Maria Carmen Palacios Prados

NOVELTY DETECTION AND EARLY CLASSIFICATION OF MALICIOUS ACTIVITIES ON INDUSTRIAL CONTROL SYSTEMS

FINAL MASTER'S PROJECT

Directed by Dr. Sergio Gómez

Master's Degree in Computer Security Engineering and Artificial Intelligence

UNIVERSITAT ROVIRA i VIRGILI

Tarragona

2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

Industrial Control Systems (ICS) are a set of industrial processes that manage, direct and regulate the behaviour of other devices. In particular, these processes are vital to service critical infrastructure, such as communications, manufacturing, and energy. An attack on these infrastructures can pose a threat to the day to day of the states. Unfortunately, in recent years ICS have been subject to an increase in the number of attacks Although Network Anomaly Detection Systems (NADS) are capable of detecting existing and zero-day attacks, it is still not universally implemented in industry and real applications, since current systems produce high False Positive Rates (FPRs) and low Detection Rates (DRs). Consequently, anomaly detection is still under-utilized in the cybersecurity arena. However, the alternative technique, the detection of abuse, is limited by the fact that it only addresses known vulnerabilities. Therefore, there is a mandatory need for anomaly detection to be operational to increase the coverage of current Intrusion Detection Systems (IDS).

The goal of this master thesis is to apply machine learning and deep learning techniques that allow the normality space definition to address novelty detection and early classification of malicious activities on ICS. The rationale behind is that if the underlying infrastructure of malware samples is similar (e.g., as a result of being controlled by the same attacker, or by reusing code from another author), their behaviours or the order in which they perform certain actions would be similar. Specifically, the solution looks at similarities shown in industrial network traffic modelized by an enhanced feature set composed of simple, high-level features, extracted from the headers and payloads of the network packets. Regarding machine learning techniques, some traditional Machine Learning methods and Deep Neural Networks well suited for high-dimensional datasets have been implemented and tested. After several trails, the final solution combines an unsupervised anomaly detection technique called AutoEncoder (for the detection of unknown attacks) with Random Forest, a supervised machine learning that supports the detection of known attacks as well to reduce the false positive rate. This novel machine learning pipeline successfully detects zero-day and specific attacks since it achieves a precision of 0.998425, recall of 0.9607375 and f1-score of 0.9733375 while reduces to a negligible value the false positive rate.

The performance evaluations of this approach have been carried out using the benchmark dataset named as CICIDS2017. This dataset was created by the Canadian Institute for Cybersecurity (CIC) and the University of New Brunswick (UNB). This dataset accommodates a variety of up-to-date multistage attacks and intruder strategies in modern normal behaviours.

Keywords

Intrusion Detection, Machine Learning, Deep Learning, AutoEncoder, Random Forest

1. Introduction

1.1 Problem statement

Industrial Control Systems (ICS) are a set of industrial processes that manage, direct and regulate the behaviour of other devices [1]. In particular, these processes are vital to service critical infrastructure, such as communications, manufacturing, and energy. An attack on these infrastructures can pose a threat to the day to day of the states. These risks may generate disastrous consequences in industrial environments, and the related impacts may grow broader with the rise of Industry 4.0 paradigm. Nowadays, ICS are increasingly adopting Information Technology (IT) solutions to promote enterprise systems connectivity and remote access capabilities, but they still retain unique characteristics. It must be considered that logic executing in ICS has a direct effect on the physical world. The consequences associated with an unappropriated logic execution include significant risk to the health and safety of human lives and serious damage to the environment, as well as serious financial consequences such as production losses, negative impact to a nation's economy, and compromise of proprietary information [2]. In practise only when infrastructures have been affected by cyber-incidents that caused serious problems, the interest in the protection of industries has started to consider the logical aspects. One cause of this situation is the development of functionalities has prevailed over a (cyber)secure design in the vast majority of industrial control systems, which makes them highly vulnerable. And another important issue comes from the undergoing digital transformation which opens new threats that until now were not relevant in the OT environment (Operation Technologies).

The article "State of Industrial Cybersecurity 2018 in Spain" published by infoPLC shows the vulnerabilities that affect different equipment and industrial systems installed in many manufacturing plants. This article uses the data coming from bulletins about industrial control systems emitted by the early alert service of INCIBE. According to this source, 228 vulnerability alerts in industrial sectors were published during 2018 against 199 in 2017 (see next figure), affecting multipurpose devices/systems used in practically all industrial sectors, and what is worse, these numbers continue to rise [3].

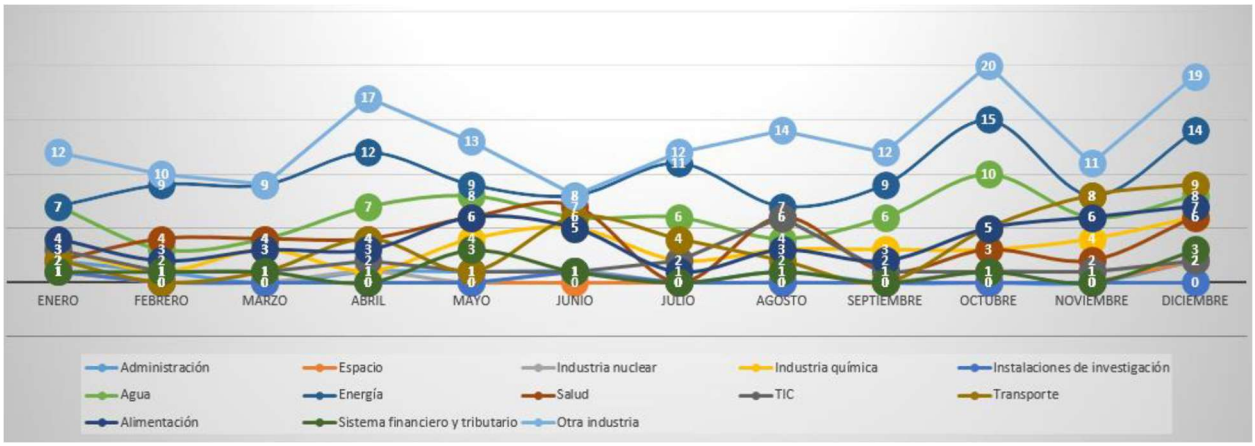


Figure 1. Industrial security incidents per sector in Spain [3]

Similarly, the report "Cybersecurity for Manufacturing" published by EEF in 2018 states that the 48% of manufacturers in the UK have suffered some kind of cyber-attack, around half of whom had also suffered financial loss as a result. Of course, this does not include those businesses who do not even realise that they have been subject to an attack. According to the same source, 91% of the manufacturers are investing or intend to invest in digitization, but 35% consider that cybervulnerability inhibits them from doing so fully [4].

From the perspective of cyber security, the European Cyber Security Organisation (ECSO) identifies 6 challenges which need to be addressed to ensure a reasonable level of security for Industry 4.0: 1) Safety-security convergence, 2) Secure Industrial IoT, 3) Intrusion/Anomaly detection on ICS, 4) Manage cyber physical threats, 5) Manage behavioural and organisational changes, 6) Ensure security throughout the value chain. Specifically, this thesis addresses the challenge 3 entitled as "Intrusion/Anomaly detection on Industrial Control Systems". According to ECSO, a mix of protocol-based and behaviour-based approaches is required to effectively detect cyber-attacks on ICS. They also highlight that detection techniques involving machine-learning may improve the detection rates and enable the detection of zero-day attacks [5].

A zero-day attack can be defined as "a traffic pattern of interest that, in general, has no matching patterns in malware or attack detection elements in the network" [6]. Many researchers have focused on addressing this issue. Although Network Anomaly Detection Systems (NADS) are capable of detecting existing and zero-day attacks (previously unseen), it is still not universally implemented in industry and real applications, since current systems produce high False Positive Rates (FPRs) and low Detection Rates (DRs). Consequently, anomaly detection is still under-utilized in the cybersecurity arena. However, the alternative technique, the detection of abuse, is limited by the fact that it only addresses known vulnerabilities. Therefore, there is a mandatory need for anomaly detection to be operational to increase the coverage of current Intrusion Detection Systems (IDS).

Under this topic, another main challenge is network intrusion detection has to be performed on streams of data that have a time-series evolution; this issue is generally tackle by performing a features' extraction procedure before the detection phase. The features' extraction phase consists of translating the informative content of time-series data into continuous variables [7]. This feature engineering phase is decisive for machine and deep learning models to learn appropriately the crucial information in order to achieve high performance.

1.2 Goals and challenges

The goal of this master thesis is to apply machine learning and deep leaning techniques that allow the normality space definition to address novelty detection and early classification of malicious activities on ICS. The rationale behind is that if the underlying infrastructure of malware samples is similar (e.g., as a result of being controlled by the same attacker, or by reusing code from another author), their behaviours or the order in which they perform certain actions would be similar. Specifically, the solution will look at similarities shown in industrial network traffic modelized by an enhanced feature set composed of simple, high-level features, extracted from the headers and payloads of the network packets.

The first challenge to face is the high imbalance between benign and malicious observations usually gathered in these environments. Therefore, it is paramount to reduce the false positive rate, or number of benign samples considered as attack, since only a reduced percentage of them can overwhelm the security engineers' daily work.

The second challenge is related to the discovery of zero-day attacks, which are advanced methods that combine unknown attacks. Taking in mind the nature of this problem, its solution requires the use of unsupervised learning methods, which unfortunately can produce either high false positive or low attack detection rates. Therefore, the provided system should be optimised to achieve high performance at detection time. On the other hand, it is advisable to reinforce it with other techniques which help to leverage the accurate results. This brings us to the third challenge which consists of the detection of

well-known attacks. This challenge should be faced by the application of supervised learning methods that will allow the detection of specific attacks. This is of vital importance for the security engineers as the countermeasures needed depend on the type of attack to contain.

And, finally, the fourth challenge is the problem of lack of datasets on real networks. Sometimes privacy concerns prevent sharing data while other times the real-world trends are not reflected as a side effect of their heavy anonymisation. Then, the first task will be to analyse and choose a public dataset which accomplishes the heavy requirements needed in this research such as the covering of heterogeneity of equipment, real traffic capture, modern attacks, and high-class imbalance.

From a holistic point of view, this solution can be considered as an evolutionary system. By covering a wide variety of intrusion detection techniques, it will be able to improve its performance as soon as it is available the feedback of security engineers about treated incidents.

1.3 Thesis organization

This thesis is organised as follows:

- Chapter 2 presents the background on the fields of intrusion detection systems and zero-day attacks. Therefore, it details several topics of interest such as relevant datasets publicly available, research on intrusion detection systems and machine learning techniques commonly used for zero-day attacks detection.
- Chapter 3 explains the innovative solution proposed in this work to address novelty detection and early classification of malicious activities in industrial environments. Therefore, it includes a description of the used dataset, a background on machine learning and deep learning, and the detailed description of the solution for anomaly detection based on a machine learning algorithms pipeline.
- Chapter 4 presents the experimental results and findings. Several techniques for the machine learning algorithms pipeline have been implemented and evaluated with the aim of detecting properly both new zero-day attacks and well-known attacks as soon as possible along with a very low false positive rate.
- Chapter 5 includes the list of key contributions provided by this thesis. Moreover, it concludes with some suggestions about research directions that could further improve the results achieved.

2. Research and related work

This chapter covers a background on the fields of intrusion detection systems and zero-day attacks. Firstly, details of the most relevant datasets publicly available are provided along with the dataset selected in this work and the corresponding rationale. Lastly, concepts related to intrusion detection systems topic and a deep description of the machine learning techniques used for zero-day attacks detection are included. Finally, it concludes with a section devoted to the selection of the machine learning (ML) techniques to be implemented in this research work.

2.1 Available datasets for Intrusion Detection Systems

This section provides a review of the datasets used in IDS research works that have been made publicly available since 1988, although the majority of them present some shortcomings [8].

DARPA (Lincoln Laboratory, 1998 - 99). DARPA includes activities such as send and receive mail, browse websites, send and receive files using FTP, the use of telnet to log into remote computers and perform work, send and receive IRC messages, and monitor the router remotely using SNMP. It contains attacks like DOS, guess password, buffer overflow, remote FTP, syn flood, Nmap, and rootkit. Unfortunately, DARPA presents issues related to the artificial injection of attacks and benign traffic and, what is worse, it is outdated for the effective evaluation of IDSs on modern networks in terms of attack types and network infrastructure [9].

KDD'99 (University of California, Irvine 1998 - 99), which was created by processing the tcpdump portion of the 1998 DARPA dataset. KDD99 includes more than twenty attacks such as neptune, buffer-overflow, rootkit, satan and teardrop among others. The records of normal and attack network traffic are merged in a simulated environment resulting in a dataset with a large number of redundant records that are studded with data corruptions that leads to skewed testing results [10]. Therefore, a new version of the KDD was created and called as *NSL-KDD* [11].

DEFCON (The Shmoo Group, 2000-2002), which includes several datasets. DEFCON8 contains port scanning and buffer overflow attacks, whereas the DEFCON10 dataset uses port scan and sweeps, bad packets, administrative privilege, and FTP by telnet protocol attacks. The main drawback is they mainly consist of intrusive traffic instead of normal background traffic [12].

CAIDA (Centre of Applied Internet Data Analysis, 2002-2016), which includes three datasets: 1) CAIDA OC48 contains different types of data observed on an OC48 link in San Jose; 2) CAIDA DDOS includes one-hour DDoS attack traffic split of 5-minute Packet CAPture (PCAP) files; and 3) CAIDA Internet 2016 contains passive traffic traces from CAIDA's Equinix-Chicago monitor on High-speed Internet backbone. This dataset is not considered as an effective benchmarking dataset due to their multiple shortcomings described in [13][14].

LBNL (Lawrence Berkeley National Laboratory and ICSI, 2004-2005), which is full header network traffic recorded at a medium-sized site. Therefore, it does not include payloads and any information which could identify an individual IP [15].

Kyoto (Kyoto University, 2009), which includes attacks directed against honeypots. It presents ten new features such as IDS Detection, Malware Detection, and Ashula Detection that are useful in NIDS analysis and evaluation. However, normal traffic is simulated by only DNS and mail traffic data. Therefore, it lacks false positives, which are important since they help to minimise the number of alerts [16][17][18].

ISCX2012 (University of New Brunswick, 2012), which includes two profiles. The Alpha profile carries out various multistage attack scenarios, while the Beta profile is the benign traffic generator that generates realistic network traffic with background noise. HTTP, SMTP, SSH, IMAP, POP3, and FTP protocols traces are included. However, the distribution of the simulated attacks is not based on real world statistics [13].

UNSW-NB15 dataset (Australian Centre for Cyber Security, 2015), which includes realistic normal activities and synthetic contemporary attack behaviour of live network traffic. This dataset represents nine categories of modern attack families such as Generic, Exploits, Fuzzers, DoS, Reconnaissance, Analysis, Backdoor, Shellcode and Worms. The dataset contains 49 features from packet headers and payload to allow the discrimination of the network packets either as normal or malicious [19].

CICIDS2017 (Canadian Institute for Cybersecurity, 2017), which is a valid dataset created by following a novel systematic approach based on two types of activity profiles. The dataset contains a variety of up to date multistage attacks such as Heartbleed and different types of DoS and DDoS attacks. Furthermore, a variety of modern protocols are included. CICIDS2017 fulfils the eleven important criteria [20] that are necessary for building a reliable benchmark dataset [21].

CSE-CIC-IDS2018 (Communications Security Establishment and Canadian Institute for Cybersecurity, 2018), which is appropriate for big data analysis and covers a wide range of attack types such as Brute-power, DoS, DDoS, Heartbleed, Botnet, Web assaults, and organization penetration from inside. This multi-class dataset is realistic since it shows a class imbalance where the 17% of the instances are anomalous [22].

2.1.1 Selection of the dataset

Nowadays, it is well-known that anomaly detection research suffers from a lack of available realistic and public datasets. Sometimes privacy concerns prevent sharing data while other times the real-world trends are not reflected as a side effect of their heavy anonymisation. For addressing such issues, the authors of [21] stablished the following 11 criteria that a dataset should meet:

- Complete computer network configuration that represents the real world, since several attacks have only revealed in a complete network with numerous PCs, servers, routers, and firewalls.
- Complete network traffic, which can be composed of real, pseudo-realistic, or synthetic traffic. Where the pseudo-realistic has partially the real-world traffic, such as having simulated human behaviour traffics with real attack scenarios.
- Labelled dataset. Not only labels have to be accurate but also more informative than merely “benign” or “malicious” annotations.
- Complete interaction for the right interpretation of the results. Therefore, having all network interactions such as within or between internal LANs is vital.
- Complete traffic capture to allow the adequate calculation of the false positive percentage.
- Available protocols. A complete dataset should have a great diversity of protocols along with normal and anomalous traffic.
- Attack diversity since it is advisable to include modern attacks types.
- Anonymity. Privacy concerns should not force to anonymize payloads since they are required by some detection mechanisms, especially deep packet inspection (DPI).
- Heterogeneity. While a homogeneous dataset using a single source type can be useful for analysing a specific type of detection systems, network traffic logs from various types of source could be used for covering all aspects of the detection process.

- Feature set. The dataset should allow the extraction of features from different type of data sources such as traffic or logs.
- Metadata. Insufficient information about the network configuration, operating systems for attacker and victim machines, attack scenarios, and other vital information detracts from the usefulness of a dataset.

According to these evaluation criteria, the best dataset candidates for IDS research are UNSW-NB15, CICIDS2017 and CSE-CIC-IDS2018. Finally, the selection has been CICIDS2017 dataset since it clearly fulfils all 11 evaluation criteria. On the one hand, it should be considered that CICIDS2017 covers 14 contemporary attack types while UNSW-NB15 only 9. On the other hand, although CSE-CIC-IDS2018 shows class imbalance and is structurally similar to CICIDS2017, CSE-CIC-IDS2018 was prepared from a much larger network of simulated client-targets and attack machines in order to address big data. Therefore, the final decision has been made taking in mind that CICIDS207 also includes a specific set of data with only normal network activities, which fully support the application of unsupervised and supervised anomaly-based detection techniques.

