


Università degli Studi di Napoli Federico II

Ph.D. Program in

Information Technology and Electrical Engineering

XXXVII Cycle

Thesis for the Degree of Doctor of Philosophy

Enhancing Internet of Things Security:

Developing and Validating Sustainable Intrusion Detection Strategies

by

De Vivo Simona

Advisor: Prof. Domenico Cotroneo

Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

To myself,
for everything.

Enhancing Internet of Things Security: Developing and Validating Sustainable Intrusion Detection Strategies

Ph.D. Thesis presented
for the fulfillment of the Degree of Doctor of Philosophy
in Information Technology and Electrical Engineering
by

De Vivo Simona

December 2024

Approved as to style and content by

Prof. Domenico Cotroneo, Advisor

Università degli Studi di Napoli Federico II
Ph.D. Program in Information Technology and Electrical Engineering
XXXVII cycle - Chairman: Prof. Stefano Russo

http://itee.dieti.unina.it

Candidate's declaration

I hereby declare that this thesis submitted to obtain the academic degree of Philosophiæ Doctor (Ph.D.) in Information Technology and Electrical Engineering is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.
Parts of this dissertation have been published in international conference articles (see list of the author's publications at the end of the thesis).

Napoli, February 24, 2025

De Vivo Simona

La borsa di dottorato è stata cofinanziata con risorse del
Programma Operativo Nazionale Ricerca e Innovazione 2014 – 2020,
risorse FSE REACT-EU
Azione IV.4 “Dottorati e contratti di ricerca su tematiche dell’innovazione”
e Azione IV.5 “Dottorati su tematiche Green”.

Abstract

This PhD thesis tackles significant cybersecurity challenges posed by the IoT, a key element of Industry 4.0 that enables real-time communication between devices. The rapid growth of IoT systems, including resource-constrained devices, renders them vulnerable to sophisticated attacks like DoS and DDoS. Traditional security solutions, such as rule-based IDS and SIEM struggle to address these challenges, particularly in the context of Industry 5.0's sustainability goals, so this work aims to develop innovative, lightweight cybersecurity strategies tailored to IoT environments. A first contribution, obtained in collaboration with Digital Platforms S.p.A. company, includes a comparative evaluation of RTA and LimaCharlie SIEMs, revealing their limitations against sophisticated DDoS attacks. This analysis points to the need for ML techniques to enhance detection accuracy. For this reason, this work proposes a lightweight IDS using K-Means clustering, achieving 90% accuracy with only 1.5W power consumption to detect DoS attacks in onboard train networks. Additionally, this dissertation introduces DDoShield-IoT, developed at the University of North Carolina at Charlotte, an IDS testbed for developing IoT-tailored security solutions. This framework generates realistic traffic patterns, allowing to address the scarcity of high-quality IoT datasets through a Data Augmentation strategy that combines real and synthetic data. Lastly, to address privacy and sustainability, this thesis presents a novel intrusion detection methodology that blends User Profiling with FL, improving intrusion detection while reducing energy consumption. This approach proved effective in the real-world scenarios of the European CyberSEAS project aimed at protecting Smart Grid infrastructures. These contributions address current IoT security limitations and align with the principles of Industry 5.0.

Keywords: Internet of Things, Intrusion Detection System, GreenAI, Federated Learning, Cybersecurity, Sustainability

Sintesi in lingua italiana

Questa tesi di dottorato affronta le sfide significative della sicurezza informatica poste dall'IoT, un elemento chiave dell'Industria 4.0 che consente la comunicazione in tempo reale tra dispositivi. La rapida crescita dei sistemi IoT, inclusi i dispositivi con risorse limitate, li rende vulnerabili ad attacchi sofisticati come DoS e DDoS. Le soluzioni di sicurezza tradizionali, come IDS basati su regole e SIEM, hanno difficoltà ad affrontare queste sfide, in particolare nel contesto degli obiettivi di sostenibilità dell'Industria 5.0, quindi questo lavoro mira a sviluppare strategie di sicurezza informatica innovative e leggere su misura per gli ambienti IoT. Un primo contributo, ottenuto in collaborazione con la società Digital Platforms S.p.A., include una valutazione comparativa di SIEM RTA e LimaCharlie, rivelando i loro limiti contro attacchi DDoS sofisticati. Questa analisi evidenzia la necessità di tecniche di ML per migliorare la precisione del rilevamento. Per questo motivo, questo lavoro propone un IDS leggero che utilizza il clustering K-Means, ottenendo una precisione del 90% con un consumo energetico di soli 1,5 W per rilevare attacchi DoS nelle reti di bordo dei treni. Inoltre, questa tesi introduce DDoShield-IoT, sviluppato presso l'Università della Carolina del Nord a Charlotte, un banco di prova IDS per lo sviluppo di soluzioni di sicurezza personalizzate per IoT. Questo framework genera modelli di traffico realistici, consentendo di affrontare la scarsità di set di dati IoT di alta qualità tramite una strategia di Data Augmentation che combina dati reali e sintetici. Infine, per affrontare la privacy e la sostenibilità, questa tesi presenta una nuova metodologia di rilevamento delle intrusioni che fonde User Profiling con FL, migliorando il rilevamento delle intrusioni e riducendo al contempo il consumo di energia. Questo approccio si è dimostrato efficace negli scenari reali del progetto europeo CyberSEAS, volto a proteggere le infrastrutture Smart Grid. Questi contributi affrontano le attuali limitazioni della sicurezza IoT e si allineano ai principi dell'Industria 5.0.

Parole chiave: Internet of Things, Intrusion Detection System, GreenAI, Federated Learning, Cybersecurity, Sostenibilità

Acknowledgements

The research presented in this dissertation was supported by the Italian Ministry of University and Research (MIUR) through the PON Research and Innovation 2014-2020 program, Action IV.5 "Doctorates on green topics," as outlined in the CDA Resolution No. 73 of 29.04.2022. This funding is associated with the project UGOV: 000010—PON - DOTT GREEN ITEE 37 CICLO 001 005.
I am deeply grateful to my supervisor, Professor Domenico Cotroneo, for his unwavering support and guidance throughout my Ph.D. journey. His insights and advice have profoundly shaped this research and influenced my personal and professional development. I would also like to thank Dr. Pietro Liguori for his availability and assistance, Professor Roberto Natella for the opportunity to collaborate on research projects and assist in his teaching activities, and Professor Stefano Russo, coordinator of the Ph.D. program, for his expertise and immense knowledge. A Special thanks also to all members of the DESSERT research group for creating a stimulating environment for my research.
I am grateful to Professors Bojan Cukic and Dong Dai for their invaluable support during my research period abroad at the University of North Carolina at Charlotte, USA, and to Digital Platforms S.p.A. for their collaboration and support, both of which significantly contributed to the success of my research.

Contents

List of Acronyms

The following acronyms are used throughout the thesis.

ANOVA	Analysis of Variance
AI	Artificial Intelligence
CPU	Central Processing Unit
C&C	Command & Control Server
CNN	Convolutional Neural Network
XSS	Cross-Site Scripting
CPS	Cyber-Physical Systems
CPPS	Cyber-Physical Production Systems
CyberSEAS	Cyber Securing Energy dAta Services
DA	Data Augmentation
DL	Deep Learning
DoS	Denial of Service
DoE	Design of Experiment
DDoS	Distributed Denial of Service
DSL	Domain Specific Language

EPES	Electric Power Systems
FN	False Negatives
FNR	False Negatives Rate
FP	False Positives
FPR	False Positive Rate
FedAvg	Federated Averaging
FL	Federated Learning
GDPR	General Data Protection Regulation
HIDS	Host-based IDS
IIoT	Industrial Internet of Things
IoC	Indicators of Compromise
IT	Information Technology
IID	Installer ID
IoT	Internet of Things
IDS	Intrusion Detection System
IPS	Intrusion Prevention Systems
ML	Machine Learning
MaaS	Model as a Service
NLP	Natural Language Processing
NIDS	Network-based IDS
NFV	Network Function Virtualization
NSM	Network Security Monitor
OID	Organization ID
PCA	Principal Component Analysis
PR-AUC	Area Under the Precision-Recall Curve

RF	Random Forest
RTA	Real-Time Analytics
RBAC	Role-based access control system
RMSE	Root Mean Square Error
SEM	Security Event Management
SIEM	Security Information and Event Management
SIM	Security Information Management
SOAR	Security Orchestration, Automation, and Response
SO	Service Orchestration
SDN	Software-Defined Networking
TCMS	Train Communication & Monitoring System
TP	True Positive
TPR	True Positive Rate
VAE	Variational Autoencoders

List of Figures

List of Tables

Chapter 1

Introduction

Sometimes it is the people no one can imagine anything of who do the things no one can imagine.

Alan Turing

1.1 Motivation

Industry 4.0, or the Fourth Industrial Revolution, is characterized by the integration of physical systems with digital technologies through Cyber-Physical Systems (CPS) and Cyber-Physical Production Systems (CPPS), which can enable intelligent manufacturing through smooth communication between physical assets (machines, sensors, robots) and digital elements (software, data analytics, Artificial Intelligence (AI)). Specifically, CPSs integrate physical processes (the "real world") with computational resources (the "digital world") to enable real-time data collection, analysis, and decision-making. These systems consist of several elements, including i) sensors and actuators that monitor and control physical parameters such as temperature, pressure, and position in real-time and provide data to digital systems that process the information and issue commands to control or optimize physical processes; ii) embedded systems that process data from sensors and make decisions that affect physical systems. For example, a CPS might use real-time data from a sensor to adjust the speed of a machine or control a robotic arm; CPSs also rely on (iii) the Internet of Things (IoT) to enable

communication among devices, sensors, and machines. IoT connects disparate devices, enabling them to share information and collaborate autonomously; finally, (iv) cloud systems store and analyze large volumes of data generated by CPSs, while edge computing brings computing closer to physical devices to reduce latency and enable faster and more efficient real-time decision-making [156]. CPPSs are a specialized subclass of CPSs focused on the manufacturing domain, enabling intelligent and flexible production lines where machines and people work together seamlessly. The main features of CPPSs are i) decentralized control where, unlike traditional centralized control systems, each component or subsystem can make independent decisions based on real-time data, enabling greater flexibility and adaptability in manufacturing; ii) use autonomous machines and robots that can make decisions based on information received from sensors and work together with human operators to improve efficiency and safety; iii) utilize digital twins, or digital representations of a physical object or a human being, that can be used to control the machine or robot; and (iii) use digital twins, or digital copies of a physical object or system, to simulate real-world operations, helping to predict failures, optimize processes, and enable remote monitoring and control of equipment; and finally, (iv) use AI algorithms and advanced analytics by processing data in real-time to predict maintenance needs, optimize supply chains, and adapt manufacturing processes. The techniques of Machine Learning (ML) and Deep Learning (DL) are particularly useful for identifying patterns and improving system performance.

The most important technology on which Industry 4.0 is based is undoubtedly the IoT and its extension, the Industrial Internet of Things (IIoT).

Internet of Things IoT is a technological paradigm that enables the connection and communication of physical devices, sensors, actuators, and digital systems over the Internet or private networks. It gives objects and devices of daily life a digital identity by enabling them to communicate with each other and with other digital platforms, collecting, processing, and sharing data in real time. Thanks to the IoT, these devices can work autonomously and cooperatively to improve processes and decisions, often without direct human intervention [107]. The IoT architecture consists of several key elements that work together to collect, transmit, process, and use data. The main components of the IoT are the connected devices and objects, namely the *sensors* and *actuators* that monitor physical parameters (temperature, pressure, position, etc.) and act on them, for example by turning on a motor or opening a valve. These IoT devices are connected through various communication technologies, including Wi-Fi networks, Bluetooth, ZigBee, LoRaWAN, NB-IoT (Narrowband IoT), 5G, and other wireless technologies. Each type of connectivity is chosen based on specific

requirements for speed, distance, and power consumption. Data collected from IoT devices is then sent to centralized **processing** and **archiving** platforms for analysis and use. These platforms can be based on **cloud computing** or **edge computing**, depending on latency and processing speed requirements. In particular, cloud computing enables scalable data management, while edge computing brings processing closer to the devices themselves, reducing latency and improving operational efficiency [75]. IoT platforms are equipped with software that enables advanced analysis of collected data and automation of decision-making processes, often using **ML** and **AI** algorithms to identify patterns, make predictions, and optimize processes. Finally, IoT applications, which can be installed on mobile or desktop devices, allow users to visualize and monitor data from devices, as well as send commands to adjust parameters or perform specific actions.

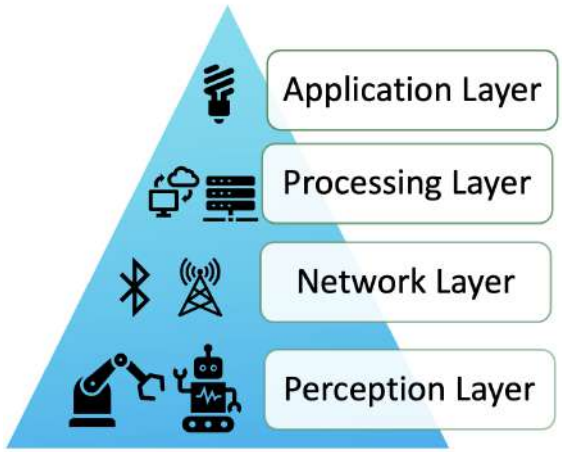

Figure 1.1. IoT Architecture

As shown in Figure 1.1, the IoT architecture has four layers [107]:

1. **Physical Layer**: This layer includes all physical devices that collect data from the environment or perform actions. Sensors measure various parameters (such as temperature, humidity, pressure, etc.), while actuators perform physical actions, such as turning on a light or activating a pump.
2. **Network Layer**: This layer is responsible for transmitting the data collected by the devices to the next layer, using a variety of network protocols and communication technologies such as Wi-Fi, 4G, 5G, LoRaWAN, ZigBee, etc. This layer is responsible for ensuring that data is transmitted securely and reliably.

3. **Processing Layer**: At this layer, transmitted data is received and processed by processing platforms (cloud or edge). This is where the data is analyzed, and stored, and useful insights are generated. AI and ML algorithms are also run at this layer to optimize operations and make informed decisions.

4. **Application Layer**: This layer includes the software and application interfaces used by users to interact with data and devices. User interfaces provide dashboards, reports, manual controls, and notifications.

IoT has applications in numerous sectors and areas, some of the most important of which include industry, where it is used for production monitoring and control, predictive maintenance, supply chain optimization, and resource management. It is also proving to be a key element in the development of smart cities, where it is being used for traffic management, smart street lighting, air quality monitoring, and urban safety. In healthcare, it is extremely useful for remote patient monitoring, smart medical devices, and telemedicine. Other areas where IoT is used include agriculture for crop monitoring, irrigation management agricultural resource optimization, and home automation, including home automation, energy management, and home security.

IoT plays a key role in both Industry 4.0 and Industry 5.0, driving their digital transformation and enabling highly connected, autonomous, and efficient industrial systems. Industry 5.0, in particular, represents a paradigm shift from Industry 4.0 toward human-centered design and sustainability. The integration of advanced technologies with human creativity is at the heart of Industry 5.0, enabling a more resilient, sustainable, and inclusive industrial ecosystem. Key characteristics of Industry 5.0 include: sustainability, human-centered design, resilience, and adaptability [7].

Sustainability Industry 5.0 focuses on sustainable production practices, such as reducing waste, conserving resources, and lowering carbon emissions. Technologies such as AI and IoT are used here to monitor and optimize energy consumption, improve supply chains, and reduce environmental impact.

Human-Centered Design This new industrial revolution also emphasizes human-machine collaboration, with robots and artificial intelligence assisting workers rather than replacing them. This approach enhances human creativity, safety, and well-being in the workplace.

Resilience and Adaptability Finally, Industry 5.0 also aims to create more resilient production systems that can adapt to disruptions such as supply chain problems, economic fluctuations, or global crises. Through decentralized control and flexible use of technology, companies can respond more effectively to change.

In particular, IoT supports one of the key drivers of Industry 5.0, namely sustainability. This technology enables manufacturers to monitor and reduce energy consumption, minimize waste, and improve resource efficiency. IoT sensors track energy consumption in real-time, allowing companies to implement energy-saving strategies, such as automatically adjusting lighting or temperature based on occupancy or usage. In addition, IoT helps manage waste and emissions, ensuring compliance with environmental regulations and contributing to overall sustainability goals. In summary, IoT is the critical enabler of both Industry 4.0 and Industry 5.0. While Industry 4.0 focuses on automation, efficiency, and real-time data analytics, Industry 5.0 takes a human-centric approach by emphasizing sustainability, creativity, and resilience. As the IoT continues to evolve, the future of manufacturing will be defined by even smarter, more adaptable, and sustainable systems that optimize both human and machine contributions. However, addressing security, privacy, and data integrity will be critical to fully unlocking the potential of the IoT across all sectors.

1.2 Problem Statement

The massive adoption of IoT has led to numerous technical and operational challenges, particularly in terms of security, sustainability, and resource management. These challenges arise from the complex and distributed nature of IoT systems, which require specific solutions to ensure the proper operation of devices, the protection of sensitive information, and the optimization of resources in dynamic and interconnected environments. Each layer of the IoT architecture (physical, network, processing, and application) has vulnerabilities that compromise the security triad: **confidentiality**, **integrity**, and **availability** of information.

At the network and protocol level, one of the most serious and widespread attacks in the IoT ecosystem is the Distributed Denial of Service (DDoS) attack, which has the potential to overload and disable network resources. This type of attack in critical infrastructure contexts (e.g., smart cities, healthcare, and industry) can severely compromise the functionality and availability of IoT systems, resulting in economic damage and disruption of critical services [101]. In addition, architectural vulnerabilities arise from a design that is often inadequate to

cope with the increasing volume of connected devices, combined with the limited computing and storage resources of IoT devices, making it difficult to implement robust security measures. The limited resources of IoT devices make it difficult to apply traditional security solutions without compromising performance and operational efficiency, such as handling complex encryption algorithms [6].

Another critical issue is the emergence of new network technologies, such as 5G and 6G, which promise to enable ultra-fast and low-latency communications for the IoT. However, these technologies increase the number of potential attack points and introduce new security challenges, such as the protection of sensitive data during transmission over high-speed wireless networks, which require innovative approaches to ensure privacy and data protection [137]. This thesis focuses on addressing the main challenges related to the cybersecurity of IoT systems, specifically analyzing the limitations of traditional protection techniques and proposing innovative solutions based on emerging techniques such as Federated Learning (FL). The problems addressed are made explicit below.

Problem 1: Traditional IDSs and SIEMs are ineffective at adapting to heterogeneous IoT threats in real time Proactive, real-time threat monitoring is critical to counter cyber attacks and intrusions into IoT systems. However, the use of traditional Intrusion Detection System (IDS) and Security Information and Event Management (SIEM), which rely on predefined firmware and rules, has shown obvious limitations when it comes to dynamically adapting in real-time to highly heterogeneous contexts such as those typical of IoT. These approaches rely on rigid models that cannot handle the volume, variety, and velocity of data generated by connected objects, especially when they come from devices of different types and manufacturers, each with its operating model and vulnerabilities [141]. In addition, the use of these traditional systems is not compatible with the limited resources of IoT devices. Signature-based IDSs and SIEMs that rely on static rules cannot detect threats in real time, nor can they respond quickly to attacks that evolve dynamically over time, creating a window of vulnerability. Therefore, research focuses on the need for lighter yet robust intrusion detection solutions that can operate effectively on resource-constrained devices while providing a high level of protection [141].

Problem 2: High computational and energy costs of traditional AI AI, and in particular the techniques of ML and DL, have become key to improving threat detection capabilities in IoT systems. Indeed, these algorithms can detect anomalous behavior patterns and predict potential attacks in real-time, significantly improving the accuracy of security systems [91]. However, the adoption of traditional AI techniques brings with it the problem of high energy

consumption and computational costs, which conflict with the sustainability goals of technologies such as Industry 5.0, which focuses on energy efficiency and reduced environmental impact. In particular, ML and DL models are very demanding in terms of computational resources (CPU, memory) and require advanced computing infrastructure such as cloud servers or the use of data centers that consume high levels of energy. This approach generates a dual problem: on the one hand, the high consumption of resources needed to train and update models, and on the other hand, the environmental impact of the computational resources needed [15]. To this end, recent research has focused on more sustainable solutions, such as FL, which allows decentralized computing and reduces the use of centralized computing power. These technologies make it possible to maintain the effectiveness of ML and DL models while reducing energy consumption [131].

Problem 3: Scarcity of quality IoT datasets and the limitations of synthetic data Another significant obstacle in developing advanced IoT security solutions is the scarcity of high-quality datasets that can be used to train AI and ML models. Collecting authentic data in complex IoT scenarios is particularly difficult, as companies often do not share their data due to concerns about privacy and protecting sensitive information. Additionally, data from IoT devices is typically unbalanced, difficult to label, and not always complete. This makes it extremely challenging to develop and test security models that can generalize to a wide range of possible scenarios and attacks [132]. The rarity of real-world attacks and the randomness with which they occur make it difficult to collect enough data to train accurate models. As a solution, some researchers propose using synthetic data generated through simulations to train AI models [67]. However, the use of simulated data raises questions about the authenticity and effectiveness of these datasets in improving threat detection capabilities in real IoT systems [84]. Furthermore, there is a risk that synthetic data may not adequately reflect the complexities of real-world behaviors and threats, reducing the reliability of security models.

1.3 Contribution

The work carried out in this thesis aims to propose solutions that can advantageously address the IoT security challenges discussed in 1.2. The first contribution of this thesis addresses **Problem 1**, through a comparative evaluation of two SIEM platforms designed for Information Technology (IT) environments, namely Real-Time Analytics (RTA) and LimaCharlie. The goal of this analysis is to support security administrators in the search for IT solutions that can

adapt to the most complex IoT environments. In collaboration with the company *DigitalPlatform S.p.A.*, we analyzed how effectively the considered platforms can detect *DDoS* attacks in IoT environments. To do so, we created a synthetic dataset using botnets of different sizes, i.e. networks of compromised devices that an attacker controls to perform malicious activities. Next, we applied detection techniques from the platforms and evaluated their performance by measuring *False Positives (FP)*, i.e., benign events incorrectly classified as anomalous, and *False Negatives (FN)*, i.e., benign events incorrectly classified as benign. The analysis highlights that since both platforms rely on deterministic rules, they are limited in their ability to detect novel or complex attacks. This result suggests the integration of *ML* models to improve detection accuracy.

To address **Problem 2**, this thesis proposes the implementation and evaluation of the performance and sustainability of a lightweight IDS specifically designed for embedded networks, characterized by limited computational resources. In particular, the proposed IDS has been implemented using the K-means model to detect, in real-time, Denial of Service (DoS) attacks on an onboard network of a train emulated by the Mininet-WiFi tool. Experimental results demonstrate that despite the limitation of Central Processing Unit (CPU) and memory usage, the reference IDS achieves a high detection accuracy while maintaining a minimum energy consumption of about 1.5 Watt [34]. It is therefore deduced that this solution satisfies the trade-off between performance and energy efficiency, aligning with the requirements of Green AI, that is, an approach to AI that aims to reduce the environmental impact of models and algorithms, optimizing their energy consumption and computational resources. To further support the search for sustainable cybersecurity solutions specifically for IoT environments, this work introduces DDoShield-IoT [35], a flexible testbed built using Docker containers and the NS3 simulator, which allows

- Generate realistic traffic, both benign (FTP, RTMP, HTTP) and malicious (Mirai DDoS);
- Evaluate the sustainability and performance of ML-based IDS by performing real-time intrusion detection;
- Support the integration of various ML models (currently, Random Forest (RF), K-Means, Convolutional Neural Network (CNN)), achieving high detection accuracy;
- Provide a scalable environment, thanks to the exploitation of Docker and NS3, ensuring reproducibility, platform independence, and flexibility for different research scenarios;
- Create high-quality datasets, generating customizable pcap files, thus addressing the scarcity of IoT-specific data;

- Collaborate with the scientific community, as the source code is publicly available, thus promoting reproducibility and enabling researchers to develop and test customized IDS solutions.

DDoShield-IoT, therefore, provides a versatile and reproducible environment to develop effective intrusion detection strategies. Indeed, by leveraging its architecture, a new intrusion detection methodology is proposed, combining User Profiling and FL. In particular, an IoT network has been considered, where heterogeneous devices communicate. Here, the User Profiling approach groups devices based on traffic patterns using clustering algorithms (e.g., K-means). Once the clusters are defined, a two-stage anomaly detection approach is implemented, first leveraging lightweight models (e.g., RF) and then applying a deeper analysis only on packets classified as malicious in the first stage and randomly on a subset of benign packets. In the second stage of the detection process, the FL approach is used to improve IDS performance while preserving the privacy of sensitive information, adapting to large-scale distributed devices, and reducing energy consumption. This system demonstrates an efficient, scalable, and environmentally sustainable approach to IoT security. The proposed methodology has been adapted and validated in real-world scenarios through its application in the European project CyberSEAS [33]. This use case focused on Smart Grid infrastructures has confirmed the value of the methodology in protecting critical IoT systems while meeting performance and sustainability objectives. Finally, to address the scarcity of high-quality IoT data for training ML and DL models mentioned in **Problem3**, this research introduces a Data Augmentation approach that combines a real-world dataset (i.e., the TON_IoT dataset) with simulated traffic-generated by DDoShield-IoT to obtain a more balanced and enriched dataset of hard-to-obtain high-quality IoT data from the real world. This work aimed to help fill the gap in data availability and improve the development of older ML and DL models that become more efficient for IoT security; indeed, experimental evaluations demonstrate that the augmented dataset improved the performance and robustness of IDS models in detecting zero-day attacks.

1.4 Thesis Structure

The thesis is structured as follows.

Chapter 2 reviews IDS and SIEM limitations in IoT environments, IoT-tailored solutions, FL and Data Augmentation approaches for IoT cybersecurity.

Chapter 3 consists of a comparative analysis of SIEM platforms: RTA and LimaCharlie. It discusses in-depth methodology, simulation results, and their

strengths and weaknesses when detecting DDoS attacks.

Chapter 4 proposes a framework that enables IoT-tailored IDS. It discusses the implementation in detail, and the performance of CPU, memory, and energy usage.

Chapter 5 discusses the design and implementation of DDoShield-IoT, an advanced testbed for evaluating IDSs in IoT environments, and introduces a Data Augmentation technique to improve IDS robustness.

Chapter 6 proposes a novel intrusion detection methodology that combines FL with user profiling for IoT anomaly detection. It also validates the proposed approach through its real-world application in the European CyberSEAS project.

Chapter 7 summarizes what can be considered the main contributions of this thesis and provides final reflections.

Chapter 2

Background and Related Work

The heterogeneous and resource-constrained nature of IoT systems implies that cybersecurity in this context faces numerous challenges. This chapter will analyze IDS and SIEM platforms, their limitations in IoT environments, tailored solutions, and emerging approaches such as Federated Learning (FL) and Data Augmentation.

2.1 IDS and SIEM platforms overview

IDS and SIEM platforms are essential technologies in the cybersecurity domain since their primary goal is to monitor, detect, and respond to cybersecurity incidents on networks and systems [110]. In particular, the IDS are dedicated systems designed to identify malicious activity within a network or host environment by analyzing traffic patterns, system logs, and other data sources to identify potential compromises and threats to the monitored system [112]. The IDS can be broadly classified into two main categories: i) Signature-based IDS and ii) Anomaly-based IDS. The first category of IDS identifies attacks monitoring the system activity and comparing it with well-known attack patterns, named *Signatures*. When a match occurs, the IDS generates an alert, thus facilitating the prompt identification of specific threats and reducing instances of false alarms. However, these systems cannot detect novel or evolving attacks efficiently, necessitating regular updates to maintain efficacy. In contrast, anomaly-based IDS

employs artificial intelligence techniques to identify unusual behavior in a network or system. Specifically, it creates a normal activity baseline by statistically analyzing network traffic or system activity over time, then monitors network traffic or system activity in real time, comparing it to the baseline, and flagging any significant deviations as potential intrusions. This IDS type is more effective at discovering unknown threats but is susceptible to higher false-positive rates [55] [19] [112].

SIEM platforms exceed the capabilities of IDSs by offering a centralized approach to log aggregation, analysis, and incident response. They provide a holistic view of an organization's security status by correlating data collected from different sources, including IDS, network devices, endpoint systems, and applications. This centralized approach facilitates real-time monitoring and alerting of security incidents, generates automatic responses to detected threats and detailed reports of the analyzed events and defenses deployed that can also be exploited for future forensic analysis [52]

IDS and SIEM thus represent essential elements in the scope of cybersecurity tools, thought to work in support of each other so that their combination creates a solid defense mechanism, enabling proactive and reactive measures against cyber threats.

2.1.1 Common Types of Cyber-Attacks in IoT: DoS and DDoS Attacks

The rapid proliferation of connected devices and the inherent weaknesses that characterize these systems render IoT environments increasingly vulnerable to several cyber threats, from the most common malware to the always more sophisticated cyberattacks tailored for IoT [69] [95]. Among the most disruptive and frequent threats are DoS attacks, which are designed to overwhelm a single target, such as an IoT device or a critical server, by flooding it with excessive traffic or sending malformed requests until it is inaccessible; and the DDoS attack, which is the distributed evolution of a DoS attack that involves a botnet, i.e., a network of compromised devices, that flood the target with traffic from numerous sources simultaneously [35] [74]. IoT devices, often poorly protected, are prime candidates for recruitment into these botnets, amplifying the reach and impact of such attacks that exploit the limited resources and widespread interconnectivity of IoT ecosystems, having critical consequences, especially in critical infrastructures (CIs), including power grids, transportation networks, water supply systems, and healthcare facilities, which are highly dependent on real-time operations and continuous availability to ensure the functionality and safety of

society. Indeed, a DDoS attack could, for example, result in power outages due to the paralysis of network management systems, the disruption of public transportation as a consequence of the overloading of IoT-based planning and monitoring systems, or even the endangerment of human lives as a result of the incapacitation of medical IoT devices or emergency communication networks. The 2016 Dyn DDoS attack, which resulted in the unavailability of online services such as Twitter and Netflix, underscored the vulnerability of broader critical infrastructures that rely on the same DNS infrastructure [42]. The notorious Mirai botnet, moreover, illustrated the potential for insecure IoT devices to be exploited to generate traffic over 1 Tbps, which could overwhelm even robust and well-structured networks [161] [90]. The combination of massive scale and sophisticated methods used by attackers also renders DoS and DDoS attacks particularly dangerous, making detection and mitigation challenging due to sophisticated attacker techniques. Some DDoS attacks indeed use a strategy of low and slow traffic generation, which closely mimics legitimate user behavior and is, therefore, difficult to identify [95]. Others target the application layer, focusing on specific services such as HTTP servers to exploit vulnerabilities in the application logic, bypassing, often, the traditional defenses focusing on traffic volume rather than application behavior [64]. In essence, DDoS attacks are becoming a significant and constantly evolving threat to IoT ecosystems and, specifically to critical infrastructures, since their capacity to exploit vulnerabilities in IoT-constrained systems, scale on a massive scale, and adapt to circumvent traditional forms of defense underscores the pressing need for more sophisticated detection and mitigation strategies development.

2.1.2 Traditional IDS and SIEM Limitations in IoT Environments

Despite their efficacy in conventional IT infrastructures, IDS and SIEM systems encounter considerable obstacles when deployed in IoT environments due to the distinctive attributes inherent to IoT, which impose constraints that must be acknowledged to guarantee comprehensive protection [43]. IoT networks are composed of several devices exhibiting discrepancies in hardware, software, and communication protocols, with standards ranging from MQTT to CoAP. Such heterogeneity presents a challenge for the uniform detection methods implementation since the traditional IDS and SIEM systems designed for more homogeneous environments could not scale effectively [100]. In addition, many IoT devices are limited in computing and storage capacity, making local IDS agent deployment impractical, while streaming data to centralized SIEM systems often consumes excessive network resources, potentially degrading performance [70]. These chal-

lenges are further exacerbated by device behavior variability, where, often legitimate outliers in data are misclassified as malicious activity, thus increasing false positives [129] [95]. The critical infrastructures, capable of generating considerable volumes of network traffic at high velocity, face additional challenges in real-time detection of attack patterns. Precision is paramount in this context, where sophisticated detection tools, SIEM or IDS platforms, must work at a millisecond granularity to analyze and neutralize threats before they cause significant harm [20] [121]. Moreover, ongoing attack evolution creates critical challenges for signature-based IDS when dealing with dynamic threats like botnets and ransomware. This can result in traditional systems exhibiting a high False Negatives Rate (FNR) in the context of zero-day or rapidly mutating threats [46] [60]. The identified shortcomings underscore the need for more advanced, and/or adaptive solutions that can meet the specific requirements of the IoT.

2.1.3 Real-Time Analytics (RTA) and LimaCharlie

To address the cybersecurity challenges in IoT mentioned above, we first evaluated several SIEM platforms used commonly in information technology (IT) environments to determine their suitability in highly heterogeneous and resource-constrained environments. Among them: i) Splunk [1] stands out for its ability to analyze large datasets and offer sophisticated event management, although it is constrained by significant challenges about the elevated costs associated with licensing and the necessity for costly infrastructure; ii) IBM QRadar SIEM [2] is recommended for its automated event correlation and threat management capabilities, but it deficits in its ability to manage distributed environments comprising disparate devices; iii) ArcSight [3] is renowned for its scalability, it is not without its shortcomings, particularly in the detection of large-scale attacks in real-time and the effective management of logs; other platforms, such as iv) LogRhythm [4] and v) AlienVault[5], provide more straightforward integration options, however, they are not designed to meet the distinctive requirements of IoT ecosystems.

In this context, Real-Time Analytics (RTA) [6] and LimaCharlie [7] solutions stand out for their innovative and flexible features. In particular, RTA provides

[1] https://www.splunk.com/
[2] https://www.ibm.com/it-it/products/qradar-siem
[3] www.microfocus.com/arcsight
[4] https://logrhythm.com/
[5] https://isec.ecohmedia.com/alienvault/
[6] https://www.cy4gate.com/it/prodotti/rta/
[7] https://limacharlie.io/

sophisticated capabilities for real-time monitoring, whereas LimaCharlie offers a range of flexible options for complex and distributed infrastructures due to its cloud-native, modular architecture. For this reason, Chapter3 of this thesis will provide a detailed analysis of these platforms, considering their potential to address the challenges inherent to IoT security constraints, finding that both solutions adopt deterministic rules that limit the detection of attacks, suggesting to include ML models to overcome this limitation, improving the detection of unknown threats and reducing false negatives, and handling of complex backend queries to allow better correlation of information over specific time intervals.

2.2 IoT Tailored IDS

While RTA and LimaCharlie offer a potentially suitable existing, though somewhat limited, solution to IoT security requirements, it is also important to note that, because of the always-growing number of IoT devices used in Critical infrastructures, there is the need for tailored security solutions for this environment which have to be able to face to the specific IoT constraints, like computational resources limitations, energy power restrictions, heterogeneous protocols, etc., and have to provide efficient defenses against a wide range of evolving threats. Existing studies, indeed, emphasize the importance of real-time detection in high-risk environments [143] [158] [88] [148], and highlight traditional IDS limitations in handling IoT-specific anomalies [129] [5] [43], focusing, for example, on the development of Lightweight IDS specifically designed for resource-constrained IoT environments, addressing the need for tailored efficient security measures. Lightweight IDSs aim to detect various attacks, including impersonation and denial of service, while minimizing computational requirements [76] [65]. Recent approaches utilize machine learning techniques, such as Support Vector Machines (SVM) and XGBoost, combined with feature selection methods to reduce the feature space and improve detection accuracy [104] [65]. Also, significant progress has been made in Lightweight IDS design and implementation in various IoT domains. Hazman et al.[57] introduce an anomaly detection model to effectively monitor IoT-based Smart City environments using ensemble learning methods, including Support Vector Machines (SVM) and Random Forests (RF). This system demonstrated high accuracy and timely anomaly detection, which is critical to ensuring the safety and efficiency of urban IoT ecosystems. Similarly, Guezzaz et al.[53] propose a hybrid IDS framework to secure edge-based Industrial IoT (IIoT) systems, combining lightweight machine learning algorithms with efficient resource management, enabling real-time intrusion detection while minimizing computational overhead on edge devices. In the domain of Smart Homes,

Gazdar et al.[49] developed an ML-based IDS trained on legitimate behavioral models. Their system successfully identifies deviations that indicate potential intrusions, providing an efficient and accurate method to protect home automation systems. Expanding on edge-based security, Idrissi et al.[62] presented a deep learning-based IDS designed to run directly on constrained edge devices. Using convolutional neural networks trained on IoT network traffic data, their solution achieved accurate anomaly detection while conserving system resources. Energy efficiency remains a critical concern in IoT security, so Raja et al.[120] address this issue by proposing Software-Defined Vehicular Networks (SDVN) that integrate lightweight cryptographic techniques with energy optimization algorithms, highlighting the importance of balancing security and energy consumption in dynamic vehicular environments; as well as, Thamilarasu et al.[145] that present an IDS focused on IoT healthcare systems, employing behavior-based analytics to protect medical devices from targeted cyberattacks. Instead, He et al.[59] implement a reinforcement learning-based anomaly detection system for IoT networks to improve adaptability. This approach dynamically adjusted detection parameters to align with evolving network conditions, improving accuracy. While studies mentioned above have made significant strides in designing and implementing lightweight IDS solutions across various IoT domains, each addressing unique challenges such as energy efficiency, real-time detection, and resource constraints, this thesis takes a different approach. Specifically, Chapter 4 focuses not on developing new IDS algorithms but on evaluating the performance and energy efficiency of existing lightweight IDS solutions tailored for IoT environments. By utilizing the K-means algorithm for anomaly detection, this research explores the practical viability of lightweight IDS in constrained environments, where both accuracy and resource efficiency are crucial. Moreover, a custom dataset simulating a Train Communication & Monitoring System (TCMS) network of a high-speed train was generated to overcome the challenges of realistic data representation in IoT contexts. The results demonstrate the feasibility of deploying a resource-aware IDS in critical environments, emphasizing the balance between performance and sustainability.

