
(b) Window Size of 30 packets.



(c) Window Size of 50 packets.



(d) Window Size of 70 packets.



(e) Window Size of 90 packets.

Figure 5.11: TTR measured by sliding windows of various sizes for the data modification attack.

Its worth noting that while the detection of MitM attacks is of great concern, the detection of sudden variances in the system is itself of great importance, since it may represent an anomaly within the system, which should be immediately assessed.

Chapter 6. Conclusions and Future Work

The scientific community researching security concerns on Cyber-Physical Systems (CPSs) has been focused on presenting newer algorithms and techniques or domain-adaptation of those algorithms from the much more researched field of securing common communication infrastructures. However, unlike the latter research, the community does not have common datasets from which it can compare their results, mostly due to the confidentiality of these systems. This work constitutes a step to providing an insight on the frailties of machine learning algorithms which may lead to proper generalization of those techniques and, consequentially, real-world application.

The effects of varying the attack and the capture timespans, when using different algorithms and attacks, were studied. It was inferred that, although small attack timeframes provide apparently good results, the classification accuracy starts to decrease as they grow in size and the imbalance of data starts to diminish. Once the data imbalance is overcome, the results start improving again. The overfitting problem is also detected and discussed. A study of hyperparametrization tuning of algorithms is also made, where it is inferred that simple variations on the parametrization of the algorithms may (or may not) have significant performance changes in the classification procedure.

This work also provides an insight at a generalized detection framework for Man-in-the-Middle attacks, based on the Time-To-Response of packets which is inherently longer when a third party gets between the traffic of two hosts in a network. This approach proved capable of detecting the attacks when data modification was applied, but was incapable when direct forwarding was applied, since the variation was not sufficient considering the natural variation of the network.

Considering the results obtained in this work, one of the possibilities of future work is the analysis of other common classification algorithms, which should analyse a wider set of attacks. Hyperparametrization tuning should also be expanded to more algorithm since time constraints prevented a deeper analysis in this dissertation. Future developments of this work should also involve an analysis of how the feature selection process may affect both the time required to create the models for detection and the resulting classification performance and generalization capabilities.

Bibliography

[1] Alfonso Valdes et al. ‘Intrusion Monitoring in Process Control Systems’. In: (2009), pp. 1–7.

[2] Igor Nai Fovino et al. ‘An experimental investigation of malware attacks on SCADA systems’. In: *International Journal of Critical Infrastructure Protection* (2009). ISSN: 18745482. DOI: `10.1016/j.ijcip.2009.10.001`.

[3] Stamatis Karnouskos. *Stuxnet Worm Impact on Industrial Cyber-Physical System Security*. Tech. rep. URL: `https://papers.duckdns.org/files/2011_IECON_stuxnet.pdf`.

[4] Christopher Johnson, Glasgow Ac and Eduardo Di Monte. *Communication network dependencies for ICS / SCADA Systems*. December. 2016. ISBN: 9789292041922.

[5] Paulo Tabuada. ‘Cyber-Physical Systems : Position Paper’. In: *NSF Workshop on Cyber-Physical Systems* (2006), pp. 1–3.

[6] K. Stouffer, J. Falco and K. Kent. ‘Guide to Industrial Control Systems (ICS) Security Recommendations of the National Institute of Standards and Technology’. In: *Nist Special Publication* 800.82 (2008). DOI: `http://dx.doi.org/10.6028/NIST.SP.800-82r1`.

[7] R. Braden. ‘Requirements for Internet Hosts - Communication Layers’. In: (). URL: `https://tools.ietf.org/html/rfc1122#page-8`.

[8] D. Plummer. ‘An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware’. In: (). URL: `https://tools.ietf.org/html/rfc826`.

[9] Luís Rosa et al. ‘Attacking SCADA systems: A practical perspective’. In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, May 2017, pp. 741–746. ISBN: 978-3-901882-89-0. DOI: `10.23919/INM.2017.7987369`. URL: `http://ieeexplore.ieee.org/document/7987369/`.

[10] Bo Chen et al. ‘Implementing attacks for modbus/TCP protocol in a real-time cyber physical system test bed’. In: *Proceedings - CQR 2015: 2015 IEEE International Workshop Technical Committee on Communications Quality and Reliability*. 2015. ISBN: 9781479985753. DOI: `10.1109/CQR.2015.7129084`.