2.2 Intrusion Detection Systems

[23] Intrusion Detection Systems (IDS) detect anomalies and intrusions. Since anomalies in data translate to significant (and often critical) actionable information about intrusions. These systems implement security mechanisms that can be hardware and/or software designed to control intrusive activities on hosts and a defined red perimeter. In case of Passive IDSs, as it is name suggests, notification messages are triggered when malicious activities are detected. Whereas Active IDSs react immediately taking actions such as rejecting malicious traffic or closing an open port.

Generally, IDSs are classified as:

- Network-based IDSs (NIDSs). A NID sniffs network traffic in real time, analyses it and can also perform defensive actions. Such automatic actions can be either trigger alerts to the security engineer or active reactions like dropping malicious traffic.
- Host-based IDSs (HIDSs). A HID inspects log files of the operating system, services and applications installed on hosts and emits alerts when suspicious activities are detected.
- Hybrid IDSs provide capabilities of NIDS and HIDS for the gathering and analysis of heterogeneous cyber threat information.

And, in recent years, the following four main detection techniques are used to detect intrusions:

- Knowledge-based detection (also known as Signature or Misuse), which looks for matches of incoming patterns in its database of known threatening signatures for accurate detection. It cannot detect zero-day attacks.
- Anomaly-based detection (also known as Profile or Heuristic), which builds statistical patterns representing normal behaviours of the supervised system to be able to identify abnormal activity at production time. An anomaly-based IDS can detect zero-day attacks but generates numerous false positives.
- Multi-agent-based detection, in which IDS agents operate in centralized, distributed, collaborative and cooperative manner to detect advanced persistent threats and multi-stage attacks.
- Hybrid detection, which is based on a combination of the previous detection methods and leverages multiple detection techniques to increase the detection accuracy.

Regarding anomaly detection, a great variety of techniques have been described, implemented and tested in the literature, as it is shown in the next figure. Anomaly-based detection, as a behavioural based intrusion detection system, observes changes in normal activity within a system by building a profile which is monitored. This baseline profile is generated over a predefined period of time while the system is behaving under normal conditions. Therefore, the most important advantage is that it offers the ability to detect attacks never seen. Anomaly detection can be categorised into self-learning and programmed main classes, depending on the method used to create the normal or baseline profile of a system [24].

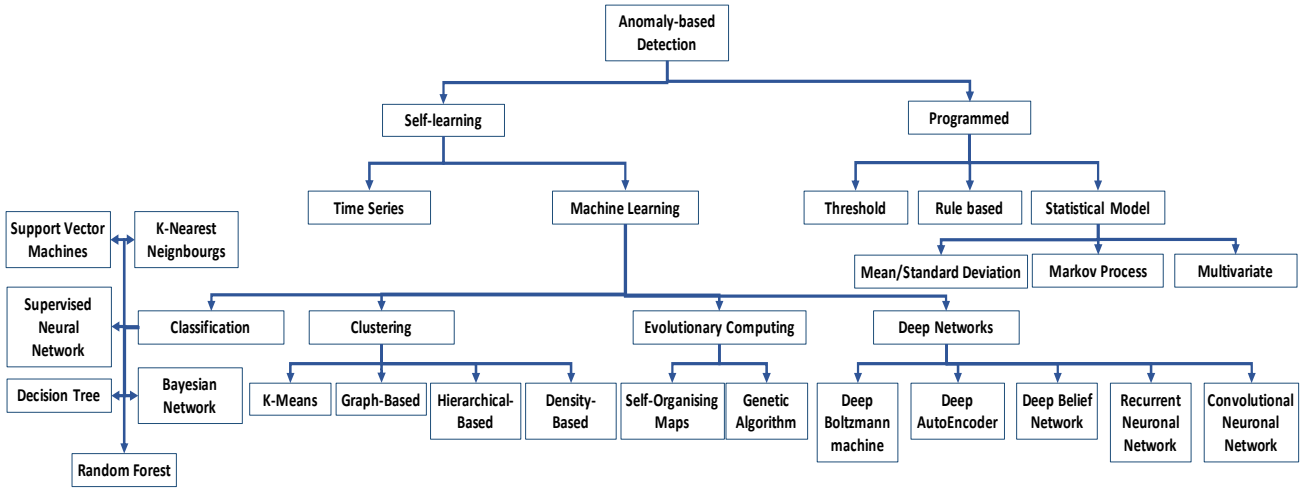


Figure 2. Anomaly-base detection techniques (adapted from [23][24])

Where programmed category refers to models that need a person to manually teach it to detect changes in system's behaviour. Hence, the human being is who decides the extent of abnormal behaviours in the system and flags them as an intrusion threat. The programmed models are grouped into three categories: threshold, simple rule based and statistical models. Threshold models can be considered as the simplest programmed descriptive statistical detector [25]. By analysing statistical data, a user can programmatically trigger an alarm when a statistical variable is under or over the pre-defined threshold. Here, careful selection of the threshold is required to minimise the false alarm rate. Since a threshold defined too high leads to a risk of missing an alarm when a malicious action is ongoing. Simple rule-based model consists in monitoring the events generated whenever a rule is fired when an abnormal or unexpected behaviour is detected. This model fails to detect a threat that is not programmed as a rule [26]. Statistics model collects data in a profile. Analysing the profile of normal statistical behaviour helps decide whether an activity is normal or abnormal. The system is continuously calculating the distance vector from the observed traffic to the normal profile. And an alarm is raised when such distance is great enough.

Under self-learning category, a model is trained with the network traffic gathered over a period of time in order to learn how the underlying processes behave. Time series and machine learning techniques belong to this category. As its name suggests, time series model considers the chronological order of appearance of observations in intervals of uniformity. Time series flags changes in normal behaviour. Therefore, it is an effective model when attacks are sequential over a period of time [27]. However, this model is more costly computationally [28]. On the other hand, machine learning techniques (ML) can be defined as a set of methods that can automatically detect patterns to predict future data trends. ML can detect current, new and subtle attacks without extensive human-based training or intervention. The ML techniques can be classified as: classification, which is a supervised approach applied for categorical

output values where data can be separated into specific classes; clustering, which is an unsupervised learning approach that targets homogeneous data clusters such that the most similar data belongs to the same cluster; evolutionary computing that studies and designs biologically-inspired algorithms (e.g., gene reproduction); and deep learning (DL), which is inspired by the human brain and allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

In the literature there are relevant research works where such ML algorithms have been applied in the intrusion detection domain and have achieved promising results. For example, authors of [21] used a Random Forest Regressor to determine the best set of features to detect each attack family. Later on, the performance of these features was examined by the applications of different algorithms such as K-Nearest Neighbour (KNN), Adaboost, Multi-Layer Perceptron (MLP), Naïve Bayes, Random Forest (RF), Iterative Dichotomiser 3 (ID3) and Quadratic Discriminant Analysis (QDA).

In [29], a small portion of the CICIDS2017 dataset samples were used to evaluate an intrusion detection system that implemented a genetic algorithm (GA) for the selection of features and multiple Support Vector Machines (SVM) for classification. In fact, this system combined multiple SVM classifiers, which were ordered based on the severity of the attacks. Since every classifier was trained on a different feature set in order to be able to discover a specific attack category.

Moreover, the authors of [30] applied two classifiers, a Multi-Layer Perceptron (MLP) and a Convolutional Neural Network (CNN), that used the PCAP files of the CICIDS2017 dataset. In this study, specified network packet header features were selected. Besides, Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) models are also used to detect intrusions [31].

In addition, in [32] an intrusion detection system for wireless sensor networks was developed by using Deep Auto-Encoder (DAE) as the basic classifier to detect the specific attack types. On the other hand, the authors in [33] proposed SU-IDS; a semi-supervised and unsupervised network intrusion detection system that used an auto-encoder-based framework. The framework augmented the usual clustering (or classification) loss with an auxiliary loss of the auto-encoder, and thus achieving a better performance.

2.2.1 Zero-day attacks detection based on Machine Learning

Nowadays, detecting zero-day attacks (previously unseen) is one of the main open issues of Intrusion Detection Systems. In addition, current IDSs suffer from high false-positive rates, thus limiting their practical adoption. As a consequence, zero-day attacks remain undetected in practice, which escalate their negative impacts [34].

A zero-day attack can be defined as “a traffic pattern of interest that, in general, has no matching patterns in malware or attack detection elements in the network” [6]. Many researchers have focused on addressing this issue. For instance, the authors of [35] analyse the impact of zero-day attacks in real-world. They highlighted that zero-day attacks are more frequent than suspected, demonstrating that out of their 18 analysed attacks, 11 were previously unknown zero-day attacks. Furthermore, they showed that zero-day attacks can exist for a long period of time (average of 10 months) before they are detected, being able to compromise systems during long periods [35].

On the other hand, a distributed diagnosis system to detect zero-day attacks in Internet of Things (IoT) networks was proposed in [36]. In the research work [37], the authors propose a Bayesian probabilistic model to detect zero-day attack paths. In addition, the attacks are visualised in a graph-like structure. While [38] evaluated different supervised ML techniques: Decision Tree, Random Forest, K-Nearest Neighbour, Multi-Layer Perceptron, Quadratic Discriminant Analysis, and Gaussian Naïve Bayes

classifiers. However, in this study, it is not clarified neither how such ML techniques are trained on benign traffic solely to be utilised for unknown attacks detection nor how zero-day attacks are simulated and detected. In other work was presented Kitsune, a plug and play NIDS capable of online attacks detection for simple network devices. The Kitsune's core algorithm (KitNET) implemented a group of AutoEncoders to distinguish normal from abnormal traffic patterns [39]. In [40] was proposed an approach that combined deep and shallow learning, which can correctly analyse a wide range of network traffic. They joined the power of stacking a Non-symmetrical Deep Auto Encoder (NDAE) and a Random Forest (RF). They made an evaluation on the KDDCup 1999 and NSLKDD datasets. Unfortunately, they model did not present the capability of handling zero-day attacks [40]. In addition, the authors of [34] proposed to use Deep Learning to serve the outlier detection purpose of new intrusions and zero-day attacks while minimising the false-positive rates and achieving high positive rates. Their main contribution was a novel use for autoencoders as a zero-day IDS.

Moreover, transfer learning technique can be used to detect zero-day attacks. For example, transfer learning was used to map the connection between known and zero-day attacks in [41]. And, [42] presented the application of deep transductive transfer learning to detect zero-day attacks.

Furthermore, ML can be used to address zero-day malware detection. For example, the effectiveness of several ML techniques (Support Vector Machine, Naïve Bayes, Multi-Layer Perceptron, Decision Trees, k-Nearest Neighbour and Random Forests) to detect zero-day malware were evaluated in [43], while Deep-Convolutional Generative Adversarial Network (DCGAN) were proposed in [44].

2.2.2 Selection of the ML techniques for zero-day and well-known attacks detection

The focus this master's thesis is put on the design and developing of an evolutive anomaly-based detection system that addresses the detection of zero-day attacks along with well-known attacks. The key challenge of anomaly detection in IDSs is the huge volume of data. Where the anomaly detection techniques need to be computationally efficient to handle these large sized inputs in real-time. Another issue which arises because of the large sized input is the false alarm rate, which can make analysis overwhelming for a security analyst. Moreover, the datasets are usually imbalanced since normal behaviour samples are usually available in this domain, while intrusions are not. The detection approaches described in the state-of-the-art have achieved encouraging results in most categories of intrusion detection data, but there are still problems with detection fails and low detection rates of some minority classes. Therefore, the proposed solution will combine an unsupervised anomaly detection technique called AutoEncoder (to be able to detect unknown attacks) with Random Forest, a supervised machine learning that will support the detection of known attacks as well to reduce the false positive rate. Since RF is one of the best algorithms for intrusion detection as it has been demonstrated by current comparative researches.

3. A solution for novelty detection and early classification of malicious activities

This chapter is devoted to the detailed description of the innovative solution proposed in this work to address novelty detection and early classification of malicious activities in industrial environments.

Firstly, a description of the CICIDS2017 dataset is provided. This dataset has been chosen since it is an up-to-date dataset that accommodates new attacks and intruder strategies.

Secondly, some background on machine learning and deep learning is introduced. Specifically, the focus is put on Naïve Bayes, Support Vector Machines, Random Forest and AutoEncoders algorithms.

Finally, the methodology followed, and the detailed description of the proposed solution is included. Its main stages are: pre-processing of the raw data, cleaning of the dataset, detection and deletion of outliers, feature engineering phase, the creation of the baseline and zero-day attacks datasets and implementation of the solution for anomaly detection based on a machine learning algorithms pipeline.

3.1 CICIDS2017 dataset

The dataset CICIDS2017 was created by the Canadian Institute for Cybersecurity (CIC) and the University of New Brunswick (UNB). It is an up-to-date dataset that accommodates new attacks and intruder strategies. CICIDS2017 was created in an emulated network environment for a period of five days, from Monday to Friday. This open dataset includes not only raw network traffic in packet format (provided as PCAP files), but also pre-processed network data as bidirectional network flows (provided as CSV files) obtained with the CICFlowMeter tool and described by 85 features. Since this pre-processed netflow data can be more easily fed into machine learning pipeline, it will be the dataset version used in this master's thesis.

In the creation of this benchmark dataset, Sharafldin et. al. [21] implemented a testbed infrastructure composed of two separated networks, namely Victim-Network and Attack-Network. The Victim-Network covered all common and necessary equipment usually presented in a modern-day highly secure infrastructure, since it included routers, firewalls, switches, three servers and ten PCs interconnected by a domain controller (DC) and active directory. While the Attack-Network included router, switch and four PCs working on multiple public IPs. These PCs were loaded with the software required to perform attacks against the Victim-Network. And all network traffic in the Victim-Network was recorded through a mirror port of its main switch.

An important characteristic of this dataset is the degree to which it represents real network traffic that would be gathered in a live network, commonly referred to as background traffic. Since it is essential for Network IDS testing to be conducted with rich background traffic. This is the reason why the abstract behaviour profiles of 25 users were implemented based on protocols such as HTTP, HTTPS, FTP, SSH and email protocols. Such abstract benign profiles, called as β-profiles, were created for the Victim-Network by using machine learning and statistical analysis techniques to properly obtain distributions of packets, packet sizes, protocol uses among others.

The CICIDS2017 dataset contains over 2.8 million flows and includes a wide range of attack types such as SSH, brute force, heartbleed, botnet, DoS, DDoS, web, and infiltration attacks, which were the most common attacks identified by the 2016 McAfee report. Where the breakdown of samples is given in the next table.

Table 1. CICIDS2017 types of network traffic

Traffic type	**Number of samples**	**Description**
Benign	2,273,097	Normal traffic behaviour.
DoS Hulk	231,073	A denial of service attack on a web server is carried out by generating volumes of unique and obfuscated traffic. Moreover, the generated traffic can bypass caching engines and strike a server's direct resource pool.
Portscan	158,930	An attacker tries to gather information from a victim machine related to the operating system and running services by sending packets with different destination ports.
DDoS	128,027	Under this attack, multiple machines that operate together are used to attack a victim machine.
DoS GoldenEye	10,293	The GoldenEye tool is used to perform a denial of service attack.
FTP-Patator	7,938	FTP Patator is used to perform a brute force attack in order to guess the FTP login password.
SSH-Patator	5,897	SSH Patator is used to perform a brute force attack in order to guess the SSH login Password.
DoS Slowloris	5,796	The Slowloris tool is used to execute a denial of service attack.
DoS Slowhttptest	5,499	An attacker exploits the HTTP Get request to exceed the number of HTTP connections allowed on a web server in order to prevent other clients from accessing.
Botnet	1,966	Trojans are used to breach the security of several victim machines, taking control of such machines, organizing them, and allowing their remote management.
Web Attack - Brute Force	1,507	An attacker tries to obtain privilege information such as password and Personal Identification Number (PIN) using a trial-and-error strategy.
Web Attack – XSS	625	Malicious scripts are injected into otherwise benign and trusted websites.
Infiltration	36	An attacker is able to infiltrate and gain full unauthorized access to the network data of a system.
Web Attack - SQL Injection	21	Injection of nefarious SQL statements into an entry field of a web form in order to attack data-driven applications.
Heartbleed	11	An attacker exploits a vulnerability of the OpenSSL protocol to insert malicious information into OpenSSL memory, enabling the attacker with unauthorized access to valuable information.
Total	2,830,743	

Therefore, the attack traffic only represents about the 19.7 % of the total network flows, as it is shown in the next picture.

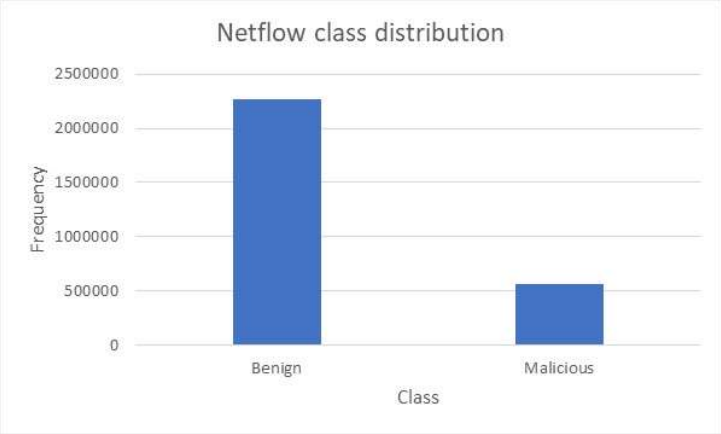


Figure 3. Class distribution in CICIDS2017 dataset

These figures confirm this dataset presents a high-class imbalance in order to reproduce common situations in real world scenarios. Class imbalance, which is a phenomenon caused by unequal distribution between majority and minority classes, can skew results in a big data study. CICIDS2017 has a high-class imbalance with respect to some of the individual attack types. Since, high class imbalance is defined by a majority-to-minority ratio between 100:1 and 10,000:1 [45].

The dataset CICIDS2017 is available at the site https://www.unb.ca/cic/datasets/ids-2017.html [21]. And, as it has been stated before, this work uses the bidirectional flows format version, which is composed of the following files and sample types.