2.3 Testbeds and Simulation Frameworks for IoT Intrusion Detection

The solutions described in the previous section demonstrate the significant progress in designing and implementing lightweight IDSs for IoT environments. However, the complexities and heterogeneity of IoT environments require further development and rigorous testing to ensure the effectiveness of these solutions.

While existing research has made great strides in proposing strategies to enhance security in critical environments, many testbeds and simulation frameworks for IoT intrusion detection reveal significant scalability, flexibility, and real-world applicability limitations. For example, Bhayo et al. [23] propose a testbed for evaluating algorithms for DDoS detection in Software Designed Networks (SDN) designed for IoT environments by simulating SDN-IoT traffic via the SDN-WISE environment and using WEKA for classification with in-depth parameter analysis, but do not fully evaluate the performance of these algorithms in real-world scenarios. Or Zolanvari et al. [167] introduce an IIoT network testbed, simulating an actual industrial plant to monitor water levels and turbidity in a reservoir, collecting realistic data to train ML models but adopting a framework configuration that remains limited in generalizability to other industrial contexts. Anthi et al. [16] then focus on smart home security, emulating a Smart Home IoT environment considering various types of devices to test a purpose-built IDS. This method provides realistic generated data but still lacks a comprehensive consideration of the diverse nature of devices and attack scenarios in an actual Smart Home environment. Nie et al. [105] implement a network testbed for IoT (IoV) networks that generates realistic traffic flows to perform DDoS attacks, and while their setup provides insight into the attack pattern in IoV scenarios, it suffers from complexity and scalability issues; similarly, Kumar et al. [73] design EDIMA testbed to detect malware in IoT networks using ML, but they encounter scalability issues and high setup complexity that makes the proposed testbed unsuitable for deployment in resource-constrained organizations. Albulayhi et al. [12] also present a testbed that includes physical IoT devices such as AI speakers, Wi-Fi cameras, and laptops to emulate a wide range of real-world attacks. This work increases realism by using hardware but suffers from limited coverage of heterogeneity in IoT scenarios and attack types. Zachos et al. [159] then present the IoT/IoMT security testbed where Raspberry Pi devices emulate sensor behavior, creating a setup appropriate for specific simulations but still far from the flexibility and robustness required in IoT environments. Finally, Obaidat et al. [108] create DDoSim, a simulation framework to study large-scale DDoS attacks on botnets, focusing on scalability and cost-effectiveness but leaving out realistic benign traffic generation and machine learning-based IDS evaluation, reducing its usefulness for comprehensive security assessments. Ultimately, while these testbeds provide valuable contributions, they expose several limitations. Most of them fail to mimic the complexity and diversity of IoT environments in terms of different devices, protocols, and attack scenarios. Some of them also lack mechanisms for in-depth evaluations of IDS performance under realistic conditions, which reduces their effectiveness in bridging the gap between research and real-world applications. High resource requirements and scalability issues further limit their accessibility and applicability in different research and industrial con-

texts.

To address these challenges, chapter 5 of this thesis presents, DDoShield-IoT , an IDS testbed, i.e., a controlled environment that simulates network conditions, devices, and attacks to test and evaluate performance IDS and effectiveness in detecting threats [11] [126] [72] [47]. DDoShield-IoT integrates Docker containers with the NS3 network simulator, thus enabling the deployment and evaluation of the IDS' implementation in a manageable environment that simplifies setup and improves usability. The proposed framework can generate real benign and malicious traffic in simulations, significantly improving the realism and practical relevance of the testbed. DDoShield-IoT supports several ML models that can be leveraged by IDS, including Random Forest, K-Means clustering, and Convolutional Neural Networks (CNN), thus allowing users to compare exhaustively their performances within the same simulation environment and choose the best model for their needs. In addition, unlike most of the existing work, the framework has been made public, ensuring reproducibility and collaboration, simplifying the replication of experiments, and allowing researchers to validate their customized IDS.

2.4 Data Augmentation

One of the most significant aspects of IDS' assessment through testbeds is the availability of representative and diverse datasets. However, data collection in the context of IoT is challenging, as it is often hampered by the ample diversity of devices, heterogeneous communication protocols, and variable network configurations. The variety and quantity of data available in existing IoT testbeds are typically used to train and test IDSs, and this is a particularly critical issue in the context of generalizing IDSs, as the paucity of data makes it challenging to simulate complex and realistic scenarios, which are crucial for the efficient training of intrusion detection models [135]. Moreover, data collection from genuine IoT devices is frequently hindered by compatibility issues, constraints about privacy, and logistical challenges [31]. The restricted accessibility of diverse and pertinent IoT data thus renders it challenging for intrusion detection models to effectively generalize to scenarios not encountered during training [25].

To overcome these difficulties, the research community increasingly embraces the concept of Data Augmentation (DA), which is a technique widely used in other fields such as image recognition [102] [68] or natural language processing [77] [29]. DA is a technique used to artificially expand training datasets for machine learning models, particularly in scenarios with limited data availabil-

ity [135]; more specifically, this technique relies on the application of transformations to already existing data to create new instances, increasing data variability and improving the ability of the models to generalize to new scenarios.

These transformations may include several techniques like geometric manipulation of images, modification of text in language, and the synthetic generation of new data in more complex contexts, such as the IoT, where augmentation encompasses not only the generation of new data but also the simulation of scenarios that may not be present in the original datasets [93] [164], thus supporting the testing of IDS' resiliency under unknown circumstances.

The main transformations used in data augmentation for IoT include changing the network traffic pattern, for example, introducing latencies or simulating packet losses, adding synthetic noise to mimic real perturbations in communications between devices, and generating simulated attacks, which might differ depending on network protocols or vulnerabilities in IoT devices [38] [162].

In general, the potential of DA's techniques has been widely examined to address the issue of imbalanced datasets in the context of IoT, on which train ML-based IDS. Approaches such as Variational Autoencoders (VAE) and conditional VAEs demonstrated that they potentially enhance the performance of IDSs [80], even if there are still obstacles to overcome to ensure the efficient application of these techniques. Indeed, IoT devices' diversity and susceptibility to attacks present significant challenges for effective IDS development [70].

Notable recent developments include the application of generative adversarial networks (GANs) for data augmentation, which have demonstrated enhanced detection accuracy on imbalanced datasets [38]. Although these attempts, obstacles persist in adaptive and intelligent IoT application creation due to the heterogeneity of infrastructure and protocol [8].

For these reasons, it could be beneficial to develop more robust DA techniques, which can address the distinctive challenges of IoT environments, such as device heterogeneity and evolving attack patterns, thus enhancing the effectiveness of IDS.

Therefore, a further contribution of the 5 chapter of this thesis focuses on a customized DA approach, which aims to simulate realistic network scenarios through the combination of the realistic IoT traffic generated by DDoShield-IoT , which can create IoT traffic in a highly configurable manner, and real traffic from the TON_IoT dataset, which includes telemetry data from IoT and IIoT services, operating system logs and network traffic, providing a comprehensive and realistic representation of IoT networks and potential attacks [14]. Due to the combination of data, the simulations are designed to replicate not only common

attacks such as DDoS attacks but also emerging threats, creating different, larger, and more varied datasets and solving the problem of unbalanced learning, which occurs when malicious or attack data is insufficient compared to legitimate data, allowing IDS to deal with unknown attacks and increase the overall effectiveness of protection in IoT networks by adapting to changing threats and reducing the risk of false negatives.

2.5 Federated Learning and User Profiling to address IoT Security Challenges

The challenges that security solutions such as IDS and SIEM face in IoT environments require innovative approaches that can address the unique characteristics of these ecosystems. Device heterogeneity, computational constraints, and IoT data dynamics require flexible and scalable solutions that balance privacy, performance, and detection accuracy.

In this context, Federated Learning (FL) is emerging as a promising paradigm for training distributed models directly on local devices without centralized data necessity. This decentralized approach effectively addresses privacy concerns because sensitive data is never transmitted, unlike only model parameters (i.e., updated weights). Moreover, FL is especially well-suited to edge computing and big data scenarios where data is generated by a multitude of heterogeneous devices, including IoT sensors and mobile devices, in a distributed manner. Centralizing this data would be logistically complex and expensive.

While FL enables privacy protection by minimizing the risk of data exposure and facilitating compliance with regulations such as General Data Protection Regulation (GDPR) [24] [48] [138]. Despite its advantages, FL presents several significant challenges, especially in handling non-independently and Identically Distributed (IID) data and optimizing inter-device communication.

Distributed learning requires advanced communication schemes to reduce the impact on limited device resources such as bandwidth and computation [48] [146]. Recent studies focusing on FL-based IDS have made significant progress. For example, Chaabene et al. [28] propose a FL-based IDS that combines network and energy data, achieving good performance and high accuracy. Similarly, Benameur et al. [22] improve the FL model using knowledge distillation techniques, thus increasing the computational efficiency and accuracy.

Other works, such as Hamdi et al. [56], perform well in accuracy and face privacy issues but do not fully address the scalability challenges in large-scale

IoT networks. Another crucial issue is non-ID data, where Guo et al. [54] introduce a data augmentation algorithm to address this difficulty by improving the performance of host intrusion detection systems (HIDS). However, many studies still fail to comprehensively handle data in heterogeneous IoT environments, thus affecting the performance of the federated model.

Managing the privacy-performance tradeoff remains a persistent challenge, as highlighted by the proposal of Anwar et al. [17], which attempts to balance privacy and accuracy with architectures such as DeepFed without fully addressing scalability issues. Existing research, therefore, demonstrates substantial progress in FL-based IDSs for IoT and IIoT environments. However, scalability, non-IID data management, privacy-performance tradeoffs, and comprehensive evaluation metrics remain challenging requirements for current solutions.

This thesis seeks to overcome these limitations by integrating FL with User Profiling in a two-stage IDS model.

User Profiling consists of creating behavioral profiles of individuals or groups of users by leveraging data representing their actions, preferences, and characteristics. The process involves collecting and analyzing users' interactions with a product, service, or platform to discover their needs, interests, and digital behavior.

User Profiling is frequently used in e-commerce, marketing, and personalized content distribution, and recently, it has been adapted for use in cybersecurity within IoT contexts. Rose et al. [125] develop an IDS that combines network traffic profiling with machine learning techniques, achieving high accuracy (98.35%) in detecting known and unknown attacks. This system highlights the potential of profiling and FL integration, improving detection capabilities while addressing computational resource constraints. However, challenges remain in ensuring its adaptability to dynamic network conditions.

Furthermore, user profiling can enhance the effectiveness of IDS, as discussed by Peng et al. [116], who explore the integration of behavioral biometrics such as keystroke dynamics and psychometrics to validate better user legitimacy. While their work offers promising results, it faces challenges such as high false positives and profile manipulation, especially in IoT contexts where behavior can change over time. Wachter [150] also highlights critical privacy challenges in IoT identification, including profiling, discrimination, and GDPR implications. His analysis highlights the tension between user identification for seamless services and the need for robust privacy protections, advocating for greater transparency, user control, and context-aware data sharing to address these risks effectively.

As noted by Safi et al. [127], profiling techniques in IoT contexts also highlight the need for continuous monitoring and the application of machine learning to

adapt to different devices and behaviors. This work aligns with the growing focus on ensuring scalability, adaptability, and privacy in IoT security. Then, Maraj et al. [89] highlight a comprehensive framework for user profiling in Internet services, offering solutions for privacy issues via dynamic data-sharing controls; their approach, although effective in balancing personalization and privacy, does not fully address IoT-specific requirements, suggesting the need for more targeted IoT profiling solutions.

Finally, Shaman et al. [133] present a novel network metadata-based user profiling method, which achieves 74% accuracy in user identification via application-level analysis. However, this method presents limitations because of the sample size and lack of real-time applicability, which may hinder its implementation in more complex IoT networks.

The reviewed works highlight significant progress in the use of User Profiling and machine learning for IoT security but have key limitations that include insufficient adaptability to dynamic networks, high false-positive rates, vulnerability to profile manipulation, lack of continuous monitoring, difficulty in scalability and real-time applicability, and tensions between personalization and privacy. These issues compromise the effectiveness of existing solutions in complex and evolving IoT environments.

For this reason, the following dissertation proposes a methodology that addresses these challenges by integrating Federated Learning and User Profiling into a two-stage IDS model. Combining real-time raw feature detection on profiled packets and advanced statistical feature analysis with FL, the methodology ensures scalability, adaptability, and privacy protection, representing an innovative contribution to improving security in IoT ecosystems.

The methodology, although initially defined merely through a Design of Experiment (DoE) [41], was later modified, adapted, and validated in the real-world context of the Cyber Securing Energy Data Services (CyberSEAS) [33] project, which has three principal objectives: i) safeguarding electricity grids from major cyber threats, ii) protecting consumer data, and iii) enhancing the energy data space security, proving to be an effective solution for anomaly detection in smart grids. Specifically, it excels in scalability and applicability to unlabeled data, offering improved accuracy by reducing false positives.

The subsequent chapters provide a detailed discussion of the proposed solution to address the limitations identified in related work and the problem statements. Specifically, I analyze the methodologies and implementation processes, along with a discussion of the results.

Chapter 3

Comparative Analysis of SIEM Platforms

This chapter presents a comparative analysis between two existing SIEM platforms, Real-Time Analytics by CY4GATE and Limacharlie. The goal is to evaluate their ability to detect DDoS attacks through experiments on a synthetic dataset generated by DDoSim [108], with a rule-based detection approach.

3.1 Security Information and Event Management

A SIEM is a software solution that combines Security Information Management (SIM) and Security Event Management (SEM) to monitor, analyze, and correlate security events from multiple sources to detect and respond promptly to cyber threats. In particular, the operation of SIEM is explained in different phases, all aimed at improving and managing the cybersecurity of an infrastructure. The first phase always comprises collecting data from various sources such as firewalls, IDS/IPS, servers, endpoints, applications, and network logs. These are then standardized and stored in a central database to enable analysis and correlation. Then, the correlation engine, a sophisticated component that uses algorithms, predefined rules, and artificial intelligence models, analyzes the transformed data to find relationships between security events, find suspicious patterns, and generate alerts or automated actions if certain conditions occur. SIEMs also provide essential features such as log retention to support forensic

analysis, regulatory compliance, long-term threat detection, and intuitive dashboards to monitor security status in real-time, thus facilitating data analysis. When a SIEM system detects anomalies or violations, it generates alerts to notify security analysts. Sometimes, these systems integrate response automation tools to address threats immediately. They often automate repetitive tasks and streamline incident and threat response workflows with Security Orchestration, Automation, and Response (SOAR) systems. SOAR systems are software solutions that enable security teams to integrate and coordinate various security tools effectively [52].

Real-Time Analytics (RTA) RTA is a modern SIEM that supports analysts in detecting anomalies and taking appropriate countermeasures. It is capable of collecting millions of data per minute, and it is a modular platform that allows you to coordinate and manage the modulations, transformations, and indexing of the vast amount of data collected. To support the management of large organizations, RTA uses a Role-based access control system (RBAC), a technique that defines access permissions based on the roles held within the organization, designed according to the principle of *least privilege*.

A crucial feature of RTA is its compatibility with many security solutions already on the market, with which it is possible to combine, thus enabling data collection from different sources through tools that we call sensors (Figure 3.1). However, since the installation, maintenance, and storage of sensors involve high costs, a preliminary assessment of the assets to be monitored by these sensors becomes necessary. The element that interfaces with the sensors to collect and input data into RTA is the JConnector module, developed in Java.

JConnector is the event collection and normalization engine structured as a processing pipeline in which processors called "Listeners" collect events processed by intermediate nodes until sent to Forwarders, as shown in Figure 3.2. Each process step in the pipeline specializes in diverse tasks such as event collection, filtering, categorization, and forwarding. Additionally, the internal cache queue allows for event handling with variable processing speed without any loss in case one of the forwarding devices compromises processing.

A parser, i.e., a configuration file (*.yaml*) that specifies the data formatting rules, normalizes all the events collected by JConnector. According to this file, events are converted into a common *Event Message* format, specifically designed to allow complete mapping of heterogeneous event information, converting it into a set of well-structured and meaningful *key-value* pairs, which, when well defined, facilitates the performance of advanced analysis such as clustering by similarity, technical queries (e.g., IP addresses), and event correlation based on criteria like

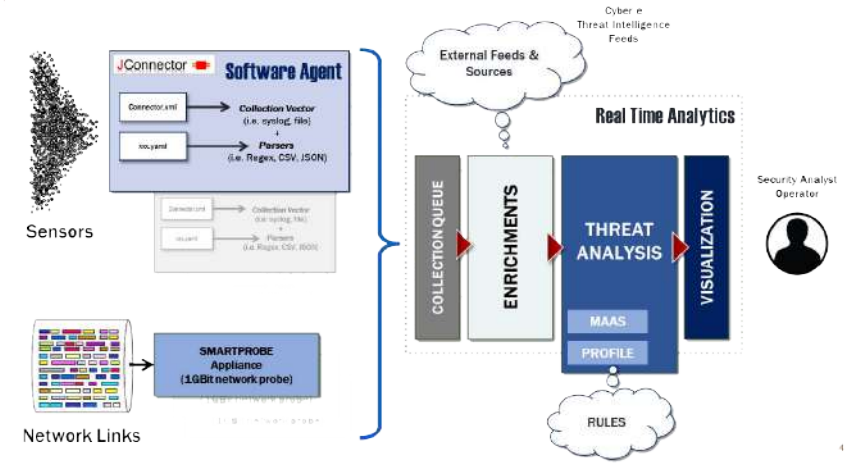

Figure 3.1. RTA Functional Diagram

frequency or match. The event message is structured in JSON format, which is flexible enough to allow customization, although this may result in a loss of accuracy. Following the normalization phase, RTA provides an enrichment data process, consisting of information addition like geolocation information via Apache Storm or offline enrichers (faster), to improve the analysis level of detail. When collecting raw data from perimeter network infrastructure, RTA also integrates this data in real-time with Threat Intelligence information from external sources to enable a better and more detailed understanding of the threat. This process, referred to as *merge & contextualization* helps give meaningful context to the information, allowing RTA to respond in a more targeted and accurate manner to identified threats.

A key component of RTA is the dashboard (Figure 3.3), which represents the platform homepage, too. It graphically displays input data through various plu-

Figure 3.2. JConnector Processing Model

gins, i.e., tools like histograms, tables, maps, and timelines, which allow analysts to visualize information and monitor alerts in real time. Such plugins, particularly *chart* and *histogram*, will be used in the Detection phase.

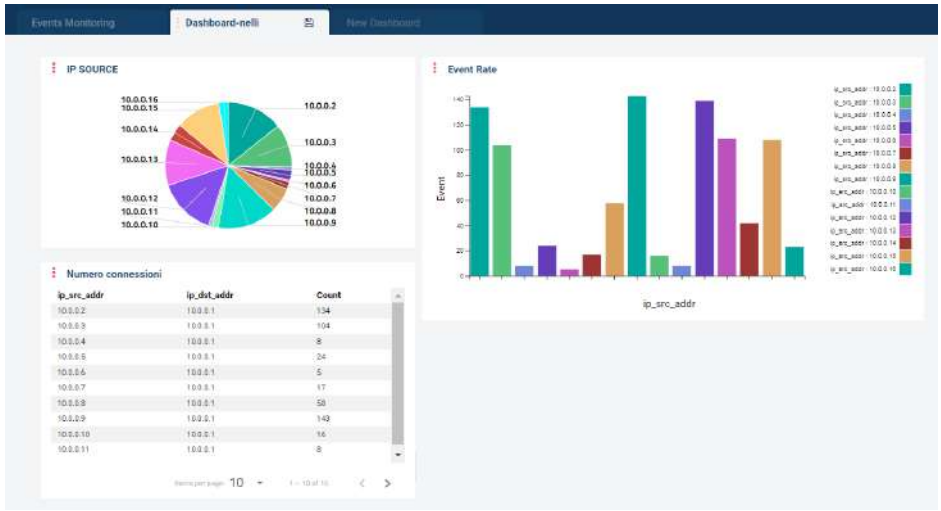

Figure 3.3. Dashboard RTA

All of these components and processes support the key function of RTA, namely *threat analysis*, designed to identify threats in real time through an advanced event analysis and correlation process. As shown in Figure 3.5, it takes place in several stages beginning when the monitored infrastructure (e.g., a firewall, an IDS, etc.) generates a *security event* not yet stored in the database, referred to as *raw data*). As soon as the event enters the system, it undergoes initial filtering, which removes data that is not useful for anomaly detection purposes, thus reducing the amount of information to be analyzed. The filtered event becomes the input of a Lunaris script (a Domain Specific Language (DSL) used to process events, where expressions end with a semicolon and defined variables become new fields in the Event Message. This language also allows for the use of temporary variables that are reset at the end of the script to optimize resources), which performs event enrichment by examining the event and looking for additional information from other sources, querying two main components: i) the Profiler, that stores information about entities, like users or devices, allowing their behavior monitoring over time, and ii) the Model as a Service (MaaS) that uses Machine Learning models to make predictions about event behavior to detect anomalies or high risks. At this point, the Lunaris engine uses the enriched information by looking for correlations with other events. Here, an alert is gen-

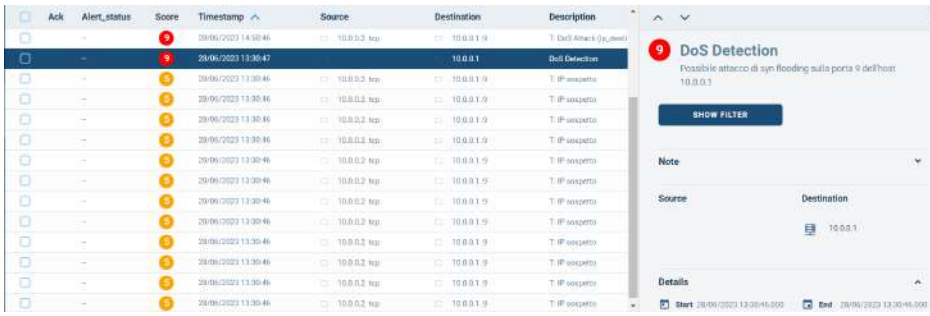

Figure 3.4. RTA DDoS Attack Alert

erated when the analyzed event has characteristics that match predefined rules, such as abnormal behavior or suspicious action, as illustrated in Figure 3.4. This correlation process bases itself on rules that can be of different types, such as:

- Standard Correlation: that immediately raises an alert if a suspicious event is detected.
- Correlation Based on Threat Intelligence: for example, a user connecting from a VPN outside the country of residence.
- Behavior-Based Correlation: statistical analysis to check whether the entity's behavior is ordinary.
- Machine Learning: used to identify previously undefined anomalies.

The correlation process result, which can be true or false, is passed to a final evaluation phase, the Triage, which highlights events with a high-risk score so that they are brought to the attention of the security analyst. Specifically, Triage assigns a threat score (e.g., low, medium, high) and adds a reason for the alert. At the end of this process, RTA updates the Profiler with new information about the event, allowing the system to learn from the entity's behavior and improve over time. This update is critical for tracking the monitored entity's future behavior.

LimaCharlie

LimaCharlie [2] is a cloud-native platform that helps organizations protect their systems from internal and external cyber threats. Using artificial intelligence, data analytics (i.e., system logs and network events), and automation it

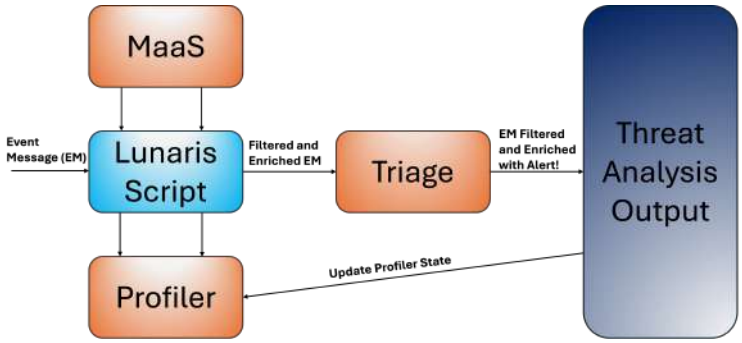

Figure 3.5. Threat Analysis Process

can detect threats, respond to incidents, and monitor anomalous activity. Moreover, the platform supports data analysis in real-time, enabling organizations to take action, such as isolating compromised systems or initiating deeper investigations, to prevent damage. LimaCharlie also offers flexible policy management, customizing pre-existing detection rules (Imported from sources such as Sigma [1], YARA [2], Zeek [3], and VirusTotal [4]) to address even complex scenarios.

The cloud-native architecture of this platform allows combining lightweight agents and scalable cloud services, offering comprehensive, flexible protection that easily adapts to different IT contexts. It includes several components, including *agents*, lightweight programs installed on endpoints (such as desktops, servers, and mobile devices) that collect data and send it to cloud services for analysis; cloud services, which analyze data through detection algorithms, generating real-time alerts; an API and a dashboard (Figure 3.6) to integrate the platform with other security tools, allowing administrators to monitor and manage better security data. Finally, among the main components are sensors and rules.

Sensors are scalable solutions that securely connect endpoints to the cloud, allowing the transmission of telemetry and artifacts from the host to the centralized organization in the cloud. They are not constantly active and can alternate between online and offline states. Therefore, the organization must define a threshold defining the maximum number of sensors connected simultaneously, which, if exceeded, will generate an event in the *deployments* stream. Each sen-

[1] **Generic Signature Format for SIEM Systems (Sigma)**: https://github.com/SigmaHQ/sigma

[2] **Yara Rules**: https://github.com/Yara-Rules/rules

[3] **Zeek**: https://zeek.org/

[4] **VirusTotal**: https://www.virustotal.com/gui/home/upload

Figure 3.6. LimaCharlie Dashboard

sor is also associated with an installation key, allowing its registration within the organization. Multiple keys can distinguish between different endpoint types, and include various properties, including the Organization ID (OID), Installer ID (IID), tags associated with sensors, and a description, providing flexible and customized control of endpoints. The rules, instead, are configurable elements in YAML format and allow you to define patterns to detect, such as indicators of compromise or suspicious behavior, and specify actions to take in response to those detections. These actions can include, for example, generating a real-time report providing a high degree of customization and control over monitoring and response activities. In addition to supporting real-time analysis, LimaCharlie offers the ability to perform post-processing of collected events using *BigQuery*, i.e., queries on historical data that allow specific aspects exploration, like the analysis of recurring IP addresses or network ports in suspicious connections.

LimaCharlie can also be extended with several add-ons to face customization and integration needs. These include collectors, which let you collect and analyze data from other sources; threat intelligence feeds, which help you spot threats by providing up-to-date info; integrations, which connect the platform to SIEM and SOAR systems; and IoT sensors, designed to monitor and protect internet-connected devices. These components extend the platform capabilities, providing a comprehensive and adaptable approach to cyber security management.

3.2 Intrusion Detection System

An IDS [144] is a computer security component that allows you to monitor the network or system to identify suspicious or potentially malicious activities. They represent a combination of hardware and software that automates all the intrusion or anomaly detection phases. In particular, they monitor network traffic or log files for suspicious events and alert when anomalous or malicious activity is detected, i.e., threats that exploit system vulnerabilities. An IDS consists of three logical components:

- *Sensors:* responsible for collecting data (network packets, log files, system call traces, etcetera) and sending it to analyzers.
- *Analyzers:* receive input data from sensors, decide whether an intrusion has occurred, and, as output, provide a report, providing some guidelines on the actions to take.
- *User Interface:* is the interface that allows the security analyst to view and control the system behavior.

In general, an IDS consists of one or more *sensors* responsible for collecting data, a *server* that analyzes the data collected by the sensors and provides leak detection, a *console* that monitors the status of the network and computers, and a *database* with a set of rules designed to identify anomalies and security breaches. IDSs are classified based on the source and type of data analyzed. So, it is possible to distinguish between Host-based IDS (HIDS), Network-based IDS (NIDS), and distributed or hybrid IDSs. In particular, HIDSs work on the information collected within a single computer system, carefully analyzing its activities and determining whose processes and users are involved in a specific attack on the operating system. The main advantages of this type of IDS are the ability to detect local attacks at the level of the single endpoint, the ability to analyze even encrypted network traffic before sending it over the network, and the ability to detect attacks that involve software integrity violations that appear as inconsistencies in the execution of processes. HIDSs can be challenging to manage because they require specific configuration and oversight for each monitored host. Additionally, since their data sources are located on the host under attack, they are vulnerable to being targeted and potentially disabled during an attack. Finally, storing the data collected by these devices requires additional local storage space on the system, causing a decrease in performance as it exploits the computational resources of the monitored hosts. NIDS, on the other hand, typically consists of several sensors positioned at various points in the network that monitor and analyze the traffic flow, reporting attacks to a central management

console. Many of these systems operate in stealth mode, meaning they do not have an IP address to avoid detection by malicious users. This feature provides a significant advantage for NIDS, allowing them to monitor wired networks without impacting system performance. However, a high traffic volume can render this tool ineffective, making the system highly vulnerable. Therefore, it becomes essential to minimize the amount of data to analyze, which can also reduce the detection accuracy. Finally, a further limitation of NIDS is the impossibility of analyzing encrypted information, which makes it necessary to analyze all the individual hosts involved to know if an attack was successful. There are also hybrid IDSs, that allow to overcome these limitations since they combine the functionality of the network and host-based IDS to provide complete and accurate protection, having visibility of both network traffic and system logs. These security tool performance evaluations regard the rate of FP, i.e., rare benign events considered anomalous, and FN, i.e., malicious events classified as benign. In particular, a high volume of false positives indicates a bad IDS configuration that leads to a high false alarm rate, causing system inefficiency. If, on the other hand, many false negatives occur, it means that the IDS is not able to detect threats effectively, so the analyst team is not informed of the ongoing attack that then exploits the system. To overcome the above limitations, IDS can be part of a wider IT security system with other components like firewalls and Intrusion Prevention Systems (IPS).

Zeek Zeek [3], formerly known as *Bro* [115], is an advanced network-based IDS designed to analyze traffic in real-time and detect suspicious or anomalous activity, acting as a true *Network Security Monitor (NSM)*. In addition to being a full-fledged IDS, Zeek offers a broad range of capabilities, including network metadata extraction, known attack detection, and host behavior analysis. Unlike many other systems, it does not just log connections but also analyzes application-level interactions.

At a high level, Zeek consists of two main components: the event engine and the script interpreter. The event engine (or *core*) transforms the incoming packet flow into higher-level events, which describe what has been observed in the network, regardless of the context or relevance of the individual event. This engine includes several essential subcomponents, such as the packet processing pipeline, composed of:

- *Input sources*: collect network traffic from configured interfaces;
- *Packet analysis*: process lower-level protocols, starting from the link layer;
- *Session analysis*: manage application-level protocols;

- *File analysis*: inspect the content of files transferred during sessions.

The engine architecture also allows plugin integration, extending Zeek's capabilities, and making it highly modular and customizable.

The script interpreter instead defines the event semantics of custom scripts in Zeek's native language, enabling security policy specification and defining actions to take in response to specific events. In addition, it organizes the collected data into well-structured log files available in tabular or JSON format, simplifying processing by external tools.

Zeek was selected for this discussion as a data source for SIEM platforms, which will use the generated logs for advanced analysis, raising alarms, and presenting the results on interactive dashboards. The choice of Zeek comes from its ability to analyze network data in a detailed and flexible way, allowing an effective extraction of information from traffic traces (*PCAP*) in the form of structured files, simplifying evaluation and integration with other platforms.

3.3 DDoSim Tool

DDoSim [108] is a simulation platform that enables large-scale DDoS botnet attack execution in realistic network environments. It models all phases of a DDoS attack, allowing real-time analysis of its effects using external tools such as Wireshark [5]. This framework helps the user to track all crucial performance metrics at runtime, such as throughput on the target server, frequency of data reception, and number of participating bots, thus indicating the development of an attack and its consequences on system performance. The network simulated by DDoSim is user-configurable and can be of the CSMA type, i.e., a shared medium access technique used in both wired and wireless networks, which allows devices to detect if the communication channel is busy and avoid data collisions or Wi-Fi, i.e., a wireless technology for connecting to the Internet and communicating between devices via radio waves, enabling researchers to explore how different attacks behave under various network conditions. The architecture of this framework is based on two core open-source technologies: Docker [6], which allows applications to be created, deployed, and managed within isolated environments called 'containers'. These containers guarantee fast, reliable, and portable execution of applications, greatly simplifying development, deployment, and scalability while maintaining a high level of consistency between development, test,

[5] **Wireshark website**: https://www.wireshark.org/
[6] **Docker website**: https://www.docker.com/

and production environments; and NS-3 [7], a widely used tool for network simulations. In DDoSim, Docker is specifically used to run each botnet component in an isolated container. While NS-3 simulates the underlying network traffic and interactions between different nodes, allowing the system to simulate a DDoS attack in a controlled but dynamic environment, to recreate realistic attack scenarios.

As shown in Figure 3.7, DDoSim simulates attacks by exploiting three main components: the **Attacker**, the vulnerable IoT devices named **Devs**, and the target server named **TServer**. In particular, the Attacker is implemented as a Docker node communicating through a "ghost" node of the NS-3 framework, is responsible for identifying vulnerabilities in devices connected to the simulated network, recruiting these devices into the botnet, and coordinating attacks against the TServer. This component includes several tools for compromising *Devs*, i.e., exploit and infection scripts, used to identify and exploit weaknesses in target devices and distribute a specific malware that turns Devs into active bots; a *Command & Control Server (C&C)* that manages this bot, coordinating the actions of the botnet, sending commands to compromised agents, and collecting information on the status of the operation, and a *File Server* that distributes resources such as configurations and updates necessary to keep the botnet up and in execution.

3.4 Experimental Setup

Dataset and Traffic Simulation To support the experiments conducted, we created a comprehensive dataset by simulating network traffic using the DDoSim tool. This dataset consists of *pcap* files that capture network traffic traces, divided into four categories: *udpplain* , *syn*, and *ack* flooding attacks, and *benign traffic*. In particular, two recording sessions of 10 and 20 minutes created different network traffic configurations characterized by transmission protocols (i.e., CSMA or Wi-Fi), number of bots (10), and attack type. This process resulted in 12 hours of network traffic, with 6 hours recorded for each protocol. Specifically, 4 hours consisted of DDoS attack traffic, while 8 hours contained benign traffic, as illustrated in [Figure 3.8].