[11] *Ettercap Home Page*. URL: `https://ettercap.github.io/ettercap/`.

[12] *Wireshark · Go Deep*. URL: `https://www.wireshark.org/`.

[13] *libmodbus.org*. URL: `http://libmodbus.org/`.

[14] J. Postel. *Internet Control Message Protocol*. Tech. rep. URL: `https://tools.ietf.org/html/rfc792`.

[15] *Modicon Modbus Protocol Reference Guide*. Tech. rep.

[16] Peter Huitsing et al. 'Attack taxonomies for the Modbus protocols'. In: *International Journal of Critical Infrastructure Protection* 1.C (2008), pp. 37–44. ISSN: 18745482. DOI: `10.1016/j.ijcip.2008.08.003`. URL: `http://dx.doi.org/10.1016/j.ijcip.2008.08.003`.

[17] IDA Modbus. 'Modbus Messaging on TCP/IP Implementation Guide v1. 0a'. In: *... Grafton, Massachusetts (www. modbus. org/specs. php ...* (2004). URL: `http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:MODBUS+MESSAGING+ON+TCP/IP+IMPLEMENTATION+GUIDE#0`.

[18] Javier Jiménez Díaz and Robert Vandenbrink. 'Using SNORT® for intrusion detection in MODBUS TCP/IP communications'. In: *SANS Institute Reading Room* (2011).

[19] Rose Tsang. 'Cyberthreats, vulnerabilities and attacks on SCADA networks'. In: *University of California, Berkeley, Working Paper ...* (2010), pp. 1–23. URL: `http://gspp.dreamhosters.com/iths/Tsang_SCADA%20Attacks.pdf`.

[20] A. Shahzad et al. 'The SCADA review: System components, architecture, protocols and future security trends'. In: *American Journal of Applied Sciences* 11.8 (2014), pp. 1418–1425. ISSN: 15543641. DOI: `10.3844/ajassp.2014.1418.1425`.

[21] P. S. M. Pires and L. A. H. G. Oliveira. 'Security aspects of SCADA and corporate network interconnection: An overview'. In: *Proceedings of International Conference on Dependability of Computer Systems, DepCoS-RELCOMEX 2006* (2007), pp. 127–134. ISSN: 0-7695-2565-2. DOI: `10.1109/DEPCOS-RELCOMEX.2006.46`.

[22] Theodore J. Williams. *The Purdue Enterprise Reference Architecture and Methodology (PERA)*. Tech. rep. URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.194.6112&rep=rep1&type=pdf`.

[23] *Russia accused of unleashing cyberwar to disable Estonia*. URL: `https://www.theguardian.com/world/2007/may/17/topstories3.russia`.

[24] Gaoqi Liang et al. 'A Review of False Data Injection Attacks Against Modern Power Systems'. In: *IEEE TRANSACTIONS ON SMART GRID* 8.4 (2017). DOI: `10.1109/TSG.2015.2495133`.

[25] Zakarya Drias, Ahmed Serhrouchni and Olivier Vogel. 'Taxonomy of attacks on industrial control protocols'. In: *International Conference on Protocol Engineering, ICPE 2015 and International Conference on New Technologies of Distributed Systems, NTDS 2015 - Proceedings* (2015). ISSN: 18745482. DOI: 10.1109/NOTERE.2015.7293513.

[26] Carlos Queiroz et al. 'Building a SCADA security testbed'. In: *NSS 2009 - Network and System Security*. 2009. ISBN: 9780769538389. DOI: 10.1109/NSS.2009.82.

[27] *Hping - Active Network Security Tool*. URL: http://www.hping.org/.

[28] Estefania Etcheves Miciolino et al. 'Communications network analysis in a SCADA system testbed under cyber-attacks'. In: *2015 23rd Telecommunications Forum, TELFOR 2015*. 2016. ISBN: 9781509000548. DOI: 10.1109/TELFOR.2015.7377479.

[29] *Nping - Network packet generation tool / ping utiliy*. URL: https://nmap.org/nping/.

[30] Lorena Cazorla, Cristina Alcaraz and Javier Lopez. 'Cyber Stealth Attacks in Critical Information Infrastructures'. In: *IEEE Systems Journal* may 1932 (2016), pp. 1–15. ISSN: 19379234. DOI: 10.1109/JSYST.2015.2487684.