Table 2. CICIDS2017 attacks per every data file

Data file	Type of traffic
Monday-WorkingHours.pcap_ISCX.csv	Benign (normal human activity)
Tuesday-WorkingHours.pcap_ISCX.csv	Benign, FTP-Patator and SSH-Patator
Wednesday-workingHours.pcap_ISCX.csv	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS Slowloris and Heartbleed
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Benign, Brute Force, SQL Injection and XSS
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Benign and Infiltration
Friday-WorkingHours-Morning.pcap_ISCX.csv	Benign and Botnet
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Benign and Portscan
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Benign and DDoS

3.2 Background on machine learning and deep learning

3.2.1 Naïve Bayes

Naïve Bayes (NB) is a probabilistic machine learning algorithm based on the Bayes Theorem, which allows to calculate the probability of an event based on the knowledge of factors that may affect it. As other statistical methods, it estimates a set of probabilistic parameters, which express the conditional probability of each class given the properties of a sample (described in the form of attributes). From

here, these parameters can be combined to assign the classes that maximize their probabilities to new samples. The Naïve Bayes algorithm classifies new samples $x = \{x_1, \ldots, x_N\}$ assigning it the class k that maximizes the conditional probability of the class given the observed attributes of the sample. Namely,

$$argmax\ P(k|x_1,\ldots,x_m) = argmax\ \frac{P(x_1,\ldots,x_m|k)P(k)}{P(x_1,\ldots,x_m)} \approx argmax\ P(k) \prod_{i=1}^{m} P(x_i|k)$$

where $P(k)$ and $P(x_i|k)$ are estimated from the training set, using the relative frequencies (maximum likelihood estimation).

In the next figure, the classification of a new data point using Naïve Bayes Classifier is explained. The samples can belong to green or yellow class. Based on the number of data points of each colour and the total number of points in the dataset it is possible to estimate the probability of any data point to belong to one class or another. To classify the new data point at prediction phase, its vicinity is considered (within black circle). Given the number of data points of each class in the vicinity and the general probability of having a data point of this respective class, the probability of the new data point to belong to each class is calculated. The highest probability is retained to classify the new data point. In this simple example, the aim is to classify the sample "?" as green or yellow. And, the final decision is green since the posterior probability P´´(? is green) is greater than P´´(? is yellow).

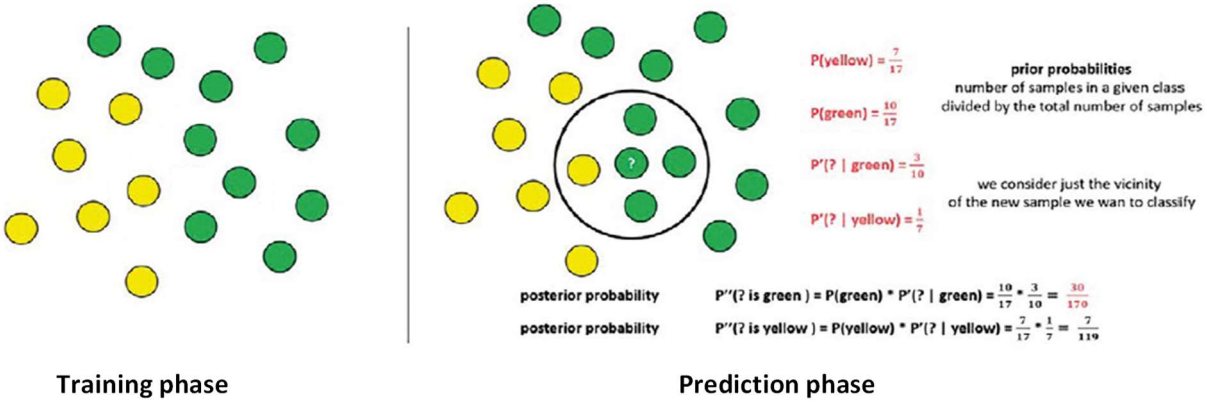


Figure 4. Example of Naïve Bayes classifier (adapted from [46])

An advantage of Naïve Bayes is that it only requires a small amount of training data to estimate the necessary probabilities and that the classifier can be trained incrementally.

3.2.2 Support Vector Machine

Support Vector Machines (SVM) solve a binary classification problem in which, a hyperplane is built from training data and capable of separate the observations into two groups. SVM is used to draw an optimal division line (or plane, depending on the number of input features) between classes so that it has the maximum possible distance to the nearest data points. The lines passing through these nearest data points are called support vectors. With the evolution of the input data set and introduction of new data points the support vectors as well as the optimal class separation line could change. It is typically used with a linear model but can be generalized to solve non-linear problems.

The next figure shows a simple example of the classification using SVM. The image on the left presents various possible hyperplanes (or lines in the bidimensional space) that allow the classification of data as black or white, being the lines H_1, H_2 and H_3 possible candidates. The image on the right presents an optimal separation line (H_3) and the maximum margin.

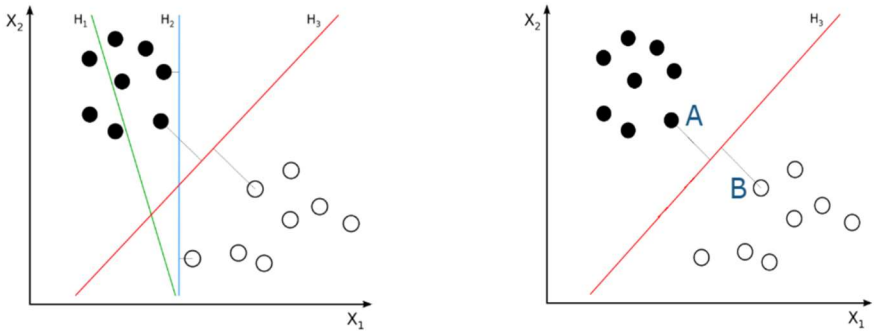


Figure 5. Example of Support Vector Machines classifier

As it is shown in the image on the left, we could find infinite hyperplanes for this classification problem. Once a hyperplane is chosen, it is possible to create a classification rule from it by defining the following equation:

$$h(x) = sign(b + \sum_{i=1}^{N} w_i\, x_i\,)$$

where N is the number of attributes, $x = \{x_1, \ldots, x_N\}$ is the observation to classify and b and the vector $w = \{w_1, \ldots, w_N\}$ define the hyperplane.

The aim of SVM is to define a hyperplane that optimally classify the data. Therefore, the hyperplane should have the maximum margin, which is the maximum distance between the samples of both classes. In this example, the figure on the right shows that the maximum margin is defined by the hyperplane H_3, which will be found by the SVM method.

3.2.3 Random Forest

Random Forest (RF) is a popular ensemble method which uses decision trees as the base classifiers. The output of Random Forest is decided by the votes given by all individual trees. Random Forest algorithm manipulates both the training dataset and input features. Therefore, each decision tree is built by classifying the bootstrap samples of the input data using a tree algorithm. At classification time, every tree will be used to classify the testing data. Where each tree emits a vote for guessing the label of every observation. Finally, the forest decides the classification result of the testing data after collecting the votes among trees.

Next figure provides a schematic explanation of how a random forest algorithm works, presenting a forest of three trees.

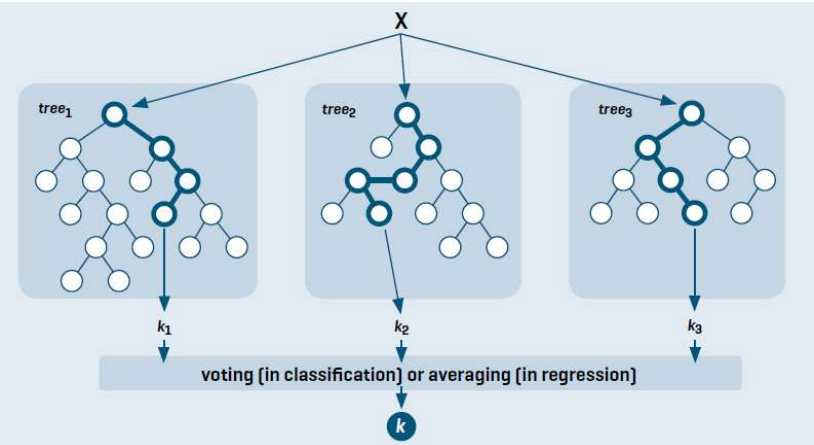


Figure 6. Example of Random Forest classifier [46]

This method aims to utilize the exibility of a single decision tree, while mitigating their drawbacks by spreading them out over an entire collection. The accuracy of Random Forest depends on the strength of the individual trees and a measure of the dependence between the trees. Moreover, the Random Forest algorithm uses bootstrap to avoid biases in tree building, such that cross validation (CV) is not needed in training and testing. However, Random Forest suffers from the class imbalance due to the maximization of the prediction accuracy in its algorithm. Since tree-based methods have a high variance, the hierarchical structure of trees can produce an unstable result. The average of many trees, e.g. using bagging (bootstrap aggregation), can improve stability of ensemble learning algorithms. These methods are efficient for large datasets and excel at multiclass problems, but deeper trees might lead to overfitting.

3.2.4 AutoEncoder

An AutoEncoder is an unsupervised neuronal network that attempts to reconstruct the input variables that have been presented to it. As it can be seen in the following image, an AutoEncoder consists of an Encoder and a Decoder connected sequentially. Where the Encoder network accepts high-dimensional input data and reduces the data to a low-dimensional latent space. While the Decoder network accepts as input such data from the latent space. The goal of the AutoEncoder is to reconstruct the original input data as faithfully as possible.

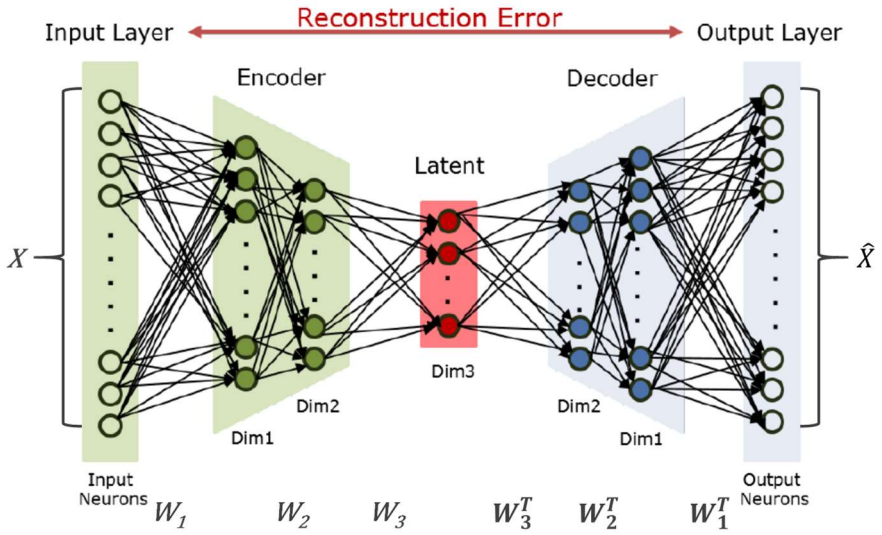


Figure 7. Basic architecture of an AutoEncoder (adapted from [48])

Formally, given an input *X*, an autoencoder is trained to minimise the Reconstruction Error, which is represented as the difference between *X* and $\hat{X}$ such that:

$\hat{X} = g(f(X))$, being *X* a vector, *f(X)* the encoding function and *g(X)* the decoding function.

The Reconstruction Error is defined by a function that represents the difference between the input *X* and its counterpart $\hat{X}$. As both *X* and $\hat{X}$ are in fact real values, the squared error can be used as the loss function. Therefore, it can be considered as a regression problem. Alternatively, cross-entropy can also be used when *X* is a value in the range [0, 1]. Regarding hidden layers and output layer, different activation functions can be applied such as Sigmoid, Relu and Linear among others.

3.3 Methodology for the implementation of the proposed solution

The methodology followed to design and implement the final solution includes the main following phases:

Figure 8. Methodology for implementing the proposed solution

On the other hand, in this research work, all processing activities have been done on a laptop with the following characteristics: Intel(R) Core™ i7-8850H CPU @ 2.60GHz, 32.0 GB of RAM and 64-bit Windows 10 Enterprise operating system. In terms of software, the implementations have been made on Python language doing an extensive use of the modules and libraries: NumPy [49], Pandas [50], Scikit-learn [51], Keras [52] and TensorFlow [53].

In the next sections, every phase is described in detail.

3.4 Pre-processing of the dataset

Data pre-processing is an essential step in any machine learning project. Realistic data typically come from heterogeneous platforms and can be noisy, redundant, incomplete, and inconsistent [54]. Therefore, once the bidirectional flow data are available, the pre-process step covers cleaning the data, removing the outliers and some other relevant data transformations required to enhance the dataset.

3.4.1 Cleaning the dataset

Data cleaning refers to identifying and correcting errors in the dataset that may negatively impact on the performance of the machine learning model. For doing so, several pre-processing tasks have been required as they are described below.

Renaming IP addresses according to the official information of the dataset

The CICIDS2017 dataset presents an inconsistency related to the source and destination IPs. Sometimes the IP address of the firewall (172.16.0.1) appears instead of the right attacker or the victim IP. Therefore, the IP 172.16.0.1 has been replaced manually in order to avoid the effects of the NAT process on the firewall and to force the overall coherence of the data.

Cleaning redundant, infinity and missing values

Several cleaning tasks have been carried out under this activity. Firstly, missing and infinity values of the observations have been treated since generally machine learning algorithms do not consume them. As these values represent a small percentage, those records from the dataset have been deleted. Secondly, features of the Monday's samples with only one value in all samples are discarded since they have no effect on any calculation on the dataset. Specifically, these features are: Fwd_URG_Flags, CWE_Flag_Count, Bwd_Avg_Packets/Bulk and Bwd_Avg_Bulk_Rate. The same action has been done on the rest of dataset files. In addition, the feature Fwd_Header_Length is duplicated; therefore, it is removed once. Finally, samples that have duplicated values (except for Timestamp and FlowID features) are also removed although they are only a few registers.

Removing samples of protocol 0

In the original baseline dataset corresponding to Monday, there are samples of three protocols: 0 (corresponding to several protocols), 6 (corresponding to TCP/IP) and 17 (corresponding to UDP), being

their number of samples 317, 305423 and 224178 respectively. In this respect, it must be highlighted all malicious activities are related to observations of protocol 6. Taking in mind that protocol 0 is not representative in all of the data files and it is usual in practice to monitor samples by protocol, it has been made the decision to remove such samples and train the models with only protocols 6 and 17.

3.4.2 Removing outliers

As it is shown in the next picture, the baseline dataset presents many outlier values for the majority of the feature space. Therefore, it is necessary to identify and remove samples with outliers, and thus ignore the deviant observations.

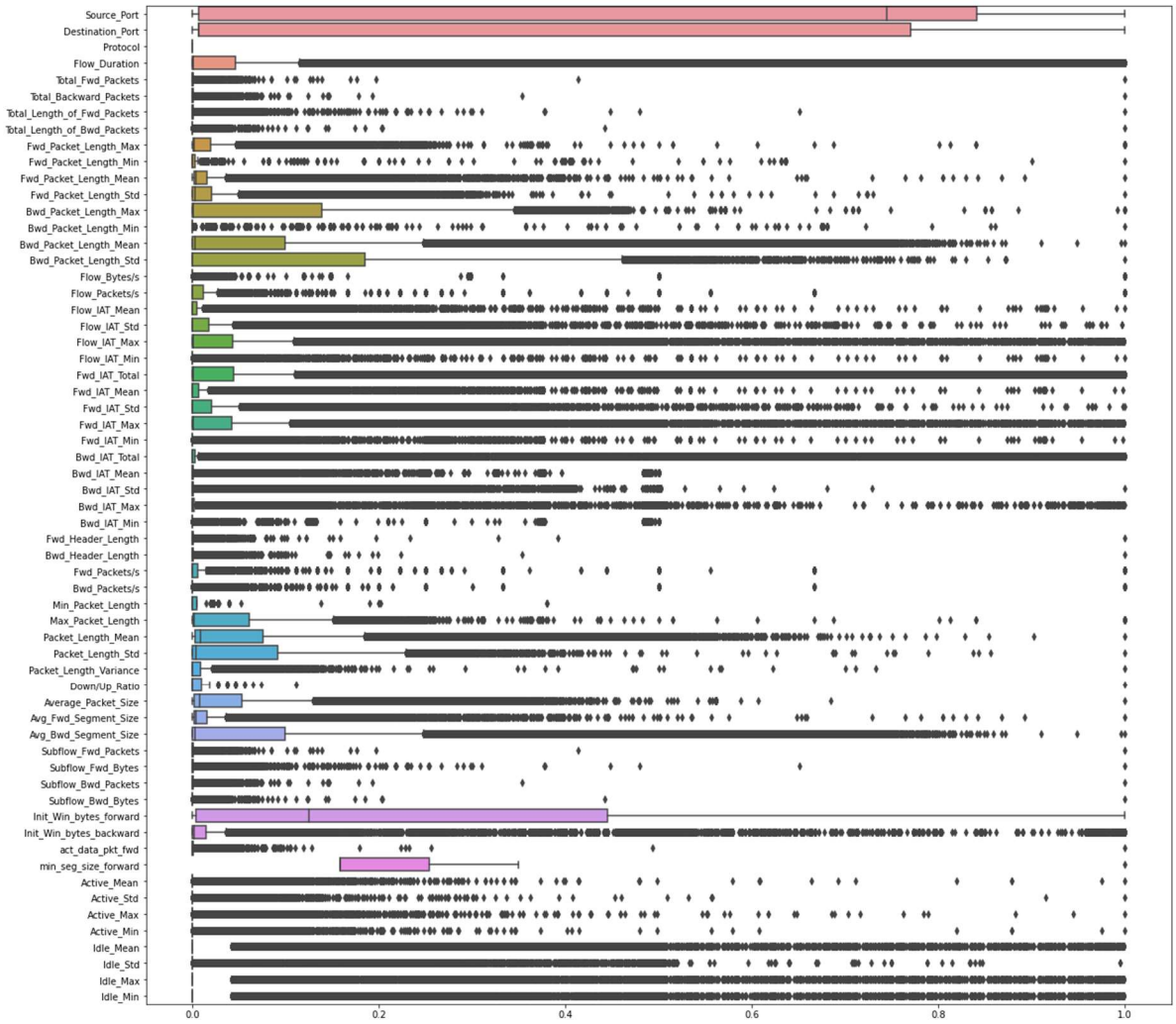


Figure 9. Median, quartiles and outliers in the baseline dataset

At a first attempt, the following unsupervised methods have been applied to detect outliers on the baseline dataset that involves a high number of features:

- Interquartile Range or IQR-based. IQR is equal to the difference between 75th and 25th percentiles. The IQR can be used to identify outliers by defining limits on the sample values that are a factor k of the IQR below the 25th percentile or above the 75th percentile. The common value for the factor k is the value 1.5. While a factor k of 3 or more can be used to identify values that are extreme outliers.
- DBScan (Density-Based Spatial clustering of applications with noise), which identifies points that are in dense regions of the feature space (i.e., where many data points are close together). The

idea behind DBSCAN is that clusters form dense regions of data, separated by regions that are relatively empty and in which outliers will be located at.