DDoSim tool, indeed, allows traffic generation by configuring the network with the following command:

```
./main.py -n <nodes> -t <time> -net <network> -l <log> create
```

[7] **NS-3 website**: https://www.nsnam.org/

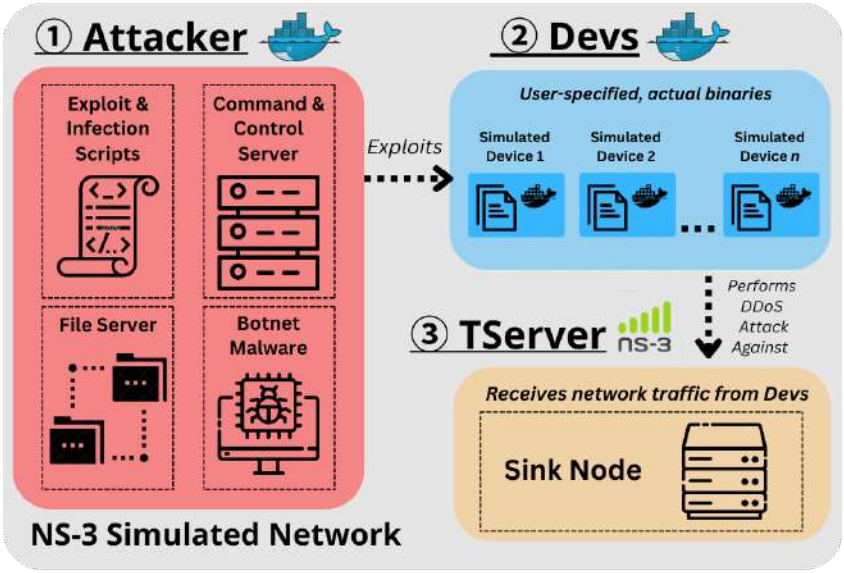

Figure 3.7. DDoSim Architecture

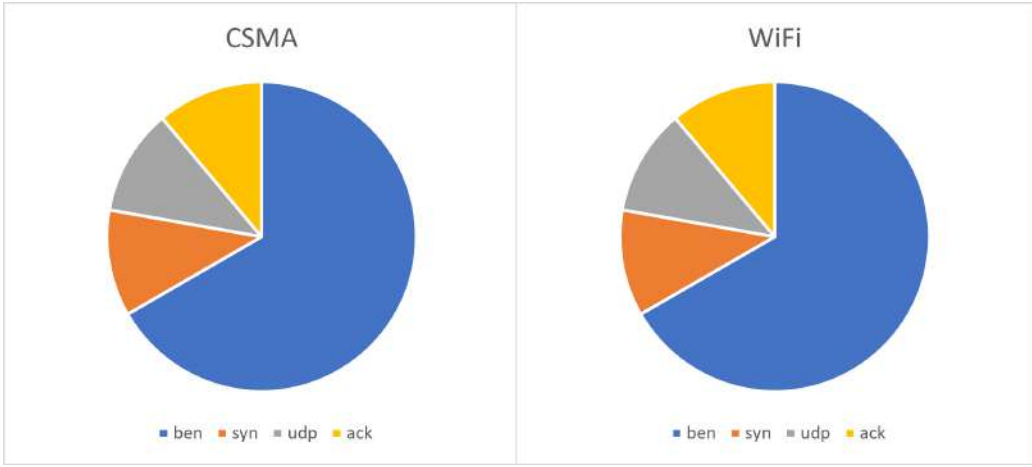

Figure 3.8. Dataset Composition

Where the key parameters are:

- **-n**: Specifies the total number of nodes, including one attacker, and *n-1*

bots.

- **-t**: Defines the simulation duration in seconds (600 or 1200 seconds).
- **-net**: Chooses the network protocol (*CSMA* or *Wi-Fi*). **-ch**: Configures node variability (*churn*). We assumed a constant number of nodes, setting *churn* to 0 (default value).
- **-l**: Enables log saving, including *pcap* files, used for our experiments, and server throughput logs (*txt* files).

After configuring the parameters, we created Docker containers to represent the network nodes, initiating the simulation by starting the ns3 simulator.

Attack Simulation To simulate attacks, we accessed the C&C container using the telnet command to launch attack commands on the connected bots. Attack commands followed the syntax:

```
<attack> <targets> <duration (seconds)> <flags>}
```

This command parameter included the attack type, target server IP, duration (100 seconds), and optional flags changing according to the transmission protocol. For example, HTTP attacks could include flags specifying methods (GET/POST), resource paths, and destination ports, while transport-layer attacks allowed settings such as TTL values and packet sizes. We repeated the attacks periodically. In particular, in 600-second simulations, we executed three attack cycles, while in 1200-second simulations, we executed five cycles. Each simulation included a bot connection phase lasting 2 to 5 minutes, depending on the number of nodes.

3.5 Methodology

This section comprehensively describes the methodology used to evaluate the two platforms, i.e., RTA and LimaCharlie. The evaluation involved a series of well-defined experiments designed to assess the platforms' ability to detect DDoS attacks, their performance under different network loads, and their ability to scale in an IoT context. We detail the experiments for both RTA and LimaCharlie, followed by a comparative analysis of the results.

3.5.1 Experiments with RTA

We used RTA to implement detection rules and monitor network traffic to detect potential DDoS attacks through the following experiments.

System Setup First, we configured RTA on Linux-based systems to capture network traffic, parse raw traffic data, and generate logs of detected anomalies. RTA configuration allows users to define custom detection rules to monitor specific network traffic patterns, such as unusually high traffic volumes.

Experiment 1: Traffic Capture and Rule Definition Then we set up RTA to capture network traffic exchanged between DDoSim Devs and TServer, defining rules to detect potential DDoS attacks based on a threshold. In particular, we set that if 30 packets were detected within 2 seconds from a single source IP to a specific destination IP, the system would flag this as a potential DDoS attack. It is important to underline that we determined the threshold of 30 packets in 2 seconds after performing empirical tests and carefully reviewing the literature on typical DDoS attack patterns. Specifically, we evaluated different threshold values to balance FP and FN, and we found that lower thresholds (e.g., 20 packets every 2 seconds) resulted in higher FP rates, signaling legitimate high-frequency traffic as malicious. Conversely, higher thresholds (e.g., 40 packets every 2 seconds) led to undetected attacks, increasing the FN rate. We therefore identified the chosen threshold as an optimal compromise, ensuring reliable detection while minimizing classification errors. Working with a rule-based SIEM, we had to choose a static threshold, although it may not fit well in IoT environments characterized by dynamic networks.

Experiment 2: Performance Testing We tested the system performance under different network load conditions. More specifically we simulated traffic from multiple IoT devices (ranging from 5 to 100 devices) to assess the platform response to varying numbers of connections and data traffic. The goal was to determine how well RTA could scale with increasing numbers of devices and to assess its ability to detect attacks when multiple devices are involved.

Experiment 3: False Positive Rate To test the system accuracy, we generated benign network traffic (such as legitimate device communication) to observe how often the system incorrectly flagged benign traffic as DDoS attacks. This experiment allowed us to assess the system sensitivity and determine if we set correctly the thresholds for attack detection.

3.5.2 Results from RTA Experiments

The results from the RTA experiments are summarized in Table 3.1.

Table 3.1. Summary of RTA Experiment Results. Detection success rate refers to the system's accuracy in correctly identifying attacks, while FP indicates the percentage of legitimate traffic misclassified as attacks and FN represents missed attack detections.

Experiment	Detection Success Rate	FP	FN
Traffic Capture	95%	2%	3%
Performance Testing	90%	4%	6%
False Positive Rate	85%	10%	5%

These experiments aimed to evaluate the platform's ability to detect DDoS attacks under various conditions. The detection success rate was highest for the *Traffic Capture* experiment at 95%, demonstrating strong detection performance in identifying anomalous traffic patterns. However, *Performance Testing* showed a slightly lower success rate at 90%, indicating some challenges in maintaining accuracy under stress conditions. The *False Positive Rate* experiment had the lowest success rate at 85%, suggesting a higher rate of misclassification. One of the main challenges observed was an increased number of FP, particularly in low-traffic scenarios where legitimate traffic was mistakenly classified as an attack. Additionally, in cases of low-volume distributed attacks, the system occasionally failed to recognize the threat, leading to an increase in false negatives, as shown in the table.

3.5.3 Experiments with LimaCharlie

We tested LimaCharlie using a different approach from RTA, which involved capturing network traffic and analyzing it with advanced tools like Zeek. We chose

LimaCharlie because of its cloud-based infrastructure and enhanced capabilities for processing large-scale traffic. The methodology for the LimaCharlie includes four experiments.

System Setup A first step of the methodology is LimaCharlie configuration to capture network traffic on Linux-based machines through its *Artifact Collection* command, which allows you to retrieve files directly from an EDR Sensor. So, using this feature, we collected raw traffic data in PCAP format. Then, we employed Zeek to analyze the captured traffic for potential signs of DDoS attacks.

Experiment 1: Artifact Collection and Traffic Capture The first experiment focused on capturing network traffic directed to TServer, where we collected PCAP files for further analysis. We used LimaCharlie to monitor traffic patterns continuously and employed Zeek to identify anomalies or signs of attack in the captured data.

Experiment 2: DDoS Detection We created a detection rule, as shown in Figure 3.9, to flag potential DDoS attacks for the second experiment. The rule identified a rapid increase in TCP connections from a single IP address to TServer port 9. In particular, we tested the rule by simulating traffic from multiple IoT devices in different configurations, observing how well LimaCharlie could detect attacks under various conditions.

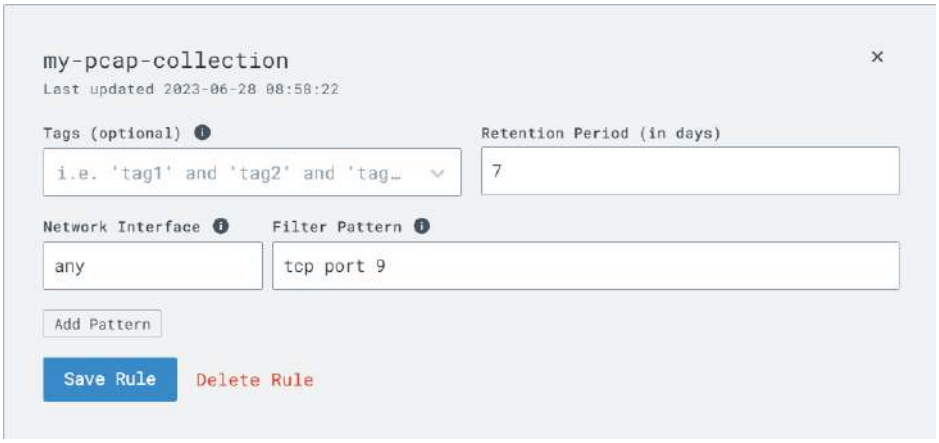

Figure 3.9. LimaCharlie Interface to Set Detection Rules

Experiment 3: Performance Testing In this third experiment, we evaluated LimaCharlie's scalability by simulating a network of IoT devices ranging from 10 to 500. More specifically, we evaluated the performance system under varying traffic loads, assessing its ability to detect DDoS attacks at different traffic volumes and numbers of devices.

Experiment 4: False Positive Rate To evaluate the system accuracy, we used DDoSim to simulate benign network traffic (such as legitimate IoT device communication), measuring the ability of LimaCharlie to distinguish between benign traffic and potential DDoS attacks by tracking the number of FP and false negatives generated during the experiment.

3.5.4 Results from LimaCharlie Experiments

Table 3.2. Summary of LimaCharlie Experiment Results. This table highlights the detection accuracy, FP, and FN for different experiment types, showing the effectiveness of LimaCharlie in various testing scenarios.

Experiment	Detection Success Rate	FP	FN
Artifact Collection	98%	1%	2%
DDoS Detection	96%	3%	4%
Performance Testing	92%	5%	3%
False Positive Rate	90%	6%	4%

Table 3.2 presents the experimental results obtained with LimaCharlie.

Compared to RTA, LimaCharlie showed consistently high detection success rates across all tests. In the Artifact Collection experiment, it achieved a 98% detection success rate with only 1% FP and 2% FN, demonstrating its effectiveness

in identifying and categorizing network traffic. The DDoS Detection experiment recorded a *detection success rate* of 96%, slightly lower than the *artifact collection phase* but still higher than RTA's equivalent test. The FN rate remained at 4%, indicating that LimaCharlie was able to maintain accurate classification even as attack patterns varied. One of the most notable differences was observed in the *Performance Testing* experiment, where LimaCharlie maintained a 92% detection success rate, with FP and FN rates at 5% and 3%, respectively. Unlike RTA, which showed a noticeable decline in accuracy under higher traffic loads, LimaCharlie's more advanced analysis techniques helped it sustain reliable detection performance. Finally, the *False Positive Rate (FPR)* experiment demonstrated that LimaCharlie had a lower FP rate (6%) compared to RTA (10%). This reduction suggests that LimaCharlie's use of external threat intelligence, behavioral analysis, and advanced correlation mechanisms helped minimize misclassifications of legitimate traffic. Overall, the results indicate that while both platforms performed well in detecting DDoS attacks, LimaCharlie's rule complexity and cloud-based architecture contributed to more stable and precise threat identification, particularly under varying traffic conditions.

3.6 RTA vs LimaCharlie: Results Considerations

Both the platforms achieved high performance in DDoS attack detection, however, LimaCharlie slightly outperformed RTA due to its cloud-based architecture that relies on extensive computational resources and sophisticated traffic analysis techniques to identify anomalies and threats effectively.

RTA exhibited a higher FP rate because it relies on static detection thresholds that led to misclassifying benign traffic as malicious, especially in complex IoT environments. LimaCharlie, on its part, uses advanced mechanisms for managing FPs that reduce their occurrence significantly. So, while both platforms experienced some FP, LimaCharlie's sophisticated analysis methods minimized this issue more effectively than RTA.

One of the key factors contributing to LimaCharlie's improved performance is the complexity of its detection rules compared to RTA. While RTA relies on a static threshold-based approach, LimaCharlie integrates more advanced rule definitions, incorporating Indicators of Compromise (IoC) and behavioral analytics. Specifically, LimaCharlie allows the use of Zeek for deep packet inspection, leveraging custom signatures and anomaly detection mechanisms. This allows for more accurate detection of attack patterns and minimizes false positives. Moreover, LimaCharlie's cloud-based infrastructure allows for the real-time enhancement of

security events using external threat intelligence feeds, which improves detection precision. In contrast, RTA mainly relies on preset rules and does not adjust dynamically to changing attack patterns. This distinction explains the observed differences in detection success rates between the two platforms.

The cloud-native design of LimaCharlie also allows it to manage large volumes of traffic and a high number of IoT devices seamlessly, thus outperforming RTA in scalability, which, instead, struggled to maintain performance under heavy network loads, making it less suitable for large-scale IoT deployments.

On the other hand, RTA offers a more user-friendly interface, particularly for users familiar with creating detection rules using a DSL. Its customizable dashboard and guided rule configuration make it approachable for analysts. However, the need for specific DSL syntax and the lack of compatibility with pre-existing detection rules may increase the learning curve and setup time. LimaCharlie's interface, instead, while less intuitive, provides advanced features such as the ability to import external detection rules and automatic incident response capabilities. Additionally, its integration with tools like Zeek and extensive documentation supported by an active community enhances its usability for experienced users.

Challenges in the IoT Context Security monitoring in IoT environments faces unique challenges due to changing traffic patterns, shifting IP addresses, and various protocols. RTA and LimaCharlie offer strong security monitoring tools, but they have specific weaknesses impacting their performance in IoT networks. RTA struggles to keep up with the changing nature of IoT settings. It uses static thresholds to detect issues, often leading to false alarms. This happens when normal device communication looks similar to possible threats. Additionally, RTA does not automate the management of these false alarms, making it less effective in large-scale IoT deployments where it needs to adapt quickly. On the other hand, LimaCharlie is good at scaling up, but it has a drawback in IoT situations. It relies on known IP addresses to detect attacks, which can be problematic because devices frequently change their IPs. Threats may come from various sources across the network, making them harder to spot. Moreover, managing large volumes of IoT traffic on a cloud platform can increase costs based on how many devices are monitored and how much data is generated.

3.7 Final Discussion

In this study, we analyzed the two SIEM platforms, RTA and LimaCharlie, to assess their adaptability in IoT environments by evaluating their ability to detect

DDoS attacks. To support the experiments, we created a representative dataset using the DDoSim tool, simulating various attack scenarios to test detection capabilities under different network conditions.

Both platforms demonstrated strengths and weaknesses. LimaCharlie proved to be highly scalable and exhibited advanced detection capabilities, particularly in handling high-traffic scenarios and integrating external threat intelligence. RTA, on the other hand, provided a more user-friendly interface, facilitating the creation of detection rules through an intuitive DSL-based rule configuration system. However, RTA showed higher FPR and lower adaptability to increasing traffic volumes, limiting its effectiveness in highly dynamic environments.

The scalability evaluation highlighted significant differences in how RTA and LimaCharlie handle increased traffic loads. RTA's detection success rate declined as network traffic increased, dropping from 95% in controlled conditions to 90% under high-load scenarios. This decline is linked to its static threshold-based detection, which does not dynamically adapt to changing traffic patterns. Additionally, the FN rate increased from 3% to 6%, indicating that low-volume distributed attacks were occasionally missed as the number of concurrent connections grew.

Conversely, LimaCharlie maintained a more stable detection performance, achieving 92% accuracy even in high-traffic conditions. This stability is attributed to its Zeek-based deep packet inspection and real-time telemetry analysis, which allow for more granular anomaly detection. Moreover, LimaCharlie demonstrated a lower FPR (6%) compared to RTA (10%), suggesting that its rule complexity and threat intelligence integration helped minimize misclassifications. However, being a cloud-native platform, LimaCharlie introduces additional considerations, such as potential latency in environments with extremely high data throughput and the need for adequate resource allocation to manage traffic spikes efficiently.

Despite these differences, both platforms rely on rule-based detection, meaning their effectiveness is dependent on how well detection rules are defined and updated. Future improvements could involve the integration of machine learning models to enhance adaptive threat detection, reducing false negatives and improving overall detection accuracy. RTA could benefit from more dynamic threshold tuning mechanisms, while LimaCharlie could further optimize resource management for large-scale deployments.

Ultimately, the choice between the two platforms depends on organizational priorities. If ease of use and local processing are key requirements, RTA offers a more accessible solution. However, if scalability, lower FPR, and advanced traffic analysis are priorities, LimaCharlie represents a more robust option for complex and dynamic IoT environments.

Chapter 4

Lightweight IDS Development

This chapter presents a lightweight IDS development and evaluation focusing on detecting DoS attacks with low power consumption. We analyze the trade-off between computational constraints and the performance of IDS through a simulation environment, focusing on the relevance of energy efficiency in IoT and IIoT applications. The contribution proposed, already discussed in our work [34], belongs under the principles of GreenAI, i.e., energy efficiency and resource optimization.

4.1 Green AI

Green AI refers to AI use that prioritizes energy efficiency and sustainability to address the environmental and societal challenges posed by the existing resource-intensive AI practices. Indeed, this new paradigm reduces AI models' environmental footprint by optimizing resource usage in training and inference. Techniques such as smaller datasets, efficient algorithms, and renewable energy sources drive this efficiency. Verdecchia et al. [149] demonstrated that with GreenAI techniques, it is possible to achieve 115% energy savings, recognizing hyperparameter tuning and model monitoring as two essential strategies to reduce energy consumption. Notwithstanding these developments, there is a compromise to be made between the principles of the environment and AI quality, as noted by Barbierato [21], who explain how lowering computational

intensity can also mean accepting lower performance. Green AI implementations allow social, environmental, and economic sustainability, key elements for Smart Cities development. For instance, researchers such as Yigitcanlar et al. [157] highlight how Green AI can facilitate a transition to smart cities and sustainable development goals to promote the development of alternative AIs that respect the environment without compromising technology development. In particular, they describe *green sensing* as a potential solution and consider its relevance for urban planning. Green AI has already transformed other industrial applications, such as the IIoT. Zhu et al. [165] illustrated how this intelligent edge computing approach slashes energy consumption by offloading tasks from centralized servers onto edge devices. The authors proposed an optimal scheduling algorithm that ensures better energy efficiency than traditional static and FIFO strategies, giving one more example of Green AI leading to sustainable industrial practices. Despite its promise, Green AI faces barriers to adoption, including the lack of readily available tools and challenges related to scalability. Verdecchia et al. [149] noted that bridging the gap between academic research and industry applications remains essential. Raising awareness and integrating Green AI principles into policies will be crucial for its widespread implementation, ensuring alignment with environmental and economic objectives.
The lightweight IDS we propose in this chapter aligns with Green AI principles. Specifically, deploying an energy-efficient IDS that functions well on resource-constrained devices aligns with Green AI's overarching goals of improving efficiency and sustainability while maintaining effective intrusion detection capabilities.

4.2 Lightweight IDSs Background

Efficient IDSs are necessary for ensuring IoT ecosystem security since they are environments with limited computational and energy resources. Indeed, IoT devices typically operate under strict constraints, so, lightweight IDSs emerged as a practical solution to balance detection performance with resource efficiency. Currently, researchers are studying the adaptation of Lightweight IDS in several contexts of IoT, underlining its advantages and limitations. For example, Hazman et al. [57] presents an anomaly detection model based on ensemble learning for IoT-based Smart cities. The model utilizes ML algorithms such as Support Vector Machines and Random Forests to accurately and timely detect anomalies in the data generated in Smart city environments. Guezzaz et al. [53] propose a lightweight and hybrid intrusion detection framework for securing edge-based IIoT systems. The framework employs ML techniques to identify intrusions

and suspicious activities while minimizing the impact on limited edge device resources. Gazdar et al. [49] introduce a ML-based IDS for smart homes. The system is trained on representative legitimate behavior data and successfully detects significant deviations, alerting potential intrusions. Idrissi et al. [62] present a lightweight, optimized deep learning-based IDS deployed directly on edge devices for IoT. The system utilizes a convolutional neural network trained on IoT network traffic data. Raja et al. [120] propose an energy-efficient end-to-end security system for Software-Defined Vehicular Networks (SDVN). The approach combines lightweight cryptographic algorithms with energy optimization techniques. Simulations demonstrate a good balance between security and energy consumption.

Despite these studies' contributions, various shortcomings remain because they may be computation-intensive or require huge volumes of data, which could be difficult to collect or process in real time, especially in resource-constrained contexts. Moreover, while certain approaches efficiently balance performance with energy efficiency, they generally require major compromises in accuracy or generalizability, thus seriously impeding broader applicability in diverse IoT contexts.

Building on these developments, we created a robust framework for evaluating the performance and energy consumption of lightweight IDSs in IoT scenarios. In particular, rather than design a new IDS, we implemented a simulation environment to evaluate a Lightweight IDS that uses the K-means algorithm for anomaly detection. Moreover, to address the challenges of working with realistic data, we generated a comprehensive dataset simulating real-world data, specifically generated by a Train Communication & Monitoring System (TCMS) network of a high-speed train. This dataset mitigates previous studies' limitations, offering a more controlled, realistic context for evaluating IDS performance, especially energy efficiency and computational constraints.

4.3 Methodology

This section presents the proposed methodology described in Figure 4.1, consisting of rating the sustainability of an IDS deployed in an embedded network, assessing the limited resources IDS' power consumption and accuracy.

Simulation Environment The first step in our methodology is to develop a sophisticated and detailed simulation environment, emulating as closely as possible the onboard TCMS of a high-speed train using Mininet-WiFi [122], to study

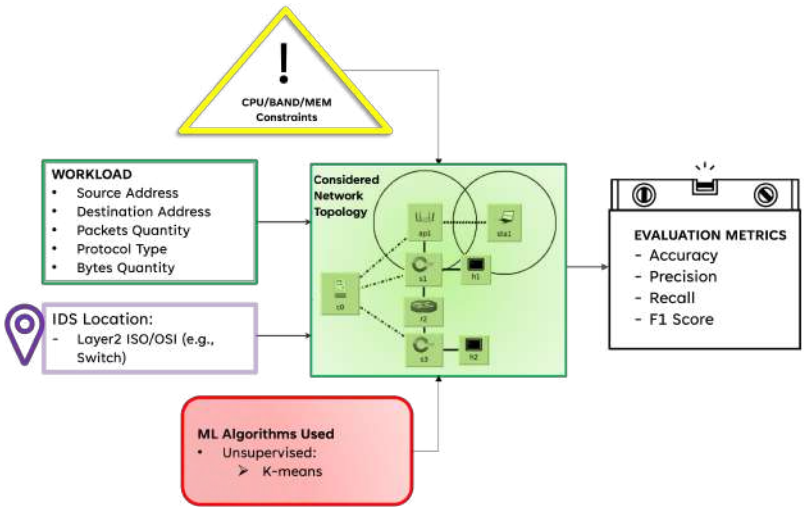

Figure 4.1. Proposed Methodology

the performance and energy efficiency of lightweight IDSs. This simulation environment creates a controlled yet realistic setting to generate and analyze diverse network traffic patterns, both benign and malicious. The system architecture models the operational intricacies of TCMS, thus enabling a very detailed emulation of traffic dynamics. This involves several components integration to emulate real-world scenarios like network latencies, protocol interactions, and typical IoT device behaviors.

Data Collection Training ML or DL models for industrial applications often face data lack issues, so we address the problem using traffic generators (i.e., httperf [111] and iperf [1]) to create a benign traffic model, from which we extract features relevant to cybersecurity attack detection, including source and destination IP address, protocol, source and destination port, etc. In particular, we chose these features by analyzing relevant attributes from data and considering network behavior and patterns. Specifically, for the evaluation, we collected network traffic traces generated within the previously simulated TCMS network, creating a dataset whose total size is approximately 12 hours of recorded traffic, comprising both benign and attack scenarios. Specifically, the dataset includes 8 hours of normal traffic and 4 hours of DDoS attack traffic, which was generated using a combination of SYN flooding, UDP flooding, and ICMP flooding attacks. We structured the dataset to ensure a representative distribution of real-world

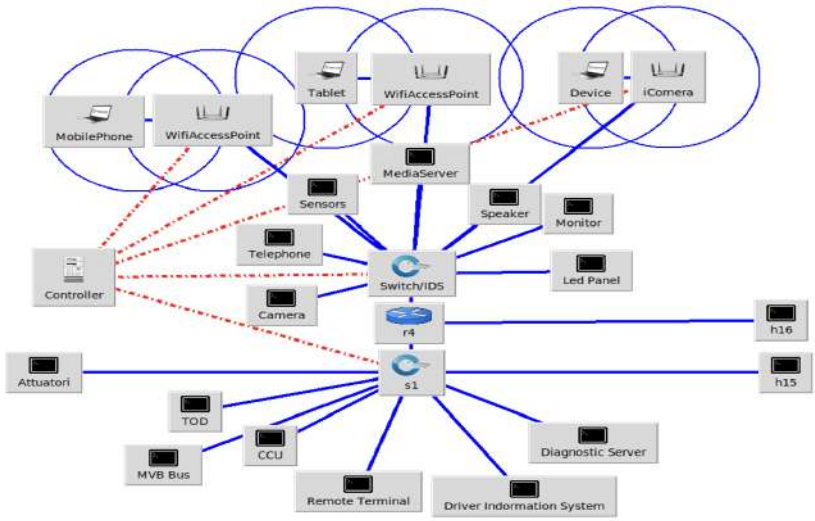

Figure 4.2. Considered Network Topology

network conditions, allowing for a balanced evaluation of the IDS in detecting anomalies while maintaining energy efficiency.

IDS location Another important step in our methodology was determining the optimal location for the IDS to capture most of the traffic between the involved components. So, considering the network topology in Figure 4.2, we identified the router (r4) as the optimal candidate to place the IDS. Indeed, since the router is responsible for forwarding packets from one device to another, it is the only device that allows the IDS to control the entire traffic circulating in the network.

Unsupervised ML for Anomaly Detection Since the collected data is unlabeled, we utilize an unsupervised clustering algorithm to differentiate between malicious activities and normal behavior. Specifically, we chose the K-means algorithm for data analysis due to its effectiveness in handling large datasets [10]. We use Silhouette analysis and train the model according to this technique to determine the optimal number of clusters. The algorithm assigns new data points to clusters by evaluating their distances from the centroids. It identifies anomalies based on a threshold derived from the maximum distances.

Implementation of the IDS To evaluate the effectiveness of the IDS in resource-constrained scenarios, we deploy it inside a Docker container, which is a widely used software package that combines the code and its dependencies, allowing the application to operate swiftly and consistently across various computing environments [39].

Attack Simulation Utilizing Scapy, a Python library capable of manipulating, decoding, and transmitting packets for different protocols [130], we modeled specific types of Denial of Service (DoS) attacks, with a focus on SYN flooding attacks, UDP flooding attacks, and ICMP flooding attacks. These attacks entail sending a high volume of SYN requests to the targeted system to exhaust sufficient resources, rendering it inaccessible to legitimate traffic.

Assessment of IDS Performance As a conclusion we conduct a thorough evaluation of the proposed Lightweight IDS accuracy considering the most widely used metrics in the field of anomaly detection, which are Accuracy, Recall, Precision, and F1-Score, detailed described in the following:

1. **Precision**: gauges the ratio of True Positive (TP) predictions (correctly identified instances) to the total instances number predicted as authentic. It highlights the model's ability to make accurate predictions.

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

2. **Recall**: also known as True Positive Rate (TPR) or Sensitivity, quantifies the ratio of TP predictions to the total number of authentic threats. It assesses the model's capability to identify all relevant positive cases.

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

   Where FN represents False Negatives.

3. **F1-Score**: is a harmonic mean of Precision and Recall, providing their balance, and offering a comprehensive model's performance evaluation in binary classification tasks.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

4. **Accuracy**: measures the proportion of correctly predicted instances over the total instances number. It provides an overall assessment of the model's

correctness in its predictions.

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

Where TN represents True Negatives.

This assessment goal is to ascertain whether the IDSs can achieve and maintain a high level of accuracy despite the limitations imposed by available resources.

Over the models' accuracy, we also assessed their sustainability and their compatibility with IoT devices. In particular, we utilize the PowerTOP tool [94], a utility for Linux, to monitor the energy consumption of the IDS. This analytical tool uses sampling and evaluation techniques to observe power usage on the system, estimating the power consumption of the devices by periodically sampling different system metrics like CPU frequency, memory usage, disk I/O, and power supply voltage.

4.4 Preliminary Evaluation of the IDS with Resource Constraints

This section outlines a preliminary evaluation focused on analyzing IDS performance under varying constraints related to CPU and RAM. Resource limitations, particularly in processing power and memory, play a crucial role in the effectiveness of the IDS in processing and analyzing network data, particularly in resource-constrained IoT environments.

For our experiments, we utilized a virtual machine configured with 4 GB of RAM and a CPU clock speed of 2.6 GHz, operating on the Ubuntu platform. To ensure reproducibility, the experimental code has been made publicly accessible through an open-source repository[1].

Optimizing computational efficiency and memory management is essential for achieving optimal performance in anomaly detection, particularly in analyzing and classifying network traffic patterns. It is crucial for lightweight IDS solutions to maintain a balance between computational efficiency and sustainability, minimizing resource consumption in the process. These considerations

[1]**Framework Repository:** https://github.com/dessertlab/SEEliDS.git

become increasingly significant in high-traffic or real-time scenarios, where system responsiveness is critical to ensuring effective intrusion detection. To explore this balance, we compared the IDS performance under both constrained and unconstrained resource conditions, focusing, in particular, on CPU and RAM limitations to identify an optimal trade-off between performance and resource utilization. In particular, concerning CPU, we analyzed how CPU speed and the number of cores impact IDS performance under different processing constraints.

The effects of CPU speed and the number of CPU cores on IDS performance mainly result from processing latency differences and workload distribution differences. Increased CPU speed allows the IDS to carry out more instructions per second, which decreases the time it takes to process the incoming network packets. As the CPU speed is lowered, the inspection of the packets and the extraction of features are delayed, which causes delays in classification as well as an increased likelihood of misclassification.

Furthermore, multi-core processing greatly improves IDS effectiveness, since network traffic analysis is a naturally parallel task. A quad-core system allocates incoming packets to various threads, enhancing throughput and real-time detection abilities. On the other hand, a dual-core system might encounter bottlenecks under heavy traffic, resulting in longer packet queuing times and possible delays in detection.

These findings emphasize the importance of matching IDS deployment configurations with available hardware resources. Deploying IDS on low-power embedded devices may require lightweight detection techniques or adaptive load-balancing strategies to optimize computational efficiency while maintaining high detection accuracy.

4.5 Experimental Setup

We limited CPU and RAM availability through specialized tools to simulate resource-constrained environments. In particular, we employed the *cpulimit* utility to regulate CPU usage, thereby effectively controlling the allocation of CPU cycles to the IDS [4]. This utility is instrumental in managing batch tasks while preventing excessive resource consumption.

For our evaluation, we emulated the hardware limitations characteristic of a Raspberry Pi [123], a cost-effective and compact device commonly utilized in Internet of Things (IoT) applications. The Raspberry Pi's capabilities in conducting analytics, enabling device connectivity, and processing sensor data

render it an appropriate benchmark for lightweight IDS scenarios. Specifically, we assessed the IDS under three distinct CPU speed configurations:

- 0.9 GHz: Corresponding to the CPU clock speed of the Raspberry Pi 2 Model B.
- 1.5 GHz: Equivalent to the CPU clock speed of the Raspberry Pi 4 Model B.
- 2.6 GHz: Reflecting the total CPU speed of the experimental virtual machine.

We also analyzed the impact of different CPU core counts on performance by assessing configurations with two (dual-core) and four cores (quad-core).

To limit RAM usage, we used the *ulimit* command, a Unix utility that manages resource allocation for processes, including memory consumption [109]. Preliminary experiments with different memory allocations showed that the IDS fails to operate with less than 200 MB of RAM, resulting in immediate memory errors. When we allocated 1 GB of RAM, the IDS could run a little before ultimately crashing. Based on these findings, we decided to set a RAM limit of 2 GB for our experiments, which is a realistic constraint that fits within the memory capacities of many embedded systems.

We conducted the IDS evaluation using a workload comprising 6 hours of simulated network traffic, incorporating both benign and malicious activities. The tested attack scenarios included ICMP, UDP, and SYN flood attacks, strategically incorporated with intervals of harmless traffic. We employed this methodology to assess the IDS's capacity to adapt to fluctuating network conditions.

4.6 Experimental Results

4.6.1 Performance Evaluation

Table 4.1 provides an overview of the performance metrics for the IDS across various CPU configurations, with the Random Access Memory (RAM) fixed at 2 GB. As expected, the results indicate a decline in performance as both CPU speed and the number of cores are reduced.

A reduction in CPU speed from 2.6 GHz to 1.5 GHz led to an average decrease of approximately 7% across all evaluated metrics, while a further decrease to 0.9 GHz resulted in a performance decline of about 15%. Similarly, a limitation

Table 4.1. IDS Performance Under Varying CPU Constraints.

Metric	CPU Speed Clock	Dual Core	Quad Core
Accuracy	0.9 GHz	72.5%	77%
	1.5 GHz	80.7%	85.1%
	2.6 GHz	85.6%	90.8%
Precision	0.9 GHz	61.7%	65.7%
	1.5 GHz	70.9%	74.1%
	2.6 GHz	80.4%	84.3%
Recall	0.9 GHz	78.7%	82.7%
	1.5 GHz	84.5%	89.7%
	2.6 GHz	90.8%	94.8%
F1-Score	0.9 GHz	69.2%	73.2%
	1.5 GHz	77.1%	81.2%
	2.6 GHz	85.3%	89.2%

in the number of CPU cores from 4 to 2 was associated with an approximate performance reduction of around 4%.

Table 4.1 highlights how the IDS adapts to resource constraints while maintaining its core detection capabilities. When facing high-volume DDoS attacks, the IDS effectively identifies its operational limits and responds accordingly. Rather than interpreting the drop in accuracy as a failure, this behavior demonstrates a built-in mechanism that signals when the system reaches capacity. In real-world applications, such a response can trigger an alert, prompting adaptive measures such as load balancing, traffic filtering, or resource scaling to ensure continuous protection. Instead of focusing on accuracy decline, a more relevant indicator of IDS performance in these scenarios is its ability to maintain timely data processing.

Further analysis of the results in Table 4.1 indicates that CPU speed as well as the number of CPU cores have a significant impact on IDS performance. As CPU speed drops from $2.6GHz$ down to $0.9GHz$, accuracy falls from 85.6% down to 72.5% for dual-core setups and from 90.8% down to 77% for quad-core setups. A similar tendency is seen in precision, recall, and F1-score, which shows that lower

processing power limits the IDS's ability to classify network traffic accurately.

The number of CPU cores is also vital in optimizing detection effectiveness. IDS performed $4-5\%$ better across all metrics when operating quad-core configurations compared to dual-core configurations. It is assumed here that multi-core processing enables enhanced allocation of incoming traffic, reducing classification latencies as well as enhancing the IDS capabilities in event detection in real time.

Another important consideration is the CPU frequency's role in influencing Recall. At the lowest level of CPU frequency (0.9 GHz, dual-core), the recall is also quite high at 78.7%, reflecting the IDS's preference for detecting threats even under low computational resources. However, the decrease in precision at low CPU levels (from 80.4% at 2.6 GHz down to 61.7% at 0.9 GHz) reflects the introduction of more FP since there is low processing power with which the IDS can identify benign from harmful traffic. These results indicate the requirement for optimizing IDS placement based on the resources available as well as the implementation of adaptive resource management policies to optimize detection even under low resources.

To assess the system's responsiveness, we evaluated latency, defined as the time between packet arrival and processing by the IDS. The results indicate that latency remains consistently low, ensuring that the system promptly analyzes incoming traffic. This is a critical factor in real-world applications where timely threat detection is essential. Future improvements will focus on optimizing resource allocation to maintain efficiency even under high-traffic conditions, further enhancing system adaptability.