[31] Ahmed Patel, Qais Qassim and Christopher Wills. 'A survey of intrusion detection and prevention systems'. In: *Information Management & Computer Security* (2010). ISSN: 0968-5227. DOI: 10.1108/09685221011079199.

[32] Bonnie Zhu. 'SCADA-specific Intrusion Detection/Prevention Systems: A Survey and Taxonomy'. In: *Network* (2010), pp. 1–16. URL: http://www.cse.psu.edu/~smclaugh/cse598e-f11/papers/zhu.pdf.

[33] Chih Fong Tsai et al. 'Intrusion detection by machine learning: A review'. In: *Expert Systems with Applications* 36.10 (2009), pp. 11994–12000. ISSN: 09574174. DOI: 10.1016/j.eswa.2009.05.029. URL: http://dx.doi.org/10.1016/j.eswa.2009.05.029.

[34] Pedro Domingos. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. 2015, p. 354. ISBN: 9780465065707. DOI: unmCentennialLowerLevel2Q387.D662015. URL: http://www.amazon.com/The-Master-Algorithm-Ultimate-Learning-ebook/dp/B012271YB2/ref=zg_bs_3887_6.

[35] Corinna Cortes and Vladimir Vapnik. 'Support-Vector Networks'. In: *Machine Learning* (1995). ISSN: 15730565. DOI: 10.1023/A:1022627411411.

[36] N. S. Altman. 'An introduction to kernel and nearest-neighbor nonparametric regression'. In: *American Statistician* (1992). ISSN: 15372731. DOI: 10.1080/00031305.1992.10475879.

[37] S. P. Luttrell. ‘Partitioned mixture distribution: an adaptive Bayesian network for low-level image processing’. In: *Vision, Image and Signal Processing, IEE Proceedings* - (1994). ISSN: 1350-245X. DOI: `10.1049/ip-vis:19941316`.

[38] L. Breiman et al. *Classification and Regression Trees*. 1984. ISBN: 0412048418. DOI: `10.1371/journal.pone.0015807`.

[39] C. M. Bishop. ‘Neural networks for pattern recognition’. In: *Journal of the American Statistical Association* (1995). ISSN: 01621459. DOI: `10.2307/2965437`.

[40] Sylvain Arlot and Alain Celisse. ‘A survey of cross-validation procedures for model selection’. In: *Statistics Surveys* 4 (2010), pp. 40–79. ISSN: 1935-7516. DOI: `10.1214/09-SS054`. URL: `http://projecteuclid.org/euclid.ssu/1268143839`.

[41] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008, p. 482. ISBN: 0521865719. URL: `https://nlp.stanford.edu/IR-book/information-retrieval-book.html`.

[42] S. Shitharth and D. Prince Winston. ‘An enhanced optimization based algorithm for intrusion detection in SCADA network’. In: *Computers and Security* 70 (2017), pp. 16–26. ISSN: 01674048. DOI: `10.1016/j.cose.2017.04.012`. URL: `https://doi.org/10.1016/j.cose.2017.04.012`.

[43] Lin Lil, Ming Wanl and Peng Zengl. ‘Industrial Communication Intrusion Detection Algorithm Based on Improved One-class SVM’. In: (2015), pp. 21–25.

[44] Chi-Ho Tsang and Sam Kwong. ‘Multi-Agent Intrusion Detection System in Industrial Network using Ant Colony Clustering Approach and Unsupervised Feature Extraction’. In: *2005 IEEE International Conference on Industrial Technology* (2005), pp. 51–56. DOI: `10.1109/ICIT.2005.1600609`. URL: `http://ieeexplore.ieee.org/document/1600609/`.

[45] Dayu Yang, Alexander Usynin and J. Wesley Hines. ‘Anomaly-based intrusion detection for SCADA systems’. In: *5th Intl. Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC&HMIT 05)* (2005), pp. 12–16. URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.1649&rep=rep1&type=pdf`.

[46] A. Valdes and S. Cheung. ‘Communication pattern anomaly detection in process control systems’. In: *Technologies for Homeland Security, 2009. HST ’09. IEEE Conference on* (2009), pp. 22–29. DOI: `10.1109/ths.2009.5168010`.