- Robust covariance or Elliptic Envelope, which creates an imaginary elliptical area around a given dataset. Values that fall inside the envelope are considered normal data while anything outside as outliers. This method works better when a dataset follows a Gaussian distribution.
- One-Class SVM (OSVM), which is a technique that was developed from the SVM (Support Vector machines). OSVM can be used to find outliers in a dataset based on a certain threshold.
- Isolation Forest (IF), which isolates observations by randomly selecting a feature and then randomly selecting a split value between its maximum and minimum values. Since recursive partitioning can be represented by a tree structure, the number of splitting required to isolate a sample is equivalent to the path length from the root node to the terminating node. This path length, averaged over a forest of such random trees, is a measure of normality. Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies.
- Local Outlier Factor (LOF), which is an algorithm that finds outliers by comparing the density of each data. Hence, by measuring the local deviation of a given sample with respect to its neighbours and determining the lowest density as abnormally.

For example, these methods are able to detect the following outliers of the samples related to the protocol 6 in the Monday dataset:

- IQR-based: 279926 (applied a factor of 3.0)
- DBScan: 5722
- Robust covariance (Elliptic Envelope): 45675
- OSVM (One-class SVM): 45673
- IF (Isolation Forest): 45674
- LOF (Local Outlier Factor): 45674

It must be highlighted that although the number of outliers is similar, the methods identify different observations. Therefore, after merging all the outliers identified, a total number of 88167 samples have been removed from the dataset. However, this approach does not have the expected positive effect on the performance of the AutoEncoder model. The reason could be the great number of outliers that still remains in the dataset as it is shown in the next picture.

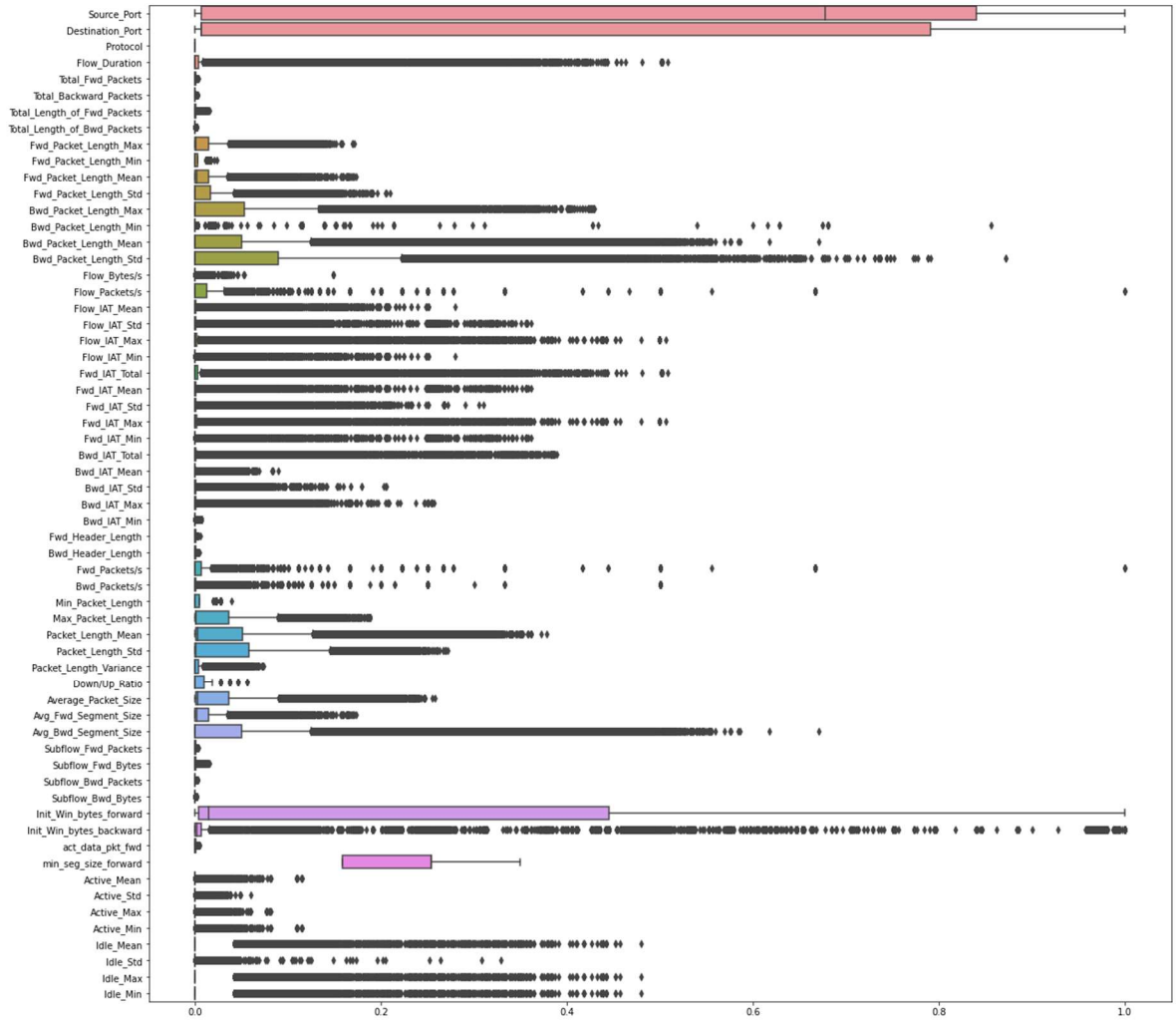


Figure 10. Distribution of the baseline dataset after removing the identified outliers

Therefore, finally, a custom process to detect and remove outliers in all data files has to be applied to overcome such problems. This process consists of the following tasks:

- Analysis of the right minimum values per feature. As a result, this analysis has identified that some features (which are: Flow_Duration, Flow_Bytes/s, Flow_Packets/s, Flow_IAT_Mean, Flow_IAT_Max, Flow_IAT_Min, Fwd_Header_Length, Bwd_Header_Length, Fwd_Packets/s, and min_seg_size_forward) should have a minimum value of zero. However, the minimum value of Init_Win_bytes_forward and Init_Win_bytes_backward should be -1 since their present this value the 42.4% and 56.4% of the samples respectively.
- Mark the observations whose minimum values are lower than zero as outlier with a new feature called as Outlier_min.
- Analysis of the right maximum values per feature. As a result, this analysis has identified the maximum recommended value for some features as it is described in the table below.
- Mark the observations whose maximum values are 1.4 times greater than the recommend maximum value as outlier with a new feature called as Outlier_max. Here, the factor has been setup to 1.4 with the aim of leaving high margins with respect to the advisable maximum values.

Table 3. Maximum value recommended per feature

Feature	Maximum value recommended	Data file with the most critical situation
Total_Fwd_Packets	112141	Wednesday-workingHours
Total_Backward_Packets	157388	Wednesday-workingHours
Total_Length_of_Fwd_Packets	1323378	
Total_Length_of_Bwd_Packets	399000000	Wednesday-workingHours
Bwd_Packet_Length_Max	13140	
Bwd_Packet_Length_Mean	2976.321839	
Bwd_Packet_Length_Std	2728.559021	
Flow_IAT_Mean	119748208	
Fwd_Header_Length	2289684	Wednesday-workingHours
Bwd_Header_Length	3147772	Wednesday-workingHours
Avg_Bwd_Segment_Size	2976.321839	
Subflow_Fwd_Packets	112141	Wednesday-workingHours
Subflow_Fwd_Bytes	929308	Thursday-WorkingHours-Afternoon-Infilteration
Subflow_Bwd_Packets	157388	Wednesday-workingHours
Subflow_Bwd_Bytes	399368696	Wednesday-workingHours
act_data_pkt_fwd	108273	Wednesday-workingHours

3.5 Feature engineering phase

Once the raw data have been cleaned, it is usual to include a feature engineering phase when applying machine and deep learning techniques. Feature engineering is the process of using domain knowledge to extract features from data. These new features can be used to improve the performance of machine learning algorithms. In words of Scott Locklin [55], “much of the success of machine learning is actually success in engineering features that a learner can understand”. Therefore, at this stage, a process for enhancing the dataset is carried out over the data flows cleaned in the previous phase. Here, the tasks performed have mainly focused on different treatments for the original source and destination IP addresses and ports features as it is described in detail below.

3.5.1 Handling IP address feature

The IP address information is one of the most important features to detect intrusions since it is strongly related to source (malicious agents’ PCs) and destination machines (mainly enterprise servers) involved in attack scenarios. This feature is gathered as an IPv4 address in the dataset CICIDS2017. An IPv4 address is a 32-bit number that uniquely identifies a network interface of a machine, and it is displayed as four numbers (or octets) separated by three dots. As a categorical feature, IPv4 address cannot be easily managed by machine learning algorithms. Therefore, five methods for encoding IP addresses have been applied as it is described in this section.

Encoding IP address feature as an integer number

This is the simplest encoding mechanism where an IP address is encoded as an integer value between 0 and number of IPs-1. Thus, the function *LabelEncoder* of Scikit-learn library has been used to accomplish this task.

Splitting IP address feature into four numbers

This method converts an IP address feature into new four features, which are integer numbers, as it is shown in the next picture. Therefore, every octet will correspond to one feature.

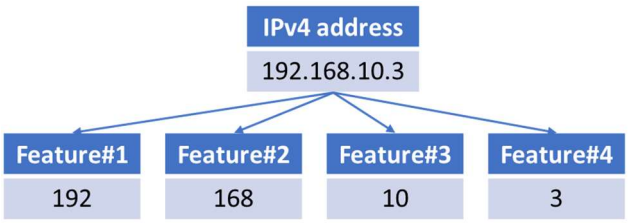


Figure 11. Encoding an IPv4 address into 4 features

This method also allows to encode an IP address feature into three features. In this case, the last octet is ignored while the first three features maintain the information of the network (features 1 and 2) and the subnet (feature 3).

Encoding the 1000 most frequent IP addresses

This method includes several steps. Firstly, all IPs (both source and destination) are ordered by their frequency of use. And later on, only the first high frequency IPs are encoded as an integer number from zero to 1012, which are about 1000 IPs. While the rest of the IPs are encoded with the value of 1014.

Flag for infrequent IPs

This flag marks low-frequency IPs since it is set to 1 for such IPs while it has a value of zero for the 1000 most frequent IP addresses identified previously.

3.5.2 Handling port feature

Port numbers are integer values from zero to 65535. Values below 1011 are related to standard communication services and are called well-known ports. While values above 1023 can be either registered or dynamic (also called private or non-reserved). Registered ports are in the range 1024 to 49151. Dynamic ports are in the range 49152 to 65535. Generally, the newest port assignments are in the range from 1024 to 49151. Although, the source and destination port features appear numerical and potentially continuous in nature, they should be considered as categorical variables since their value has a meaning. For instance, the value ‘80’ corresponds to the HTTP protocol and not the continuous value of 80. Hence, relevant values such as 21, 22, 80 and 444 corresponds to specific communication protocols which are usually attacked with different adversary tactics and techniques. Therefore, good encoding of port-based features is vital for achieving good results in intrusion detection from machine and deep learning models. In this research work, several encoding methods have been applied as it is described below.

Encoding port feature as an integer number

This is the simplest encoding mechanism where a port feature is treated by its integer value. In fact, no encoding is performed in this case.

Own custom encoding of the port feature

This encoding gives more relevance to the well-known ports used by UDP and TCP services and functions. This method encodes the port features by applying the following formula:

a) value = $\frac{port\ number * 0.5}{500}$, if port number <= 500

b) value = $0.5 + \frac{port\ number * 0.5}{65535}$, if port number > 500

Where 65535 is the maximum port number by definition.

Flag for well-known destination ports

This feature flags ports with values from zero to 1023 since the communications standards reserve this range of values for well-known services.

Flag for registered destination ports

This feature flags ports with values from 1024 to 49151 since the communications standards reserve this range of values for registered services.

One-hot encoding of the 20 most frequent source ports

One-hot encoding is a common technique for converting categorical feature variables into continuous features for their use with a machine learning algorithm. This process works by creating a new binary feature for every unique possible value of the original categorical feature.

This method includes several steps. Firstly, source ports are ordered by their frequency of use. And later on, only the first high frequency ports are encoded as an integer number from zero to 21. While the rest of the ports are encoded with the value of 22.

Flag for infrequent source ports

This flag marks non high-frequency source ports since it is set to zero for the source ports identified previously while it has a value of one for the rest of them.

One-hot encoding of the 20 most frequent destination ports

This method includes several steps. Firstly, destination ports are ordered by their frequency of use. And later on, only the first high frequency ports are encoded as an integer number from zero to 20. While the rest of the ports are encoded with the value of 21.

Flag for infrequent destination ports

This flag marks non high-frequency destination ports since it is set to zero for the destination ports identified previously while it has a value of one for the rest of them.

One-hot encoding of the 20 most frequent source and destination ports

This method includes several steps. Firstly, all ports (both source and destination) are ordered by their frequency of use. And later on, only the first high frequency ports are encoded as an integer number from zero to 20. While the rest of the ports are encoded with the value of 21.

Flag for infrequent source and destination ports

This flag marks non high-frequency source and destination ports since it is set to zero for the source and destination ports identified previously while it has a value of one for the rest of them.

3.6 Creating the baseline and zero-day attacks datasets

Firstly, at the beginning of this process, all samples marked as outliers previously have been discarded.

At a second step, it has been necessary to normalize or scale the features with continuous values. Since, the numerical features of the dataset are of different ranges, which poses some challenges to machine learning methods during the training to compensate these differences in magnitudes. Therefore, it is necessary to produce more homogeneous values while maintaining the relativity among the values of every feature. This allows for faster convergence on learning and more uniform influence for all weights [56]. Common approaches for scaling are: MinMax, Standard (or Z-score), Robust, Normalizer and Quantile.

In this work, MinMax and Standard scaler have been applied to normalize the features, where the Monday's samples have been taken as the reference values. The MinMax scaling converts the range of a given feature to [0, 1] by applying the following formula given a feature f with a range [f_{min}, f_{max}]:

$$f_{scaled} = \frac{f - f_{min}}{f_{max} - f_{min}}$$

And the Standard scaling removes the mean μ and scales to unit variance σ by applying the following formula given a feature f:

$$f_z = \frac{f - \mu_{min}}{\sigma_f}$$

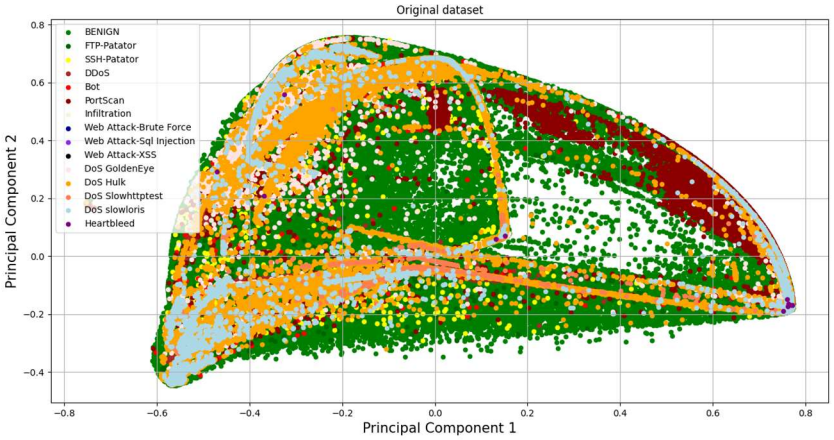
At the third step, the zero-day attacks dataset has been created by the union of all available data from Tuesday to Friday. While the baseline or normal dataset corresponds to the Monday data file.

Finally, observations in both datasets have been chronologically ordered since they are not in the original CICIDS2017 dataset.

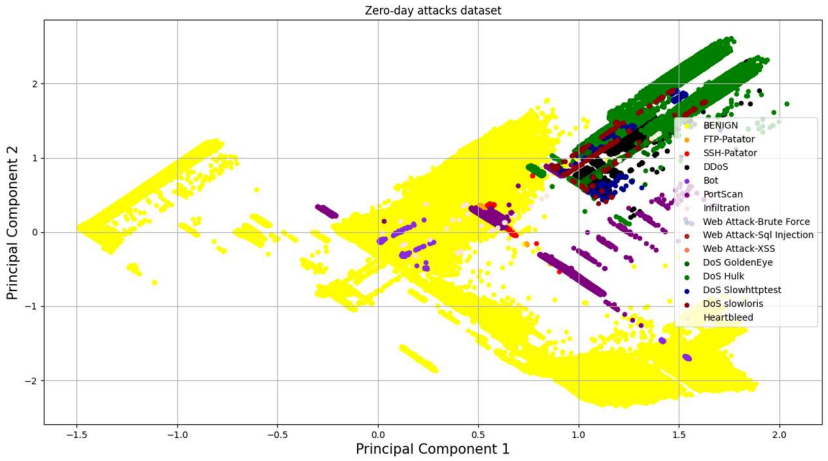
In the next table, some summary statistics of the baseline and zero-day attacks datasets are given. While in the images in the figure, the original and the zero-day attacks datasets (with samples from Tuesday to Friday) have been projected with PCA (Principal Component Analysis) dimensionality reduction technique. These images show how the enhanced dataset presents a variance which is better explained through its features. However, it is still a complex task to detect zero-day attacks with low false positive and high detection rates.