4.6.2 Energy Consumption Analysis

Since power consumption is a crucial factor for IDS deployment in low-power IoT environments, we monitored energy usage under different experimental conditions. Table 4.2 presents a detailed breakdown of power consumption for each test scenario.

These measurements illustrate that CPU power consumption is highly sensitive to both CPU frequency and workload intensity. Under normal operation, power consumption is $0.25W$ when running at full processing capacity (2.6 GHz, 4 cores), decreasing to $0.15W$ when the CPU is throttled to 0.9 GHz.

In high-intensity attack scenarios like SYN flooding, power consumption hits 1.5W—sixfold higher than typical traffic conditions. This increase is probably caused by heightened CPU utilization and more frequent memory accesses as the IDS handles a notably greater volume of packets in real-time. In cases where

Experiment	CPU Speed	Core Count	Power Consumption (W)
Benign Traffic	2.6 GHz	4 cores	0.25
	1.5 GHz	2 cores	0.19
	0.9 GHz	2 cores	0.15
High-volume SYN flooding attack	2.6 GHz	4 cores	1.50
	1.5 GHz	2 cores	1.12
	0.9 GHz	2 cores	0.98
Mixed attack scenario (SYN + UDP + ICMP)	2.6 GHz	4 cores	1.45
	1.5 GHz	2 cores	1.05
	0.9 GHz	2 cores	0.92

Table 4.2. IDS Performance Under Varying CPU Constraints.

the attack involves a mix of SYN, UDP, and ICMP flooding, the power usage is marginally lower compared to a situation of purely SYN flooding, likely due to the different methods by which various attack types create and spread network traffic.

These results emphasize the significance of optimizing IDS for power efficiency through strategies like dynamic frequency scaling (DFS) and engaging low-power modes in idle times. When implementing IDS on battery-operated IoT devices, adaptive power management strategies must be taken into account, allowing the system to adjust processing resources dynamically according to observed network conditions.

4.7 Final Discussion

This chapter has outlined the design, implementation, and evaluation of a lightweight IDS framework tailored for IoT and IIoT environments, emphasizing the balance between energy efficiency and intrusion detection performance. Through the simulation of real-world scenarios, such as the TCMS of high-speed trains, and the incorporation of Green AI principles, we demonstrated that resource constraints can be managed effectively without compromising the core functionality of IDS.

The experimental results confirmed the feasibility of deploying lightweight IDSs in resource-limited environments. The trade-offs between computational resources, detection accuracy, and energy efficiency underline the importance of optimizing IDS configurations based on hardware capabilities. Specifically, CPU speed and the number of cores directly impacted IDS performance, with a reduction from 2.6 GHz to 0.9 GHz leading to an average 15% performance drop, while quad-core setups consistently outperformed dual-core configurations by approximately 4-5%. Additionally, power consumption analysis revealed that IDS energy usage increased from 0.25W to 1.5W during attack scenarios, emphasizing the need for energy-aware optimizations in low-power IoT devices. Future research could explore dynamic power scaling strategies to optimize energy consumption without compromising detection accuracy.

Despite these challenges, the proposed solution achieves scalability, adaptability, and sustainability in dynamic IoT ecosystems by leveraging unsupervised machine learning techniques, such as K-means clustering, and employing containerized deployment. These results provide a foundation for the continued development of efficient, low-power security solutions for resource-constrained systems.

Chapter 5

DDoShield-IoT IDS Testbed and Data Augmentation Technique

As cyber threats evolve, robust solutions to protect networks are becoming increasingly necessary, especially in IoT contexts where the large-scale distribution and heterogeneity of the components involved exacerbate the whole thing. In this context, IDSs play a key role in detecting and mitigating these threats, among which, as mentioned above, the most common include DDoS attacks. Therefore, to effectively develop and evaluate IDSs to be used in these environments and to optimize their potential, it becomes essential to have test platforms capable of faithfully simulating IoT traffic and environments. To this end, the following chapter presents a contribution already discussed in our work [35], or DDoShield-IoT, a testbed designed to meet these specific needs. It is based on Docker containers and the NS-3 network simulator and allows the creation and analysis of IDSs in realistic contexts, facilitating both reproducibility and scalability of the tests. As use cases, leveraging the proposed framework, we implemented and evaluated several ML-based IDSs, obtaining significant results, including an accuracy of over 90% in detecting malicious traffic generated by botnets such as Mirai. Furthermore, the DDoShield-IoT allows us to analyze the performance metrics, e.g., CPU and memory consumption, highlighting its usefulness in optimizing security solutions. In this chapter, we also addressed another problem of IoT environments, namely the lack of representative data. So, we proposed a Data Augmentation approach that leverages DDoShield-IoT to

improve the efficiency of IDSs. This strategy is helpful, especially in detecting new threats that IDSs face, such as zero-day attacks, by widening the gaps and improving ML models' training data quality. Ultimately, the integrated approach of the Data Augmentation technique with the framework leads to building more sophisticated and robust security solutions to protect IoT networks.

5.1 IDS Testbeds Background

Researchers have recently proposed diverse testbeds and methodologies to enhance IoT security and implement more efficient IDS. For example, Bhayo et al. [23] develop a testbed based on SDN-IoT traffic emulation to evaluate DDoS detection algorithms, enabling detailed parameter analysis. Zolanvari et al. [167] create a testbed for IIoT environments, incorporating ML techniques to analyze data from a simulated industrial plant monitoring water levels and turbidity. This testbed integrates hardware and software to simulate benign and malicious traffic for evaluating ML models. Anthi et al. [16] design a smart home testbed to evaluate IDS for home IoT devices. While Nie et al. [105] develop a testbed emulating an Internet of Vehicles (IoV) scenario with DDoS attack simulations, including TCP, UDP, and HTTP flooding. Albulayhi et al. [12] propose a testbed incorporating real IoT devices like AI speakers and Wi-Fi cameras to simulate diverse attacks. While Kumar et al. [73] introduce *EDIMA*, a testbed leveraging ML to detect malware by analyzing IoT network traffic. Zachos et al. [159] present the IoT/IoMT Security Testbed, which uses Raspberry Pi devices to emulate IoT sensor behavior, offering simulation capabilities but lacking comprehensive features for IDS testing. Other contributions include Eskandari et al. [44] Passban IDS, which focuses on anomaly-based intrusion detection for IoT edge devices, and Obaidat et al. [108], which emphasizes large-scale botnet DDoS attack simulation but does not provide real-world traffic or an IDS evaluation component.

While these approaches significantly enhance IoT security research, they also face limitations. Indeed, many testbeds struggle to i) fully replicate complex IoT environments, ii) emulate attack and traffic scenario diversity, and iii) lack integrated features for deep IDS evaluation. So, we propose DDoShield-IoT to address these challenges. Indeed, by combining Docker containers with the NS3 network simulator, we gave it the flexibility needed to simplify IDS setup and configuration, and we also enhanced the realism in IoT environment simulation, allowing DDoShield-IoT to generate real benign and malicious traffic.

5.1.1 DDoShield-IoT Architecture

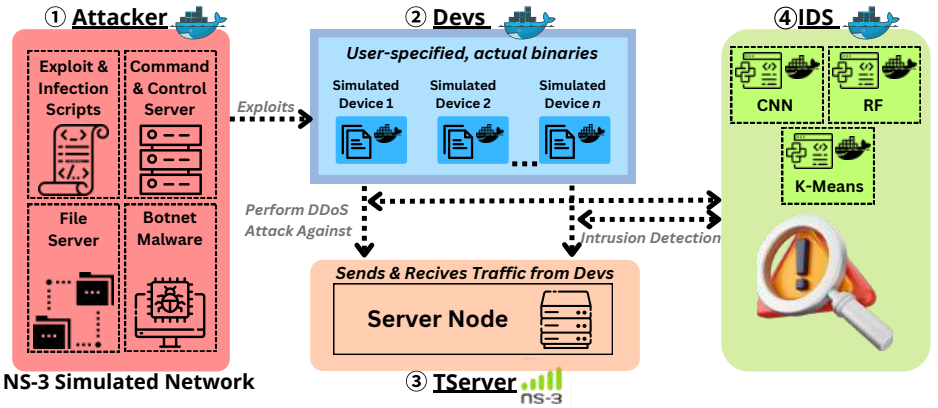

Figure 5.1. DDoShield-IoT Overview

In this section, we introduce DDoShield-IoT, an IDS simulation testbed derived from the pre-existing DDoSim framework [108], of which we have retained the main features, adding the ability to generate a representative dataset including both benign and malicious traffic, and also adding the ability to deploy and evaluate the performance of IDSs in IoT environments.

5.1.2 DDoSim Overview

DoSim [108] is a simulation environment designed to mimic large-scale botnet DDoS attacks in a realistic context. It facilitates modeling of all attack phases, allowing real-time analysis at different stages, such as assessing the severity of the impact, changes in the throughput of the target server, the average rate of data reception, and the number of connected bots, using external tools such as Wireshark.

DDoSim allows all its components to communicate via a configurable network adapted to various applications, including CSMA and Wi-Fi networks. This is possible thanks to the integration into the framework of two external components such as i) Docker, which is an open-source platform for building, deploying, and managing applications in isolated settings known as containers, and ii) NS-3, which is an open-source simulation framework for communication networks. Indeed, DDoSim leverages the dynamic capabilities of Docker containers to emulate

the essential components of a standard botnet DDoS attack scenario, namely, the attacker, the Devs, and the TServer.

In particular, the attacker component has the role of orchestrating the attack by exploiting vulnerabilities found in the Devs and distributing botnet malware. Specifically, the attacker infiltrates and compromises Devs using exploit and infection scripts specific to leverage their vulnerabilities. As a consequence of this infiltration, the Attacker leads to the installation and execution of botnet malware in Devs, establishing a foothold for command and control activities. The Command and Control Server, i.e., a critical subcomponent of the attacker, acts as a central hub to coordinate the actions of the compromised Devs, issuing synchronized commands to launch botnet DDoS attacks on the targeted TServer.

The Devs represent diverse internet-connected IoT devices, which we have already deployed as Docker containers with vulnerable binaries. These containers emulate the vulnerability of real devices to cyber threats, acting as unwitting participants in the botnet.

The TServer is the target of this attack. In the NS-3 network, it is a virtual node whose purpose is to bear the heavy load of malicious traffic generated by the infected Devs.

Ultimately, each Docker container, representing an attacker or a developer, connects to the simulated NS-3 network via specialized bridge configurations. These configurations enable seamless communication and data transfer between simulated components, accurately reflecting real-world network dynamics.

Additionally, DDoSim enables you to evaluate several factors typical for complex IoT networks, such as the abandonment rates of different devices involved and the duration of attacks. For example, by adjusting abandonment rates, DDoSim makes it easy to evaluate how device mobility and connectivity impact the resilience of the TServer to botnet DDoS attacks. Similarly, modifying the duration of attacks enables the evaluation of the progression of attack intensities over time and develop proactive defense strategies as needed. The data collected from these experiments provides critical benchmarks for evaluating the effectiveness of defense mechanisms, which may include IDS, traffic filtering, and mitigation techniques. By replicating experimental conditions and configurations, DDoSim ensures that its simulations closely resemble the behavior of physical network environments.

5.1.3 DDoShield-IoT's Improvements to DDoSim

As illustrated in Figure 5.1, our framework enhances DDoSim architecture and functioning (described in detail in Figure 3.7) by improving the TServer and adding a new component, through a new Docker container that simulates an IDS able to identify botnet DDoS attacks in real-time performed by Devs against the TServer. Specifically:

- The TServer in DDoShield-IoT instead of receiving only traffic from Devs becomes able to generate benign traffic to exchange with the Devs by including three primary servers Apache, Nginx, and a custom FTP-Server. More servers inclusion that generate diverse types of traffic namely HTTP, RTMP, and FTP traffic, is important to have a variety of standard activities in the simulation environment to emulate the complex IoT scenarios and avoid unbalanced datasets. In particular, the benign traffic produced by the TServer helps intrusion detection algorithms recognize valid traffic patterns, creating a baseline for anomaly detection. The presence of benign traffic in the dataset also lets us i) evaluate the algorithms' adaptability to changes in dynamic network activity, and ii) minimize false positives by allowing systems to distinguish between legitimate and potentially malicious actions. Including both legitimate and malicious traffic in the botnet simulation, DDoS attack detection becomes a more realistic challenge and helps improve the effectiveness and resilience of ML security solutions.
- The Real-Time IDS component uses ML models to detect botnet DDoS attacks. In particular, depending on user requirements, leverages one of the following algorithms: *CNN*, *RF*, or K-Means.

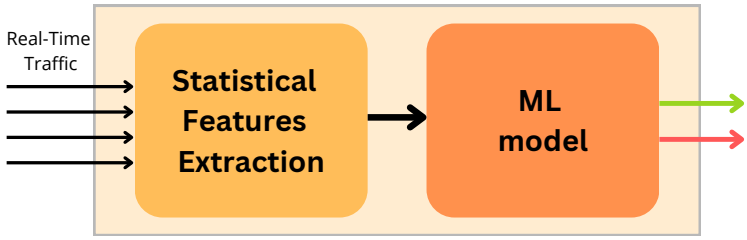

Figure 5.2. IDS Component Overview

As shown in Figure 5.2, the IDS functions in three phases: i) monitoring traffic in real-time, ii) data pre-processing, and iii) detecting the attack.

Network traffic analysis starts with extracting features, by aggregating incoming data within a user-defined time frame (for instance, we chose a 1-second time window in our experiments). This stage computes the statistical features consistent for each packet within the time window, combining them with the basic ones. This step is crucial to prevent misclassifying packets from different classes (malicious or benign) within the same time window, thereby enhancing overall accuracy. Once the framework completes the data preprocessing phase, the previously trained ML model conducts the intrusion detection task on the new data in real time. It supports traffic classification as benign or malicious and provides accuracy for each intrusion detection task within the specified time window.

5.1.4 Experimental Evaluation

This section evaluates the effectiveness and adaptability of the DDoShield-IoT as an IDS testbed. In particular, we evaluate the framework's ability to prepare data, extract features, and undergo evaluation metrics and resource consumption analysis to understand whether it can be considered a complete platform for implementing and evaluating IDS in IoT contexts. To achieve our goal, we perform several experiments using a laptop with 16 GB of memory and a 2.7 GHz Intel Core i5 CPU. We run DDoShield-IoT on a virtual machine with Ubuntu 22.04 LTS as the guest operating system and implement DDoShield-IoT with Docker version 20.10 and NS-3 simulator version 3.38.

Data Preparation Data preprocessing is a critical part of the intrusion detection process, which helps improve the effectiveness of predictive models. It includes the feature extraction procedure essential to reduce the number of input variables used in developing a predictive model. It also filters out irrelevant variables and selects only the most useful ones for accurate predictions [26]. This process is key to our study and is a critical step in analyzing network traffic for intrusion detection. As in the studies by Dong et al. [40], Ma et al. [85], and Javaid et al. [66], we do manual feature extraction, selecting only the relevant features, useful in identifying unusual network activity, such as botnet DDoS attacks. It is notable to underline that we execute the Feature Selection process only considering the scientific literature taking into account established studies in intrusion detection for IoT environments. In particular, Dong et al. [40]'s study highlighted the importance of features related to packet rate variation and TCP flags in detecting DDoS attacks, Ma et al. [85] showed that metrics like destination port entropy and the rate of broken connections are strongly indicative of malicious traffic, and Javaid et al. [66] pointed out that combining features based on traffic

patterns and statistical indicators improves the accuracy of IDSs. Therefore, our selection was guided by the convergence of these experimental results, choosing the features most frequently used in reference works.

In our case study, feature extraction first captures several attributes in network packets, such as timestamps, source and destination IP addresses, protocol types (e.g., TCP, UDP), and source and destination ports. These raw features provide insight into the basic characteristics of network communications between Devs and TServer.

To enhance our examination, we also incorporate statistical features to identify unpretentious patterns and anomalies in network traffic. A key component is the calculation of packet counts over designated time intervals, which offers insights into traffic volume and variations over time, enabling us to differentiate between typical traffic patterns that show steady packet counts and irregularities that may signify potential attacks, such as sudden increases or decreases in packet counts. Additionally, we analyze destination port entropy to evaluate the diversity and randomness of port utilization. Standard situations typically showcase various destination ports with a relatively balanced distribution, reflecting multiple communication activities. Attack scenarios, instead, may reveal skewed or concentrated distribution patterns, indicating targeted or malicious port activities. Such anomalies in regular port entropy values could signal possible scanning or probing activities linked to malicious actions. To bolster our intrusion detection capabilities, we execute a frequency analysis of specific port usage, which helps us uncover trends and anomalies in port communication patterns. For instance, recognizing short-lived connections and spotting repeated connection attempts are vital for identifying aggressive or suspicious actions suggestive of various attacks, including DDoS, as these differ from regular traffic behavior, which displays stable and predictable port usage patterns marked by consistent communication frequencies across multiple ports. The feature extraction process also entails network scanning activities through examination, concentrating on SYN flags that lack corresponding ACK flags, often preceding severe attacks. We attain a refined understanding of network dynamics by scrutinizing these and other statistical features like flow rates and variations in sequence numbers, effectively differentiating between benign and harmful activities. With a combination of raw and statistical features, the IDS within DDoShield-IoT gains a comprehensive representation of network behavior. This multi-dimensional feature extraction approach lays the groundwork for the following stages of the intrusion detection process, contributing to the IDS's accuracy and reliability in distinguishing between benign and malicious activities.

The Raw Features and Statistical Features discussed earlier are summarized in Table 5.1 and Table 5.2. Their combination involved a temporal aggregation

process, where data collected from network packets was processed over fixed 1-second time windows. The feature fusion process consisted of the following steps:

1. **Traffic Segmentation into Time Windows**
   Network traffic was divided into 1-second time windows, within which all received packets were collected. Then, we split the network traffic into 1-second time windows, within which we collected all received packets, treating each window as a single sample in the final dataset.

2. **Computation of Statistical Features for Each Window**
   At this stage, we computed all the statistical features in Table 5.2 based on the packets received in that specific time window.

3. **Alignment and Fusion with Raw Features**
   Each window w^t represents a feature vector including the statistical features calculated in the previous step, and the most representative basic features of the received packets (e.g., most frequent source and destination IPs, observed protocols, number of detected SYN flags). We chose the final value of each basic feature as the most frequent value in the time window, to represent the typical traffic behavior in that time.

4. **Structure of the Final Dataset** Each sample in the final dataset represents a 1-second time window described by a vector containing the basic and statistical features. So, the resulting structure of the dataset is a table with N rows (time windows) and M columns (combination of features).

This approach allows the integration of low-level granular information (individual packets) with aggregate metrics, improving the model's ability to detect malicious traffic patterns without losing critical details.

The choice of a 1-second time window comes from an experimental study assessing the impact of time window size on model performance. In particular, we conducted several experiments using four different configurations, i.e., 500 ms, 1 s, 2 s and 5 s. For each window, we evaluated the detection metrics (Precision, Recall, F1-score, and Accuracy), the computational efficiency, in terms of CPU and memory utilization, and the ability of the model to detect short-lived attacks. We generated a dataset using DDoShield-IoT, containing benign and malicious traffic (SYN Flood, UDP Flood, and ACK Flood attacks), splitting it into time windows of $500ms$, $1s$, $2s$, and $5s$, and repeating the process for each configuration. Then, we trained ML models on each dataset, evaluating the performance in the four temporal configurations. The analysis of the results shows that the 1-s time window represents the best trade-off between accuracy, ability to detect short attacks, and computational impact. A smaller window ($< 1s$) introduces

too much variability, while longer windows ($> 1s$) reduce the effectiveness of the model in detecting temporal patterns of attacks. Therefore, this configuration was chosen for the final implementation.

Table 5.1. Basic Features

Feature	**Description**
Timestamp	Time of packet observation.
IP Source Address	IP address of the source device.
IP Destination Address	IP address of the destination device.
Protocol	Protocol type (e.g., TCP, UDP).
Source Port	Port number of the source device.
Destination Port	Port number of the destination device.
TCP Flags	Includes ACK, SYN, FIN, PSH, URG, and RST flags.
Sequence Number	Sequence number of the packet in a connection.
Acknowledgment Number	Acknowledgment Number indicate received data.
Packet Size	Size of the packet in bytes.
Payload Size	Size of the payload in bytes.

Models To evaluate the efficiency of the IDS component in the IoT context emulated by DDoShield-IoT, we exploited three algorithms that we chose for their different ways of approaching intrusion detection, namely RF, K-Means, and CNN, which are described in detail below:

- **CNNs** are neural networks widely used in image processing. The key components of this model include i) filters or convolution kernels that we can define as small matrices that run over the input data to detect specific features through the convolution operation, which is a mathematical operation that allows the automatic extraction of features from the input data without the need for manual intervention, ii) convolution layers, which are sections of the network that apply filters to the input to extract interesting information, whose output is a *feature map*, then there are the iii) pooling layers, which serve to reduce the feature maps in spatial dimensions, keeping only the relevant information, reducing the computational complexity and avoiding overfitting, and finally, there is the iv) activation layer consisting of mathematical function (usually non-linear) applied at the end of each convolutional pass, such as the ReLU function, which allows the

Table 5.2. Statistical Features.

Feature	Description
Pkt Count	Total packets per window; spikes suggest DDoS.
Dst Port Entropy	Entropy of destination ports; high values suggest scanning.
Src Port Freq	Most common source port; detects anomalies.
Dst Port Freq	Most targeted destination port; key for port-specific attacks.
Short Conn	Rapid, short-lived connections; indicate DDoS.
Rep Conn Attempts	Duplicate destination counts; signs of attack.
Scan Activity	SYN-only packets; scanning precursor to attacks.
Flow Rate	Packets per second; essential for DDoS detection.
Src Entropy	Entropy of source addresses; high values suggest spoofing.
Conn Errors	RST flag counts; frequent resets signal disruption.
Pkt Size Freq	Most common packet size; uniformity suggests DDoS.
Abn Size Freq	Oversized packets; detects anomalous payloads.
Seq Num Var	Variance in sequence numbers; irregularities suggest spoofing.
Avg Pkt Num	Average packets per window; spikes suggest flooding.
SYN Freq	SYN flags per second; high rates indicate SYN floods.
ACK Freq	ACK flags per second; high rates indicate ACK floods.
TCP Freq	Proportion of TCP packets; detects TCP attacks.
UDP Freq	Proportion of UDP packets; identifies UDP floods.
Proto Freq	Most common protocol; shifts may signal attacks.
Pkt Size Var	Variance in packet sizes; abnormal values suggest attacks.
Payload Size Freq	Most common payload size; uniformity indicates automation.
Avg Payload	Average payload size; deviations suggest flooding.
Pkt Size Std	Standard deviation of packet sizes; variability signals anomalies.
Avg Pkt Size	Average packet size; deviations suggest attacks.

model to learn more complex relationships between features. These *structures* are combined in CNN to form a network that can learn complex and hierarchical representations of data [79]. In our case, we presented a 1D CNN architecture, whose components interact as follows: the input layer receives scaled and reshaped data as sequences with a single channel. A convolutional layer applies 64 filters with a kernel size of 3 to extract local features, followed by ReLU activation to capture complex patterns. The MaxPooling layers reduce the size of the feature maps with a window size of 2 useful for retaining valuable information and minimizing overfitting.

A Flatten layer transforms multidimensional data into a vector, then we use a Dense layer composed of 50 neurons and with ReLU as the activation to uncover complex feature relationships. In addition, the Dropout layer uses a 0.5 rate to prevent overfitting. Finally, the output layer contains one neuron with a Sigmoid activation function to generate binary classification probabilities. We trained the model using the Adam optimizer and binary cross-entropy loss, optimizing for metrics, like accuracy, precision, and recall. We trained it for 10 epochs with a batch size of 32, validating 20% of the training data.

- **K-Means** is a clustering algorithm that allows similar data to be grouped into clusters. According to Sinaga et al. [139], to solve the problem of selecting the optimal number of clusters, often present in traditional methods, it is possible to improve them by adding a penalty term based on entropy. The algorithm evolves in several steps. It initially assigns inputs to clusters randomly, then calculates the centroids of each cluster and reassigns the points to the cluster with the closest center, repeating the process until convergence. Besides the dynamic assignment of cluster optimization, this approach makes K-Means fast and efficient characteristics that make it particularly suitable for unsupervised clustering in contexts such as IoT, where adaptability and stability are fundamental.
- **RF**, finally, is an ensemble algorithm, a method that combines multiple base models (in this case, decision trees) to obtain a more accurate and robust overall prediction than a single model; in the case of RF, it uses a technique called bagging (bootstrap aggregation), which consists of creating multiple random subsets of data (with replacement) from the original dataset to train each decision tree separately, thus adding randomness and reducing the risk of overfitting; this diversity among trees improves the generalization ability of the model and, during inference, the trees collaborate by voting or averaging their outputs to provide a robust final prediction, even in the presence of noise or outliers. Parallel tree construction allows efficient training on large datasets [114].

To implement these models, we used Scikit-Learn [1] for RF and K-means, as it offers a wide range of ML algorithms and tools for model evaluation. For the CNN implementation, we instead used TensorFlow [2], an open-source library developed by Google that is ideal for numerical computation and deep learning. It has optimized CPU and GPU performance, making it an excellent choice for building and training advanced neural network models.

[1] **Scikit-learn website:** https://scikit-learn.org/stable/
[2] **Tensorflow website:** https://www.tensorflow.org/learn?hl=en

5.1.5 Evaluation Metrics

We evaluated the performance of the adopted DDoS attack detection models by leveraging the evaluation metrics described in 4.3, namely Precision, Recall, F1-score, and Accuracy. The computational resources available in IoT devices are highly variable across IoT devices, but in general, they all have limited processing power, memory, and power supply. This creates specific constraints for machine learning algorithms designed for IoT applications [103], and it is critical to optimize these models to ensure compatibility with IoT devices and to operate efficiently within these constraints [87]. Based on these observations, in this contribution, in addition to the accuracy of the models, we also evaluated their sustainability and compatibility with IoT devices, calculating the percentage of CPU usage (%), the RAM occupied in *Kilobytes* (Kb) and the model size in Kb. These metrics provide valuable information on the resource consumption of ML models on IoT devices [140].

We calculated these parameters because resource-aware ML models are necessary to reduce costs and achieve longer lifecycles for IoT devices. In fact, through this kind of ML model, we contribute to reducing the energy impact of IoT deployments, assessing the environmental and economic sustainability of ML applications in IoT contexts, and promoting the development of more efficient and eco-friendly solutions [163].

5.1.6 Performance Evaluation

We performed several experiments to evaluate the performance of the IDS component in the DDoShield-IoT environment, focusing on training the proposed models and real-time DDoS attack detection. We started by running the IoT environment simulation through DDoShield-IoT for 10 minutes, to obtain a dataset that included benign and malicious traffic. We then used the obtained dataset to train the different RF, CNN, and K-Means models offline. This phase is essential for the subsequent real-time detection activities. In particular, in this phase, the TServer generates benign traffic as described in 5.1.3. We also generate malicious traffic using the Mirai malware, which can perform several botnet [71] DDoS attacks. Specifically, the attacks analyzed include SYN Flood, which overwhelms servers by flooding them with SYN requests and draining resources, ACK Flood, which floods systems with numerous ACK packets to disrupt normal network operations, and UDP Flood, which floods servers with a high volume of UDP packets, causing network congestion and potential service disruption. We selected these specific attacks to demonstrate the basic functionality of the IDS within the DDoShield-IoT framework while avoiding more complex application-layer

attacks such as HTTP Flood or DNS Flood, which require additional analysis at the application layer. The resulting dataset is nearly balanced, comprising 3,012,885 malicious packets and 2,243,634 benign packets.

We then trained the selected models on the resulting dataset and evaluated their performance. Specifically, we measured Accuracy, Precision, Recall, and F1-score, as these metrics offer a clearer understanding of model performance, especially in scenarios with imbalanced datasets. Table 5.3 presents the evaluation results for each ML model after the training phase.

Table 5.3. ML Models Performance Evaluation on the Training Dataset.

Model	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F1-score** (%)
RF	96.3	95.7	94.2	94.9
K-Means	92.1	89.4	91.1	90.2
CNN	**97.8**	**97.3**	**96.9**	**97.1**

As shown in Table 5.3, all models achieved high accuracy scores, with CNN obtaining the highest performance across all metrics. The F1-score values confirm a balanced trade-off between Precision and Recall, ensuring that the model does not favor one class over the other.

After the training phase, we saved each model in a PKL file[3] to achieve persistence of the learned parameters, allowing them to be used later without retraining. We used these saved models for real-time detection.

After the first training phase, we ran the DDoShield-IoT for 5 minutes, during which we performed the real-time intrusion detection process, extracting statistical features from the packets collected in a 1-second time window and evaluating the accuracy of the models for that specific period. It is important to note that users can customize the time window's length according to their analysis needs.

Once the real-time detection simulation was also finished, we calculated the average accuracy for each ML model. Unlike the training phase, where we consider multiple evaluation metrics, for real-time detection performance we only calculate accuracy. We chose this approach due to the specific characteristics of the simulation environment, where, although benign and malicious traffic are

[3]A **PKL file**, or Pickle, is a serialized file format used in Python to store objects, including data structures, models, or any Python object.

Table 5.4. ML Models Performance Evaluation in Real-Time Detection.

Model	Accuracy (%)
RF	61.22
K-Means	94.82
CNN	**95.47**

generated simultaneously, there are cases during the real-time detection process where only one type of traffic is detected. For example, there may be intervals where only malicious traffic is identified, followed by periods with benign traffic only. As a result, calculating precision and recall metrics during these isolated periods may result in division by zero, making these metrics unreliable. Therefore, we decided to focus exclusively on accuracy to ensure consistent and meaningful evaluation, even in scenarios with only one type of traffic.

Table 5.4 shows the average accuracy for each ML model at the end of the simulation time.

Analyzing the accuracy score related to each second during the simulation, we notice that the first and last seconds of an attack duration report a drop in the model accuracy. The minimum registered is 35% for the K-Means model. This is due to statistical features that make noise since they are equal for each packet collected in that specific second. This happens because we execute manual feature extraction without evaluating the actual usefulness of each feature after basic and statistical feature aggregation. This will be part of future work.

5.1.7 Sustainability Evaluation

Table 5.5, shows CPU usage (%), the occupied RAM (Kb), and the model size (Kb) evaluated during the real-time detection process.

The sustainability assessment reveals increased CPU usage, mainly due to the IDS calculating statistical features every second. A strategic method to reduce this high CPU usage involves changing the frequency at which these statistical features are calculated. So, it is possible to achieve lower CPU utilization by lengthening the interval for feature computation.

Additionally, observing the allocated RAM and considering the model size indicates that the deep learning model, specifically the CNN, is expected to be the largest. In comparison, the k-Means model is the lightest in resource usage.

Table 5.5. ML Models Sustainability.

Model	CPU (%)	Memory (Kb)	Model Size (Kb)
RF	**65.46**	98.07	712.30
K-Means	67.88	**86.83**	**11.20**
CNN	65.94	275.85	736.30

5.2 Enhancing IDS Training with Data Augmentation in IoT Contexts

The capabilities demonstrated by DDoShield-IoT have allowed us to develop a Data Augmentation strategy to address the problem of the lack of representative data for training ML models. Through this approach, we aim to enrich the training data by generating synthetic traffic patterns that represent a broader spectrum of potential attack scenarios, because by diversifying the data, IDSs can be trained to recognize a wider range of attacks, including emerging or never-before-seen ones. The next section will explain more about the Data Augmentation technique and how to use it with the framework to make IDS models better at detecting threats in IoT contexts.

5.3 Data Augmentation

Data Augmentation is a technique that expands the amount and variety of available data and is particularly useful for training more robust ML models. It consists of modifying existing data to generate new data without collecting often unexisting additional information. The usefulness of this approach is especially seen in contexts where data is limited or difficult to obtain, such as image recognition, natural language processing Natural Language Processing (NLP), and emerging sectors such as cybersecurity and IoT. For example, to expand and diversify image datasets, transformations such as rotations, scale changes, or color changes can be applied to simulate situations different from those represented by the original data, helping models to recognize the same objects in variable contexts. For texts, instead, some techniques replace words with synonyms or translate them into other languages, thus creating new versions of the content. Finally, in the case of tabular data, the most common techniques to modify or

enrich the data include varying the numerical values or generating new synthetic examples to balance the less-represented classes. Data Augmentation has several advantages, the most important of which is the capability to reduce the risk of overfitting. When the training data are limited the model risks adapting too much to them, losing generalization capabilities; instead, by varying the data, the model learns to identify more general characteristics that can be applied to new contexts. Furthermore, this technique is particularly effective for addressing the imbalance between classes, for example in datasets where some categories of events or attacks are less represented than others. However, Data Augmentation can be detrimental if the data transformations are too extreme or unsuitable for the specific domain because there is a risk of introducing errors or distorting the data's original meaning. Furthermore, some advanced techniques may require significant computational resources, which are not always available, especially in environments like IoT[86].

Based on this, we propose a new Data Augmentation strategy by combining simulated traffic from DDoShield-IoT with real-world traffic from the TON_IoT dataset, a dataset of network traffic from IoT devices, used for attack detection in IoT contexts, to evaluate the effectiveness of this approach in training robust intrusion detection models.

5.4 Dataset Selection

To support the Data Augmentation strategy proposed in the IoT context, we conducted a literature review to select the most suitable dataset considering the following specific requirements:

- *Representativeness of real IoT contexts*: the dataset must include realistic traffic generated by heterogeneous IoT devices.
- *Balance between benign and malicious traffic*: this balance is essential to ensure effective model training.
- *Availability of labeled data*: knowing the nature of data is essential for accurate performance comparison of sensor models.
- *Ease of integration with data enrichment strategies*: for the proposed methodology, this characteristic is essential to enrich data and improve model robustness.

Considering these requirements, we selected and analyzed the datasets commonly used in the literature, summarized in Table 5.6.

Dataset	IoT Scenarios	Labeled Data	Attack Variety	Class Balance	Recency
KDDCup99	No	Yes	Limited	No	Low
NSL-KDD	No	Yes	Limited	Partial	Medium
UNSW-NB15	No	Yes	High	Partial	High
CICIDS2017	Partial	Yes	High	Yes	High
TON_IoT	Yes	Yes	High	Yes	Very High

Table 5.6. Comparison of Popular Datasets for Intrusion Detection.

In detail, we considered:

- **KDDCup99** [142]: A widely used historical intrusion detection dataset. However, it has significant limitations, such as an outdated representation of attacks and a lack of real-world IoT scenarios.
- **NSL-KDD** [37]: An update of KDDCup99 that addresses some limitations but still does not reflect modern IoT contexts.
- **UNSW-NB15** [98]: Includes newer network traffic with a mix of modern attacks but is not specifically designed for IoT devices.
- **CICIDS2017** [113]: Provides a good variety of modern attacks and network traffic but does not include specific data from IoT devices.
- **TON_IoT** [97]: An advanced dataset specifically designed for IoT scenarios. It includes telemetry, network traffic, and operating system information, making it particularly suitable for our research context.

In light of the conducted analysis, we consider the TON_IoT dataset to be the definitive choice for strengthening intrusion detection strategies in the IoT context. Indeed, its in-depth representation of realistic scenarios, along with a wealth of labeled data and a modern design, makes it exceptionally effective for evaluating the proposed solutions.

The TON_IoT Dataset A literature review showed that the TON_IoT dataset is one of the main sources for training and evaluating intrusion detection models in IoT contexts. This dataset, developed by the University of New South Wales (UNSW) in Canberra, represents realistic IoT network scenarios. It was generated using a complex testbed that emulates communications between edge, fog, and cloud devices, reflecting the typical interactions of modern IoT systems.

This dataset is a popular reference point because it includes different types of data, like telemetry data from IoT devices, information from operating systems (Windows and Linux), and network traffic. This makes it particularly useful

in cybersecurity unlike other outdated and less varied datasets, such as those mentioned above. Even the name TON_IoT shows the variety of data sources included. It comes from the acronym Telemetry, Operating Systems, and Network Internet of Things, which precisely describes it [97].

In Figure 5.3 we can observe the network architecture simulated by this dataset. It is distributed on three main levels, i.e., edge, fog, and cloud. It is realized using advanced technologies such as Software-Defined Networking (SDN), Network Function Virtualization (NFV), and Service Orchestration (SO). In particular, the Edge Level includes physical IoT and IIoT devices that collect sensor data and communicate with the fog and cloud levels. Examples of devices include sensors for environmental monitoring, smart TVs, and smartphones, managed by local gateways that send data to the fog level. At the Fog Level, network function virtualization is implemented through virtual machines, which perform preliminary data analysis to reduce latency and cloud dependence, while the Cloud Level provides remote monitoring services and advanced analysis of network traffic and IoT data. In particular, the cloud deals with the processing of large volumes of data coming from edge and fog, using platforms such as Microsoft Azure or AWS.