[47] Noam Erez and Avishai Wool. 'Control variable classi fi cation , modeling and anomaly detection in Modbus / TCP SCADA systems'. In: *International Journal of Critical Infrastructure Protection* 10 (2015), pp. 59–70. ISSN: 1874-5482. DOI: 10.1016/j.ijcip.2015.05.001. URL: http://dx.doi.org/10.1016/j.ijcip.2015.05.001.

[48] Niv Goldenberg and Avishai Wool. 'Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems'. In: *International Journal of Critical Infrastructure Protection* 6.2 (2013), pp. 63–75. ISSN: 18745482. DOI: 10.1016/j.ijcip.2013.05.001. URL: http://dx.doi.org/10.1016/j.ijcip.2013.05.001.

[49] Steven Cheung and Keith Skinner. 'Using Model-based Intrusion Detection for SCADA Networks'. In: *Science And Technology* 329.7461 (2006), pp. 1–12. ISSN: 09598138. DOI: 10.1136/bmj.329.7461.331.

[50] Igor Nai Fovino et al. 'Modbus/DNP3 state-based intrusion detection system'. In: *Proceedings - International Conference on Advanced Information Networking and Applications, AINA* (2010), pp. 729–736. ISSN: 1550445X. DOI: 10.1109/AINA.2010.86.

[51] Abdulmohsen Almalawi et al. 'An unsupervised anomaly-based detection approach for integrity attacks on SCADA systems'. In: *Computers and Security* 46 (2014), pp. 94–110. ISSN: 01674048. DOI: 10.1016/j.cose.2014.07.005. URL: http://dx.doi.org/10.1016/j.cose.2014.07.005.

[52] *UCI Machine Learning Repository: Water Treatment Plant Data Set*. URL: https://archive.ics.uci.edu/ml/datasets/water+treatment+plant.

[53] Man-ki Yoon and Gabriela F. Ciocarlie. 'Communication Pattern Monitoring : Improving the Utility of Anomaly Detection for Industrial Control Systems'. In: *Sent '14* February (2014), pp. 1–10. DOI: 10.14722/sent.2014.23012.

[54] Marco Caselli, Emmanuele Zambon and Frank Kargl. 'Sequence-aware Intrusion Detection in Industrial Control Systems'. In: *Proceedings of the 1st {ACM} {Workshop} on {Cyber}-{Physical} {System} {Security}* (2015), pp. 13–24. DOI: 10.1145/2732198.2732200. URL: http://doi.acm.org/10.1145/2732198.2732200.

[55] Mustafa Faisal, Alvaro A Cardenas and Avishai Wool. 'Modeling Modbus TCP for Intrusion Detection'. In: (2016), pp. –4.

[56] Rishabh Samdarshi, Nidul Sinha and Paritosh Tripathi. 'A triple layer intrusion detection system for SCADA security of electric utility'. In: *12th IEEE International Conference Electronics, Energy, Environment, Communication, Computer, Control: (E3-C3), INDICON 2015* (2016), pp. 1–5. DOI: 10.1109/INDICON.2015.7443439.

[57] Raymond C. Borges Hink et al. ‘Machine learning for power system disturbance and cyber-attack discrimination’. In: *7th International Symposium on Resilient Control Systems, ISRCS 2014* (2014). DOI: `10.1109/ISRCS.2014.6900095`.

[58] ‘MODBUS Application Protocol Specification V1.1b3 Modbus’. In: (2012). URL: `http://www.modbus.org`.

[59] *Rapid SCADA | Free, Open Source, Full Featured SCADA Software*. URL: `https://rapidscada.org/`.

[60] *SMOD*. URL: `https://github.com/enddo/smod`.

[61] *tshark - The Wireshark Network Analyzer 2.6.2*. URL: `https://www.wireshark.org/docs/man-pages/tshark.html`.

[62] *MATLAB - MathWorks - MATLAB & Simulink*. URL: `https://www.mathworks.com/products/matlab.html`.

[63] *Statistics and Machine Learning Toolbox - MATLAB*. URL: `https://www.mathworks.com/products/statistics.html`.

[64] *Deep Learning Toolbox - MATLAB*. URL: `https://www.mathworks.com/products/deep-learning.html`.