Table 4. Breakdown of observations in baseline and zero-day attacks datasets

Label	Baseline (Monday)	Zero-day attacks (Tuesday-Friday)	Total
Benign	528,629	1,552,458	2,081,087
Heartbleed		6	519,617
Web Att - Brute Force		1,507	
Web Att - XSS		652	
Web Att - Sql Inj		21	
PortScan		158,798	
DDoS		102,649	
DoS Hulk		219,681	
DoS GoldenEye		9,195	
DoS slowloris		5,796	
DoS Slowhttptest		5,499	
Bot		1,956	
FTP-Patator		7,931	
SSH-Patator		5,895	
Infiltration		31	
Total	528,629	2,072,075	2,600,704



Original dataset (observations from Tuesday to Friday)



Zero-day attacks dataset

Figure 12. Projection of the original and zero-day attacks datasets with PCA

On the other hand, the correlation matrix of the zero-day attacks dataset is shown in the next figure in order to depict the patterns of relationships among their features.

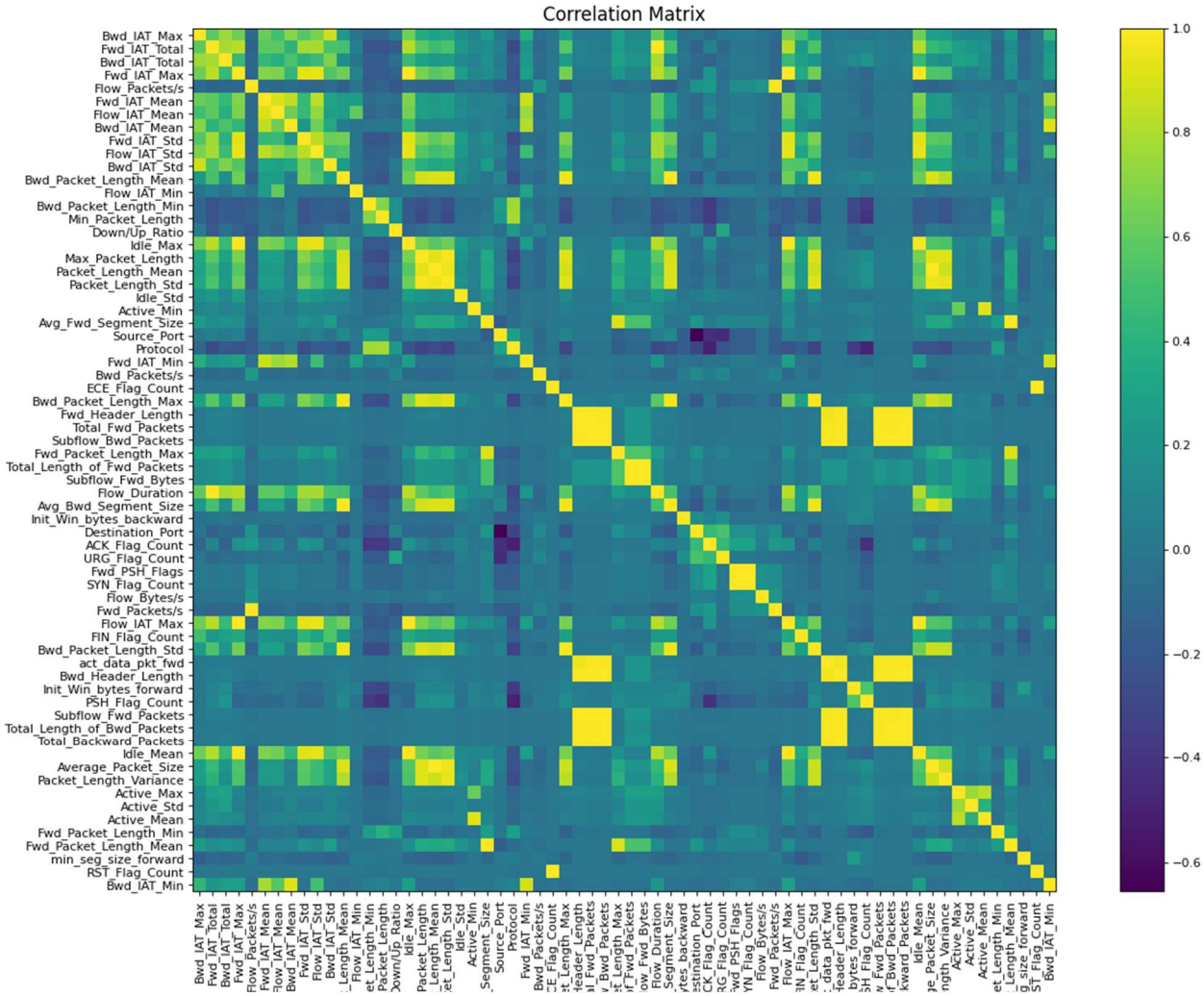
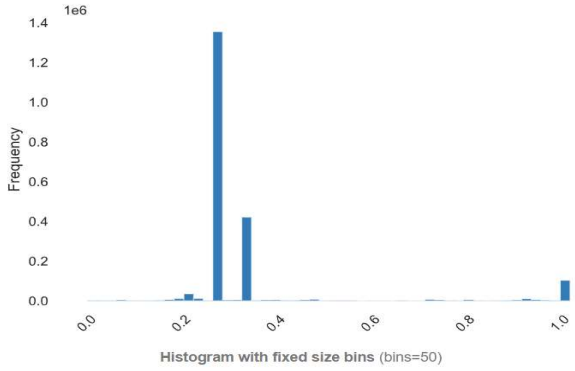
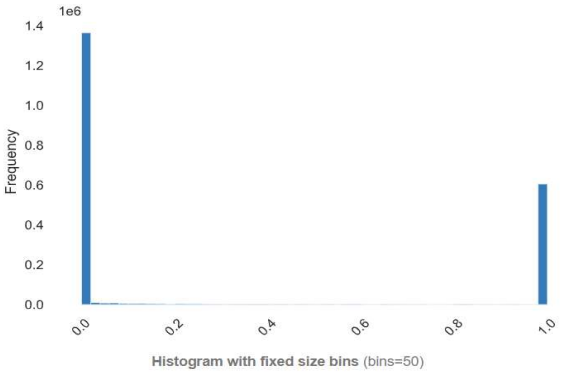


Figure 13. Correlation matrix of the zero-day attacks dataset

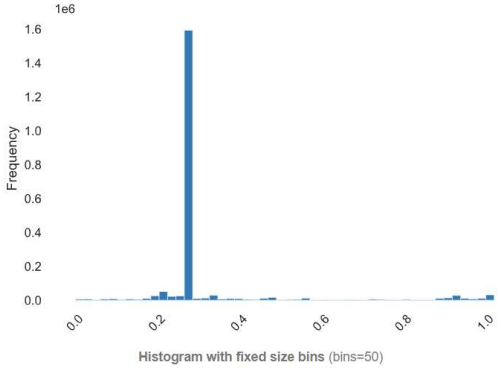
Moreover, it is noticeable that many features do not follow Gaussian distributions as it is shown in the next pictures.



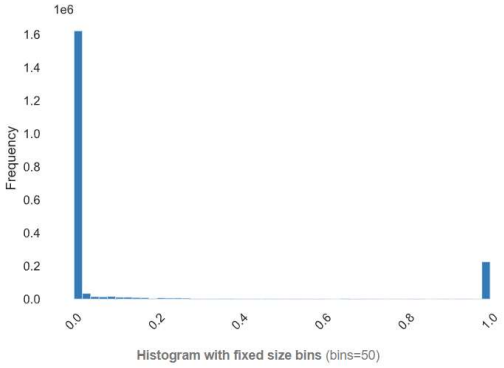
Source IP encoded as an integer number



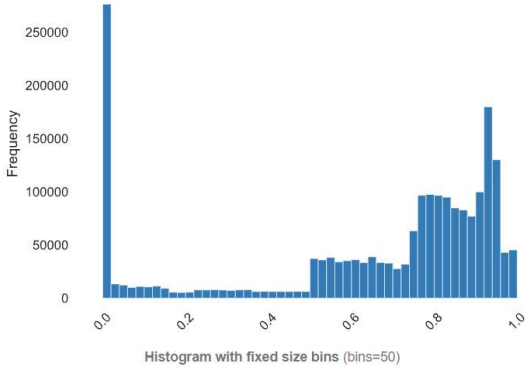
Source IP encoded as the 1000 most frequent



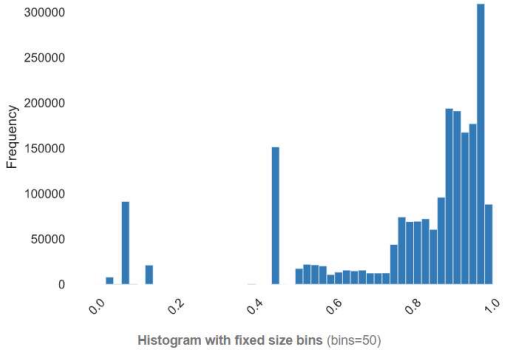
Destination IP encoded as an integer number



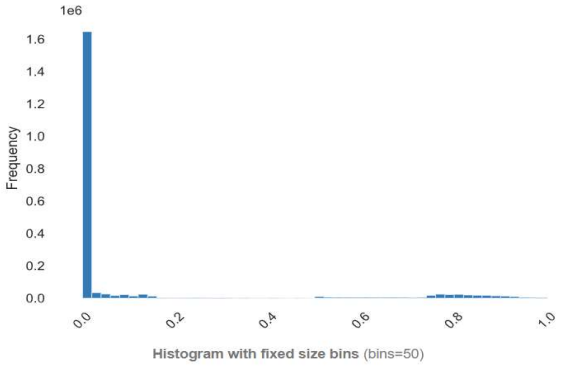
Destination IP encoded as the 1000 most frequent



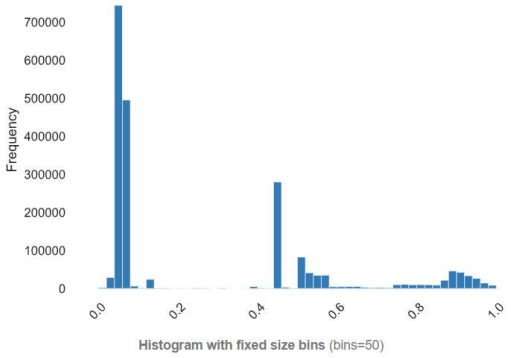
Source port encoded as an integer number



Source port encoded as own custom approach



Destination port encoded as an integer number



Destination port encoded as own custom approach

Figure 14. Distribution of several IPs and port features in zero-day attacks dataset

3.7 Adopted solution for zero-day and known attacks detection

This work proposes a solution based on the holistic combination of machine and deep learning models for the detection of zero-day and known attacks to Industrial Control Systems and Critical Infrastructures, as it is shown in the next picture.

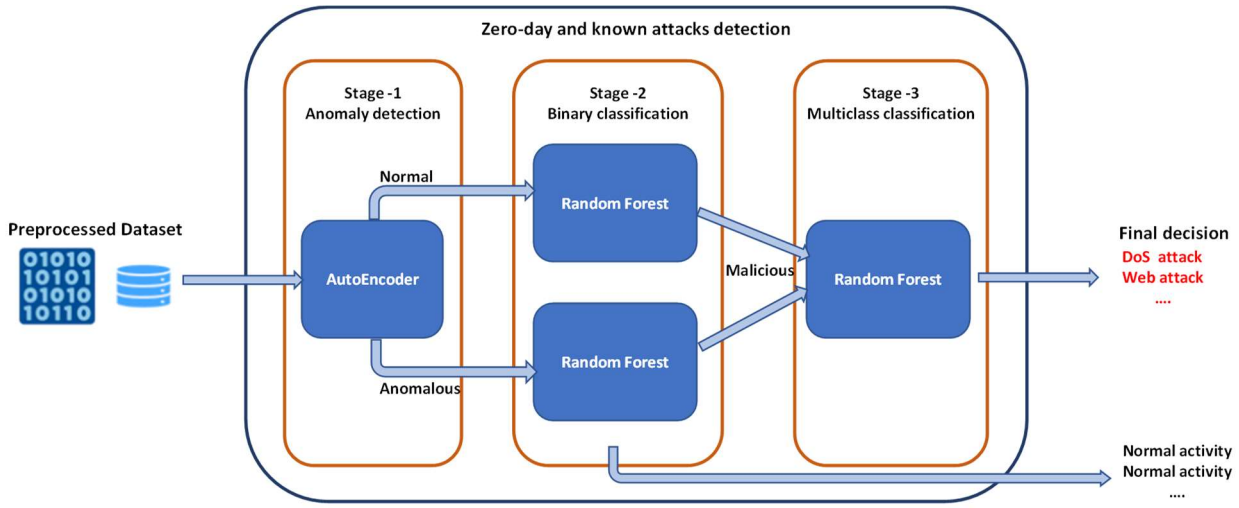


Figure 15. Machine learning pipeline for zero-day and known attacks detection

The machine learning pipeline consists of 3 stages. Firstly, an AutoEncoder is applied to detect anomalies in the traffic network once it has been only trained on benign traffic flows. This model is trained on the baseline dataset (only with samples of Monday). And, as unsupervised training is performed, no labelled dataset is used.

At the second stage, a binary classifier is used in order to determine whether samples correspond to normal activities or malicious behaviours. For doing so, three algorithms have been candidates: Naïve Bayes, Support Vector Machines and Random Forest. In fact, two models per candidate have been training on the dataset partition generated previously by the AutoEncoder. The selection of the best techniques has been made by applying a cross-validation approach. Where cross-validation is mainly used to estimate how accurately a predictive model will perform in practice. The goal of cross-validation is to test the model's ability to predict new data that was not used in its training, in order to give an insight on how the model will generalize them. In this work, a stratified 10-Folds cross-validation approach has been applied. This cross-validation returns 10 stratified folds where such folds are made by preserving the percentage of samples for each class. After testing the candidates, the Random Forests was chosen since it presented the best performance.

And at the third and final stage, a multiclass classifier is used in order to determine which type of malicious activity is carrying on. Similarly, to the previous case, three algorithms have been candidates: Naïve Bayes, Support Vector Machines and Random Forest. The selection of the best technique has been made by applying a stratified 10-Folds cross-validation approach. After testing the candidates, the Random Forests was chosen since it presented the best performance.

In summary, all models work together. The first model helps to split the observations into similar to benign traffic and very different. While the second model is able to detect normal activity and the third model the type of malicious behaviours.

3.7.1 Anomaly detection based on AutoEncoders

This is the first stage of the machine learning pipeline. This model is based on an AutoEncoder, which is an unsupervised deep learning technique. An AutoEncoder is able to detect anomalies in the traffic network once it has been trained on benign traffic flows. As it has been explained in section 3.2.4, an AutoEncoder attempts to reconstruct the input variables that have been presented to it. It consists of an Encoder and a Decoder networks connected sequentially. The Encoder accepts high-dimensional input data and reduces the data to a low-dimensional latent space. While the Decoder accepts as input such data from the latent space. The goal of the AutoEncoder is to reconstruct the original input data as faithfully as possible. The hypothesis behind is the reconstruction error will be higher when the neuronal network sees malicious flows, therefore, such error can be used as an anomaly score. Therefore, anomaly scores higher than a predefined threshold will be considered as malicious activities. The aim of the AutoEncoder is to perform a binary classification. Therefore, the labels will be encoded as zero for benign samples and as one in case of attacks.

At a first attempt, it was implemented the symmetric architecture proposed at the work [34]. There, the authors defined an optimised AutoEncoder architecture for the CICIDS2017 dataset comprised from an ANN network with 3 hidden layers with 15, 9, 15 neurons respectively. Although all other hyperparameters and training conditions were also reproduced, this solution was discarded due to the poor results achieved. Similarly, it was also tested and discarded the KitNET solution [39] since it is too simple to be able to address this complex problem.

Finally, after several trials, the AutoEncoder architecture for CICIDS 2017 proposed by Fernandez [57] has been implemented, as it is shown in the next picture. This AutoEncoder model has been implemented with Keras, the Python deep learning API.

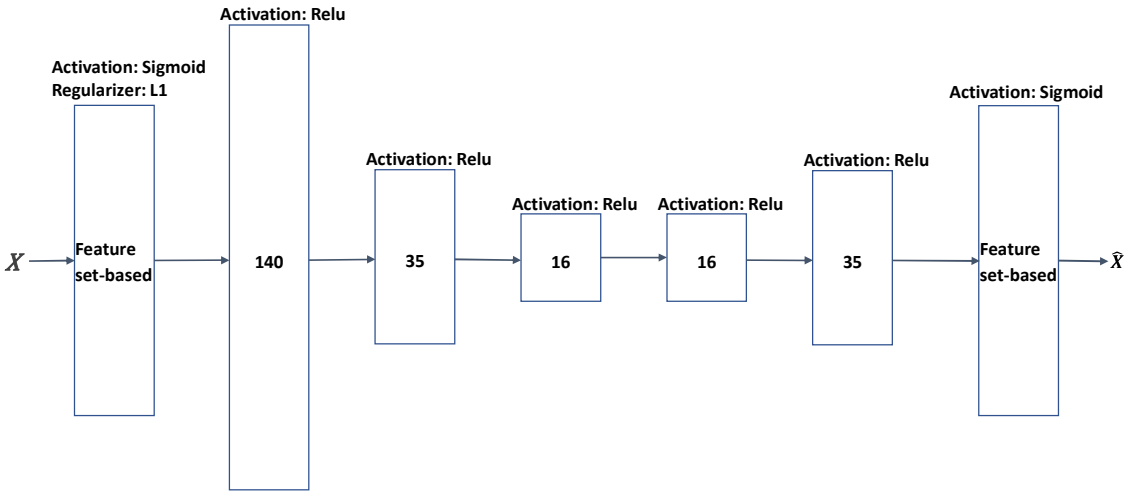


Figure 16. Structure of the AutoEncoder model (adapted from [57])

It is interesting to mention that the number of the neurons at input and output layers is underdetermined because they depend on the feature set used at the training phase. Since several tests have been carried out in order to train this algorithm with different combination of features in order to discover the most appropriate ones. A detailed description is provided in the chapter 4.2.

On the one hand, the model has been trained on the baseline dataset (only with samples of Monday). And, as unsupervised training is performed, no labelled dataset is used. Here, Min-Max scaling is applied to normalize the features since sigmoid is used as the activation function in the output layer of this deep neuronal network. Where anomaly scores lower than a specific threshold are considered benign

samples because they will present small reconstruction error. Otherwise, they will be considered as malicious. After several trials, the threshold has finally been set to 0.15003404215567 as it will be explained in the chapter 4.2.