5.5 Data Augmentation Workflow

The main goal of the proposed strategy, described in detail in Figure 5.4, is to evaluate the impact of the use of the Data Augmentation technique in the training phase on the performance of an IDS in detecting anomalies and attacks in IoT contexts.

The workflow is divided into the different phases described below.

Data Collection The first phase of the strategy involves simulated and real data collection. In particular, the simulated data were acquired through the *tcpdump* tool, part of the Wireshark suite supported by
DDoShield-IoT. Using this tool, we collected benign and malicious network traffic, necessary for training and evaluating intrusion detection models. The real data, instead, were taken from the pcap files from the TON_IoT dataset.

Definition and creation of different scenarios Once we had all the data available, we divided and combined them in diverse proportions to represent a variety of real IoT network situations and contexts, including different types of

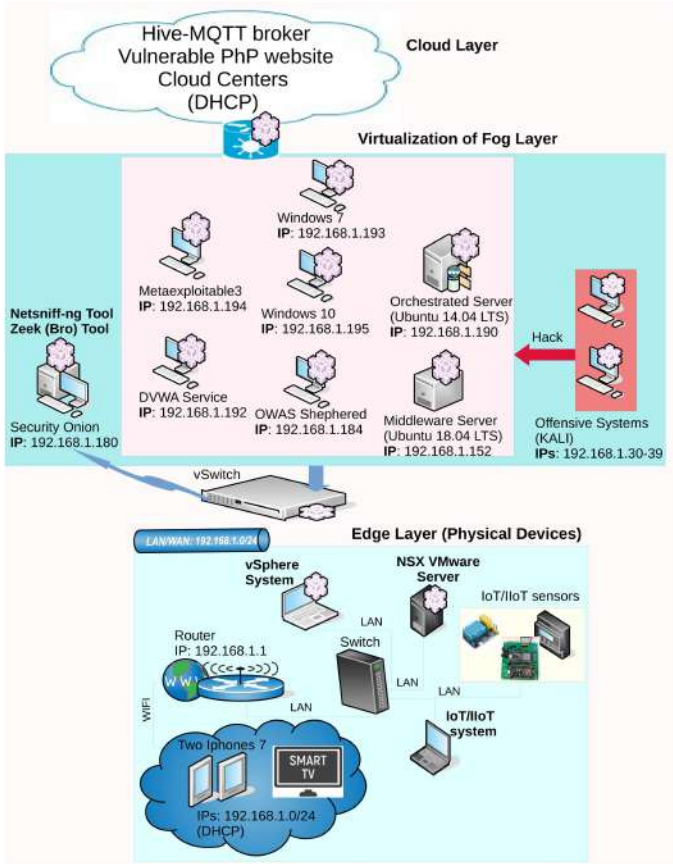

Figure 5.3. TON_IoT Dataset Architecture [97]

network traffic and different DDoS attack intensities. The scenarios thus obtained allow us to evaluate the performance of the IDS in heterogeneous environments.

Extraction of features of interest In each scenario's dataset, we performed a feature extraction process, obtaining those useful for detecting potential DDoS attacks. In particular, These features include traffic-related parameters, such as packet size, transmission time, and the number of packets sent in a defined time window.

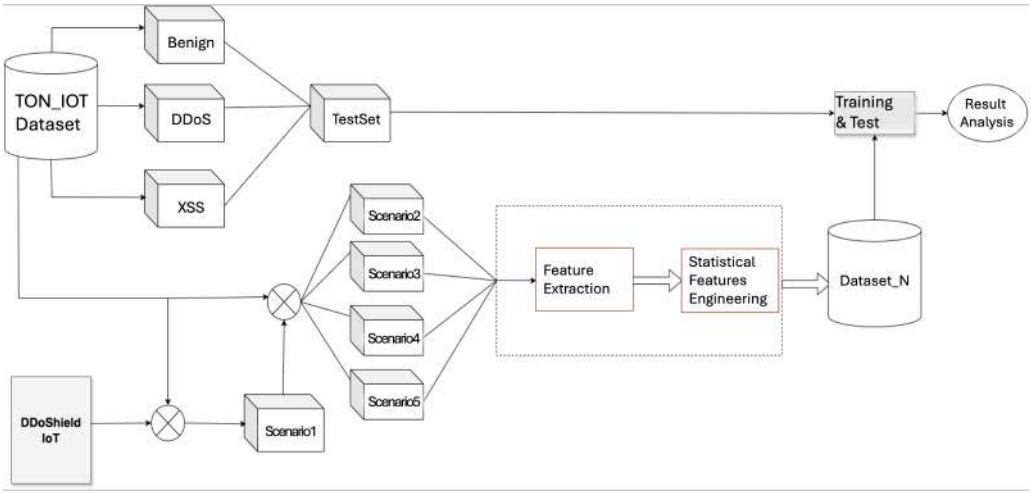

Figure 5.4. Proposed Data Augmentation Strategy Workflow

Feature engineering process At this workflow's point, we applied a feature engineering process to improve the data quality and optimize the features used for training. We chose the most relevant features using correlation analysis and preliminary models, avoiding redundant or useless ones. In particular, we used Pearson's correlation coefficient to measure the linear relationship between numerical features. This allowed us to identify and remove highly correlated features (e.g., with a coefficient greater than 0.8) to avoid multicollinearity and improve model interpretability. In addition, we applied Principal Component Analysis (PCA) to reduce dimensionality and improve generalization, which created new features as linear combinations of the originals, reducing complexity and overfitting. This process made the sensing systems in IoT environments more effective and robust. This step is crucial to increase the model's effectiveness and reduce overfitting, thus enhancing its ability to generalize new data.

Test Set Creation To allow the evaluation of the proposed strategy, we created a separate test set, composed of data not used in the training phase to evaluate the performance of the detection models.

Analysis and discussion of the results Once the models were trained and tested, we performed an in-depth results analysis. This phase includes the evaluation of the performance metrics (i.e., accuracy, precision, recall, and F1-

score), and the discussion on the impact of Data Augmentation on the IDS performance, highlighting the advantages and limitations of the adopted technique.

The described workflow ensures a rigorous and reproducible methodology to evaluate the effectiveness of Data Augmentation in improving the performance of IDS in IoT environments. Each step has been performed to maximize the model generalization and obtain a detailed picture of its capabilities in complex and variable scenarios.

Traditional Data Augmentation techniques, such as applying transformations to existing samples (e.g., noise injection, scaling, or feature perturbation), have been widely used in fields like computer vision and natural language processing to improve model robustness [136, 45]. However, in network security and intrusion detection context, these conventional methods have limitations due to the highly structured nature of network traffic, where arbitrary transformations can disrupt meaningful patterns and degrade performance [124].

To address this challenge, recent studies have explored alternative data augmentation strategies tailored for cybersecurity applications. One effective approach is the combination of real and synthetic traffic, which has been shown to improve model generalization by exposing it to a more diverse range of attack variations [134, 78]. Simulated traffic, generated through tools such as *tcpdump* and network emulation frameworks, allows for controlled experimentation with different attack intensities and variations, complementing the variability found in real-world datasets [51].

The proposed augmentation strategy aligns with this emerging trend by combining real IoT traffic (TON_IoT dataset) with synthetic traffic generated via simulation. This methodology ensures that the IDS is trained on a dataset that captures real-world attack patterns and includes also controlled variations that might not be fully represented in the original dataset. Such an approach has been recommended in multiple studies as a way to enhance the robustness of IDSs, particularly in IoT environments where real attack data can be scarce or biased [99, 82].

By adopting this hybrid augmentation strategy, we aim to create a more balanced and comprehensive dataset, improving the IDS's ability to detect anomalies across different deployment conditions while mitigating overfitting to a specific traffic distribution. Future research could further refine this approach by systematically evaluating the impact of different ratios of real and synthetic data, as suggested in [27].

5.6 Reproduced Scenarios

We created five scenarios for the experiments, each represented by a specific dataset. We organized the five created datasets in distinct configurations, each characterized by different combinations of traffic simulated via DDoShield-IoT and real traffic taken from TON_IoT, both benign and malicious.

We selected these five scenarios to evaluate IDS performance under different conditions, that closely resemble real-world IoT network environments. Specifically, we designed these scenarios to analyze the impact of varying proportions of synthetic and real network traffic on intrusion detection. Particularly, **Scenario 1** serves as a baseline, primarily composed of simulated traffic, allowing an initial assessment of model performance on synthetic data, **Scenario 2** introduces real malicious traffic to examine how models handle mixed data distributions, **Scenario 3** increases the proportion of simulated attacks, testing the generalization capability of IDSs when trained with synthetic data, **Scenario 4** does the opposite, emphasizing real attack data to evaluate IDS robustness against real-world threats, and **Scenario 5** further complicates the classification task by introducing a second type of attack (XSS), testing the ability of IDSs to distinguish between multiple attack types. We carefully chose these scenarios to reflect different levels of realism and complexity, ensuring a comprehensive evaluation of the impact of data augmentation on model performance. The five dataset compositions are described in detail below.

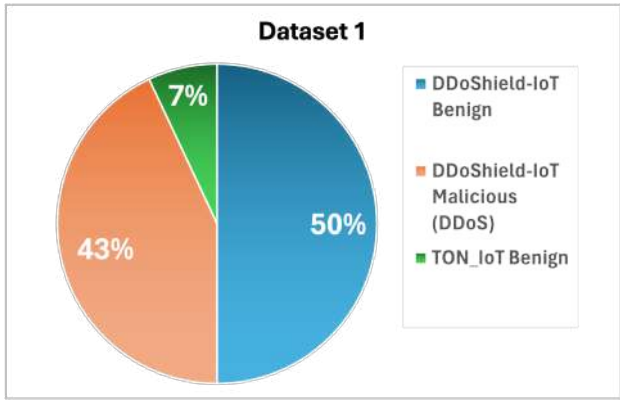

Figure 5.5. Dataset 1 Composition

Dataset 1 The first dataset (Figure 5.5) includes 93% data simulated with DDoShield-IoT, of which 50% are benign and 43% represent DDoS attacks. The remaining 7% includes benign data from the TON_IoT dataset. Comprehensively, this dataset contains about 1,670,000 packets, with a distribution of about 57.34% of benign packets and 42.66% of malicious packets belonging to DDoS attacks.

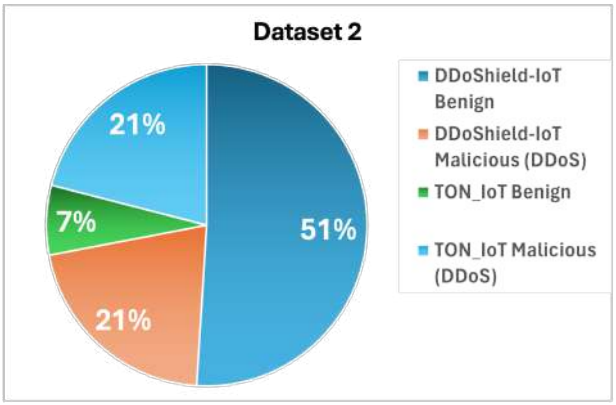

Figure 5.6. Dataset 2 Composition

Dataset 2 This dataset was created keeping the traffic unchanged benign of *Dataset 1* and splitting the malicious traffic so that 50% malicious packets are generated by the simulator, while 50% are from the real dataset. This dataset, represented in Figure 5.6, is particularly useful for testing the ability of IDS to detect DDoS attacks distributed across different sources.

Dataset 3 The third dataset follows a similar logic to Dataset 2, but with a different distribution of malicious traffic. 70% of malicious traffic is synthetic, while the remaining 30% is real. This configuration increases the emphasis on simulated attacks over real ones, creating a more complex scenario to test the generalization ability of IDSs in the face of different attack sources. The detailed traffic composition is shown in Figure reffig:Da3

Dataset 4 The dataset in Figure 5.8, corresponding to the fourth scenario, inverts the proportion of malicious traffic compared to *Dataset 3*, including 30% of traffic simulating DDoS attacks generated by DDoShield-IoT, and 70% of real DDoS attacks from the TON_IoT dataset. This configuration is particu-

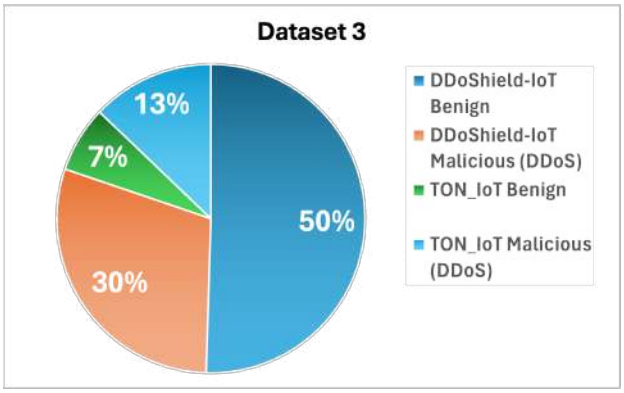

Figure 5.7. Dataset 3 Composition

larly significant for testing the robustness of IDSs against real attacks with low interference from simulated data. The prevalence of real DDoS traffic makes this dataset useful for analyzing the system's ability to detect realistic threats in IoT environments.

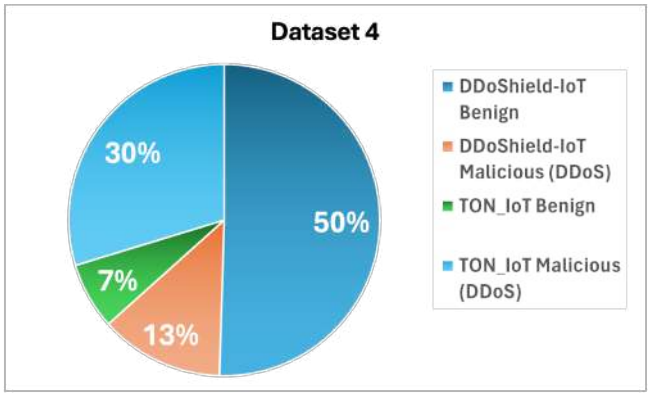

Figure 5.8. Dataset 4 Composition

Dataset 5 Finally, for the last scenario, we introduce more complexity by creating a dataset that includes real DDoS and Cross-Site Scripting (XSS) attacks included in the TON_IoT dataset. In this configuration, the malicious traffic is split equally between 50% packets generated by the simulator and 50% real, further split into 25% DDoS and 25% XSS. This dataset (Figure 5.9) represents

a complex multi-attack scenario, which tests the ability of IDS to simultaneously manage multiple types of threats, analyzing their ability to distinguish between attacks of different natures, particularly interesting.

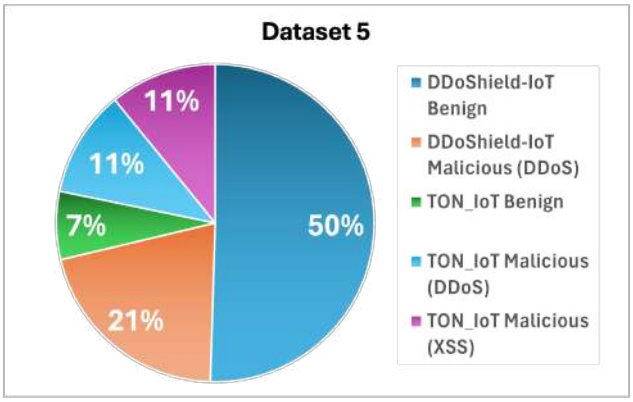

Figure 5.9. Dataset 5 Composition

The choice of diverse and specific percentages in the different scenarios aims to reproduce different conditions and therefore favorable and unfavorable scenarios to isolate the simulator impact and find the right compromise for training ML models for intrusion detection.

We generate a test set using 20% of the dataset to test the five scenarios created above. We did this by randomly choosing packets from the TON_IoT dataset that were not included in the training set. This 20% is a good sample size to test the model while having enough data for training. This ratio, often used in ML, lets the model see different patterns during training. It also keeps a separate dataset to evaluate how well the model works.

The random data sampling keeps the test set balanced between good traffic and bad traffic. This helps reduce bias and gives a more reliable idea of the model's performance. Moreover, it is useful to accurately predict how well the model will perform with new data not seen before, which is important for IoT network protection [61].

5.7 Results Analysis and Discussion

In this section, we analyze the performance of the ML models: K-Means, RF, and CNN in detecting intrusions in the five scenarios created, through the eval-

uation metrics Accuracy, Precision, Recall, and F1-Score. For the RF model, we made a further adjustment, evaluating its performance, by varying the threshold value. This model, indeed, uses a threshold value to determine the final class of an observation, based on the average probability obtained from the various trees. By default, the threshold is 0.5: if the probability of a class is greater than or equal to 0.5, the observation is classified in that class, otherwise in the other. The decision to set the threshold at 0.5 follows a standard convention in binary classification tasks, assuming a balanced dataset and equal misclassification costs for false positives and false negatives. However, in practical applications, the optimal threshold is usually chosen based on the specific objectives of the IDS. For instance, in cybersecurity contexts, where missing an attack can be more critical than raising a false alarm, the threshold can be lowered to favor recall, ensuring a higher detection rate of malicious traffic. The threshold can be modified to optimize the model, for example, to reduce false positives or false negatives, but it is generally applied to establish the final class based on the overall probability. In this case, we modify the threshold value to understand how much this can affect the amount of false positives or false negatives in the detection. The results were analyzed for each dataset and model and summarized in tables, describing how the models performed in different scenarios.

Table 5.7 shows that in the first scenario, K-Means performed well with an Accuracy of 0.84, a good value of Recall, i.e., 0.98, but a relatively low precision of 0.76. This indicates that it detected malicious traffic well, having some false positives. In contrast, CNN despite a very high precision, i.e., 0.99, had difficulties correctly detecting malicious traffic, with a low F1-Score equal to 0.50. Finally, the RF model performed well, but only when the threshold was lower than 0.5. Table 5.7.

An important result is that the K-means model demonstrates superior performance compared to RF on labeled training sets, as illustrated in Table 5.7. Specifically, K-Means performs better than RF on labeled datasets because it finds patterns in the data without relying only on the labels. This could be seen along several lines. First, RF is sensitive to the quality of the labels. If the labels are noisy or biased, RF can have poor predictions, leading to a greater number of false positives or negatives. On the other hand, K-Means considers the similarity between data points, which makes it more robust even when labels are not perfect. Second, RF will struggle to find boundaries in such data if classes are not well defined or features are overlapping. K-Means groups the data by natural distribution demonstrating to be better at picking up such intricacies, which increases recall but drops some accuracy. Thirdly, K-Means is lightweight in structure and thus faster in training and inference, especially with numerous features, like in IoT situations with small computing powers. Last but not least,

K-Means will perform well on balanced or near-balanced datasets, showing strong accuracy and recall. Conversely, RF tends to favor the majority classes and does not detect less well-represented classes.

Model	Accuracy %	Precision %	Recall %	F1-Score %
K-Means	84.09	76.54	98.95	86.31
(RF-0.25)	70.74	100	42.17	59.32
CNN	66.34	99.66	33.42	50.06

Table 5.7. Detection Performance on Dataset 1

Moreover, in the case of the RF model, we performed an analysis by varying the threshold to understand its impact on detection performance. The results (see Table 5.7) show that reducing the threshold below 0.5 increases recall but at the cost of lower precision. This highlights the importance of adjusting the threshold based on the specific use case and operational requirements of the IDS.

The performance of the models reported in Table 5.8, showed a general improvement in the second scenario. Here, K-Means achieved an accuracy of 0.86 and an F1-Score of 0.88. RF obtained excellent results with an accuracy of 0.99, while CNN showed performances similar to those of RF, reaching an accuracy value of 0.99.

Model	Accuracy %	Precision %	Recall %	F1-Score %
K-Means	86.53	79.60	98.75	88.14
RF-0.25	97.40	95.11	100	97.49
RF	99.95	99.90	100	99.95
CNN	99.98	100	99.97	99.98

Table 5.8. Detection Performance on Dataset 2

Table 5.9 related to the third dataset, shows how, with a predominance of malicious traffic generated by DDoShield-IoT, K-Means maintained performances similar to those obtained in Scenario 2, while CNN showed a significant drop, with an accuracy of 0.67 and an F1-Score of 0.53, suggesting difficulties in managing unbalanced datasets.

In the fourth scenario, malicious traffic mainly came from the real dataset, TON_IOT. Here, as Table 5.10 shows, K-Means performance slightly dropped, while RF maintained excellent performance, reaching an accuracy close to 1.0.

Model	Accuracy %	Precision %	Recall %	F1-Score %
K-Means	86.62	80.71	96.74	88.00
RF-0.25	91.18	1.0	42.17	95.38
RF	100	100	100	100
CNN	67.90	100	36.41	53.38

Table 5.9. Detection Performance on Dataset 3

Also, CNN achieving an accuracy value of 0.99, showed to be able to well handle unbalanced traffic, outperforming the performance obtained in Scenario 3.

Model	Accuracy %	Precision %	Recall %	F1-Score %
K-Means	84.07	78.26	94.97	85.81
RF-0.25	93.30	88.28	100	93.78
RF	99.91	99.83	100	99.91
CNN	99.60	100	99.22	99.60

Table 5.10. Detection Performance on Dataset 4

Finally, in the fifth Scenario, we further complicated the classification by introducing XSS traffic. However, K-Means maintained good performance (accuracy 0.86), while RF and CNN reached an accuracy of 0.99 and 0.96 respectively, demonstrating their ability to handle more complex datasets with mixed traffic (Table 5.11).

Model	Accuracy %	Precision %	Recall %	F1-Score %
K-Means	86.31	86.42	94.48	90.27
RF-0.25	76.48	74.04	100	85.08
RF	99.99	99.99	100	99.99
CNN	96.96	99.96	95.50	97.68

Table 5.11. Detection Performance on Dataset 5

The performance variations among models across the five scenarios reflect differences in the distribution of malicious and benign traffic, the overlap of patterns, and the complexity of the attack types. In scenarios where detection performance was lower, a corresponding increase in both false positives and false negatives was often observed.

For example, the unsupervised clustering method K-Means relies on intrinsic groups within the data, making it less effective in scenarios where attack patterns are not easily distinguishable from regular traffic and lead to a high number of false positives. On the other hand, RF, a decision-tree-based model, depends on distinct attack signatures and is not effective in situations with high traffic variability and evolving attack patterns. CNNs perform well on representative datasets because they learn hierarchical features in the traffic. However, they are susceptible to skewed data distributions, which can compromise recall in specific setups. Furthermore, the presence of XSS traffic in the fifth scenario added complexity to classification since there was an additional layer of variability in network usage. This required the models to differentiate between multiple types of anomalies, which could hinder their ability to maintain high precision.

These findings underscore the importance of balancing model characteristics and data distribution when evaluating performance in IDS. Hybrid models that combine rule-based and machine-learning approaches in dynamic contexts may help mitigate performance degradation by leveraging the strengths of different methodologies.

5.8 Model Analysis

K-Means Being an unsupervised clustering algorithm, K-Means showed good performance in scenarios with well-distinct classes, such as in Datasets 1 and 2, but struggled in complex or unbalanced scenarios, such as in Datasets 3 and 4, where it showed a reduction in recall and an increase in false negatives.

RF This model performed excellently on almost all datasets, with an accuracy close to 100%, thanks to its ability to handle complex and unbalanced data, capturing non-linear interactions. It showed the best behavior on the most unbalanced and complex datasets.

CNN This model performance was variable. It excelled on the most complex datasets (such as Dataset 5) but struggled with unbalanced datasets, such as in Dataset 3, where it showed low recall despite high precision. This indicates a propensity of the CNN to miss more subtle intrusions, leading to false negatives.

5.9 Impact of the Strategy

To evaluate the effectiveness of our approach contribution, this section analyzes model performance without utilizing the DDoShield-IoT simulator. The goal is to recreate scenarios where the simulator is unavailable, allowing us to measure its actual added value in real-world conditions.

The comparison of results with and without the simulator is crucial in highlighting the significance of this tool, particularly when working with highly imbalanced datasets. In these situations, we will observe how the absence of synthetic data generated by the simulator results in datasets that are challenging to use for model training, ultimately diminishing their effectiveness in anomaly detection.

The metrics in the tables that follow will be presented in this format:

$$evaluation\_metirc\_DATASET\_TON\_Only_i[metrica\_DATASET_i]$$

where $DATASET_i$, with $i = 1, 2, 3, 4, 5$, represents the previously defined scenarios. This notation enables a direct comparison of the performance obtained using only the real-world dataset (TON_IoT) against the augmented datasets enriched with synthetic data generated by the simulator.

5.9.1 Dataset 2 - Only Real-World Data

Model	Accuracy %	Precision %	Recall %	F1-Score %
K-Means	98.47 [86.53]	97.49 [79.60]	99.57 [98.75]	98.52 [88.14]
RF-0.25	49.34 [97.40]	100 [95.11]	0.02 [100]	0.05 [97.49]
RF-0.5	49.34 [99.95]	0 [99.90]	0 [100]	0 [99.95]
CNN	93.16 [99.98]	99.68 [100]	86.77 [99.97]	92.78 [99.98]

Table 5.12. Performance Evaluation Metrics for Detection on Dataset 2

In Dataset 2, the simulator seems to positively impact the models except for K-Means, where the improvement is minimal. Notably, the RF model without the simulator is inaccurate, achieving virtually no recall. However, when the simulator is utilized, the model reaches optimal performance across all metrics. A similar trend is observed with the CNN model, where the simulator significantly enhances accuracy, precision, and recall.

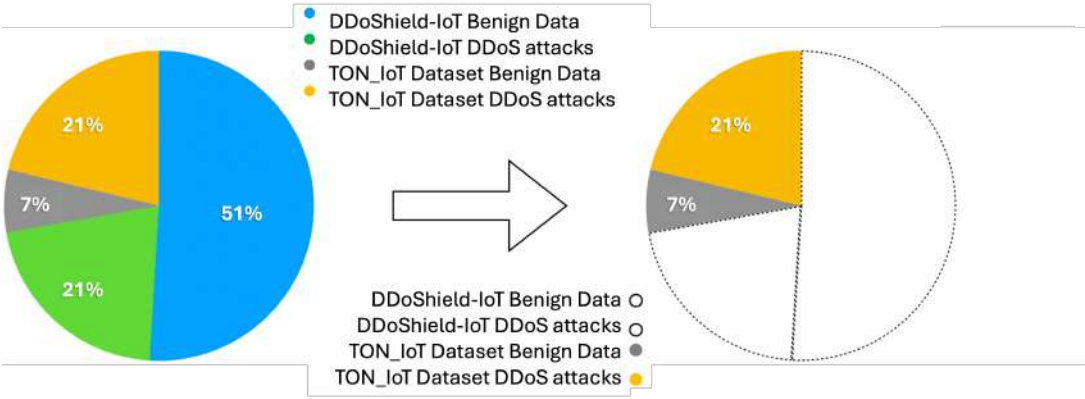

Figure 5.10. Dataset 2 - 21% of ToN_IoT Malicious Data and 7% of ToN_IoT Benign Data

5.9.2 Dataset 3 - Only Real-World Data

Model	Accuracy %	Precision %	Recall %	F1-Score %
K-Means	97.40 [86.62]	95.15 [80.71]	100 [96.74]	97.51 [88.00]
RF-0.25	57.72 [95.11]	54.51 [91.18]	100 [100]	70.56 [95.38]
RF-0.5	86.44 [100]	78.12 [100]	100 [100]	88.29 [100]
CNN	98.89 [67.90]	97.87 [100]	100 [36.41]	98.92 [53.38]

Table 5.13. Performance Evaluation Metrics for Detection on Dataset 3

In this third version of Dataset, the simulator notably enhances the performance of the RF algorithm. Conversely, the results obtained for the k-means algorithm demonstrate a higher performance when conducted without the simulator. The CNN exhibits a noteworthy behavior, indeed, while maintaining a high level of accuracy, the recall, and consequently, the F1-score experiences a substantial decline in the absence of the simulator. It is imperative to acknowledge that this dataset exhibits a pronounced skew toward simulator data.

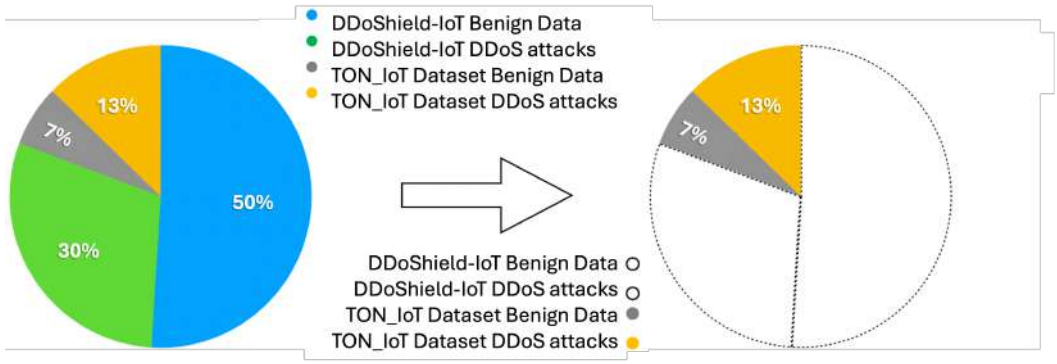

Figure 5.11. Dataset 3 - 13% of ToN_IoT Malicious Data and 7% of ToN_IoT Benign Data

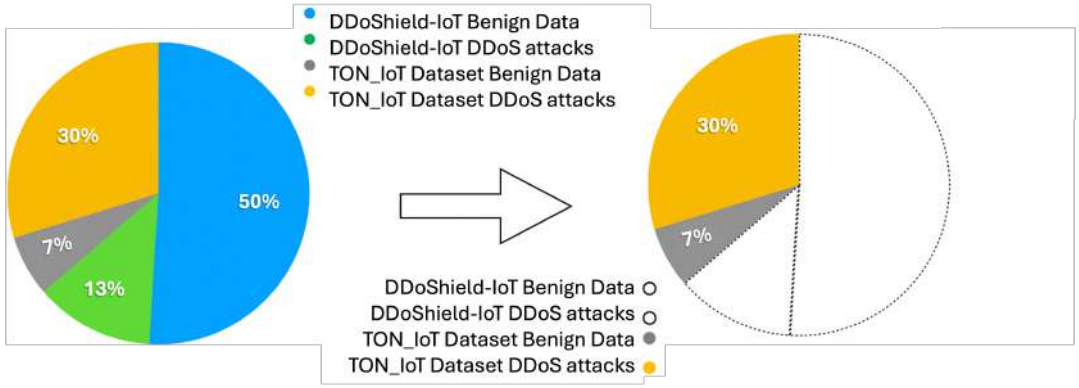

Figure 5.12. Dataset 4 - 30% of ToN_IoT Malicious Data and 7% of ToN_IoT Benign Data

Model	Accuracy %	Precision %	Recall %	F1-Score %
K-Means	97.03 [84.07]	94.50 [78.26]	100 [94.97]	97.17 [85.81]
RF-0.25	57.18 [93.30]	54.14 [88.28]	100 [100]	70.00 [93.78]
RF-0.5	86.21 [99.91]	78.46 [99.83]	100 [100]	88.46 [99.91]
CNN	98.82 [99.60]	97.72 [100]	100 [99.22]	98.85 [99.60]

Table 5.14. Performance Evaluation Metrics for Detection on Dataset 4

5.9.3 Dataset 4 - Only Real-World Data

We can observe a general improvement in performance for Dataset 4 when the simulator is employed, particularly for RF and CNN models. In contrast, K-Means demonstrates stability even in the absence of the simulator. These findings suggest that the simulator facilitates the management of more complex models in datasets characterized by heterogeneous distributions, which are notoriously challenging to predict.

5.9.4 Dataset 5 - Only Real-World Data

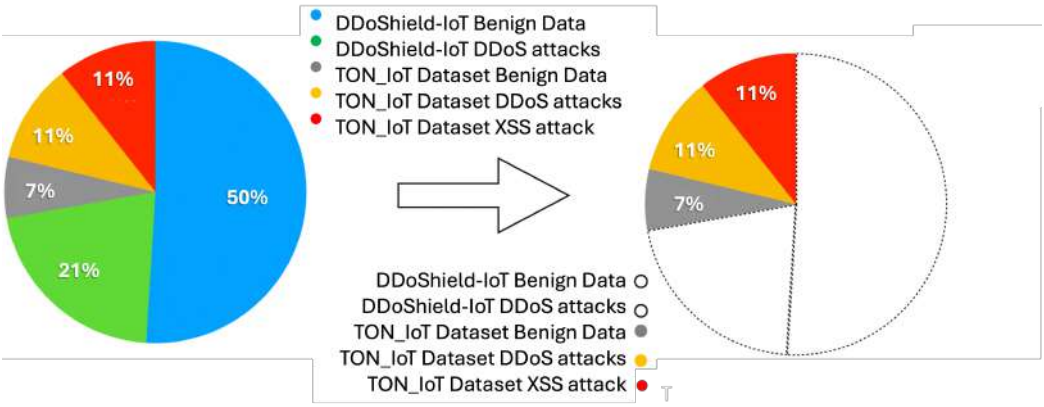

Figure 5.13. Dataset 5 - ToN_IoT Malicious Data: 11% of DDoS and 11% of XSS, 7% of ToN_IoT Benign Data

Model	Accuracy %	Precision %	Recall %	F1-Score %
K-Means	97.85 [86.31]	96.03 [86.42]	99.92 [94.48]	97.94 [90.27]
RF-0.25	52.28 [76.48]	51.50 [74.04]	100 [100]	67.99 [85.08]
RF-0.5	99.90 [99.99]	99.92 [99.99]	100 [100]	99.91 [99.99]
CNN	98.50 [96.96]	98.25 [99.96]	99.85 [95.50]	98.92 [97.68]

Table 5.15. Performance Evaluation Metrics for Detection on Dataset 5

It seems that the simulator in Dataset 5 leads to significant improvements for all models, except for K-Means, which shows higher performance without the

simulator. This behavior could be due to K-Means' ability to work better with raw and less structured data.

5.10 Final Observations

The proposed Data Augmentation strategy is a solid starting point for optimizing attack detection systems based on simulated data. The constant evolution of the threat landscape and technologies requires continuous updates to the techniques and tools employed. Research has shown the effectiveness of an innovative approach for integrating advanced simulations to improve detection capability.

The impact of data augmentation has been significant, showing how some datasets initially unusable in real conditions could be significantly improved thanks to the inclusion of a tool such as the DDoShield-IoT simulator. This methodology has proven particularly useful for dealing with situations where it is difficult to collect sufficient real data, such as in cases where the system is new or has not yet faced a significant number of threats. The ability to simulate scenarios has expanded the database, enhancing the system's readiness to handle complex attack scenarios. The study also emphasized that leveraging simulated data for training machine learning models can significantly bridge the gap between the availability of real data and the need for high-quality data to improve system performance. The adopted approach has strengthened the robustness and reliability of attack detection systems, especially in environments with a limited number of real attack examples.

The analysis over multiple datasets also supports the significant contribution made by the DDoShield-IoT simulator in optimizing model performance. In general, its usage has made a significant contribution to the performance efficiency in complex models such as RF and CNNs, particularly in imbalanced datasets where it has been crucial in balancing the distribution of classes and improving the performance of models in detecting complex and low-frequency attacks. The precision, recall, and F1-score metrics (5.12, 5.13, 5.14, 5.15) identify a significant loss in performance without the presence of the simulator, particularly in precision and F1-score, representing difficulties in distinguishing correctly attack classes. In contrast, the K-Means model was not heavily reliant on the simulator, with consistently better performance without simulated information. The RF model was the most beneficial among them with the highest impact, going from substandard performance without simulated data to optimum performance with such data available. It shows high sensitivity concerning balancing classes and diversification in the data, and the simulator was able to offer these.

In some cases, without the simulator, the model was not even able to identify attack classes, and the resulting recall and F1-score was around zero (5.12, 5.13). Similarly, the CNN model was also improved a lot with the usage of the simulator, with improved performance on nearly every parameter. In extremely imbalanced datasets, however, a reduction in recall was observed, which shows that even though the usage of the simulator improved the model's learning process, it was not completely able to eliminate the model's susceptibility to underrepresented and complex attack classes (5.14). On the other hand, the K-Means algorithm performed more satisfactorily without simulated data, suggesting that naturally occurring clusters in original datasets are more clearly identifiable by unsupervised clustering models compared to simulated datasets. Specifically, in Dataset 4, performance was good even when there were fewer samples, whereas incorporating the simulator introduced noise in the data reduced clustering accuracy, and impeded predictions (5.15).

These findings support the requirement to pick suitable augmentation techniques based on the type of the detection model. While supervised models and models with high demand for labeled data are highly reliant on simulation-based augmentation, unsupervised models like clustering would be vulnerable to simulated data. Following research is therefore imperative to fine-tune augmentation techniques in a way that better improves performance in detection without inducing unwanted side effects and permitting supervised and unsupervised models on equal grounds to enjoy the advantages presented by simulated data.