On the other hand, the hyperparameters (or parameters of the model that are set before the learning process begins) that provide the best results are the following:

- Optimisation algorithm: Adam
- Objective or loss function: Mean Squared Error
- Evaluation metric: Accuracy
- Learning rate: 1e-3
- Momentum decay: 0.9
- Regularizer: L1 regularization penalty, which is computed as loss = L1 * reduce_sum(abs(x)) with L1=0.01.
- Number of epochs: 200
- Batch size: 1024

3.7.2 Binary classification based on Naïve Bayes

At the second stage of the machine learning pipeline, three supervised methods have been chosen as candidates for the binary classification of the dataset partition generated by the AutoEncoder model. The first method tested has been the Naïve Bayes algorithm, which is a probabilistic machine learning algorithm based on the Bayes Theorem. In fact, two Naïve Bayes models have been trained over this partition by applying a stratified 10-Folds cross-validation approach. This cross-validation returns 10 stratified folds where such folds are made by preserving the percentage of samples for each majority class. However, the 60% of data are for training and the remaining 40% for testing in the case of classes with a reduced number of malicious samples (Bot, Infiltration and Heartbleed). In addition, the observations of similar attacks like Dos (Hulk, GoldenEye, Slowloris and Slowhttptest) and Web (Brute Force, XSS and Sql Injections) have been previously grouped together. At this point, it is important to ensure there is the same proportion of malicious flows in each training and test samples as they are in the dataset as a whole since it is highly imbalanced. This is reason why the dataset is stratified.

Specifically, these algorithms have been trained with a special set of features composed of:

- Flow Duration
- Flow Packets/s and Flow Bytes/s
- Subflow Fwd and Bwd Packets
- Subflow Fwd and Bwd Bytes
- Fwd and Bwd Packets/s
- Fwd and Bwd Header Length
- Packet Length Min, Max, Mean, Std and Variance
- Fwd Packet Length Max, Min, Mean and Std
- Bwd Packet Length Max, Min, Mean and Std
- Total Fwd and Backward Packets
- Total Length of Fwd and Bwd Packets
- Average Packet Size
- Avg Fwd and Bwd Segment Size
- Fwd IAT Total, Max, Min, Mean and Std
- Bwd IAT Total, Max, Min, Mean and Std

- Flow IAT Max, Min, Mean and Std
- Fwd PSH Flags
- FIN, SYN, RST, PSH, ACK, URG and ECE Flag Count
- Down/Up Ratio
- Init Win Bytes Forward and Backward
- Act Data Pkt Fwd
- Min Seg Size Forward
- Active Min, Max, Mean and Std
- Idle Max, Min, Mean and Std
- Well-known and Registered Flag Destination Port

On the one hand, it is remarkable that relevant features such as Protocol, Source Port, Source IP, and Destination IP are not used. On the other hand, as a supervised training is performed, the Label feature has been used as the feature that the model has to accurately predict. These labels have been considered as zero for benign samples and one for malicious observations since a binary classification is carried out.

Finally, it is worthy to mention that standard scaling is applied to normalize the dataset.

3.7.3 Binary classification based on Support Vector Machines

The second supervised method tested has been the SVM algorithm in which a hyperplane is built from training data and capable of separate the observations into two groups. Here, the hyperparameters are the kernel, which is chosen as RBF, and the C or regularization parameter that has been set to 100.

The procedure followed and the selected features have been the same described in the section devoted to the Naïve Bayes algorithm. In summary, two SVM models have been trained by applying a stratified 10-Folds cross-validation for each majority class. However, the 60% of data are for training and the remaining 40% for testing in the in the case of classes with a reduced number of malicious samples. In addition, the observations of similar attacks like Dos (Hulk, GoldenEye, Slowloris and Slowhttptest) and Web (Brute Force, XSS and Sql Injections) have been previously grouped together.

On the other hand, standard scaling is used to normalize the dataset.

3.7.4 Binary classification based on Random Forest

The third supervised method tested has been the Random Forest algorithm, which uses decision trees as the base classifiers. Random Forest algorithm manipulates both the training dataset and input features. A Random Forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

The procedure and features used to train the model have been the same described previously in detail in the Naïve Bayes algorithm section, where a stratified cross-validation approach has been applied. On the other hand, standard scaling is used to normalize the dataset.

After doing several trials, the hyperparameters that allow the achievement of the best performance of the model are the following:

- Bootstrap flag: True
- Criterion to measure the quality of a split: Gini impurity
- Number of features to consider when looking for the best split: Sqrt(number of features)
- The number of trees in the forest: 100

Once the Naïve Bayes, Support Vector Machines and Random Forest have been trained under the same conditions and following a stratified cross-validation approach, the performance comparison has proved that the Random Forest model presents the best results. Therefore, this technique has been finally selected. And, in a second step, the Random Forest has been trained again with the 10% of the dataset, leaving the remaining 90% as test dataset.

In this work, the main advantages of using Random Forest are:

1. Random Forest tends to perform better when the set of features to select from is large. This allows the construction of de-correlated individual trees, which in turn, improves the prediction performance of Random Forests.
2. The speed of the training is fast because, at any given tree node, Random Forest only considers a randomly selected subset of features that are much smaller in number than the entire feature space. Hence, reducing the training time.

3.7.5 Multiclass classification based on Naïve Bayes, SVM and Random Forest

At the third and final stage of the machine learning pipeline, the three supervised methods (NB, SVM and RF) have been also chosen as candidates for the multiclass classification of the dataset partition generated by the Random Forest model used previously. For multiclass classification, information flows that represent normal network traffic are ignored and only the attack information flows are used to evaluate the proposed methods.

The strategy followed has been the same described in the binary classification step. In other words, it has been used the same features set, the same stratified 10-Folds cross-validation approach for majority class, the 60% of data for training in the case of minority classes and the same model hyperparameters. Since a multi-class classification is carried out, the only difference is the labels are the attack names associated to the malicious observations defined in the original dataset.

Finally, it is worthy to mention that standard scaling is applied to normalize the dataset.

4. Results and evaluation of the proposed solution

This chapter details the evaluation of the innovative solution proposed in this work to address novelty detection and early classification of malicious activities in industrial environments. The main discussions concern different techniques for the machine learning algorithms pipeline that have been implemented and evaluated. In brief, it can be stated that the final aim of the adopted solution is to detect properly both new zero-day attacks and well-known attacks as soon as possible along with a very low false positive rate.

In addition, for the sake of completeness, a general description of the most common evaluation metrics is provided. Since they are useful for the validation and comparison of the efficiency and performance of the models tested under different conditions.

4.1 Evaluation metrics

The primary goal of a classification algorithm in the context of network intrusion detection is to achieve the highest level of accuracy with the lowest number of false positives [57]. In addition to the overall accuracy, there are other important metrics for the evaluation of the proposed solution such as recall, precision, F1-score, AUC (Area under the ROC curve) and confusion matrix. In this work, the accuracy is not enough to properly assess the reliability of the system since the dataset presents a high imbalance in the anomalous traffic. These performance metrics are defined on top of the following concepts:

- Negative label or class refers to benign or normal traffic
- Positive label refers to intrusion or malicious traffic
- True Negative (TN) refers to the correctly predicted normal traffic
- False Negative (FN) refers to samples of the intrusion traffic wrongly predicted as normal
- True Positive (TP) refers to the correctly predicted malicious traffic
- False Positive (FP) refers to samples of the normal traffic wrongly predicted as intrusion

These terms can be depicted as a confusion matrix, where each row represents an actual class, while each column represents a predicted class, as it is shown in the next picture.

		Prediction	
		Benign	Malicious
Ground Truth	Benign	True Negative (TN)	False Positive (FP)
	Malicious	False Negative (FN)	True Positive (TP)

And the mathematical formulas of the metrics are the following:

$$FP\ rate = \frac{FP}{TN + FP}$$

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

$$recall^{1} = \frac{TP}{TP + FN}$$

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

$$F1 - score = 2 * \frac{precision * recall}{precision + recall}$$

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

Finally, the Receiver Operating Characteristic curve (ROC) is a probability curve for testing the performance of a classification model at different threshold settings, and the Area Under de Curve (AUC) represents the degree or measure of classes separability. Higher the AUC, better the model predictions. The ROC curve is plotted with True Positive Rate (TPR) against False Positive Rate (FPR) where TPR is on the y-axis and FPR is on the x-axis.

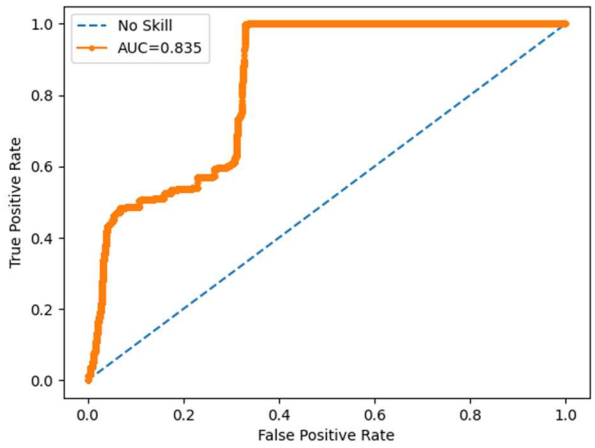
In brief, the accuracy is not sufficient by itself since it can often be a misleading oversimplification. The recall, or the true positive rate, tries to answer the question: What proportion of actual positives was identified correctly? While the precision, or positive predictive value, the question: What proportion of positive identifications was actually correct? The F1-score is considered as a harmonic mean that combines the precision and recall values. AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

4.2 Anomaly detection based on AutoEncoders

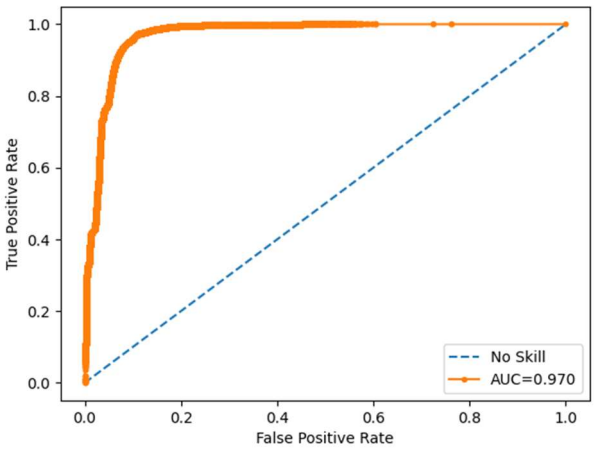
At the first stage of the machine learning pipeline, an AutoEncoder has been trained to improve the performance of IDS solutions at identifying new zero-day attacks. As it has been explained at chapter 3, the CICIDS2017 dataset is highly imbalanced. Therefore, some attacks will be more difficult to detect compared to more prevalent classes since we are carrying out an unsupervised learning task, in which data augmentation or other strategies cannot be applied.

At this point, the first aim was to detect the set of features along with the right architecture of the AE that provided the best results. Regarding the selection of the feature set, it was discovered that the most influent features where the ones related to the encoding of IPs and ports. While, in terms of AE architecture, the main issue was to choose between a symmetric or asymmetric architecture. After several trials, the main results are shown in the next images where the Area Under de Curve metric has been used to compare the performance achieved by the trained algorithms during the tests carried out.

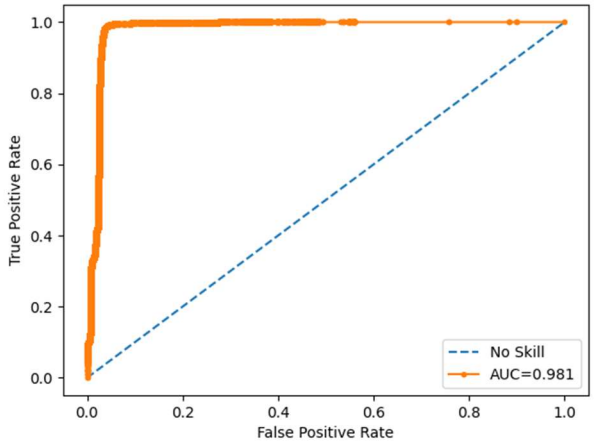
[1] Recall is also known as True Positive Rate (TPR) and Detection Rate



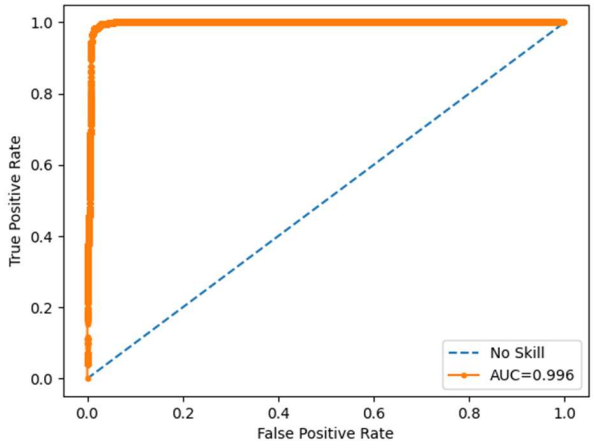
Test#A: Symmetric AE architecture (170, 35, 35, 170)
1000 most frequent IPs and infrequent IPs flag
one-hot for the 20 most frequent ports and infrequent ports flag



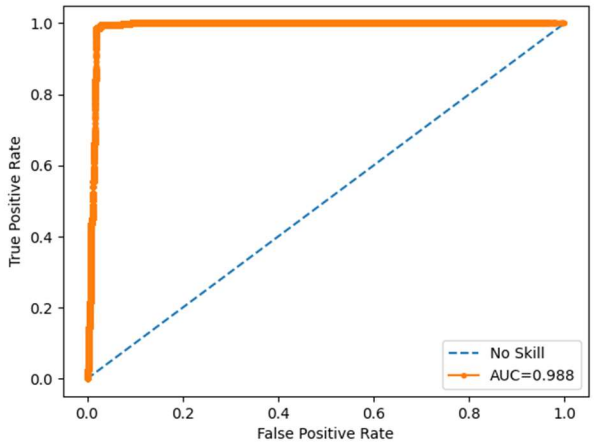
Test#B1
1000 most frequent IPs and infrequent IPs flag
custom encoding of ports and infrequent ports flag



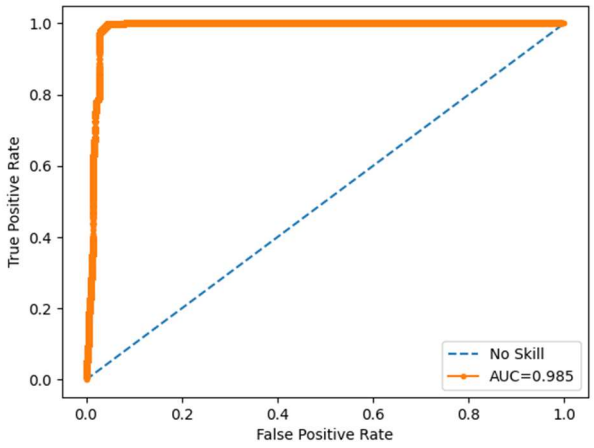
Test#B2
IPs as four independent numbers
custom encoding of ports and infrequent ports flag



Test#B4
1000 most frequent IPs and infrequent IPs flag
one-hot for the 20 most frequent ports and infrequent ports flag



Test#B5
1000 most frequent IPs and infrequent IPs flag
one-hot for the 20 most frequent origin and destination ports and
infrequent origin and destination ports flags



Test#B4.1.2
1000 most frequent IPs
one-hot for the 20 most frequent ports and infrequent ports flag

Figure 17. Results of some tests carried out with the AutoEncoder model

From these results, firstly, it is concluded that asymmetric architectures are more useful to detect anomalous behaviours. The reason is asymmetric architectures multiply the discordances between real observations and their provided reconstructions. We have to take in mind that the AutoEncoder has been training on the normal or benign samples. And, therefore, small variations in the predictions by the neurons at the hidden layers will grow through the next layers to the point of highlighting large reconstruction errors.

Secondly, traditional encoding of IPs as sequence numbers by using the *LabelEncoder* class reports the worst results since no information related to the architecture of the communication network is stored. On the other hand, the feature "encoding of IPs as four independent numbers" is a pretty good option. However, the combination of the features "encode of the 1000 most frequent IPs" and "infrequent IPs flag" are the best choice since they improve considerably the value of the AUC metric. These great results show that the model is not overfitted thanks to the selection of 1000 IPs.

Thirdly, traditional encoding of ports as integer numbers (i.e. considering their values) reports the worst results since no information related to the services they represent is not stored. While the combination of features "custom encoding of ports" and "infrequent ports flag" achieves pretty good results. However, the best option is the application of features "one-hot for the 20 most frequent ports" and "infrequent ports flag". These great results show that the model is not overfitted thanks to the selection of 20 ports.

In summary, indeed, the best solution is given by the TestB#4 which consists of training the AutoEncoder with the asymmetric architecture described previously and the original feature set with two main modifications. On the one hand, the IPs features have been replaced by the "1000 most frequent IPs" and "infrequent IPs flag" features. On the other hand, the port features have been replaced by the "one-hot for the 20 most frequent ports" and "infrequent ports flag" features.

Therefore, once the feature set and the hyperparameters of the AutoEncoder have been fixed, the second aim has been to train the AutoEncoder under such conditions with the baseline dataset and to evaluate its performance with the zero-day attacks dataset. As results, the reconstruction errors of the baseline and zero-day attacks datasets are shown in the next pictures where blue colour is used for benign samples and red for the anomalous. The threshold is represented by the red horizontal line. As it is expected, the reconstruction errors of the benign samples are the lowest. And, surprisingly, in the second picture, it is very relevant that some anomalous observations have low reconstruction error on Thursday (06/07/17) and Friday (07/07/17). In fact, these samples belong to Infiltration and Botnet attacks respectively, which are not well detected by the AutoEncoder.

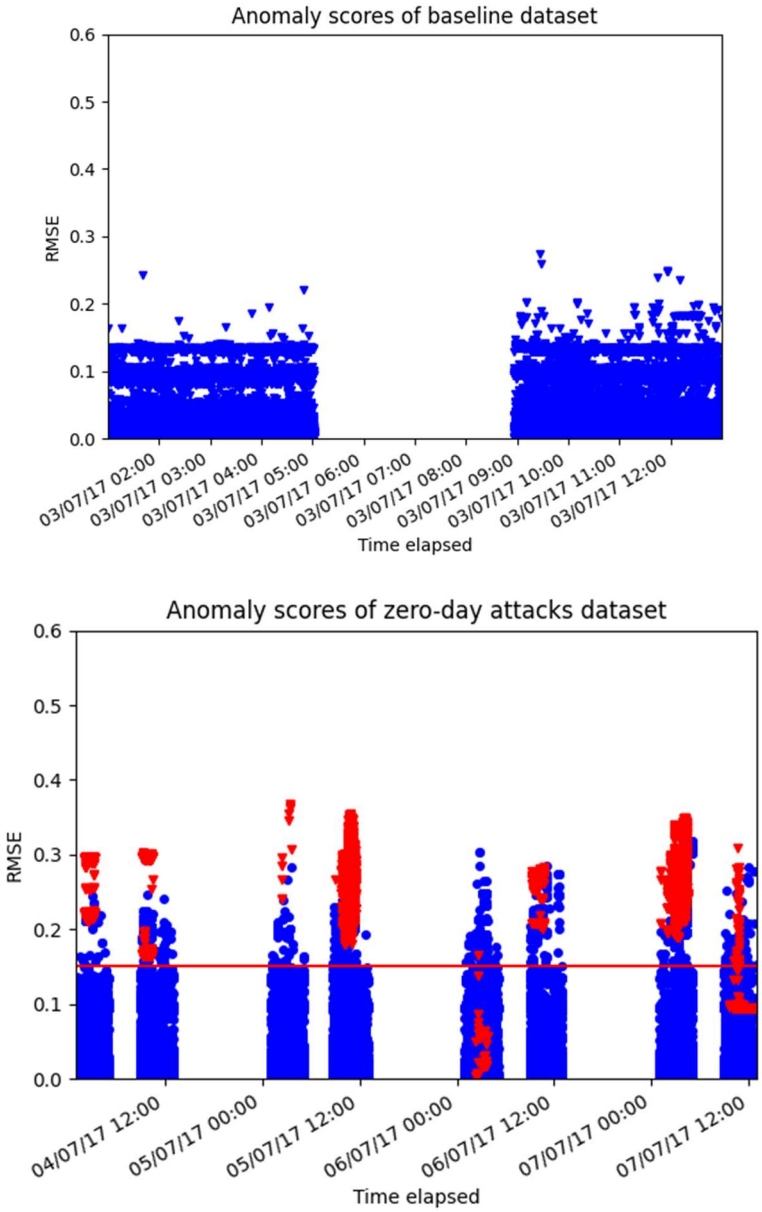


Figure 18. AutoEncoder reconstruction errors with threshold

In the picture above, the red horizontal line represents the best threshold to separate adequately benign observations from anomalous, which has been set up to a value of 0.15003404215567. By applying this threshold, the AutoEnconder presents the following evaluation metric results.

Table 5. AutoEncoder evaluation results for the zero-day attacks dataset

Metric	**Cluster BENIGN**	**Cluster ANOMALOUS**
Precision	0.9987	0.9229
Recall	0.9721	0.9963
F1	0.9852	0.9582
Global Accuracy	0.9782	

Regarding the confusion matrix, the AutoEncoder delivers very good performance since it produces a rate of false positives of 2.78% and a rate of false negatives of 0.37%, being the attack types that are undetected Infiltration (30 samples) and Botnet (1911 samples).

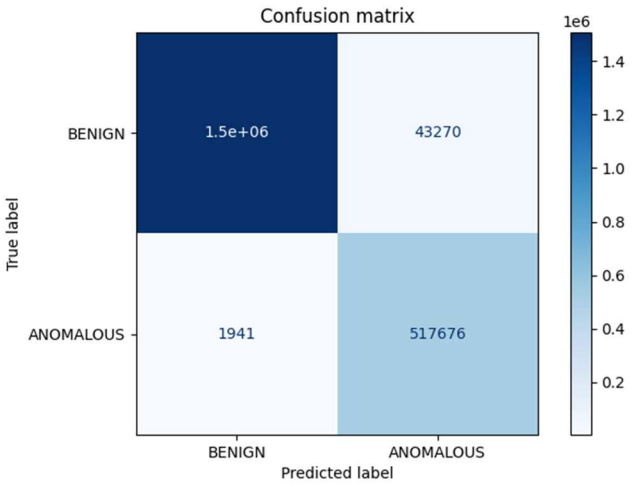


Figure 19. Confusion matrix of the AutoEncoder for the zero-day attacks dataset

4.3 Binary classification based on Naïve Bayes, SVM and Random Forest

This section includes the results of the training and evaluation of several models carried out during the second stage. Firstly, the binary classification is carried out over the observations marked as Benign by the AutoEncoder. And later on, the same process is preformed over the observations marked as Anomalous by the AutoEncoder.

4.3.1 Binary classification of samples predicted as Benign

For making the decision about the best algorithm to classify the observations as Benign or Attacks, the three methods: Naïve Bayes, Support Vector Machines and Random Forests have been trained over the partition of data marked previously as Benign by the AutoEncoder. It has been applied a stratified 10-folds cross-validation approach where 90% of data is used for training. This cross-validation method returns 10 different stratified folds in which the percentage of samples for each majority class is preserved. However, the 60% of data are for training and the remaining 40% for testing in the case of Infiltration attacks since there are very few samples. In fact, this dataset is composed of the following observations:

Table 6. First subset of the zero-day attacks dataset

Label	**Zero-day attacks (Tuesday-Friday)**
Benign	1509188
Bot	1911
Infiltration	30

A number of experiments and trials were conducted to train the models with the right configurations and hyperparameters as it has been described at chapter 3. And the resulting evaluation metrics of precision, recall, F1 and global accuracy per every method are shown below. As it can be observed, the NB model presents the worst performance. However, the RF is the best method since it presents the best F1-score which demonstrates that it is able to classify adequately both benign and attacks samples.

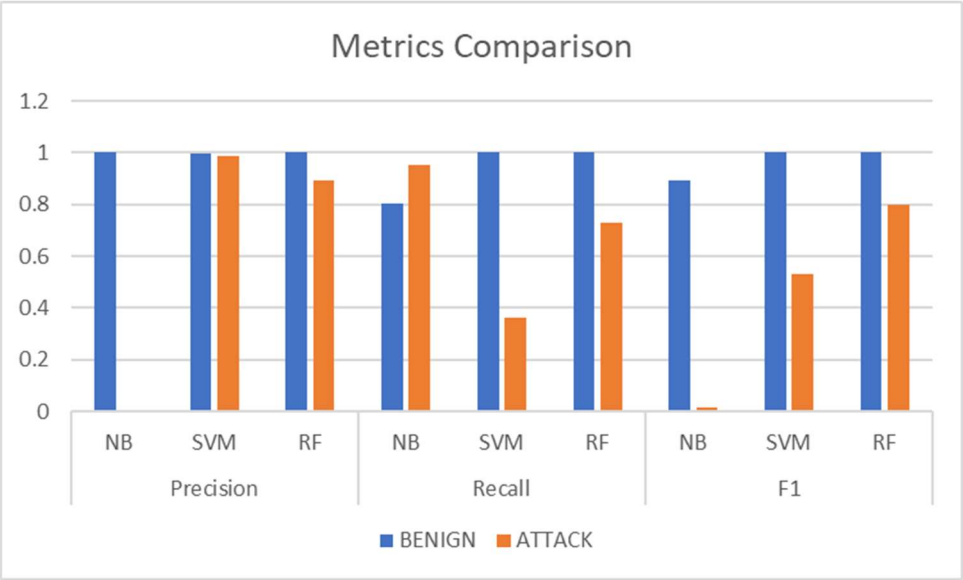


Figure 20. Evaluation metrics for binary classification of samples predicted as Benign by the AE

Table 7. Global expected accuracy

Model	Expected Accuracy
NB	0.8051
SVM	0.9991
Random Forest	0.9995

Once the Random Forest technique has been chosen, a new Random Forest model has been trained over the first partition of the zero-day attacks dataset. But this time, only the 10% of this subset has been used for training. The idea behind has been to reserve as much as possible observations for testing and, consequently, for the detection of their specific attacks at the third stage of the machine learning pipeline. As a result of performing the train with such reduced dataset, the classification of malicious observations is negatively affected. Since, as it can be seen by comparing the given data, the F1 of the malicious class has decreased from 0.8005 to 0.7061.

Table 8. Evaluation results of the Random Forest model (trained with 10% of samples)

Metric	Class BENIGN	Class ATTACK
Precision	0.9995	0.8442
Recall	0.9999	0.6068
F1	0.9997	0.7061
Global Accuracy	0.9994	

Regarding the confusion matrix, the Random Forest delivers good performance since it produces a rate of false positives of 0.01% and a rate of false negatives of 39.32%, being the attack types that are undetected Infiltration (5 samples) and Botnet (676 samples).

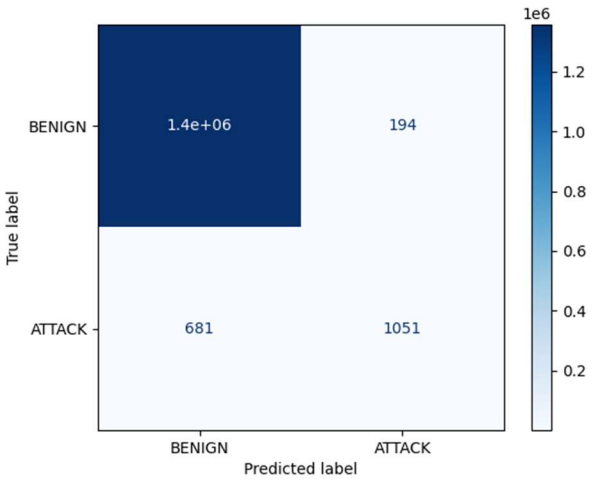


Figure 21. Confusion matrix of the Random Forest for the partition predicted as Benign by the AE

4.3.2 Binary classification of samples predicted as Anomalous

For making the decision about the best algorithm to classify the observations as Benign or Attacks, the three methods: Naïve Bayes, Support Vector Machines and Random Forests are trained over the partition of data marked previously as Anomalous by the AutoEncoder. It has been applied a stratified 10-folds cross-validation approach where 90% of data is used for training. This cross-validation method returns 10 different stratified folds in which the percentage of samples for each majority class is preserved. However, the 60% of data are for training and the remaining 40% for testing in the case of Heartbleed and Bot (which also includes the Infiltration sample) attacks since there are very few samples. In fact, this dataset is composed of the following observations

Table 9. Second subset of the zero-day attacks dataset

Label	**Zero-day attacks (Tuesday-Friday)**
Benign	43270
Heartbleed	6
Web Att	2180
PortScan	158798
DDoS	102649
DoS	240171
Bot	45
FTP-Patator	7931
SSH-Patator	5895
Infiltration	1

A number of experiments and trials were conducted to train the models with the right configurations and hyperparameters as it has been described at chapter 3. And the resulting evaluation metrics of precision, recall, F1 and global accuracy per every method are shown below. As it can be observed, the NB model presents again the worst performance. However, SVM and RF are the best methods since they are able to classify almost perfectly both benign and attacks samples. However, the training of the SVM model requires very long time. Taking in mind this fact and the aim of providing a uniform solution, the Random Forest technique has been finally chosen.

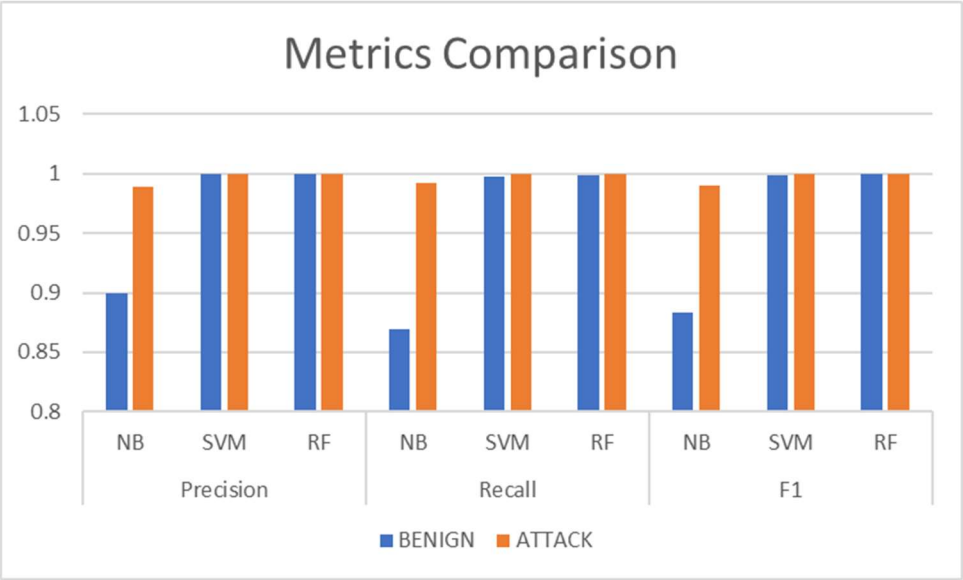


Figure 22. Evaluation metrics for binary classification of samples predicted as Anomalous by the AE

Table 10. Global expected accuracy

Model	Expected Accuracy
NB	0.9823
SVM	0.9998
Random Forest	0.9999

Once the Random Forest technique has been selected, a new Random Forest model has been trained over the second partition of the zero-day attacks dataset. But this time, only the 10% of this subset has been used for training. The idea behind has been to reserve observations for testing as much as possible and, consequently, for the detection of their specific attacks at the third stage of the machine learning pipeline. It is noticeable that performing the train with such reduced dataset has not any negative impact on the classification results.

Table 11. Evaluation results of the Random Forest model (trained with 10% of samples)

Metric	Class BENIGN	Class ATTACK
Precision	0.9999	0.9997
Recall	0.9963	1.0000
F1	0.9981	0.9998
Global Accuracy	0.9997	

Regarding the confusion matrix, the Random Forest delivers good performance since it produces a rate of false positives of 0.37% and a rate of false negatives of nearly zero (8.6e-04) %. Therefore, this classifier achieves excellent results. The only 4 samples of attack that are not well classified belong to DoS Slowhttptest (2 samples), Web Attack – XSS (1 sample) and Bot (1 sample).

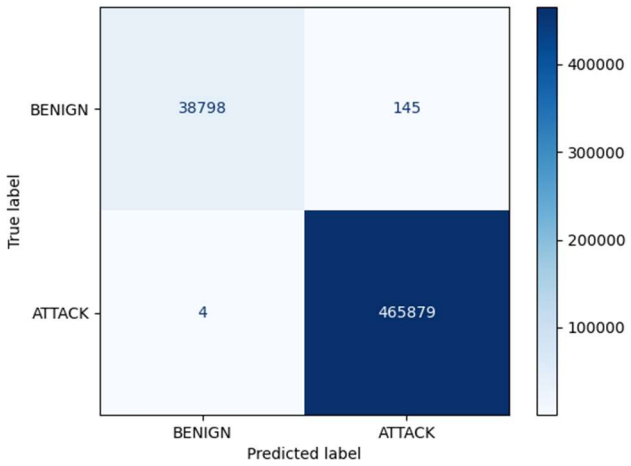


Figure 23. Confusion matrix of the Random Forest for the partition predicted as Anomalous by the AE

4.4 Multiclass classification based on Naïve Bayes, SVM and Random Forest

At the third stage of the machine learning pipeline, the application of several models to predict specific types of attack has been carried out. Therefore, a multiclass classification has been trained on the observations marked as ATTACK at the previous stage. At this point, Naïve Bayes, Support Vector Machines and Random Forests models have been evaluated.

During the tests, it has been applied a stratified 10-folds cross-validation approach where 90% of data is used for training. This cross-validation method returns 10 different stratified folds in which the percentage of samples for each majority class is preserved. However, the 60% of data are for training and the remaining 40% for testing in the case of Heartbleed and Infiltration attacks since there are very few samples. In fact, this dataset is composed of the following observations:

Table 12. Dataset of attack samples at the third stage

Label	**Known attacks (Tuesday-Friday)**
Heartbleed	3
Web Att	1961
PortScan	142918
DDoS	92384
DoS	216152
Bot	1061
FTP-Patator	7138
SSH-Patator	5306
Infiltration	7

At this phase, as there are a little number of Heartbleed and Infiltration samples, these two attack classes are treated jointly in the experiments.

After conducting the trials to train the models with the right configurations and hyperparameters, the resulting evaluation metrics of precision, recall, F1 and global accuracy per every method are shown below. As it can be observed, the NB model once again presents the worst performance. Although SVM and RF present similar results in general, it should be highlighted that SVM is not able to detect the

Infiltration and Heartbleed attacks, which are minority attacks. Therefore, the RF is considered the best method since it is able to classify almost perfectly all attack samples.