Chapter 6

Federated Learning and User Profiling for IoT Anomaly Detection

The scalable and customizable nature of DDoShield-IoT perfectly supports advanced ML techniques such as FL, which is an innovative approach to ML that allows training AI models on large amounts of data distributed across multiple devices (e.g., smartphones, IoT sensors) without the need to centralize these data. The framework's scalability allows networks of variable size management and the distribution of the computational load among devices. It is also highly customizable and ensures the adaptation of models to specific network needs and local data. Thanks to these features, DDoShield-IoT facilitates collaborative model training without centralizing data, preserving privacy and reducing the risks of exposing sensitive information. Based on these observations, the following chapter proposes an intrusion detection methodology that leverages an IDS based on FL and User Profiling adapted to a real-world use case in the Smart Grids context.

6.1 Federated Learning

FL is a decentralized ML approach designed to train models collaboratively across distributed devices or servers without sharing raw data. This method

enables privacy-preserving machine learning, which is critical for applications where data confidentiality and security are of paramount importance [13]. Indeed, with FL, data remains on local devices and only model updates, rather than the data itself, are shared with a central server for aggregation, ensuring that sensitive information is never exposed and making FL an ideal solution for privacy-sensitive applications in distributed environments such as IoT [50].

In detail, the FL workflow involves multiple device training models independently using their local data, as represented in Figure 6.1. Each device computes model updates and shares them with a central aggregator, i.e. the server, through methods such as Federated Averaging (FedAvg), which then sends the updated model to the devices for further training [24]. This iterative process continues until the model reaches the desired accuracy. The FL approach helps reduce computational and communication overhead, which is especially useful in resource-constrained environments such as IoT.

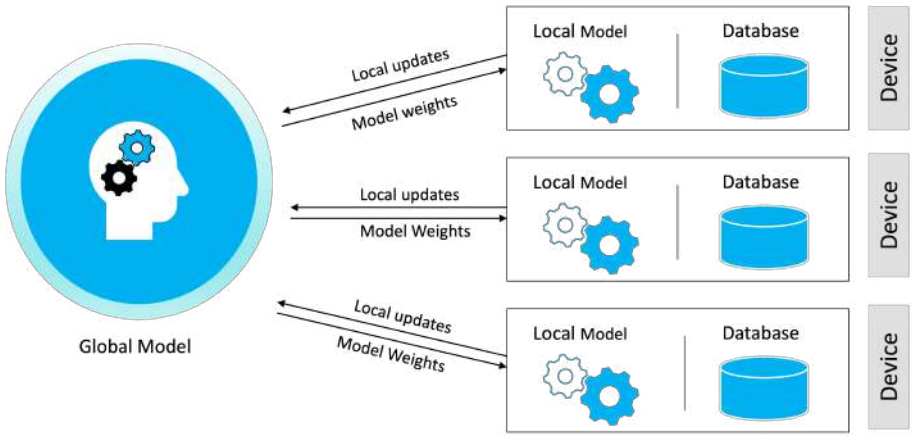

Figure 6.1. Federated Learning Workflow

In cybersecurity FL has shown great potential, improving the security and privacy of connected devices. Several FL-based models have been used in anomaly and intrusion detection systems, where multiple IoT devices collaborate to detect intrusions. In this manner, we obtain more robust models by leveraging the collective knowledge of distributed devices [96]. Indeed, the advantages of FL in cybersecurity include its ability to detect threats while keeping data decentralized, thus preserving privacy and reducing the risk of data breaches, and the possibility of improving the robustness of anomaly detection systems by continuously learning from real-time data across multiple devices [147]. However, there

are challenges, including the risks related to the model update sharing, which may still expose sensitive information. Data heterogeneity, network topology, and computational limitations of IoT devices remain significant barriers to their widespread adoption [154] [117]. In the GreenAI context, the FL reduces energy consumption and improves the ML models' sustainability in IoT and edge computing environments. Traditional centralized models require transmitting large amounts of raw data to central servers, which uses a lot of energy and is inefficient, especially in IoT applications. FL minimizes the need for data transmission by training models locally on edge devices, which reduces network traffic and energy consumption [18]. Researchers are working to make FL more energy-efficient for IoT, leveraging methods like joint quantization, model pruning, and adaptive resource allocation [128]. Their goal is to reduce the environmental impact of AI systems without sacrificing accuracy or effectiveness. So, they found that combining the FL with edge computing and fog nodes makes it possible to improve energy efficiency by reducing communication delays and energy usage [118]. This shows that FL can help us create AI solutions that are more sustainable and environmentally friendly, while also improving the performance of ML in IoT environments.

6.2 Proposed Methodology

In this section, we present a methodology encompassing the concepts of FL and User Profiling in a Two-Phase IDS to address scalability, real-time monitoring, privacy concerns, and autonomous detection in IoT environments.

In particular, we propose an IDS that executes, in several phases, the Intrusion Detection in a complex IoT environment simulated by DDoShield-IoT. First, thanks to the ability of the IDS testbed to generate diverse types of network traffic, we group devices into clusters based on traffic types through clustering algorithms like K-Means. Applying this User Profiling technique we can obtain a more efficient intrusion detection process tailored to the characteristics of each device's traffic patterns. As shown in Figure 6.2, in the first detection phase, in each cluster, a ML classifier, such as RF, analyzes the raw features of each incoming packet and classifies it as either benign or anomalous. The classification process of the first detection phase contributes 30% to the overall decision. Malicious packets identified in this phase are sent to the second detection phase for further analysis. Additionally, a small proportion of benign packets is randomly selected and sent to the second phase to prevent the occurrence of false negatives.

In the second detection phase, represented in Figure 6.3, there is an IDS for each cluster that performs local anomaly detection, analyzing packets in the

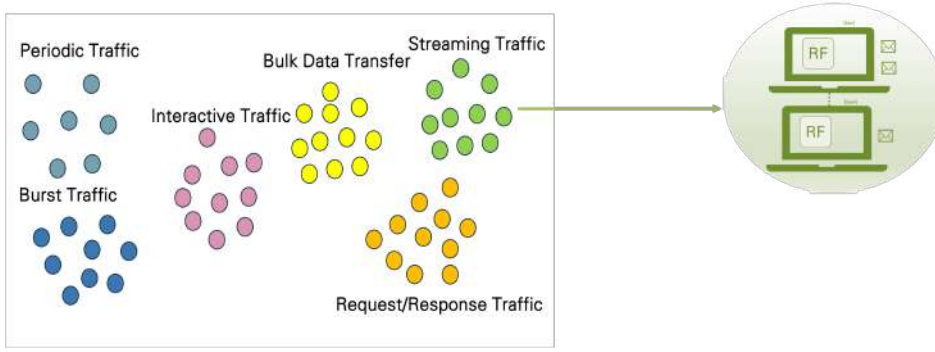

Figure 6.2. Phase One of Intrusion Detection Process

specific context of the cluster, and applying the statistical features extraction from packets collected within a particular window of time. This local analysis leverages FL, where the individual IDS within each cluster shares learned patterns and global detection model (e.g., CNN) weights with a central server that aggregates the weights, retrains the model, and redistributes it to the devices until convergence is reached. In the second phase, classifying packets as benign or anomalous contributes to the final decision at 70%.

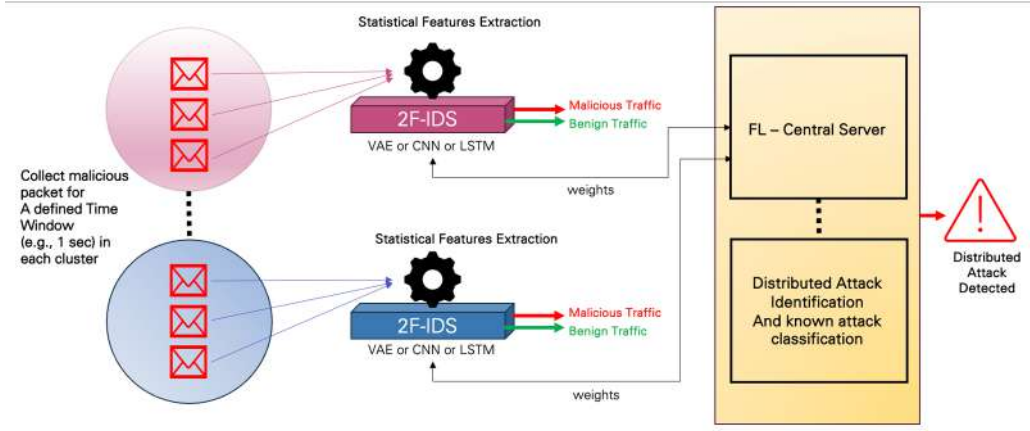

Figure 6.3. Phase Two of Intrusion Detection Process

Finally, the results from both detection phases are combined and compared against a predefined threshold to classify packets as benign or malicious. This two-phase approach enables an efficient and scalable solution for intrusion detec-

tion in IoT networks, improving real-time monitoring capabilities while minimizing the environmental impact and ensuring privacy. Using FL ensures that data does not need to leave the local devices, maintaining user privacy while continuously improving the detection model.

6.2.1 Decision Process for Federated Learning

As shown in Figure 6.3, the model updates from the IDS nodes are aggregated at a central server responsible for making the final decision in the FL framework. Here are the essential steps in the decision-making process:

1. **Local Model Updates**: After training local models based on observed network traffic, each IDS node in a cluster only needs to send the updated parameters (i.e., gradients or weights) instead of sending raw data to the central server.
2. **Federated Aggregation**: The central server executes FedAvg (described in detail in 1) to aggregate received model updates. To provide robustness in the final model, a weighted approach is performed where the IDS nodes with greater data reliability or larger datasets are prioritized.
3. **Global Model Tuning**: The global model is validated against a key dataset after aggregation and tuned to mitigate potential model drift while verifying that the new updates do not reduce detection accuracy.
4. **Final Anomaly Classification**: Using the combined model, the server classifies the oddities in the network traffic. The model confidence is used to calculate an anomaly score, and a dynamic threshold adaptable to traffic patterns is used for decision-making.
5. **Distributed Model Refinement and Continuous Learning**: The refined model is redistributed to all IDS nodes, ensuring that the system continuously learns and adapts to evolving attack patterns without requiring raw data exchange.

By taking a more integrated approach, the two-phase detection process makes a collaborative decision based on the complementary advantages of both stages. Stage one serves as a general filtering mechanism, culling potentially malicious traffic at a low computational cost. This provides an initial classification that, while not as precise, reduces the volume of data and computation required for the second stage. In the second phase, this classification is refined using advanced anomaly detection techniques to achieve high accuracy. By integrating the outputs of both phases and comparing them against a predefined threshold,

Algorithm 1 Federated Averaging (FedAvg)

Input: Local model updates from N IDS nodes: $\{w_i\}_{i=1}^{N}$
Output: Aggregated global model w_g

Initialization:
Set initial global model w_g
Define learning rate η
Aggregation Process:
for each global training round $t = 1, 2, ..., T$ **do**
  for each IDS node $i = 1, ..., N$ in parallel **do**
    Train local model w_i^t on local dataset D_i
    Compute model update $\Delta w_i = w_i^t - w_g$
    Send Δw_i to central server
  end for
  Compute weighted average of local updates:
    $w_g^{t+1} = w_g^t + \eta \sum_{i=1}^{N} \frac{|D_i|}{\sum_{j=1}^{N} |D_j|} \Delta w_i$
  Redistribute w_g^{t+1} to all IDS nodes
end for

the system strikes an optimal balance between speed and accuracy. This results in an efficient and scalable intrusion detection method, particularly beneficial in IoT contexts where vast amounts of data must be analyzed.

The Algorithm 2 3 summarizes the overall process.

6.2.2 CyberSEAS Project

The Cyber Securing Energy dAta Services (CyberSEAS) Project is a strategic initiative funded by the European Union. Its goal is to make energy supply chains more resistant to cyberattacks. It aims to combat the most severe cyber risks for Electric Power Systems (EPES), protect consumer data, and secure the energy data space. It focuses on protecting supply chains in many ways, working with different industries, and all parts of a cyber attack.

A key innovation of CyberSEAS [33] is the adoption of FL as a privacy-preserving technique for anomaly detection in smart grids. Traditional centralized methods for ML often put consumer data at risk of privacy breaches. While, FL keeps sensitive data, like energy usage patterns, on the user's device. This way, user profiling and advanced algorithms for detecting anomalies, VAE are deployed

Algorithm 2 Phase 1, 2, 3, 4 - TP-FL-IDS

Input: Incoming network traffic packets P
Output: Each Packet Classification as {Benign, Malicious}

Phase 1: Traffic Profiling and Clustering
Initialize IDS with Clustering Algorithm (e.g., K-Means)
Group IoT devices into clusters based on network traffic patterns
for each packet $p \in P$ **do**
    Assign p to its corresponding cluster C_i
end for
Phase 2: First Detection Phase (Local Classification)
for each cluster C_i **do**
    Apply ML Classifier (e.g., Random Forest) on raw packet features
    Compute anomaly probability $P1(p)$ from Phase 1
    if p is classified as Benign **then**
        With probability α, forward to Phase 2 for re-evaluation
    else
        Mark p as "Potentially Malicious" and forward to Phase 2
    end if
end for
Phase 3: FL and Global Model Aggregation
for each cluster C_i **do**
    Each IDS_i trains Local Model (e.g., CNN) on processed traffic data
    IDS_i sends model updates (gradients/weights) to Central Server
end for
Central Server aggregates updates using FedAvg
Central Server fine-tunes Global Model using the validation dataset
Central Server redistributes updated Global Model to all IDS nodes
Phase 4: Continuous Model Adaptation
Redistribute the improved Global Model to all IDS nodes and retrain it untile convergence
Repeat process as new traffic flows in

Algorithm 3 Phase 5 - TP-FL-IDS

Phase 5: Final Anomaly Classification
for each packet p **do**
  Compute anomaly probability $P2(p)$ from Phase 2 (FL-based detection)
  Compute final anomaly score $S(p)$ as:
    $S(p) = 0.3 \times P1(p) + 0.7 \times P2(p)$
  if $S(p) \geq$ Threshold **then**
    Flag p as "Malicious" and trigger security alert
  else
    Retain p as "Benign"
  end if
end for

across different nodes. This helps identify unusual energy consumption patterns without compromising privacy.

6.2.3 Privacy and Cybersecurity Challenges in Smart Grids

Smart grids are modern and advanced electricity distribution systems. They enable the transition from a centralized electricity generation system to a digital distributed energy resource system. They allow for communication between consumers and service providers and ensure that energy usage, distribution, and generation data are collected and analyzed in almost real-time [81]. By studying this data, smart grids can provide useful information and suggestions to everyone involved, like consumers, utilities, and suppliers, to help them use energy more efficiently. These structures use IoT technologies [151] to reliably collect, monitor, and analyze data from many electrical devices, such as smart appliances and smart meters. Each energy producer has operations centers that receive information from around their area. These devices report usage readings to operations centers using networks such as RFID, Wi-Fi, and ZigBee, as well as WAN networks like GPRS and 4G/5G wireless cellular networks. Utilities then manage the transmission and perform billing based on these readings [92]. However, IoT-based smart grids have the same problems as many other IoT systems. This is because the sensors have limited capabilities and resources in terms of computational power, bandwidth, and storage. These devices, like sensors and smart meters, can use high energy and resources when collecting and sending real-time

data. This can lead to performance issues. Additionally, the increased connectivity of smart grids can expose them to several challenges from both cybersecurity and privacy perspectives [106]. A smart grid is vulnerable to physical attacks by a human insider attacker, malicious software that can interfere with the control system, or by consuming system resources to perform the attacker's tasks or launch a DoS attack [81]. Each of these forms of disruption can be highly dangerous. For example, the manipulation of the billing information of some users can lead to serious economic disruptions if the smart grid is not equipped with an appropriate monitoring and/or anomaly detection strategy. Moreover, frequent data collection can infringe upon consumers' privacy, as such data is directly related to users' lifestyle habits [58] [160]. Any energy usage data linked to personal information must be protected and monitored appropriately. Electricity usage data relates to customers' energy usage patterns, which are closely related to their private lives; therefore, their improper management can cause disclosure of customers' privacy [83]. Energy usage data obtained from a third party may reveal personal information. The storage of energy usage information on the meter and its subsequent distribution can act as a rich side channel, thereby exposing customer habits and behaviors. Certain activities like watching television have discernible energy usage signatures [92].

According to NIST [106], the primary areas of privacy concern that must be addressed in the context of smart grids are as follows:

- The metering system must be designed to prevent any form of abuse by personnel or modification of the collected data to prevent fraud.

- The data stored in the smart meter may reveal certain activities of smart home appliances. Additionally, it can be used to track specific times and locations of energy usage in particular areas of the home, which could further indicate the appliances used and/or types of activities.

- Collecting energy usage data could allow inferences about whether a facility is occupied, where people are located, what they are doing, and so on.

- Personal lifestyle information from energy usage data could be valuable to some vendors or parties. For example, vendors could use this information for targeted marketing.

6.3 Combine User Profiling with Federated Learning for Anomaly detection

The proposed framework uses a User Profiling and FL combination to detect unusual energy consumption, protect smart grids from cyber attacks, and address privacy issues. This strategy ensures the security of consumers' data, complies with European regulations such as GDPR and the Electricity Directive, and protects the energy supply chain from attacks that exploit consumers in their role as prosumers.
User profiling is essential to the proposal. This technique defines each user's behavior under normal energy usage conditions by analyzing their energy consumption patterns in typical situations. This analysis creates a detailed profile for each user. By comparing users' behavior against their standard profiles, the system can accurately identify any anomalies in energy consumption, which may indicate the presence of attacks or other suspicious activities [30]. This level of customization allows the system to improve its detection capability while reducing false positives.
The FL enables the training of ML models while keeping sensitive data on users' devices, specifically smart meters, without centralizing it. In this way, we reduce the likelihood of personal data being sent or stored in one place, reducing the risk of privacy breaches and making the system more secure. Specifically, smart meters collect and process data locally, train the model, and only send updates to the central server. The central server then combines these updates to create a more accurate global model.
Combining FL with User Profiling, we solve important data security and operational resilience issues in the energy sector. Indeed, FL significantly reduces the risks associated with centralized data storage by distributing data processing and keeping sensitive information where needed. This decentralized model provides privacy protection aligned with GDPR and reduces the risk of cyber attacks on centralized data sources. It also strengthens the energy supply chain by protecting it from threats targeting consumer devices, which is important for prosumers, for whom maintaining the integrity of the energy infrastructure is crucial. The approach also aligns with the European Data Strategy, which aims to strengthen the security and governance frameworks that regulate data exchanges between interconnected European energy systems.

Refining User Profiling for Anomaly Detection The user profiling approach in this study was designed to efficiently detect deviations in normal energy consumption patterns. However, energy usage can vary significantly based on temporal factors such as the hour of the day, day of the week, or season. To

improve accuracy, the framework incorporates additional contextual factors into the anomaly detection process:

- **Time-aware Profiling**: Energy consumption profiles are adjusted dynamically based on historical trends, distinguishing between peak hours (e.g., high usage in winter evenings due to heating) and off-peak periods.
- **Anomaly Sensitivity Adjustment**: The model assigns different risk levels to anomalies depending on the context. For example, a sudden increase in power usage at night may be more suspicious than during normal working hours.
- **Contextual Awareness**: The system considers special conditions (e.g., public holidays or extreme weather events) that may justify deviations in energy consumption.

By integrating these refinements, the framework improves its ability to distinguish between true security threats and natural variations in user behavior. Future work could explore reinforcement learning techniques to dynamically adjust anomaly detection thresholds based on long-term consumption trends.

The User Profiling technique enhances anomaly detection capabilities, facilitating the early identification of suspicious activities and potential cyber threats. This proactive approach improves response times and ensures the overall resilience of the energy infrastructure. It also creates a strong balance between data protection and real-time threat detection, providing comprehensive security for critical energy infrastructure against evolving cyber threats.

6.4 The Smart Grid Scenario: System and Threats Model

A smart grid system includes the following main components [152], which define our reference model:

- **Trust Authority (TA)**: This entity provides cryptographic materials, such as public and private key pairs, to system participants via registration services. The TA remains offline unless a participant is added or removed.
- **Users (U)**: In a Smart Grid we can find domestic or industrial users. Each user has a smart meter to periodically monitor the electricity consumption

generated by devices in their home or business. After completing registration with the TA, users can securely send their data to a remote control center.

- **Gateways (GW) or Concentrators**: Gateways act as regional intermediaries, collecting and aggregating data from smart meters in their area and transmitting it to the control center. Equipped with computing and storage capabilities, gateways verify the authenticity of the data received, aggregate it, and forward it to the control center.
- **Control Center (CC)**: The control center receives aggregated data from the gateways, analyzes it to estimate energy demand, and optimizes power generation and distribution.

In this model, Smart Grids collect energy consumption data from users. Then gateways process and aggregate these data. Finally, the control center analyzes these data to ensure reliable energy services.

FL approaches for anomaly detection require proper assignment of FL server and client roles to the various system components. In this configuration, the gateways can be considered the FL clients, while the control center serves as the FL server [153]:

- **Gateways as FL Clients**: Each gateway receives processed data from user devices and acts as the data owner. Since gateways have computational and memory capabilities, they can train models locally for anomaly detection. During the training phase, gateways receive initial model parameters (such as the weights of a neural network) from the control center and send updated parameters (such as gradients) to the server.
- **Control Center as FL Server**: The control center, which receives aggregated data from several gateways, coordinates the training process. During initialization, it broadcasts the model and basic parameters to all gateways and, at the end of each training cycle, collects the updates sent by the gateways, aggregates them, and calculates new parameters to be sent back to the gateways.

The main risk in the model described is energy theft, which can have a significant economic impact since it is critical to identify through consumption data. Annual losses due to energy theft are estimated at $100 million in Canada [9], $170 million in the United Kingdom [155], and $6 billion in the United States [92]. Energy theft can also compromise power generation and distribution, making its detection a key priority for smart grids [119].

Unlike traditional grids, energy theft in a smart grid is not limited to conventional methods such as undercurrent, undervoltage, or phase shifting [36].t also includes more sophisticated threats, such as data falsification attacks [32], which manipulate sensor measurements by circumventing traditional defense mechanisms, and distributed energy theft [63], in which malicious users can tamper with meter readings to reduce their bills or falsify data from their distributed generation facilities (e.g., solar panels or wind turbines) to claim higher energy production and charge undue costs to the supplying company. While different threats affect smart grid components, our main focus is to identify whether energy theft occurs at the level of consumer devices.

6.5 Proposed framework

6.5.1 Architecture

As mentioned above, the CyberSEAS framework leverages User Profiling and FL techniques to detect anomalous energy consumption. This framework includes three main components which allow it to gain its goal They are:

- **Local anomaly detectors (LAD)**. These components leverage the VAE model and K-means clustering approach to analyze local energy consumption data and detect potential anomalous energy consumption.
- **FL clients**. These elements implement the client side of the FL scheme, sharing the local training weights with the central server, and then waiting for the updated weights from it.
- **FL server**. This component collects the weights provided by the FL clients and then aggregates the weights to build a global model, and then provides the updated weights to the FL clients.

Considering the Smart Grid system model, the CyberSEAS framework's components can be deployed as depicted in Figure 6.4, where there is the i) ***Edge Nodes***, i.e., the smart meters and concentrators. Smart Meters measure and record electricity use, while concentrators aggregate data from multiple meters. These devices collect real-time data on energy consumption (step 1) and can run a local anomaly detector, which applies the combined VAE and K-mean approach to detect behavioral anomalies. In addition, edge nodes also run an FL client, which shares the VAE training weights with the FL server (step 2), and waits for the updated parameters from the server; and ii) the ***Central Server***, i.e., the

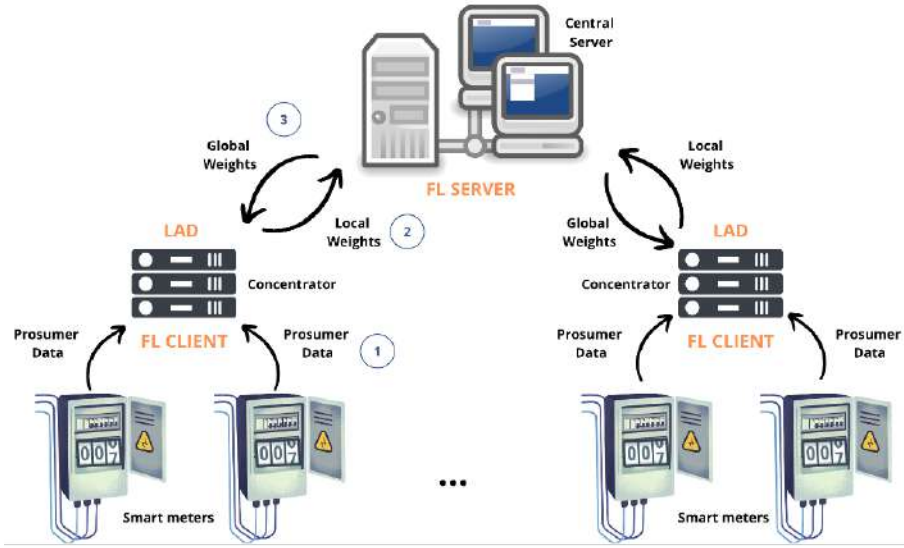

Figure 6.4. CyberSEAS Framework Architecture and Deployment

company's SIEM, runs the FL server, which aggregates the weights of the VAE model obtained by the edge nodes, i.e., the FL clients, and updates the global model weights. Indeed, after receiving the local updates, the FL server combines these weights to improve the global model. Then redistributes the update weights to the FL clients (step 3), via a continuous cycle that improves the accuracy of the attack detection system without compromising the users' data privacy.

6.5.2 Anomaly Detection Algorithms

The proposed framework initially uses the K-Means algorithm to profile users based on their energy consumption. In particular, to determine the optimal number of clusters, we apply the elbow method to evaluate the quality and consistency of the formed clusters, and then classify the training and test datasets based on these clusters. User profiling helps distinguish between typical and atypical consumption behaviors.

Next, we train a VAE using the normalized training data that learns a compact representation of the data and reconstructs it. Specifically, during the training phase, the VAE encodes the input data into a lower-dimensional latent space and decodes it back into the original space. After training, we use the VAE to

reconstruct the test data and compute reconstruction errors, highlighting the difference between the original data and its reconstruction. These reconstruction errors define how well the model captures the underlying data distribution.

We then proceed with anomaly identification via a multi-step approach. First, we identify the less populated clusters obtained after running K-Means as potentially anomalous since they represent the group of users with the least common consumption patterns, which may therefore indicate unusual or suspicious behavior. Then, we compute a threshold for reconstruction errors, using the training data, which helps us distinguish between normal and anomalous reconstruction errors. Finally, we combine the cluster labels obtained from K-Means with the reconstruction errors calculated by VAE to identify anomalies in the test set. By integrating these two methods, we improve the robustness and accuracy of our anomaly detection system to effectively identify unusual user profiles and significant deviations in data patterns, setting them as potential threats.

K-Means for user profiling and the combination of VAE for anomaly detection provide a powerful and flexible approach to identifying anomalies in unlabeled data. This synergy offers several benefits, including:

1. **User profiling and personalization**: K-Means allows us to identify distinct user profiles, allowing the anomaly detection model to adapt to specific user behaviors. This adaptation can improve detection accuracy by reducing false positives, as it considers behavioral variations across different user groups.
2. **Advanced anomaly detection**: VAE is a robust, unsupervised learning model that can capture complex data structures and identify anomalies based on reconstruction errors. VAE can learn latent data representations that improve anomaly detection accuracy compared to traditional methods.
3. **Robustness and flexibility**: The combination of K-Means and VAE enables anomaly detection at both the cluster and reconstruction levels, making the system more resilient. This methodology can be applied to different data types and does not require labeled datasets, making it suitable for scenarios where labeled data is unavailable.
4. **Federated Learning**: The ability to train the model locally on multiple clients and then aggregate the results improves the privacy and security of user data. This approach enables the development of a more robust and accurate global model by leveraging distributed information without centralizing the data. However, when implementing this system it is essential to consider the computational complexity, the need for configuration and optimization, and the challenges associated with federated learning.

6.5.3 Implementation Details

Components development and processing flow

We developed the proposed framework using PySyft, a Python library for FL. This library allows ML model training on distributed data, making their centralization not mandatory. PySyft is, in fact, an open-source platform that enables conducting data science experiments on remote data, combining federated learning techniques and differential privacy [166].

In this framework, several federated learning clients have been defined, each representing a smart meter or a concentrator, depending on the type of implementation. Each client has local data representing energy consumption measurements and uses a local anomaly detector to analyze such data.

The local anomaly detector uses PySyft to implement a VAE-based architecture and K-means clustering techniques for user profiling. First, energy consumption data is normalized using a standard scaler. Next, K-means clustering is used to identify user profiles, determine the optimal number of clusters through silhouette score analysis, and classify the training and test data according to the identified clusters.

The anomaly is detected by configuring and training a VAE using the normalized training data. This model is then used to reconstruct the test data and calculate the reconstruction errors, establishing a threshold for identifying anomalies. The system then finds the least populated clusters, considering them potentially anomalous, by comparing the reconstruction error with the calculated threshold. If the error exceeds the threshold, the samples are marked as anomalous. Finally, the system integrates the cluster information with the reconstruction errors to identify any anomalies in the test data. In particular, we consider a sample as anomalous only if both criteria, i.e., cluster and reconstruction error, indicate it as such. The trained models, the scaler, and information useful for anomaly detection are saved and made available to the federated learning daemon.

For the federated learning process, clients use PySyft to collect VAE model weights and securely share them with the federated server. Clients start the process by training the global model on their local data, sending each local VAE model's weights to the central FL server. The server collects these VAE model weights, aggregates them using specific aggregation functions, updates weights, thus enhancing the global model accuracy, and then securely distributes the updated weights to clients for local use.

Edge Nodes Configuration and Data CollectionNormalization

Edge nodes, i.e. smart meters, are an integral part of this project's distributed framework, functioning as localized data processing units. Each Edge node is configured to perform different tasks that ensure efficient and secure management of energy consumption data. Initially, these nodes collect data from local Enel meters at 15-minute intervals. Once the data is collected, the Edge node performs pre-processing and raw data normalization. During pre-processing, the data is cleaned to eliminate any inconsistencies or errors that may have occurred during collection. Normalization involves scaling the data to a standard range or standardizing it to a mean of zero and a standard deviation of one. This process mitigates issues related to different scales of input features. After pre-processing and normalization, the Edge node trains a local machine-learning model using the prepared data, leveraging the localized data to detect anomalies specific to each environment. This decentralized approach reduces the computational load on the central server while preserving data privacy, as the raw data remains on the local node and is not transmitted to a central location.

6.5.4 Communication between Nodes

Communication between edge nodes and the central server is facilitated using the PySyft library, which ensures secure and efficient data exchange. Once local models are trained, edge nodes send model updates, such as weights or gradients, to the central server without sharing the raw data, thus maintaining data privacy. The central server aggregates updates from all edge nodes to refine the global model, which is then redistributed to each edge node. This iterative process of updating and redistributing the global model allows each node to benefit from the collective knowledge gained from the data across all nodes, improving the accuracy and robustness of the model over time.

6.6 Experimental results

The experiments, formulated through a Design of Experiment (DoE), were carried out on a MacBook Pro computer with a 2GHz Intel Core i5 quad-core processor and 16 GB 3733 MHz LPDDR4X memory.

6.6.1 Case Study

We evaluated the proposed model on a real, unlabeled dataset containing energy consumption data from 288 consumers, recorded over six months from December 2023 to May 2024. Each row in the dataset represents a consumer's energy consumption reading, identified by a unique smart meter (POD) code. These readings, taken at 15-minute intervals, represent the energy consumption in kilowatt-hours (kWh) during each interval. The dataset structure includes the following main fields:

- **POD**: representing the unique smart meter code;
- **Type**: indicates the energy tariff, that can be classified as:
  - *R1* - for primary residences, often associated with lower costs for efficient use.
  - *R3* - for secondary residences or holiday homes, typically with higher costs.
  - *A1*- for non-residential entities such as businesses or industries, where costs depend on contracted power and time of use.
- **State**: represents the status of the reading, for example *RECV* inficates the reading received status.
- **Date**: is the date and time of the reading recorded.
- **Samples**: indicates a set of energy consumption readings, representing data collected by the identified user's smart meter. These readings have not yet been pre-processed.

An example of the dataset structure, including some dummy entries, is provided in Table 6.1. Here, POD is unreal because of privacy concerns.

POD	Type	State	Date	Samples
Pod#1	R1	RECV	05/04/24 00:00	0|0$0|0$0|0$0|
Pod#2	R3	RECV	13/04/24 00:00	0|0$0|0$0|0$0|
Pod#3	R3	RECV	24/04/24 00:00	0|0$0|0$0|0$0|
Pod#4	A1	RECV	27/04/24 00:00	0|0$0|0$0|0$0|
Pod#5	R1	RECV	01/04/24 00:00	0|0$0|0$0|0$0|

Table 6.1. Example Dataset (April 2024)

For our experiment, we considered a subset of the provided dataset, consisting of energy consumption data from 10 smart meters, to evaluate the performance of different FL schemes and the proposed approach's ability to detect anomalous energy consumption.

The dataset used for the experiment was unbalanced since it contained benign data at most. So, to further improve the experiment results we expanded it by emulating anomalous energy consumption patterns. This expansion allowed a more comprehensive evaluation of the anomaly detection capabilities of the proposed FL model. The use of real-world energy consumption data, combined with simulated anomalies, ensures the robustness and applicability of the proposed framework in practical scenarios.

6.6.2 Design of Experiment

DoE is a methodology used to plan, conduct, and analyze experiments to obtain meaningful and reliable information, optimize resources, and minimize errors. This discipline allows us to study how various factors influence a given outcome. The central idea of DoE is that, through careful experiment planning, it is possible to isolate the effects of independent variables (the factors) on a dependent variable (the outcome or output). Specifically, the design focuses on the variables to study selection, the determination of their variation modes, and the construction of a sequence of experiments that allows for collecting sufficient data to draw valid conclusions. A fundamental aspect of DoE is a systematic approach that reduces the number of necessary experiments, avoiding redundant tests. This is achieved by defining a plan that strategically explores all possible combinations of factors and levels using advanced statistical techniques to reduce the margin of error and improve the conclusion's reliability, increasing the efficiency of the process.

In practice, an experiment designed according to the DoE often includes techniques such as factorial designs, which allow the examination of multiple factors and their interactions, or shallow response designs, which explore the system behavior related to the chosen factors. Another important concept concerns the control of disturbing variables: the DoE helps to reduce or eliminate the influence of factors that are not wanted to be studied, improving the quality of the data obtained.

Use Case DoE

In the considered use case, we conducted 27 experiments, using JMP software [1], summarized in Table 6.2, providing following the factors and response variable details:

- **Factors:**
  - **Time Band (h):** indicates the time range for detecting any anomalies. It was divided into: *daytime* (08:00–18:00) and *night* (18:00–08:00);
  - **FL Aggregation:** indicates the Federated Learning algorithm chosen for anomaly detection. The algorithms used and analyzed were: *FedAvg* and *FedYogi*;
  - **Consumption Growth (%):** indicates, in percentage, the client energy consumption increasing. It simulates the presence of anomalies in the dataset;
  - **Duration Increment (h):** indicates the period during which each client's consumption profile is altered. This factor was divided into: *hourly* (the increase of energy consumption occurs for 2h [2] consecutive) and *half day* (the increase of energy consumption occurs for half a day, therefore, for the entire time zone);
  - **Users with Anomaly (%):** indicates the percentage of users to which the anomaly is injected;
- **Response Variable: PR-AUC**, is presented in more detail in the following section.

6.6.3 Evaluation Metrics

The unlabeled nature of the provided dataset limited the choice of evaluation metrics of the proposed framework, so it was not possible to use traditional metrics. So, we used the Area Under the Precision-Recall Curve (PR-AUC) metric to evaluate the quality and effectiveness of the proposed model. PR-AUC is an evaluation metric typically used in binary classification problems, with unbalanced classes and unlabeled datasets, such as the use case in question, and is used to understand how the model handles predictions of the positive class, i.e. the class with anomalies. In particular, a distinction is made between:

[1] **JMP website**: https://www.jmp.com/it_it/home.html

[2] 2h is an agreed value but modifiable.

Factors					Response Variable
Time Band (h)	**FL-Aggregation**	**Consumption Growth (%)**	**Duration Increment (h)**	**Users with Anomaly (%)**	**PR-AUC**
18:00-08:00	FedYogi	15%	2:00	20%	0.9000
18:00-08:00	FedYogi	5%	10:00	10%	0.8960
08:00-18:00	FedAvg	10%	10:00	20%	0.8902
18:00-08:00	FedYogi	10%	10:00	5%	0.8765
08:00-18:00	FedYogi	15%	10:00	20%	0.8632
18:00-08:00	FedYogi	10%	2:00	20%	0.9008
08:00-18:00	FedYogi	15%	2:00	10%	0.8998
08:00-18:00	FedAvg	5%	2:00	20%	0.9047
08:00-18:00	FedAvg	15%	10:00	10%	0.8949
18:00-08:00	FedAvg	5%	10:00	5%	0.8867
18:00-08:00	FedAvg	10%	10:00	10%	0.8996
08:00-18:00	FedYogi	10%	2:00	10%	0.8923
08:00-18:00	FedYogi	5%	2:00	5%	0.8978
08:00-18:00	FedYogi	10%	10:00	5%	0.8965
18:00-08:00	FedAvg	10%	10:00	20%	0.8965
08:00-18:00	FedAvg	15%	10:00	5%	0.9004
18:00-08:00	FedAvg	15%	2:00	5%	0.8951
08:00-18:00	FedYogi	15%	10:00	10%	0.8904
08:00-18:00	FedAvg	10%	10:00	10%	0.8979
08:00-18:00	FedYogi	15%	10:00	5%	0.8979
18:00-08:00	FedAvg	10%	2:00	5%	0.8951
08:00-18:00	FedAvg	5%	10:00	5%	0.8983
08:00-18:00	FedYogi	5%	10:00	20%	0.8978
08:00-18:00	FedAvg	5%	10:00	20%	0.8952
08:00-18:00	FedAvg	15%	10:00	20%	0.8771
18:00-08:00	FedYogi	5%	10:00	10%	0.8825

Table 6.2. Design of Experiments

- *Positive classes*: in the proposed dataset, the positive classes include the anomalous behavioral profiles. Each anomaly is, therefore, a positive sample;
- *Negative classes*: the negative classes, on the other hand, are formed by the normal behavioral profiles. In our case, each altered sample (anomalous sample) is a negative sample;

This metric is also particularly suitable in cases where the dataset is unbalanced and therefore has one class (positive, negative) with many more samples than the other, just like in the case in question, where the negative class (normal behaviors) is more numerous than the positive class (anomalous behaviors).

In particular, for a binary classification problem, the PR-AUC measure of 0 is a perfectly incorrect classifier. The classifier predicts the opposite class of the actual class for every sample. However, this situation is manipulatable by the simple trick of flipping the model's decisions, which would drive the PR-AUC to a score of 1. This property provides PR-AUC with the usefulness of being an effective metric in evaluating models by placing importance on the prediction quality of the positive class in vastly imbalanced data or with a lack of class labels.

PR-AUC aims to understand how well the model predicts the positive class through model evaluation that exploits the combination of the Precision and Recall metrics. In a completely unlabeled dataset, such as the one under analysis, we have no information on which samples are anomalous or normal. Therefore, we cannot directly calculate traditional metrics such as precision and recall, which require real and predicted label comparison by the model. However, it is possible to use PR-AUC as an indirect measure of the model's performance, based on the anomaly scores generated by the model for each sample. These scores indicate the probability that each sample belongs to the anomalous (positive) class, but without an explicit label. In particular, the evaluation with PR-AUC is done by sorting the samples based on the anomaly scores and plotting a Precision-Recall curve for each possible decision threshold. The area under this curve (PR-AUC) provides a measure of the ability of the model to correctly separate anomalies from normal behaviors, even in the absence of labels. Precision and recall metrics for an unlabeled dataset depend on the anomaly scores assigned by the model to each data point. The anomaly score measures the likelihood that a sample is anomalous based on the model's detection capabilities. To compute precision and recall, samples are first ranked in descending order of their anomaly scores. Then, various thresholds are applied to classify the samples as positive (anomalous) or negative (benign). The model iteratively adjusts these thresholds to evaluate its performance across a range of sensitivity levels. This process provides insight into the balance between precision and recall while constructing the Precision-Recall curve. The area under this curve, named PR-AUC, serves as a comprehensive metric, evaluating how effectively the model distinguishes anomalies in unlabeled data. In this way, PR-AUC becomes a useful indicator to estimate how well the model identifies anomalies, without the need for real labels.

The PR-AUC value ranges from 0 to 1. A value close to 1 indicates that the model effectively detects anomalies, while a value close to 0 suggests that the model struggles to differentiate between normal and anomalous behaviors.

In detail, to fully understand the results, it is necessary to note these numerical references:

- **PR-AUC=1:** indicates that the model can detect all anomalies (maximum *recall* value) without reporting false positives (maximum *precision* value);
- **PR-AUC=0.5:** indicates that the model predicts *randomly*;
- **PR-AUC=0:** indicates that the model can't distinguish between the two behavioral profiles (the two classes).

6.6.4 Obtained Results

6.6.5 PR-AUC

	PR-AUC		PR-AUC		PR-AUC
Exp#1	0.9000	**Exp#2**	0.8960	**Exp#3**	0.8902
Exp#4	0.8766	**Exp#5**	0.8632	**Exp#6**	0.8990
Exp#7	0.9008	**Exp#8**	0.9047	**Exp#9**	0.8949
Exp#10	0.8867	**Exp#11**	0.8996	**Exp#12**	0.8923
Exp#13	0.8978	**Exp#14**	0.8965	**Exp#15**	0.8965
Exp#16	0.9004	**Exp#17**	0.8951	**Exp#18**	0.8904
Exp#19	0.8740	**Exp#20**	0.8979	**Exp#21**	0.8951
Exp#22	0.8983	**Exp#23**	0.8978	**Exp#24**	0.8939
Exp#25	0.8952	**Exp#26**	0.8771	**Exp#27**	0.8825

Table 6.3. PR-AUC Results for Experiments

The execution of the experiments, as reported in Table 6.2, produced the results summarized in Table 6.3 regarding the PR-AUC score. The corresponding PR-AUC curves are illustrated from Figure 6.5 to Figure 6.13. The results obtained are overall satisfactory for most of the experimental configurations.

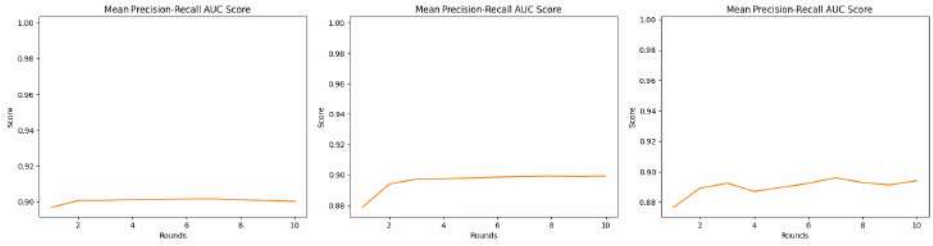

Figure 6.5. Experiment 1, Experiment 2, Experiment 3

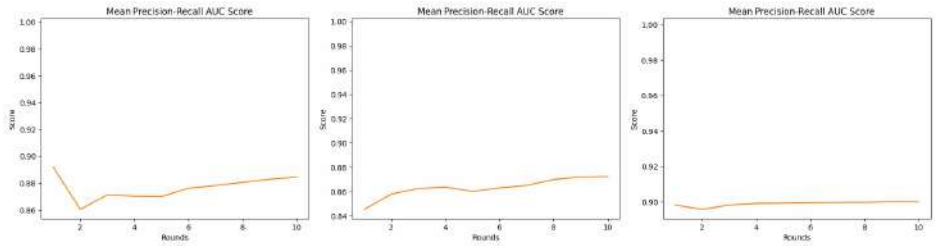

Figure 6.6. Experiment 4, Experiment 5, Experiment 6

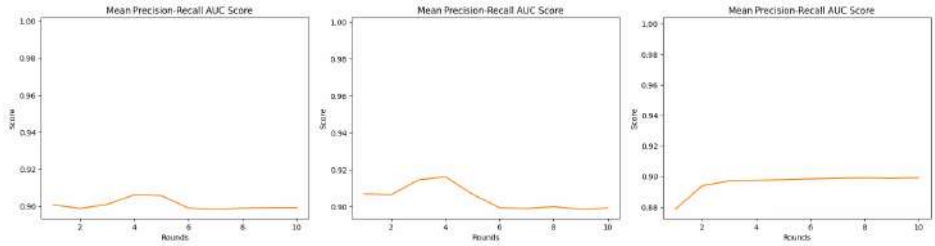

Figure 6.7. Experiment 7, Experiment 8, Experiment 9

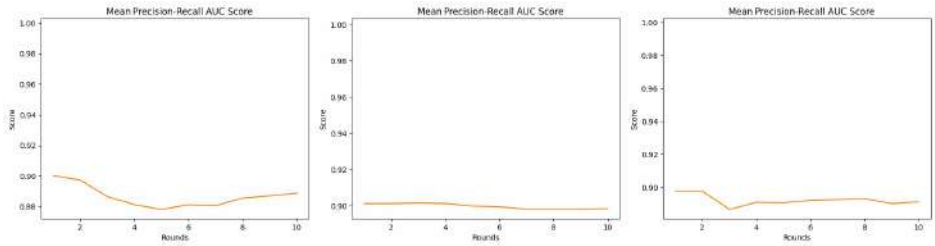

Figure 6.8. Experiment 10, Experiment 11, Experiment 12

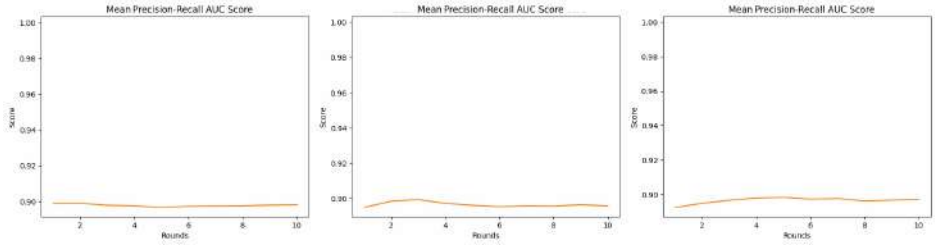

Figure 6.9. Experiment 13, Experiment 14, Experiment 15

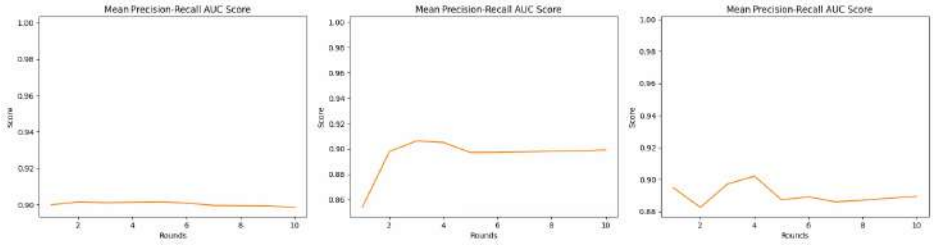

Figure 6.10. Experiment 16, Experiment 17, Experiment 18

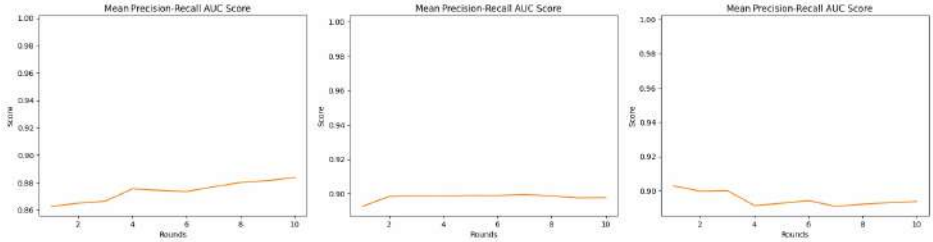

Figure 6.11. Experiment 19, Experiment 20, Experiment 21

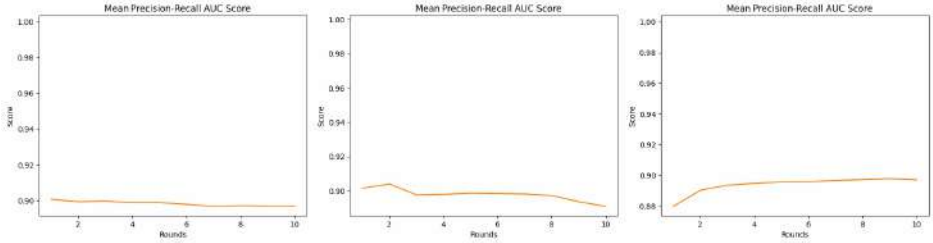

Figure 6.12. Experiment 22, Experiment 23, Experiment 24

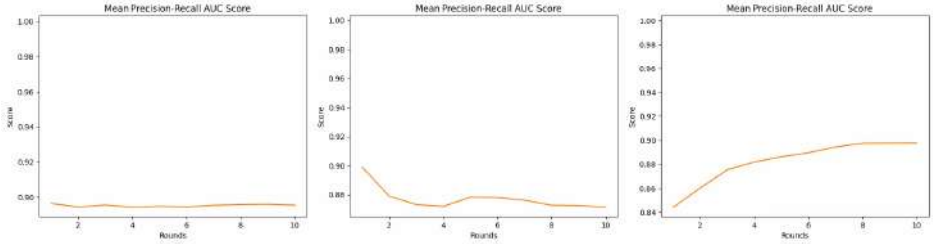

Figure 6.13. Experiment 25, Experiment 26, Experiment 27

Following the structure of the experimental plan defined in the DoE, it emerged that the precision and recall metrics are particularly affected by scenarios in which the anomalies have the following characteristics:

i. **Low evidence:** anomalies characterized by small increases in consumption;

ii. **Limited temporal distribution:** anomalies concentrated in short time windows, such as in the case of hourly increments;

iii. **High diffusion among users:** anomalies involving a higher percentage of users.

The results also highlight that the *FedYogi* aggregation algorithm provides greater stability and better performance, especially in more critical detection contexts, such as those characterized by minimal consumption increments distributed over small time intervals.

An important aspect to highlight concerns the observed PR-AUC values. As discussed in subsection 6.6.3, this metric is particularly suitable for datasets with unbalanced classes, since it focuses on the quality of predictions related to the positive class (anomalies). However, analyzing its constituent components, **Precision** and **Recall**, a particular vulnerability emerges: *Precision* tends to degrade rapidly in the presence of a limited number of samples of the positive class (anomalies). This phenomenon occurs because, while effectively detecting true positives, the model may generate many false positives, resulting in a PR-AUC value that, in our case, does not exceed 0.9047.

In summary, the degradation of the PR-AUC score can be attributed to specific factors, as follows:

- **Low percentage of users with anomalies:** A small number of anomalous users implies a smaller number of positive samples, which increases the probability of false positives and degrades the *Accuracy*.
- **Limited duration of anomalies:** anomalies of short duration, such as hourly ones (2 hours in the experiments), are more difficult to detect, with a negative impact on the *Accuracy* and, consequently, on the PR-AUC score. On the contrary, anomalies of longer duration (*Half Day*) improve the overall model performance.
- **Federated Aggregation Algorithm:** *FedYogi* stands out for its ability to handle non-uniformly distributed anomalies, such as small increments or short time windows, achieving higher PR-AUC scores than *FedAvg*, which shows a higher vulnerability in these configurations.

Despite the limitations, the proposed framework achieved good performance, with both the *FedAvg* and *FedYogi* algorithms. The analysis of the results will

be further explored in the 6.6.6 section, where an ANOVA test was applied to validate the observations.

6.6.6 ANOVA Test

The Analysis of Variance (ANOVA) test is a robust statistical method that allows us to analyze and interpret the results obtained from the DoE. Specifically, the ANOVA test assesses the influence of individual factors and their interactions on the response variable, highlighting those with the most significant impact on the experiment's outcomes. We conducted the ANOVA test for this use case on the results presented in Table 6.2, aiming to identify the factors and interactions that significantly influence the response variable. A critical component of this analysis is the *p-value*, which measures the probability that the observed effects in the data occurred purely by chance, assuming the null hypothesis is true.

The null hypothesis in this context posits that the factor or interaction being tested has no effect on the response variable—that is, any observed variation is purely due to random chance or inherent variability in the data. Factors with a $p\text{-value} < 0.05$ were deemed statistically significant. This threshold of 0.05 indicates a 5% risk of incorrectly rejecting the null hypothesis, meaning there is strong evidence that the factor or interaction has a genuine impact on the response variable. Interactions up to the third degree were considered to ensure a comprehensive analysis. The table listing these factors and their respective *p-values* is provided in Figure 6.14.

Factors and Their Interactions	Log Valence	P-value
Time Bands*FL-Aggregation	1,728	0,01871
Time Bands*FL-Aggregation*Users with Anomaly	1,725	0,01884
FL-Aggregation*Users with Anomaly	1,664	0,02166 ^
FL-Aggregation*Consumption Increment*Users with Anomaly	1,663	0,02175
FL-Aggregation	1,661	0,02183 ^
FL-Aggregation*Consumption Increment	1,659	0,02192 ^
Time Bands*FL-Aggregation*Increment Duration	1,227	0,05932
Time Bands* Increment Duration*Users with Anomaly	1,200	0,06313
Users with Anomaly(5,20)	1,127	0,07468 ^
Consumption Increment*Users with Anomaly	1,120	0,07578 ^
Consumption Increment(5,15)	1,113	0,07701 ^
Time Bands	1,065	0,08604 ^
FL-Aggregation*Consumption Increment*Increment Duration	1,029	0,09347
FL-Aggregation*Increment Duration*Users with Anomaly	1,029	0,09362
Time Bands* Consumption Increment	0,971	0,10695
Consumption Increment*Increment Duration	0,963	0,10881 ^
Increment Duration* Users with Anomaly	0,698	0,20044 ^
Increment Duration	0,408	0,39042 ^
Time Bands* Consumption Increment*Users with Anomaly	0,307	0,49367

Figure 6.14. Summary of Effects

From Figure 6.14, it is evident that the most significant influence on the response variable comes from the interaction between the factors **Time Band*FL-Aggregation**, which has a *p-value* of 0.01871. Furthermore, the table highlights that **FL-Aggregation**, individually and in combination with other factors, plays a major role in influencing the DoE output. For a more nuanced assessment of the significance of the factors (or their interactions), the concept of **log-valence** is introduced as an auxiliary parameter. Specifically:

- A **low log-valence** (values close to or lower than zero) corresponds to a high *p-value*, indicating a limited statistical significance of the factor.
- A **high log-value** (positive values greater than zero) reflects a low *p-value*, which signals a strong statistical significance of the factor.

Observed Responses vs. Predicted Responses The plot in Figure 6.15 visually compares the observed (actual) values of the response variable with those predicted by the model, serving as a tool to assess the model's accuracy in fitting the data. The plot includes a diagonal line with a slope of a unit, representing the ideal scenario in which the predicted responses match the observed responses perfectly. The closer the data points are to this diagonal line, the more accurate the model's predictions. From Figure 6.15, it is evident that the data points generally align well with the diagonal, indicating a satisfactory accuracy level in the model's predictive capabilities.

The *Actual by Predicted Plot* 6.15 is often accompanied by performance metrics such as the Coefficient of Determination ($\mathbf{R^2}$) and Root Mean Square Error (RMSE) to provide a comprehensive evaluation of the model's quality. In particular the $\mathbf{R^2}$ is a metric that measures the percentage of the variance in the response variable Y that is explained by the factors X. The value obtained is $\mathbf{R^2 = 0.87}$, indicating that the model explains 87% of the variance, reflecting a good fit. However, the remaining 13% of the variance is unexplained, likely due to other factors or interactions not considered in the model. The RMSE, instead, metric quantifies the absolute error by representing the standard deviation of the residuals (the distance between predicted and observed values). Lower values suggest that the predictions are close to the observed data, while higher values indicate a greater deviation. The obtained value of RMSE, equal to 0.0066 is very close to zero, suggesting that the model fits the data well and produces accurate predictions.

Lack of Fit The lack of fit refers to the model's deviation from the observed data. It measures how much of the variation is not explained by the model,

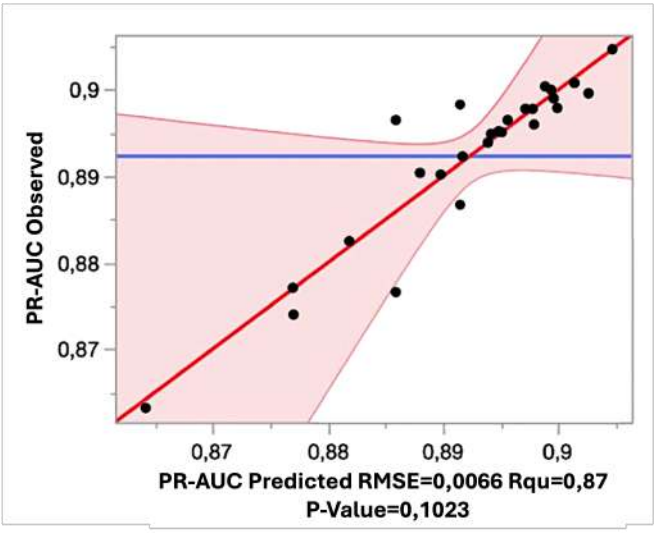

Figure 6.15. Actual by Predicted Plot

comparing this residual variance to the natural variability in the data (pure error). The results, shown in Table 6.4, suggest the following:

- **Origin:** This refers to the various sources of variance:
  - **Missing Estimation:** The variance in the data that the model cannot explain, potentially due to incorrect model specification or omitted factors.
  - **Pure Error:** Variance arising from natural randomness in the data, independent of the model or the factors analyzed.
  - **Total Error:** The total unexplained variance, combining both *missing estimation* and *pure error*.
- **DF (Degrees of Freedom):** Degrees of freedom represent the number of independent pieces of information available for estimating variance components. The DFise is calculated based on the number of parameters estimated in the model and the number of observations. In the context of ANOVA:
  - **DF for Origin:** This represents the degrees of freedom for each source of variation, such as missing estimation or pure error.

  - **Total DF:** The overall degrees of freedom, usually the total number of observations minus one.

- **Sum of Squares:** This measures the variance associated with each source of variation:
  - **Sum of Squares for Origin:** This represents the variation due to each source, such as the missing estimation or pure error.
  - **Total Sum of Squares:** The overall variance in the data before any model fitting.
  - **Error Sum of Squares:** The variance remaining after fitting the model, represents the unexplained variance.
- **Square Averages:** These represent the unexplained variance per degree of freedom and are calculated by dividing the sum of squares by the corresponding degrees of freedom. Low values of square averages indicate that the model explains most of the variance, leaving little residual variance.
- **F-ratio:** The **F-ratio** is a statistic used to compare the explained variance (due to the model and its factors) with the residual (unexplained) variance. It is calculated as:

$$\text{F-ratio} = \frac{\text{Mean Square for Factors}}{\text{Mean Square for Residuals}}$$

  A higher **F-ratio** suggests that the factors in the model explain a significant portion of the variance, making the model more effective. The obtained value is **F-ratio = 0.0631**, which indicates that the unexplained variance is small compared to the natural variability in the data.
- **Prob > F:** This value represents the p-value associated with the **F-ratio**, testing the null hypothesis that all model factors do not affect the response variable. A **high p-value** (typically greater than 0.05) suggests that the observed variation is likely due to random chance. The obtained value is **Prob > F = 0.9932**, indicating a high probability that the misestimation is due to chance, rather than a problem with the model.
- **Max R-Square:** This value indicates the theoretical maximum $\mathbf{R^2}$ that could be achieved by minimizing the pure error. The obtained value of **Max R-squared = 0.8919** is quite high and not far from the observed $\mathbf{R^2}$ value of 0.87, confirming that the model fits the data well.

It is important to note that a **low F-ratio** and a **high Prob > F** indicate there is no significant evidence of a misestimation problem, meaning the model's fit to the data is satisfactory.

Parameter Estimation				
Term	**Estimate**	**Std Error**	**T-Ratio**	**Prob> |t|**
Intercept	-225.5906	108.7861	-2.07	0.0768
Time Band[08:00-18:00]	1.3530196	0.677629	2.00	0.0860
FL-Aggregation[FedAvg]	-315.079	109.6826	-2.87	0.0239*
Consumption Increment[5,15]	-113.7127	54.88484	-2.07	0.0770
Increment Duration[2:00]	0.6508273	0.710973	0.92	0.3904
Users with Anomaly[5,20]	-138.2247	66.05189	-2.09	0.0747*
Time Band[08:00-18:00]*FL-Aggregation[FedAvg]	1.5657191	0.514163	3.05	0.0187*
FL-Aggregation[FedAvg]*Consumption Increment	-159.2332	55.38297	-2.88	0.0238*
FL-Aggregation[FedAvg]*Users with Anomaly	-191.8615	66.62591	-2.88	0.0237*
Consumption Increment*Increment Duration[2:00]	0.7047728	0.383642	1.84	0.1088*
FL-Aggregation[FedAvg]*Consumption Increment*Users with Anomaly	-96.96187	33.64122	-2.88	0.0236*
Time Band[08:00-18:00]*FL-Aggregation[FedAvg]*Increment Duration[2:00]	0.0146437	0.006512	2.25	0.0593
Time Band[08:00-18:00]*Consumption Increment*Users with Anomaly	0.1049321	0.157931	0.66	0.5277
Time Band[08:00-18:00]*FL-Aggregation[FedAvg]*Users with Anomaly	0.9416534	0.309721	3.04	0.0188*
Time Band[08:00-18:00]*Consumption Increment	0.8532871	0.46152	1.85	0.1069
Increment Duration[2:00]*Users with Anomaly	-0.45132	0.319326	-1.41	0.2004
Consumption Increment*Users with Anomaly	-69.40445	33.32397	-2.08	0.0758
Time Band[08:00-18:00]*Increment Duration[2:00]*Users with Anomaly	-0.008458	0.003845	-2.21	0.0631
FL-Aggregation[FedAvg]*Increment Duration[2:00]*Users with Anomaly	-0.445808	0.332672	-1.34	0.2221
FL-Aggregation[FedAvg]*Consumption Increment*Increment Duration[2:00]	0.3734889	0.278069	1.34	0.2211

Table 6.5. Parameter Estimation

Lack of Fit				
Origin	**DF**	**Sum of Squares**	**Square Averages**	**F-ratio**
Missing Estimation	5	0,00004184	8,367e-6	0,0631
Pure Error	2	0,00026528	0,000133	**Prob > F**
Total Error	7	0,00030712		0,9932
				Max R-quadro
				0,8919

Table 6.4. Lack of Fit

Parameter Estimation The table presented in Table 6.5 offers a detailed overview of how each factor, and their interactions, influence the response variable.

- **Estimation:** This column presents the coefficient estimate for each factor (or their interactions) in the scenario, indicating how much the response variable changes when the factor changes. Positive values suggest that an increase in the factor leads to an increase in the response variable. Negative values indicate that the response variable decreases as the factor increases.
- **Std Error:** The standard error represents the estimate precision. Smaller

values of standard error suggest that the model has made a more accurate estimate regarding the effect of that factor on the response variable.

- **T-ratio:** This statistic is used to determine whether the coefficient estimate for a given factor is significantly different from zero. Positive values suggest a positive influence of the factor on the response variable, while negative values indicate a negative effect. The greater the distance from zero, the more statistically significant the factor becomes.
- **Prob > |t|:** This column provides the p-value, which helps assess whether the factor's influence is statistically significant. Low p-values suggest strong evidence to reject the null hypothesis (that the factor has no effect), indicating the factor is relevant. Values marked in red and followed by an asterisk highlight which factors are statistically significant.

The results obtained allow us to conduct the following evaluation[3]:

i. The FL-Aggregation factor, concerning the FedAvg algorithm, has a negative coefficient (-315.079) and $p = 0.0239$. This indicates that using the *FedAvg* algorithm for aggregation decreases the PR-AUC metric. This result aligns with expectations, as the FedAvg algorithm is generally less stable and accurate than the optimized FedYogi algorithm.

ii. The interaction between FL-Aggregation (FedAvg) and Time Band (08:00-18:00) shows a positive estimate (1.5657) and $p = 0.0187$. This indicates that the performance of the FedAvg algorithm improves during daylight hours.

iii. The combination of FL-Aggregation (FedAvg) and Consumption Growth yields a negative estimate (-159.2332) and $p = 0.0238$. This suggests that the performance of the FedAvg algorithm worsens as the percentage of abnormal samples injected into the system increases.

iv. The interaction between FL-Aggregation (FedAvg) and Users with Anomaly shows a negative estimate (-191.8615) and $p = 0.0237$. This implies that the performance of the FedAvg algorithm declines as the percentage of anomalous users increases. This result supports the hypothesis that a higher abnormal user percentage leads to a deterioration in performance, which is particularly pronounced when using the FedAvg algorithm.

v. The combination of FL-Aggregation (FedAvg), Consumption Growth, and Users with Anomaly has a negative estimate (-96.96187) and $p = 0.0236$. This confirms the observations made in points iii. and iv.

[3]Factors are analyzed in the order they appear in the table.

vi. Lastly, the interaction between Time Band (08:00-18:00), FL-Aggregation (FedAvg), and Users with Anomaly shows a positive estimate (0.9417) and $\mathbf{p = 0.0188}$. This further supports the finding that the FedAvg algorithm performs better during daylight hours, even when the percentage of anomalous users varies.

Effects Test Looking at the last two columns in Table 6.6, we observe the following:

- **F-ratio:** This statistic compares the variance explained by the factor with the unexplained residual variance. A higher F-ratio indicates a greater impact of the factor on the response variable.
- **Prob > F:** Low values for this probability suggest that the factor is significant. Conversely, high values indicate that the factor is not significant and could be attributed to random variability.

Effects Test					
Source	**Nparm**	**DF**	**Sum of Squares**	**F-Ratio**	**Prob > F**
Time Band	1	1	0.00017492	3.9868	0.0860
FL-Aggregation	1	1	0.00036206	8.2521	0.0239*
Consumption Increment[5,15]	1	1	0.00018833	4.2925	0.0770
Increment Duration	1	1	0.00003677	0.8380	0.3904
Users with Anomaly[5,20]	1	1	0.00019214	4.3793	0.0747
Time Band*FL-Aggregation	1	1	0.00040685	9.2731	0.0187*
FL-Aggregation*Consumption Increment	1	1	0.00036268	8.2664	0.0238*
FL-Aggregation*Users with Anomaly	1	1	0.00036383	8.2926	0.0237*
Consumption Increment*Increment Duration	1	1	0.00014807	3.3748	0.1088
FL-Aggregation*Consumption Increment*Users with Anomaly	1	1	0.00036448	8.3073	0.0236*
Time Band*FL-Aggregation*Increment Duration	1	1	0.00022185	5.0564	0.0593
Time Band*Consumption Increment*Users with Anomaly	1	1	0.00001937	0.4414	0.5277
Time Band*FL-Aggregation*Users with Anomaly	1	1	0.00040556	9.2436	0.0188*
Time Band*Consumption Increment	1	1	0.00014998	3.4183	0.1069
Increment Duration*Users with Anomaly	1	1	0.00008764	1.9976	0.2004
Consumption Increment*Users with Anomaly	1	1	0.00019032	4.3377	0.0758
Time Band*Increment Duration*Users with Anomaly	1	1	0.00021360	4.8684	0.0631
FL-Aggregation*Increment Duration*Users with Anomaly	1	1	0.00007879	1.7958	0.2221
FL-Aggregation*Consumption Increment*Increment Duration	1	1	0.00007915	1.8041	0.2211

Table 6.6. Effects Test

The considerations discussed in 6.6.6 align with the results obtained from the effects test. Furthermore, it is worth noting that the most significant factors and their interactions demonstrate a high F-ratio in comparison to the factors that are not significant.

6.7 Final Observation

In conclusion, the proposed framework, which combines FL, K-Means Clustering, and VAE, represents an innovative and effective solution for anomaly detection in the context of smart grids. The model's ability to work with unlabelled data and its scalability makes it highly applicable in real-world scenarios. The combined approach of K-Means and VAE improves anomaly detection accuracy and significantly reduces false positive rates, thereby improving the system reliability. Furthermore, FL ensures greater data security and privacy compared to centralized solutions.

The results highlight the effectiveness of the analyzed federated aggregation algorithms, FedAvg and FedYogi, which recorded PR-AUC values ranging from 0.8632 to 0.9047. The observed performance differences between the two algorithms underline the ability of the framework to adapt to different operational contexts: FedYogi stands out for its stability under variable conditions, such as small consumption increments and a high percentage of users with anomalies, while FedAvg shows better results under less demanding conditions. In particular, both algorithms perform better at night when background noise is reduced, confirming the robustness of the solution in different scenarios. These results demonstrate the potential of the framework to provide a reliable and adaptable approach to monitoring and managing anomalies in modern energy systems.

Chapter 7

Conclusions

In recent years, the Internet of Things (IoT) has radically reshaped the technological landscape in various sectors, creating numerous opportunities for digital evolution and benefits in every aspect of daily life. However, this innovation has also introduced several significant challenges, especially regarding cybersecurity and environmental impact, which must necessarily be addressed because they can cause even serious damage. The intrinsically distributed, vulnerable, and heterogeneous nature of IoT systems amplifies these challenges, making it essential to develop innovative protection solutions that provide a complete balance of effectiveness, sustainability, and efficiency.

In this thesis, therefore, an in-depth investigation is conducted on these critical aspects, highlighting the limits of traditional security tools and the growing demand for security models that are not only scalable but also sustainable and privacy-oriented. In collaboration with the company Digital Platforms S.p.A., an in-depth comparative analysis of two Security Information and Event Management (SIEM) platforms, identified among the most adaptable to the new complex context of IoT, namely Real-Time Analytics (RTA) and LimaCharlie, is conducted. This analysis reveals their inadequacies in effectively responding to sophisticated threats, such as Distributed Denial of Service (DDoS) attacks, i.e. distributed attacks that undermine the availability of services, making them inaccessible to legitimate users, causing, in critical contexts, such as Critical Infrastructures, significant damage that can also affect people's lives. These findings underline the urgent need for advances in security technologies, in particular the integration of machine learning algorithms capable of significantly improving the accuracy of threat detection while effectively minimizing false positives and negatives, i.e. events incorrectly classified as malicious or benign, respectively.

In response to the identified challenges, this thesis proposes the development of a lightweight Intrusion Detection System (IDS) based on K-Means clustering methodology, developed in a railway context, in particular, this proposed solution, which has been limited in CPU and memory usage to reproduce the limitations of IoT devices, is tasked with detecting DoS attacks, appropriately simulated, in a realistic train onboard network. This innovative approach is designed to leverage the power of machine learning to deliver high performance in threat detection while ensuring minimal energy consumption. This alignment with sustainability principles is particularly vital in the context of Industry 5.0, where the emphasis is on creating intelligent systems that are not only effective but also environmentally friendly. By addressing these pressing issues, the proposed IDS can improve the overall security posture of IoT systems, ultimately contributing to a safer and more sustainable technological environment.

To ensure the validity of the proposed solutions, in this thesis we developed an IDS testbed, the DDoShield-IoT platform, which allows the simulation of realistic scenarios, the generation of representative datasets of the IoT context and test models in controlled environments, making a significant contribution to the evaluation of security solutions. Furthermore, the combination of real and synthetic data led to more balanced and complete datasets, improving the models' ability to detect zero-day attacks and demonstrating the importance of data augmentation to increase system resilience. A further step forward was taken with user profiling and federated learning integration, successfully addressing privacy and scalability issues and demonstrating the validity of decentralized approaches in real contexts, as shown by the results of the European CyberSEAS project. This work has consistently emphasized the importance of integrating security, sustainability, and efficiency. It aims to provide innovative solutions that not only address immediate technical challenges but also establish a basis for future development of security in IoT environments. The contributions developed, from the design of the sustainable system to the creation of advanced validation platforms, represent a significant step forward in IoT infrastructure protection, providing concrete tools to address increasingly complex and interconnected scenarios.

7.0.1 Main Findings

In summary, this dissertation provides multiple contributions to the domain of IoT security. They are schematized below:

- **Lightweight IDS for IoT:** A detection solution designed for environments with limited resources, achieving a balance between performance

and energy effectiveness.

- **Evaluation Testbed for IDS, DDoShield-IoT:** An IDS-testbed that produces realistic network traffic, enabling thorough assessment of IDS models. Its docker-based structure makes it suitable for simulating heterogeneous real-world contexts.
- **Data Augmentation Strategy and Federated Learning methodology:** We propose a new strategy of data augmentation to face the lack of realistic dataset problems and propose a distributed learning approach that facilitates intrusion detection while safeguarding user privacy.
- **User Profiling and Federated Learning for Anomaly Detection:** A technique for examining energy usage trends to enhance intrusion detection within Smart Grid systems.

7.0.2 Limitations and Threats to Validity

Although this research offers valuable insights, it also has certain limitations that must be recognized. For example, the success of FL depends on reliable and safe communication between devices and the main server. In situations with inconsistent network connectivity, performance could be impacted. Moreover, the assessment was mainly carried out in Smart Grid situations. Although the suggested method is versatile, further evaluations in various IoT settings (such as healthcare and industrial automation) would enhance its applicability.