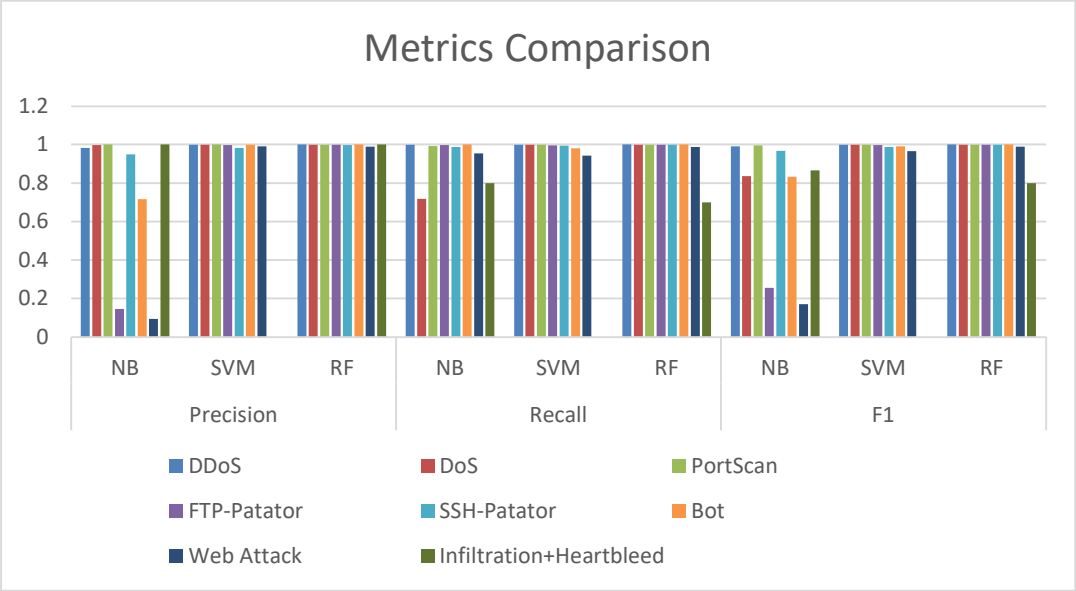


Figure 24. Evaluation metrics for multi-class classification of samples predicted as Attack

Table 13. Global expected accuracy

Model	Expected Accuracy
NB	0.8665
SVM	0.9994
Random Forest	0.9998

From the figures above, it is concluded the RF shows a very good performance and contributes properly to construct a uniform machine learning pipeline. It is also remarkable that the Random Forest technique will allow not only to detect specific known attacks but also benign samples by applying a thumb rule, since it is possible to calculate the attack class probabilities of a sample thanks to the function *predict_proba()* of the *sklearn.ensemble.RandomForestClassifier* class. Where the final predicted class is the one that has the highest probability. However, the highest probability always is below 0.5 for benign samples. Therefore, it will be possible to reduce the number of false positives by marking such benign samples in this way.

4.5 Summary

In this chapter, the results and full evaluation of the proposed machine learning pipeline have been presented in detail. The aim of this holistic solution is twofold: to detect zero-day attacks as early as possible and, once attacks are known , to identify specific attacks that are ongoing. Moreover, it is a primary requirement to reduce the false positive rate since they overwhelm the security engineers' daily work. From a holistic point of view, this solution could be considered as an evolutionary system which covers a wide variety of intrusion detection techniques while it can evolve as security engineers give feedbacks about treated incidents.

As the first stage of the machine learning pipeline, the AutoEncoder model proposed shows the ability to detect accurately the majority of the unknown attacks presented in the CICIDS 2017 dataset.

Therefore, it has been demonstrated the utility of this unsupervised technique as a first layer of defence, which is very useful in an ever-growing attack landscape.

At the second and third stages of the machine learning pipeline, the Naïve Bayes, Support Vector Machines and Random Forest techniques have been evaluated. In both cases, the experiments show that Random Forest outperforms the other techniques when classifying network flows as benign or attack (and also specific attacks). These two sequential phases are of fundamental importance to detect specific well-known attacks while reducing considerably the false positive rate. Therefore, this supervised technique has demonstrated its utility as a second layer of defence.

On the other hand, all these good results prove the new features proposed by this work complement perfectly the features provided in the network flow statistics of the CICIDS 2017 dataset. These new features are mainly related to the right treatment of the IP address and port categorical variables, which allow to learn relevant information related to the communications architecture and servers that are usual targets of adversaries.

In summary, the proposed solution reduces to a negligible value the false positive rate while achieves very good evaluation metrics in specific attack detection: precision of 0.998425, recall of 0.9607375 and f1 score of 0.9733375. Therefore, the system performs well for those categories that have a representative number of samples. However, it achieves a limited detection regarding the infiltration and botnet classes when these samples are very similar to benign ones. In fact, the 31.25% of infiltration and the 77.52% of botnet samples are wrongly evaluated. But it must be considered that the model was trained with only the 10% of samples under such evaluation tests. Therefore, these figures would be improved as more attack data become available.

5. Conclusions

5.1 Key contributions

The goal of this master thesis is to apply machine learning and deep leaning techniques that allow the normality space definition to address novelty detection and early classification of malicious activities on ICS. The proposed solution operates in 3 stages looking for similarities shown in industrial network traffic modelized by an enhanced feature set.

Specifically, the following key contributions have been provided in this final master's project:

- Selection of the open dataset CICIDS2017 that is based on bidirectional network flow data. This benchmark intrusion detection dataset covers 14 contemporary attack types and fully supports the application of both unsupervised and supervised anomaly-based detection techniques.
- A clean and enhanced version of the CICIDS2017 dataset. After analysing deeply this dataset, firstly, it has been clean by removing redundant, incomplete, and inconsistent data. In addition, some data errors have been repaired. Secondly, a custom process has been setup to detect and remove outliers with respect to values registered in the baseline dataset, which contains the state of normal system operation. Finally, the feature set has been extended through the application of different treatments to the source and destination IP addresses and ports features. These new datasets called as baseline and zero-day attacks also have the bidirectional flows ordered chronologically, what is not available in the original dataset. The projection of the original and zero-day attacks datasets with the PCA technique shows graphically how these efforts will allow a better separation of benign and malicious observations.
- An AutoEncoder model for anomaly detection based on an unsupervised deep learning technique. This novel model learns normal behaviours and is able to detect abnormal activities without generating numerous false positives. Moreover, several tests have been carried out to prove that asymmetric architectures produce considerably better results than symmetric architectures commonly used in the state-of-the-art.
- Two binary classifiers based on Random Forests for reducing the false positive and false negative rates. By applying this supervised technique is possible to improve the detection of benign and attack observations done previously by the AutoEncoder. These models will refine the results and, thus, detect almost all benign samples. Therefore, it will allow security engineers to focus only on real attacks.
- A multi-class classifier based on Random Forests for making the final decision about intrusion detection. This model determines the specific type of attack carried out. Therefore, it is of paramount importance since it is necessary to known exactly the type of an ongoing attack to be able to react properly from the security operation point of view. Additionally, this model can discard benign samples considered as attack at the previous detection phase.
- In summary, the solution addresses the challenges of detecting both zero-day and known-well attacks. From a holistic point of view, this solution could be considered as an evolutionary system which covers a wide variety of intrusion detection techniques while it can evolve as security engineers provide feedbacks about treated incidents to the system.

5.2 Future lines of research

This work could be further improved by researching on the following directions among others:

- Detect intrusions following a time-series analysis approach since attackers usually follow a sequence of actions to be able to compromise an infrastructure. These adversaries' tactics, techniques, and procedures (TTP) have been described in the knowledge base of ATT&CK for Industrial Control Systems [58]. Therefore, it could be interesting to improve the models by including new categorical features related to the timestamp of samples. It would be also interesting to replace the AutoEncoder by a hybrid model like, for example, Long Short-Term Memory AutoEncoder (LSTM-AE) to identify anomalies by computing the difference between the reconstructed and the respective real sequences. Following this recommendation, it could be improved the detection of attack samples that are very similar to benign observations and, thus, the AutoEncoder does not recognize properly.
- Explore techniques that promote the explainability and interpretability in the intrusion detection systems. Nowadays, the adoption of solutions based on Artificial Intelligence is not appropriate. Moreover, there are two relevant issues in the security of ICS domain: the ever-growing attacks to critical infrastructures and the lack of security engineers. Therefore, it is paramount to provide and promote the adoption of AI systems in this area. In this sense, it will be relevant to provide at some extent the explainability in intrusion detection systems. In this way, security engineers will probably embrace this technology since their comprehension about them will improve considerable.
- The new dataset CSE-CIC-IDS2018 [22], which is delivered by the same organisation that CICIDS2017, could be explored and used to analyse the behaviour of the solution proposed in this work. This dataset is huge and needs more hardware and computing resources. Although, it would be required to retrain the system, indeed, it could be a good test bench for the evaluation of the scalability of the adopted solution.

6. Bibliography

[1] Maglaras, Leandros A., et al. "Cyber security of critical infrastructures." Ict Express 4.1 (2018): 42-45.

[2] Stouffer, Keith, Joe Falco, and Karen Scarfone. "Guide to industrial control systems (ICS) security." NIST special publication 800.82 (2011): 16-16.

[3] infoPLC. "Estado de la Ciberseguridad Industrial 2018 en España." (2019). Available at: https://www.infoplc.net/actualidad-industrial/item/106194-estado-ciberseguridad-industrial-2018-espana

[4] "EEF. 2018. Cyber Security for Manufacturing. Available at: https://www.eef.org.uk/resources-and-knowledge/research-and-intelligence/industry-reports/cyber-security-for-manufacturers"

[5] European Cyber Security Organisation . "Cyber Security for the Industry 4.0 and ICS Sector." ECSO; Brussels, Belgium: 2018.

[6] Chapman, C. "Chapter 1—Introduction to Practical Security and Performance Testing." Network Performance and Security; Chapman, C., Ed.; Syngress: Boston, MA, USA (2016): 1-14.

[7] Susto, Gian Antonio, Angelo Cenedese, and Matteo Terzi. "Time-series classification methods: Review and applications to power systems data." Big data application in power systems (2018): 179-220.

[8] Sharafaldin, Iman, Arash Habibi Lashkari, and Ali A. Ghorbani. "A detailed analysis of the CICIDS2017 data set." International Conference on Information Systems Security and Privacy. Springer, Cham, 2018.

[9] Brown, Carson, et al. "Analysis of the 1999 DARPA/Lincoln laboratory IDS evaluation data with NetaDHICT." 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. IEEE, 2009.

[10] Tavallaee, Mahbod, et al. "A detailed analysis of the KDD CUP 99 data set." 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE, 2009.

[11] McHugh, John. "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory." ACM Transactions on Information and System Security (TISSEC) 3.4 (2000): 262-294.

[12] Nehinbe, Joshua Ojo. "A Simple Method for Improving Intrusion Detections in Corporate Networks." International Conference on Information Security and Digital Forensics. Springer, Berlin, Heidelberg, 2009.

[13] Shiravi, Ali, et al. "Toward developing a systematic approach to generate benchmark datasets for intrusion detection." computers & security 31.3 (2012): 357-374.

[14] Proebstel, Elliot Parker. "Characterizing and Improving Distributed Network-based Intrusion Detection Systems(NIDS): Timestamp Synchronization and Sampled Traffic." Diss. University of California, Davis, 2008.

[15] Nechaev, B., et al. "Lawrence Berkeley National Laboratory (LBNL)/ICSI Enterprise Tracing Project." Berkeley, CA: LBNL/ICSI (2004).

[16] Song, Jungsuk, et al. "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation." Proceedings of the first workshop on building analysis datasets and gathering experience returns for security. 2011.

[17] Sato, Masaaki, Hirofumi Yamaki, and Hiroki Takakura. "Unknown attacks detection using feature extraction from anomaly-based ids alerts." 2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet. IEEE, 2012.

[18] Chitrakar, Roshan, and Chuanhe Huang. "Anomaly based intrusion detection using hybrid learning approach of combining k-medoids clustering and naive bayes classification." 2012 8th International Conference on Wireless Communications, Networking and Mobile Computing. IEEE, 2012.

[19] Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." 2015 military communications and information systems conference (MilCIS). IEEE, 2015.

[20] Gharib, Amirhossein, et al. "An evaluation framework for intrusion detection dataset." 2016 International Conference on Information Science and Security (ICISS). IEEE, 2016.

[21] Sharafaldin, Iman, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward generating a new intrusion detection dataset and intrusion traffic characterization." ICISSp. 2018.

[22] CSE-CIC-IDS2018 on AWS. A collaborative project between the Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC). Available at: https://www.unb.ca/cic/datasets/ids-2018.html

[23] Tidjon, Lionel N., Marc Frappier, and Amel Mammar. "Intrusion detection systems: A cross-domain overview." IEEE Communications Surveys & Tutorials 21.4 (2019): 3639-3681.

[24] Hodo, Elike, et al. "Shallow and deep networks intrusion detection system: A taxonomy and survey." arXiv preprint arXiv:1701.02145 (2017).

[25] Axelsson, Stefan. "Intrusion detection systems: A survey and taxonomy. " Vol. 99. Technical report, 2000.

[26] Lunt, Teresa F. "A survey of intrusion detection techniques." Computers & Security 12.4 (1993): 405-418.

[27] Qayyum, A., M. H. Islam, and M. Jamil. "Taxonomy of statistical based anomaly detection techniques for intrusion detection." Proceedings of the IEEE Symposium on Emerging Technologies, 2005. IEEE, 2005.

[28] Garcia-Teodoro, Pedro, et al. "Anomaly-based network intrusion detection: Techniques, systems and challenges." Comput. Secur. 28.1-2 (2009): 18-28.

[29] Vijayanand, R., D. Devaraj, and B. Kannapiran. "Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection." Comput. Secur. 77 (2018): 304-314.

[30] Watson, Gavin. "A Comparison of Header and Deep Packet Features When Detecting Network Intrusions." Technical Report; University of Maryland: College Park, MD, USA, 2018.

[31] Kim, Jihyun, et al. "Long short-term memory recurrent neural network classifier for intrusion detection." 2016 International Conference on Platform Technology and Service (PlatCon). IEEE, 2016.

[32] Zhu, Jiang, et al. "Mechanism of situation element acquisition based on deep auto-encoder network in wireless sensor networks." International Journal of Distributed Sensor Networks 13.3 (2017): 1550147717699625.

[33] Min, Erxue, et al. "SU-IDS: A Semi-supervised and Unsupervised Framework for Network Intrusion Detection." International Conference on Cloud Computing and Security. Springer, Cham, 2018.

[34] Hindy, Hanan, et al. "Towards an Effective Zero-Day Attack Detection Using Outlier-Based Deep Learning Techniques." arXiv preprint arXiv:2006.15344 (2020).

[35] Bilge, Leyla, and Tudor Dumitraş. "Before we knew it: An empirical study of zero-day attacks in the real world." Proceedings of the 2012 ACM conference on Computer and communications security. 2012.

[36] Sharma, Vishal, et al. "A framework for mitigating zero-day attacks in IoT." arXiv preprint arXiv:1804.05549 (2018).

[37] Sun, Xiaoyan, et al. "Using Bayesian networks for probabilistic identification of zero-day attack paths." IEEE Transactions on Information Forensics and Security 13.10 (2018): 2506-2521.
[38] Zhou, Q., and D. Pezaros. "Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection–An Analysis on CIC-AWS-2018 dataset. arXiv 2019." arXiv preprint arXiv:1905.03685.
[39] Mirsky, Yisroel, et al. "Kitsune: an ensemble of autoencoders for online network intrusion detection." arXiv preprint arXiv:1802.09089 (2018).
[40] Shone, Nathan, et al. "A deep learning approach to network intrusion detection." IEEE transactions on emerging topics in computational intelligence 2.1 (2018): 41-50.
[41] Zhao, Juan, et al. "Transfer learning for detecting unknown network attacks." EURASIP Journal on Information Security 2019.1 (2019): 1-13.
[42] Sameera, Nerella, and M. Shashi. "Deep transductive transfer learning framework for zero-day attack detection." ICT Express 6.4 (2020): 361-367.
[43] Abri, Faranak, et al. "The Performance of Machine and Deep Learning Classifiers in Detecting Zero-Day Vulnerabilities." arXiv preprint arXiv:1911.09586 (2019).
[44] Kim, Jin-Young, Seok-Jun Bu, and Sung-Bae Cho. "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders." Information Sciences 460 (2018): 83-102.
[45] Leevy, Joffrey L., and Taghi M. Khoshgoftaar. "A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data." Journal of Big Data 7.1 (2020): 1-19.
[46] ILNAS. "Artificial Intelligence Technology, Use Cases and Applications, Trustworthiness and Technical Standardization V. 1.1" (2021) Available at: https://portail-qualite.public.lu/dam-assets/publications/normalisation/2021/ilnas-white-paper-artificial-intelligence.pdf
[47] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.
[48] Chaabouni, Nadia. Intrusion detection and prevention for IoT systems using Machine Learning. Diss. Université de Bordeaux, 2020.
[49] NumPy. The fundamental package for scientific computing with Python. Available at: https://numpy.org/
[50] Pandas - Python Data Analysis Library. Available at: https://pandas.pydata.org/
[51] Scikit-learn: machine learning in Python. Available at: https://scikit-learn.org/stable/
[52] Keras: the Python deep learning API. Available at: https://keras.io/
[53] TensorFlow. Available at: https://www.tensorflow.org/?hl=es-419
[54] Li, Jundong, et al. "Feature selection: A data perspective." ACM Computing Surveys (CSUR) 50.6 (2017): 1-45.
[55] Locklin, S. "Neglected machine learning ideas." (2017). Available at: https://scottlocklin.wordpress.com/2014/07/22/neglected-machine-learning-ideas/
[56] Jake Teo. "Data Science in Python: Chapter 5. Feature Normalization." Available at: https://python-data-science.readthedocs.io/en/latest/normalisation.html
[57] Fernandez, Gabriel. "Deep Learning Approaches for Network Intrusion Detection." Diss. Ph. D. Thesis, The University of Texas at San Antonio, San Antonio, TX, USA, 2019.
[58] The MITRE Corporation. "ATT&CK for Industrial Control Systems." (2021). Available at: https://collaborate.mitre.org/attackics/index.php/Overview

Annex Acronyms

AE	AutoEncoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
APT	Advanced Persistent Threat
AUC	Area Under de Curve
CI	Critical Infrastructure
CIC	Canadian Institute for Cybersecurity
CPS	Cyber Physical System
DBScan	Density-Based Spatial clustering of applications with noise
DL	Deep Learning
DR	Detection Rate
FN	False Negative
FP	False Positive
FPR	False Positive Rate
HID	Host-based IDS
ICS	Industrial Control System
IDS	Intrusion Detection System
IF	Isolation Forest
IP	Internet Protocol
IQR	Interquartile Range
LOF	Local Outlier Factor
LSTM	Long Short-Term Memory
LSTM-AE	Long Short-Term Memory AutoEncoder
ML	Machine Learning
NADS	Network Anomaly Detection System
NB	Naïve Bayes
NIDS	Network-based IDS
OSVM	One-Class SVM
PCA	Principal Component Analysis
PCAP	Packet CAPture
RF	Random Forest
ROC	Receiver Operating Characteristic curve

SVM	Support Vector Machines
TN	True Negative
TP	True Positive
TPR	True Positive Rate
TTP	Tactics, Techniques and Procedures
UNB	University of New Brunswick