7.0.3 Directions for Future Research

To enhance the efficiency and usability of IDS in IoT settings, various research avenues can be investigated:

- **Adaptive IDS Models:** Exploring reinforcement learning methods that allow IDS to independently adjust to emerging attack patterns instantly.
- **Energy-Conserving Security Solutions:** Creating model compression methods to minimize the computational and energy impact of ML-driven IDS systems.
- **Decentralized Federated Learning:** Investigating blockchain-enabled FL frameworks to enhance security and robustness against poisoning threats.
- **Cross-Domain Validation:** Evaluating the proposed framework across various IoT applications to confirm its efficacy outside of Smart Grid contexts.

7.0.4 Final Remarks

This thesis advances the development of lightweight, scalable, and privacy-protecting IDS solutions for IoT and IIoT environments. By combining unsupervised learning, federated learning, and user profiling, the suggested framework improves intrusion detection effectiveness while tackling critical issues concerning privacy and energy efficiency.

As IoT networks continue to grow, security measures must adapt. This study's approaches lay the foundation for more flexible, smart, and resource-efficient intrusion detection techniques, guaranteeing strong protection against new cyber threats in the constantly evolving IoT environment.

Bibliography

[1] iperf(1) - linux man page. https://linux.die.net/man/1/iperf. Accessed: 2024-11-07.

[2] Limacharlie docs.

[3] Zeek documentation.

[4] Github:cpulimit. https://github.com/opsengine/cpulimit, 2015. Accessed: 2024-11-07.

[5] Ghada Abdelmoumin, Danda B Rawat, and Abdul Rahman. On the performance of machine learning models for anomaly-based intelligent intrusion detection systems for the internet of things. *IEEE Internet of Things Journal*, 9(6):4280–4290, 2021.

[6] Oludare Isaac Abiodun, Esther Omolara Abiodun, Moatsum Alawida, Rami S Alkhawaldeh, and Humaira Arshad. A review on the security of the internet of things: Challenges and solutions. *Wireless Personal Communications*, 119:2603–2637, 2021.

[7] Amr Adel. Future of industry 5.0 in society: human-centric solutions, challenges and prospective research areas. *Journal of Cloud Computing*, 11(1):40, 2022.

[8] Erwin Adi, Adnan Anwar, Zubair A. Baig, and Sherali Zeadally. Machine learning and data analytics for the iot. *Neural Computing and Applications*, 32:16205 – 16233, 2020.

[9] Tanveer Ahmad, Huanxin Chen, Jiangyu Wang, and Yabin Guo. Review of various modeling techniques for the detection of electricity theft in smart grid environment. *Renewable and Sustainable Energy Reviews*, 82:2916–2933, 2018.

[10] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295, 2020.

[11] Muna Al-Hawawreh and Elena Sitnikova. Developing a security testbed for industrial internet of things. *IEEE Internet of Things Journal*, 8(7):5558–5573, 2020.

[12] Khalid Albulayhi, Qasem Abu Al-Haija, Suliman A Alsuhibany, Ananth A Jillepalli, Mohammad Ashrafuzzaman, and Frederick T Sheldon. Iot intrusion detection using machine learning with a novel high performing feature selection method. *Applied Sciences*, 12(10):5015, 2022.

[13] Mohammed Aledhari, Rehma Razzak, Reza M Parizi, and Fahad Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8:140699–140725, 2020.

[14] Abdullah Alsaedi, Nour Moustafa, Zahir Tari, Abdun Mahmood, and Adnan Anwar. Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems. *Ieee Access*, 8:165130–165150, 2020.

[15] Fatima Alwahedi, Alyazia Aldhaheri, Mohamed Amine Ferrag, Ammar Battah, and Norbert Tihanyi. Machine learning techniques for iot security: Current research and future vision with generative ai and large language models. *Internet of Things and Cyber-Physical Systems*, 2024.

[16] Eirini Anthi, Lowri Williams, Małgorzata Słowińska, George Theodorakopoulos, and Pete Burnap. A supervised intrusion detection system for smart home iot devices. *IEEE Internet of Things Journal*, 6(5):9042–9053, 2019.

[17] Sayeda Suaiba Anwar, Asaduzzaman, and Iqbal H. Sarker. A differential privacy aided deepfed intrusion detection system for iot applications. *SECURITY AND PRIVACY*, 2024.

[18] Joseph Bamidele Awotunde, Samarendra Nath Sur, Rasheed Gbenga Jimoh, Dayo Reuben Aremu, Dinh-Thuan Do, and Byung Moo Lee. Fl_giot: Federated learning enabled edge-based green internet of things system: A comprehensive survey. *IEEE Access*, 11:136150–136165, 2023.

[19] Zahedi Azam, Md Motaharul Islam, and Mohammad Nurul Huda. Comparative analysis of intrusion detection systems and machine learning based model analysis through decision tree. *IEEE Access*, 2023.

[20] Nikolaos Bakalos, Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Avi Ostfeld, Elad Salomons, Juan Caubet, Victor Jimenez, and Pau Li. Protecting water infrastructure from cyber and physical threats: Using multimodal data fusion and adaptive deep learning to monitor critical systems. *IEEE Signal Processing Magazine*, 36(2):36–48, 2019.

[21] Enrico Barbierato and Alice Gatti. Towards green ai. a methodological survey of the scientific literature. *IEEE Access*, 2024.

[22] Rabaie Benameur, Amine Dahane, Sami Souihi, and Abdelhamid Mellouk. A novel federated learning based intrusion detection system for iot networks. *ICC 2024 - IEEE International Conference on Communications*, pages 2402–2407, 2024.

[23] Jalal Bhayo, Syed Attique Shah, Sufian Hameed, Awais Ahmed, Jamal Nasir, and Dirk Draheim. Towards a machine learning-based framework for ddos attack detection in software-defined iot (sd-iot) networks. *Engineering Applications of Artificial Intelligence*, 123:106432, 2023.

[24] Kallista Bonawitz, Peter Kairouz, Brendan McMahan, and Daniel Ramage. Federated learning and privacy: Building privacy-preserving systems for machine learning and data science on decentralized data. *Queue*, 19(5):87–114, 2021.

[25] Tim M Booij, Irina Chiscop, Erik Meeuwissen, Nour Moustafa, and Frank TH Den Hartog. Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets. *IEEE Internet of Things Journal*, 9(1):485–496, 2021.

[26] Jason Brownlee. *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python*. Machine Learning Mastery, 2020.

[27] Anna L. Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.

[28] Riadh Ben Chaabene, Darine Ameyed, Fehmi Jaafar, Alexis Roger, Esma Aïmeur, and Mohamed Cheriet. A privacy-preserving federated learning for iot intrusion detection system. *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 351–356, 2023.

[29] Jiaao Chen, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. An empirical survey of data augmentation for limited data learning in nlp. *Transactions of the Association for Computational Linguistics*, 11:191–211, 2023.

[30] Chung Ming Cheung, Sanmukh Rao Kuppannagari, Rajgopal Kannan, and Viktor K Prasanna. Load demand user profiling in smart grids with distributed solar generation. In *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5. IEEE, 2020.

[31] Tek Raj Chhetri, Chinmaya Kumar Dehury, Blesson Varghese, Anna Fensel, Satish Narayana Srirama, and Rance J DeLong. Enabling privacy-aware interoperable and quality iot data sharing with context. *Future Generation Computer Systems*, 157:164–179, 2024.

[32] Lei Cui, Youyang Qu, Longxiang Gao, Gang Xie, and Shui Yu. Detecting false data attacks using machine learning techniques in smart grid: A survey. *Journal of Network and Computer Applications*, 170:102808, 2020.

[33] CyberSEAS. Cyber securing energy data services (cyberseas). https://cyberseas.eu, 2024. Progetto finanziato dall'Unione Europea nell'ambito del programma Horizon 2020, grant agreement n. 101020560, DOI=10.3030/101020560.

[34] Simona De Vivo and Pietro Liguori. Simulation environment for the evaluation of lightweight intrusion detection systems. In *2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 132–135. IEEE, 2023. © 2023 IEEE. Reprinted, with permission.

[35] Simona De Vivo, Islam Obaidat, Dong Dai, and Pietro Liguori. Ddoshield-iot: A testbed for simulating and lightweight detection of iot botnet ddos attacks. In *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 1–8. IEEE, 2024. © 2024 IEEE. Reprinted, with permission.

[36] Soma Shekara Sreenadh Reddy Depuru, Lingfeng Wang, and Vijay Devabhaktuni. Electricity theft: Overview, issues, prevention and a smart meter based approach to control theft. *Energy policy*, 39(2):1007–1015, 2011.

[37] L Dhanabal and SP Shantharajah. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. *International journal of advanced research in computer and communication engineering*, 4(6):446–452, 2015.

[38] Hongwei Ding, Yu Sun, Nana Huang, Zhidong Shen, and Xiaohui Cui. Tmg-gan: Generative adversarial networks-based imbalanced learning for network intrusion detection. *IEEE Transactions on Information Forensics and Security*, 19:1156–1167, 2023.

[39] Docker. Docker website. https://www.docker.com/, 2023. Accessed: 2024-11-07.

[40] Na Dong, Li Zhao, Chun-Ho Wu, and Jian-Fang Chang. Inception v3 based cervical cell classification combined with artificially extracted features. *Applied Soft Computing*, 93:106311, 2020.

[41] Benjamin Durakovic. Design of experiments application, concepts, examples: State of the art. *Periodicals of Engineering and Natural Sciences (PEN)*, 5(3), 2017.

[42] Sam Egbo. The 2016 dyn ddos cyber attack analysis: The attack that broke the internet for a day, 2018.

[43] Mohamed Faisal Elrawy, Ali Ismail Awad, and Hesham FA Hamed. Intrusion detection systems for iot-based smart environments: a survey. *Journal of Cloud Computing*, 7(1):1–20, 2018.

[44] Mojtaba Eskandari, Zaffar Haider Janjua, Massimo Vecchio, and Fabio Antonelli. Passban ids: An intelligent anomaly-based intrusion detection system for iot edge devices. *IEEE Internet of Things Journal*, 7(8):6882–6897, 2020.

[45] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, 2016.

[46] Damien Warren Fernando, Nikos Komninos, and Thomas Chen. A study on the evolution of ransomware detection using machine learning and deep learning techniques. *IoT*, 1(2):551–604, 2020.

[47] Mohamed Amine Ferrag, Othmane Friha, Djallel Hamouda, Leandros Maglaras, and Helge Janicke. Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning. *IEEE Access*, 10:40281–40306, 2022.

[48] Tomer Gafni, Nir Shlezinger, Kobi Cohen, Yonina C Eldar, and H Vincent Poor. Federated learning: A signal processing perspective. *IEEE Signal Processing Magazine*, 39(3):14–41, 2022.

[49] Tahani Gazdar. A new ids for smart home based on machine learning. In *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 393–400. IEEE, 2022.

[50] Yazeed Yasin Ghadi, Tehseen Mazhar, Syed Faisal Abbas Shah, Inayatul Haq, Wasim Ahmad, Khmaies Ouahada, and Habib Hamam. Integration of federated learning with iot for smart cities applications, challenges, and solutions. *PeerJ Computer Science*, 9:e1657, 2023.

[51] Mohammad Gharib, Edgar Weippl, Dieter Breitenbacher, and Wolfgang Kastner. Towards a realistic dataset for intrusion detection systems. *International Journal of Information Security*, 15:129–148, 2016.

[52] Gustavo González-Granadillo, Susana González-Zarzosa, and Rodrigo Diaz. Security information and event management (siem): analysis, trends, and usage in critical infrastructures. *Sensors*, 21(14):4759, 2021.

[53] Guezzaz, Azidine and Azrour, Mourade and Benkirane, Said, and Mohy-Eddine, Mouaad and Attou, Hanaa and Douiba, Maryam. A lightweight hybrid intrusion detection framework using machine learning for edge-based iiot security. *Int Arab J Inf Technol*, 19(5), 2022.

[54] Wei Guo, Zhiwei Yao, Yongfei Liu, Lanxue Zhang, Liangxiong Li, Tong Li, and Bingzhen Wu. A new federated learning model for host intrusion detection system under non-iid data. *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 494–500, 2023.

[55] Suzan Hajj, Rayane El Sibai, Jacques Bou Abdo, Jacques Demerjian, Abdallah Makhoul, and Christophe Guyeux. Anomaly-based intrusion detection systems: The requirements, methods, measurements, and datasets. *Transactions on Emerging Telecommunications Technologies*, 32(4):e4240, 2021.

[56] Najet Hamdi. Federated learning-based intrusion detection system for internet of things. *International Journal of Information Security*, 22:1937–1948, 2023.

[57] Hazman, Chaimae and Benkirane, Said and Guezzaz, Azidine and Azrour, Mourade and Abdedaime, Mohamed. Building an intelligent anomaly detection model with ensemble learning for iot-based smart cities. In *Advanced Technology for Smart Environment and Energy*, pages 287–299. Springer, 2023.

[58] Debiao He, Neeraj Kumar, Sherali Zeadally, Alexey Vinel, and Laurence T Yang. Efficient and privacy-preserving data aggregation scheme for smart grid against internal adversaries. *IEEE Transactions on Smart Grid*, 8(5):2411–2419, 2017.

[59] Mingshu He, Xiaojuan Wang, Peng Wei, Liu Yang, Yinglei Teng, and Renjian Lyu. Reinforcement learning meets network intrusion detection: a transferable and adaptable framework for anomaly behavior identification. *IEEE Transactions on Network and Service Management*, 2024.

[60] Hanan Hindy, David Brosset, Ethan Bayne, Amar Kumar Seeam, Christos Tachtatzis, Robert Atkinson, and Xavier Bellekens. A taxonomy of network

threats and the effect of current datasets on intrusion detection systems. *IEEE Access*, 8:104650–104675, 2020.

[61] Goodfellow I., Bengio Y., and A. (2016) Courville. *Deep Learning*. MIT Press, 2016.

[62] Idrissi Idriss, and Mostafa Azizi Mostafa, and Moussaoui Omar. A lightweight optimized deep learning-based host-intrusion detection system deployed on the edge for iot. *International Journal of Computing and Digital System*, 2021.

[63] Muhammad Ismail, Mostafa F Shaaban, Mahesh Naidu, and Erchin Serpedin. Deep learning detection of electricity theft cyber-attacks in renewable distributed generation. *IEEE Transactions on Smart Grid*, 11(4):3428–3437, 2020.

[64] Ghafar A Jaafar, Shahidan M Abdullah, and Saifuladli Ismail. Review of recent detection methods for http ddos attack. *Journal of Computer Networks and Communications*, 2019(1):1283472, 2019.

[65] Sana Ullah Jan, Saeed Ahmed, Vladimir Shakhov, and Insoo Koo. Toward a lightweight intrusion detection system for the internet of things. *IEEE access*, 7:42450–42471, 2019.

[66] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 21–26, 2016.

[67] James Jordon, Lukasz Szpruch, Florimond Houssiau, Mirko Bottarelli, Giovanni Cherubin, Carsten Maple, Samuel N Cohen, and Adrian Weller. Synthetic data–what, why and how? *arXiv preprint arXiv:2205.03257*, 2022.

[68] Nour Eldeen Khalifa, Mohamed Loey, and Seyedali Mirjalili. A comprehensive survey of recent trends in deep learning for digital images augmentation. *Artificial Intelligence Review*, 55(3):2351–2377, 2022.

[69] Shapla Khanam, Ismail Bin Ahmedy, Mohd Yamani Idna Idris, Mohamed Hisham Jaward, and Aznul Qalid Bin Md Sabri. A survey of security challenges, attacks taxonomy and advanced countermeasures in the internet of things. *IEEE access*, 8:219709–219743, 2020.

[70] Ansam Khraisat and Ammar Alazab. A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. *Cybersecurity*, 4:1–27, 2021.

[71] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[72] Nickolaos Koroniotis, Nour Moustafa, Francesco Schiliro, Praveen Gauravaram, and Helge Janicke. The sair-iiot cyber testbed as a service: A novel cybertwins architecture in iiot-based smart airports. *IEEE Transactions on Intelligent Transportation Systems*, 24(2):2368–2381, 2021.

[73] Ayush Kumar and Teng Joon Lim. Edima: Early detection of iot malware network activity using machine learning techniques. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 289–294. IEEE, 2019.

[74] Pooja Kumari and Ankit Kumar Jain. A comprehensive study of ddos attacks over iot network and their countermeasures. *Computers & Security*, 127:103096, 2023.

[75] Asif Ali Laghari, Kaishan Wu, Rashid Ali Laghari, Mureed Ali, and Abdullah Ayub Khan. A review and state of art of internet of things (iot). *Archives of Computational Methods in Engineering*, pages 1–19, 2021.

[76] Seo Jin Lee, Paul D Yoo, A Taufiq Asyhari, Yoonchan Jhi, Lounis Chermak, Chan Yeob Yeun, and Kamal Taha. Impact: Impersonation attack detection via edge computing using deep autoencoder and feature abstraction. *IEEE Access*, 8:65520–65529, 2020.

[77] Bohan Li, Yutai Hou, and Wanxiang Che. Data augmentation approaches in natural language processing: A survey. *Ai Open*, 3:71–90, 2022.

[78] Y. Li, S. Liu, Z. Hu, and L. Yu. Enhancing intrusion detection models using data augmentation techniques. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 3203–3210. IEEE, 2021.

[79] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[80] Chang Liu, Ruslan Antypenko, Iryna Sushko, and Oksana Zakharchenko. Intrusion detection system after data augmentation schemes based on the vae and cvae. *IEEE Transactions on Reliability*, 71(2):1000–1010, 2022.

[81] Jing Liu, Yang Xiao, Shuhui Li, Wei Liang, and CL Philip Chen. Cyber security and privacy issues in smart grids. *IEEE Communications surveys & tutorials*, 14(4):981–997, 2012.

[82] Manuel Lopez-Martin, Belén Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. Application of deep reinforcement learning to intrusion detection for supervised problems. *Expert Systems with Applications*, 141:112963, 2020.

[83] Rongxing Lu. *Privacy-enhancing aggregation techniques for smart grid communications*. Springer, 2016.

[84] Yingzhou Lu, Minjie Shen, Huazheng Wang, Xiao Wang, Capucine van Rechem, Tianfan Fu, and Wenqi Wei. Machine learning for synthetic data generation: a review. *arXiv preprint arXiv:2302.04062*, 2023.

[85] Tao Ma, Fen Wang, Jianjun Cheng, Yang Yu, and Xiaoyun Chen. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. *Sensors*, 16(10):1701, 2016.

[86] Kiran Maharana, Surajit Mondal, and Bhushankumar Nemade. A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, 3(1):91–99, 2022.

[87] Mohammad Saeid Mahdavinejad, Mohammadreza Rezvan, Mohammadamin Barekatain, Peyman Adibi, Payam Barnaghi, and Amit P Sheth. Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*, 4(3):161–175, 2018.

[88] Georgios Michail Makrakis, Constantinos Kolias, Georgios Kambourakis, Craig Rieger, and Jacob Benjamin. Industrial and critical infrastructure security: Technical analysis of real-life security incidents. *Ieee Access*, 9:165295–165325, 2021.

[89] Dardan Maraj, Marin Vuković, and Petar Hotovec. A survey on user profiling, data collection, and privacy issues of internet services. In *Telecom*, volume 5, pages 961–976. MDPI, 2024.

[90] Joel Margolis, Tae Tom Oh, Suyash Jadhav, Young Ho Kim, and Jeong Neyo Kim. An in-depth analysis of the mirai botnet. In *2017 International Conference on Software Security and Assurance (ICSSA)*, pages 6–12. IEEE, 2017.

[91] Tehseen Mazhar, Dhani Bux Talpur, Tamara Al Shloul, Yazeed Yasin Ghadi, Inayatul Haq, Inam Ullah, Khmaies Ouahada, and Habib Hamam. Analysis of iot security challenges and its solutions using artificial intelligence. *Brain Sciences*, 13(4):683, 2023.

[92] Patrick McDaniel and Stephen McLaughlin. Security and privacy challenges in the smart grid. *IEEE security & privacy*, 7(3):75–77, 2009.

[93] Francisco S Melícias, Tiago FR Ribeiro, Carlos Rabadão, Leonel Santos, and Rogério Luís de C Costa. Gpt and interpolation-based data augmentation for multiclass intrusion detection in iiot. *IEEE Access*, 2024.

[94] intel Michael A Pearce. Powertop primer. https://www.intel.com/content/www/us/en/developer/articles/tool/powertop-primer.html?wapkw=powertop, 2015. Accessed: 2024-11-07.

[95] Nivedita Mishra and Sharnil Pandya. Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. *IEEE Access*, 9:59353–59377, 2021.

[96] Viraaji Mothukuri, Prachi Khare, Reza M Parizi, Seyedamin Pouriyeh, Ali Dehghantanha, and Gautam Srivastava. Federated-learning-based anomaly detection for iot security attacks. *IEEE Internet of Things Journal*, 9(4):2545–2554, 2021.

[97] Nour Moustafa. A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. *Sustainable Cities and Society, Volume 72*, 2021.

[98] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[99] Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 dataset and the comparison with the kdd99 dataset. *Information Security Journal: A Global Perspective*, 25(1-3):18–31, 2019.

[100] Nour Moustafa, Benjamin Turnbull, and Kim-Kwang Raymond Choo. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, 6(3):4815–4830, 2018.

[101] Hichem Mrabet, Sana Belguith, Adeeb Alhomoud, and Abderrazak Jemai. A survey of iot security based on a layered architecture of sensing and data analysis. *Sensors*, 20(13):3625, 2020.

[102] Alhassan Mumuni and Fuseini Mumuni. Data augmentation: A comprehensive survey of modern approaches. *Array*, 16:100258, 2022.

[103] MG Sarwar Murshed, Christopher Murphy, Daqing Hou, Nazar Khan, Ganesh Ananthanarayanan, and Faraz Hussain. Machine learning at the network edge: A survey. *ACM Computing Surveys (CSUR)*, 54(8):1–37, 2021.

[104] Renya Nath N and Hiran V Nath. A generalized lightweight intrusion detection model with unified feature selection for internet of things networks. *International Journal of Network Management*, page e2291, 2024.

[105] Laisen Nie, Zhaolong Ning, Xiaojie Wang, Xiping Hu, Jun Cheng, and Yongkang Li. Data-driven intrusion detection for intelligent internet of vehicles: A deep convolutional neural network-based method. *IEEE Transactions on Network Science and Engineering*, 7(4):2219–2230, 2020.

[106] U.S. NIST. Guidelines for smart grid cyber security (vol. 1 to 3). http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7628, Aug. 2010. NIST IR-7628.

[107] Mudhafar Nuaimi, Lamia Chaari Fourati, and Bassem Ben Hamed. Intelligent approaches toward intrusion detection systems for industrial internet of things: A systematic comprehensive review. *Journal of Network and Computer Applications*, 215:103637, 2023.

[108] Islam Obaidat, Bennett Kahn, Fatemeh Tavakoli, and Meera Sridhar. Creating a large-scale memory error iot botnet using ns3dockeremulator. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 470–479. IEEE, 2023.

[109] Oleg Dulin. How-to: Configuring linux usage limits with docker and aws ecs. https://www.infoworld.com/article/3067303/how-to-configuring-linux-usage-limits-with-docker-and-aws-ecs.html, 2016. Accessed: 2024-11-07.

[110] Rafael Oliveira, Tiago Pedrosa, José Rufino, and Rui Pedro Lopes. Parameterization and performance analysis of a scalable, near real-time packet capturing platform. *Systems*, 12(4):126, 2024.

[111] open source. httperf. https://github.com/httperf/httperf, 2020. Accessed: 2024-11-07.

[112] Merve Ozkan-Okay, Refik Samet, Ömer Aslan, and Deepti Gupta. A comprehensive systematic literature review on intrusion detection systems. *IEEE Access*, 9:157727–157760, 2021.

[113] Ranjit Panigrahi and Samarjeet Borah. A detailed analysis of cicids2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology*, 7(3.24):479–482, 2018.

[114] Aakash Parmar, Rakesh Katariya, and Vatsal Patel. A review on random forest: An ensemble classifier. In *International conference on intelligent data communication technologies and internet of things (ICICI) 2018*, pages 758–763. Springer, 2019.

[115] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463, 1999.

[116] Jian Peng, Kim-Kwang Raymond Choo, and Helen Ashman. User profiling in intrusion detection: A review. *Journal of Network and Computer Applications*, 72:14–27, 2016.

[117] Kilian Pfeiffer, Martin Rapp, Ramin Khalili, and Jörg Henkel. Federated learning for computationally constrained heterogeneous devices: A survey. *ACM Computing Surveys*, 55(14s):1–27, 2023.

[118] Pavana Prakash, Jiahao Ding, Rui Chen, Xiaoqi Qin, Minglei Shu, Qimei Cui, Yuanxiong Guo, and Miao Pan. Iot device friendly and communication-efficient federated learning via joint model pruning and quantization. *IEEE Internet of Things Journal*, 9(15):13638–13650, 2022.

[119] Rajiv Punmiya and Sangho Choe. Energy theft detection using gradient boosting theft detector with feature engineering-based preprocessing. *IEEE Transactions on Smart Grid*, 10(2):2326–2329, 2019.

[120] Raja Gunasekaran, and Anbalagan Sudha, and Vijayaraghavan Geetha, and Dhanasekaran Priyanka, and Al-Otaibi Yasser D, and Bashir Ali Kashif. Energy-efficient end-to-end security for software-defined vehicular networks. *IEEE Transactions on Industrial Informatics*, 17(8):5730–5737, 2020.

[121] Slavica V Boštjančič Rakas, Mirjana D Stojanović, and Jasna D Marković-Petrović. A review of research work on network-based scada intrusion detection systems. *IEEE Access*, 8:93083–93108, 2020.

[122] Christian Rodolfo Esteve Rothenberg Ramon dos Reis Fontes. Mininet wifi website. https://mininet-wifi.github.io/, 2019. Accessed: 2024-11-07.

[123] Raspberry. Raspberry pi. https://www.raspberrypi.com/, 2023. Accessed: 2024-11-07.

[124] Markus Ring, Stefan Wunderlich, Daniel Scheuring, Dieter Landes, and Andreas Hotho. A survey of network-based intrusion detection data sets. *Computers & Security*, 86:147–167, 2019.

[125] Joseph R Rose, Matthew Swann, Gueltoum Bendiab, Stavros Shiaeles, and Nicholas Kolokotronis. Intrusion detection using network traffic profiling and machine learning for iot. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pages 409–415. IEEE, 2021.

[126] Xabier Sáez-de Cámara, Jose Luis Flores, Cristóbal Arellano, Aitor Urbieta, and Urko Zurutuza. Gotham testbed: a reproducible iot testbed for security experiments and dataset generation. *IEEE Transactions on Dependable and Secure Computing*, 21(1):186–203, 2023.

[127] Miraqa Safi, Sajjad Dadkhah, Farzaneh Shoeleh, Hassan Mahdikhani, Heather Molyneaux, and Ali A Ghorbani. A survey on iot profiling, fingerprinting, and identification. *ACM Transactions on Internet of Things*, 3(4):1–39, 2022.

[128] Adeb Salh, Razali Ngah, Lukman Audah, Kwang Soon Kim, Qazwan Abdullah, Yahya M Al-Moliki, Khaled A Aljaloud, and Hairul Nizam Talib. Energy-efficient federated learning with resource allocation for green iot edge intelligence in b5g. *IEEE Access*, 11:16353–16367, 2023.

[129] Leonel Santos, Ramiro Gonçalves, Carlos Rabadao, and José Martins. A flow-based intrusion detection framework for internet of things networks. *Cluster Computing*, pages 1–21, 2023.

[130] Scapy. Scapy website. https://scapy.net/, 2023. Accessed: 2024-11-07.

[131] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

[132] Kinza Shafique, Bilal A Khawaja, Farah Sabir, Sameer Qazi, and Muhammad Mustaqim. Internet of things (iot) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5g-iot scenarios. *Ieee Access*, 8:23022–23040, 2020.

[133] Faisal Shaman, Bogdan Ghita, Nathan Clarke, and Abdulrahman Alruban. User profiling based on application-level using network metadata. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–8. IEEE, 2019.

[134] Alireza Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374, 2012.

[135] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[136] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:60, 2019.

[137] Sabrina Sicari, Alessandra Rizzardi, and Alberto Coen-Porisini. 5g in the internet of things era: An overview on security and privacy challenges. *Computer Networks*, 179:107345, 2020.

[138] Paula Raissa Silva, João Vinagre, and João Gama. Towards federated learning: An overview of methods and applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2):e1486, 2023.

[139] Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727, 2020.

[140] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[141] Usman Tariq, Irfan Ahmed, Ali Kashif Bashir, and Kamran Shaukat. A critical cybersecurity analysis and future research directions for the internet of things: a comprehensive review. *Sensors*, 23(8):4117, 2023.

[142] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009.

[143] Chee-Wooi Ten, Koji Yamashita, Zhiyuan Yang, Athanasios V Vasilakos, and Andrew Ginter. Impact assessment of hypothesized cyberattacks on interconnected bulk power systems. *IEEE Transactions on Smart Grid*, 9(5):4405–4425, 2017.

[144] Ankit Thakkar and Ritika Lohiya. A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artificial Intelligence Review*, 55(1):453–563, 2022.

[145] Geethapriya Thamilarasu, Adedayo Odesile, and Andrew Hoang. An intrusion detection system for internet of medical things. *IEEE Access*, 8:181560–181576, 2020.

[146] Philip Treleaven, Malgorzata Smietanka, and Hirsh Pithadia. Federated learning: The pioneering distributed machine learning and privacy-preserving data technology. *Computer*, 55(4):20–29, 2022.

[147] Thavavel Vaiyapuri, Shabbab Algamdi, Rajan John, Zohra Sbai, Munira Al-Helal, Ahmed Alkhayyat, and Deepak Gupta. Metaheuristics with federated learning enabled intrusion detection system in internet of things environment. *Expert Systems*, 40(5):e13138, 2023.

[148] Sampath Kumar Venkatachary, Jagdish Prasad, Annamalai Alagappan, Leo John Baptist Andrews, Raymon Antony Raj, and Sarathkumar Duraisamy. Cybersecurity and cyber-terrorism challenges to energy-related

infrastructures-cybersecurity frameworks and economics–comprehensive review. *International Journal of Critical Infrastructure Protection*, page 100677, 2024.

[149] Roberto Verdecchia, June Sallou, and Luís Cruz. A systematic review of green ai. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(4):e1507, 2023.

[150] Sandra Wachter. Normative challenges of identification in the internet of things: Privacy, profiling, discrimination, and the gdpr. *Computer law & security review*, 34(3):436–449, 2018.

[151] Haoxiang Wang, Jiasheng Zhang, Chenbei Lu, and Chenye Wu. Privacy preserving in non-intrusive load monitoring: A differential privacy perspective. *IEEE Transactions on Smart Grid*, 12(3):2529–2543, 2020.

[152] Jing Wang, Libing Wu, Sherali Zeadally, Muhammad Khurram Khan, and Debiao He. Privacy-preserving data aggregation against malicious data mining attack for iot-enabled smart grid. *ACM Transactions on Sensor Networks (TOSN)*, 17(3):1–25, 2021.

[153] Mi Wen, Rong Xie, Kejie Lu, Liangliang Wang, and Kai Zhang. Feddetect: A novel privacy-preserving federated learning framework for energy theft detection in smart grid. *IEEE Internet of Things Journal*, 9(8):6069–6080, 2021.

[154] Jiajun Wu, Fan Dong, Henry Leung, Zhuangdi Zhu, Jiayu Zhou, and Steve Drew. Topology-aware federated learning in edge computing: A comprehensive survey. *ACM Computing Surveys*, 56(10):1–41, 2024.

[155] Xiaofang Xia, Yang Xiao, and Wei Liang. Sai: A suspicion assessment-based inspection algorithm to detect malicious users in smart grid. *IEEE Transactions on Information Forensics and Security*, 15:361–374, 2019.

[156] Xun Xu, Yuqian Lu, Birgit Vogel-Heuser, and Lihui Wang. Industry 4.0 and industry 5.0—inception, conception and perception. *Journal of manufacturing systems*, 61:530–535, 2021.

[157] Tan Yigitcanlar, Rashid Mehmood, and Juan M Corchado. Green artificial intelligence: Towards an efficient, sustainable and equitable technology for smart cities and futures. *Sustainability*, 13(16):8952, 2021.

[158] Shen Yin, Juan J Rodriguez-Andina, and Yuchen Jiang. Real-time monitoring and control of industrial cyberphysical systems: With integrated plant-wide monitoring and control framework. *IEEE Industrial Electronics Magazine*, 13(4):38–47, 2019.

[159] Georgios Zachos, Georgios Mantas, Ismael Essop, Kyriakos Porfyrakis, Joaquim Manuel CS Bastos, and Jonathan Rodriguez. An iot/iomt security testbed for anomaly-based intrusion detection systems. In *2023 IFIP Networking Conference (IFIP Networking)*, pages 1–6. IEEE, 2023.

[160] Sherali Zeadally, Al-Sakib Khan Pathan, Cristina Alcaraz, and Mohamad Badra. Towards privacy protection in smart grid. *Wireless personal communications*, 73:23–50, 2013.

[161] Xiaolu Zhang, Oren Upton, Nicole Lang Beebe, and Kim-Kwang Raymond Choo. Iot botnet forensics: A comprehensive digital forensic case study on mirai botnet servers. *Forensic Science International: Digital Investigation*, 32:300926, 2020.

[162] Xiaokang Zhou, Yiyong Hu, Jiayi Wu, Wei Liang, Jianhua Ma, and Qun Jin. Distribution bias aware collaborative generative adversarial network for imbalanced deep learning in industrial iot. *IEEE Transactions on Industrial Informatics*, 19(1):570–580, 2022.

[163] Chunsheng Zhu, Victor CM Leung, Lei Shu, and Edith C-H Ngai. Green internet of things for smart world. *IEEE access*, 3:2151–2162, 2015.

[164] Qinsong Zhu, Bintao Sun, Yuqing Zhou, Weifang Sun, and Jiawei Xiang. Sample augmentation for intelligent milling tool wear condition monitoring using numerical simulation and generative adversarial network. *IEEE Transactions on Instrumentation and Measurement*, 70:1–10, 2021.

[165] Sha Zhu, Kaoru Ota, and Mianxiong Dong. Green ai for iiot: Energy efficient intelligent edge computing for industrial internet of things. *IEEE Transactions on Green Communications and Networking*, 6(1):79–88, 2021.

[166] Alexander Ziller, Andrew Trask, Antonio Lopardo, Benjamin Szymkow, Bobby Wagner, Emma Bluemke, Jean-Mickael Nounahon, Jonathan Passerat-Palmbach, Kritika Prakash, Nick Rose, et al. Pysyft: A library for easy federated learning. *Federated Learning Systems: Towards Next-Generation AI*, pages 111–139, 2021.

[167] Maede Zolanvari, Marcio A Teixeira, Lav Gupta, Khaled M Khan, and Raj Jain. Machine learning-based network vulnerability analysis of industrial internet of things. *IEEE Internet of Things Journal*, 6(4):6822–6834, 2019.

Author's publications

1. Pietro Liguori, Cristina Improta, Simona De Vivo, Roberto Natella, Bojan Cukic, Domenico Cotroneo,

   Can NMT understand me? towards perturbation-based evaluation of NMT models for code generation,

   1st International Workshop on Natural Language-based Software Engineering, (NLBSE)

   Pittsburgh, Pennsylvania, May 2022, pp. 59-66, ACM,
   DOI: 10.1145/3528588.3528653

2. Alessandra Rizzardi, Raffaele Della Corte, Jesús F. Cevallos M., Vittorio Orbinato, Simona De Vivo, Sabrina Sicari,

   RaiIRED: a Node-RED-Based Framework for Modeling Train Control Management Systems,

   2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob),

   Paris, France, 2024, October 2024, pp. 671-674, IEEE,
   DOI: 10.1109/WiMob61911.2024.10770345.

3. Simona De Vivo, Islam Obaidat, Dong Dai, Pietro Liguori,

   DDoShield-IoT: A Testbed for Simulating and Lightweight Detection of IoT Botnet DDoS Attacks,

   54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W),

   Brisbane, Australia, June 2024, pp. 1-8, IEEE,
   DOI: 10.1109/DSN-W60302.2024.00014.

4. Simona De Vivo, Pietro Liguori,

Simulation Environment for the Evaluation of Lightweight Intrusion Detection Systems

34th International Symposium on Software Reliability Engineering Workshops (ISSREW)

Florence, Italy, October 2023, pp. 132-135, IEEE,
DOI: 10.1109/ISSREW60843.2023.00061.