Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
SAAD DAHLAB UNIVERSITY OF BLIDA 1
Faculty of Sciences
Department of Computer Science

FINAL YEAR PROJECT REPORT

Speciality : Computer Systems and Network

Presented by :

Houari Amel

THEME

Federated Learning For Distributed Intrusion Detection Systems

Mme.	Arkam Meriem	MAA	USDB	Thesis supervisor
Dr.	Remmide Mohamed Abdelkarim		USDB	Thesis co-supervisor

Acknowledgements

First and foremost, we express our profound gratitude to Almighty Allah for granting me the strength and patience to complete this humble project. His guidance and blessings have been a constant source of support throughout this challenging journey.

We would like to convey our deepest appreciation to our supervisor, Mme. Arkam Meriem, and co-supervisor, Mr. Remmide Mohamed Abdelkarim, for their continuous support , professionalism and tireless guidance. Their valuable advices during the entire project have been enhancing the quality of this thesis.

Our heartfelt thanks go to the members of the jury for their interest in our work and for taking the time to review it with care and attention.

We are truly grateful to all our professors whose knowledge and dedication have guided us throughout our studies.

To our family and friends,we would like to thank them for their unconditional love , constant encouragement, support, and prayers that helped us overcome every challenge.

Finally, we extend our thanks to everyone who contributed, directly or indirectly, to the completion of our project.

Abstract

The rapid growth of the internet in recent years has made cybersecurity a significant challenge. The traditional and standard Intrusion Detection Systems (IDS) which work based on known attack patterns are not effective enough and not sufficient to detect modern threats nowadays. For this reason, in this project, we aimed to enhance the functionality of IDS using either Machine Learning (ML) or Deep Learning (DL) to detect attacks.To reach our goal, we compared several models to decide which one is the best and gives best performance .However, to ensure that individuals' data stay safe, we adopted Federated Learning (FL), which enables the model to learn from different distributed data sources and devices without sharing private data. We evaluated our work using a real-world dataset UNSW-NB15 , we implemented both a Federated MLP and a Federated Random Forest (RF) that returned best results among Ml and DL algorithms, using different aggregation strategies. Our final federated MLP model achieved over 98% across accuracy, precision, recall, and F1-score, proving that federated deep learning can deliver state-of-the-art results while preserving data confidentiality.

Keywords: Cybersecurity, Intrusion Detection System (IDS), Machine Learning (ML), Deep Learning (DL), Federated Learning (FL), Multi-Layer Perceptron (MLP), Random Forest (RF).

Résumé

La croissance rapide d'Internet ces dernières années a rendu de la cybersécurité un défi majeur. Les systèmes de détection d'intrusion (IDS) traditionnels et standard, qui fonctionnent sur la base de schémas d'attaque connus, ne sont pas suffisamment efficaces pour détecter les menaces modernes. C'est pourquoi, dans ce projet, nous avons cherché à améliorer les fonctionnalités des IDS en utilisant soit l'apprentissage automatique (ML), soit l'apprentissage profond (DL) pour détecter les attaques. Pour atteindre cet objectif, nous avons comparé plusieurs modèles afin de déterminer celui qui est le plus performant.Cependant, afin de garantir la confidentialité des données des utilisateurs, nous avons adopté l'apprentissage fédéré (FL), qui permet au modèle d'apprendre à partir de sources de données et d'appareils distribués sans partager les données sensibles. Nous avons évalué notre travail à l'aide du jeu de données réel UNSW-NB15. Nous avons implémenté à la fois un MLP fédéré et une forêt aléatoire (RF) fédérée, qui ont montré les meilleurs résultats parmi les algorithmes ML et DL, en utilisant différentes stratégies d'agrégation. Notre modèle MLP fédéré final a atteint plus de 98% en exactitude, précision, rappel et F1-score, prouvant que l'apprentissage profond fédéré peut fournir des résultats de pointe tout en préservant la confidentialité des données.

Mots-clés : Cybersécurité, Système de détection d'intrusion (IDS), Apprentissage automatique (ML), Apprentissage profond (DL), Apprentissage fédéré (FL), Perceptron multicouche (MLP), Forêt aléatoire (RF).

P l

Tm\ s .t˝ryb2 ˝2 z trbymrt a t a ry t ãtwnmrt 9 rt r g rmrt wmnrt 3` Tr2`9 z` r ,T9 r`mrt ã2m hrt a2m s ,l2 zmt` trt ,(IDS) T zylqtrt 3lmtrt u2 z z` ,r! 2ny`m ,ç rumrt t@œ 9 , r@r .Ti z rt ãtz zhtrt 2 uzlr T 2fzrt ay9 2m ė2ut2 (DL) ym`rt l`trt s (ML) r t l`trt tz tm2 3lmtrt u2 Tm\ s 2Z uyc a 3ȧ9 t iwmnrt ,2yt i2m z2 T ,2qm 2nm9 ,2n9zœ ,r! 3w}wlr .ã2m hrt ,(FL) rt,zyfrt l`trt 2 zmt2t ,Tnaĩ yaz tmmrt ã2 2y 2q 2mȧr , ri ga . t2 t ã2 2ybrt T2,2ua 2 T2Ewa zh s ã2 2y ,22aa a l`tr2 iwmnlr ,mm @rt 2nm9 uyc ,UNSW-NB15 Ty`9twrt ã2 2ybrt T2wm a tz tm2 2nlm2 yyqt 2nm9 .Tm2m rt ´@lrt , rt,zyfrt RF Ty twu`rt T 2 rt iwm rt,zyfrt MLP iwm a 32 @yfnt ã2y y trtmt tz tm2 , ym`rt l`trt r t l`trt ã2yaE,tw y 2tnrt 3ȧ9s trhZs ,y 2qmrt 9 %89 E 2 t T92 2hnrt rt,zyfrt MLP iwm qc z9 .Tflt a gym ,229 rt,zyfrt ym`rt l`trt s bi 2ma ,F1-Score _rt ç2 rtm t T9zrt 3ia Tym2m t .ã2 2ybrt Ty}wa ,l2 _2f rt ga wtmmrt Tyr22 2t zq ,l2

l`trt ,(ML) r t l`trt ,(IDS) 3lmtrt u2 2\ , trbymrt a t :ãysGtfqt GqlKt T 2 rt ,(MLP) ã2qbbrt 2z`ta m rt t,2 t ,(FL) rt,zyfrt l`trt ,(DL) ym`rt .(RF) Ty twu`rt

Contents

List of Figures

List of Tables

General Introduction

Today in our world, it's very important to make sure that our computers and on-line systems highly safe because of bad actors who try to sneak in and cause troubles. These threats are becoming more advanced and dangerous over time due to the rapid development of internet tools.

To help protect it, experts use many and different special tools that monitor network and computers and look for any signs of danger.We mention one of them which is called Intrusion Detection Systems (IDS) . However, older versions of these systems sometimes fail to catch new types of attacks they haven't encountered before.

This is where Machine Learning (ML) and Deep Learning (DL) comes in ,they offer intelligent ways that help detect unusual activities in network especially in our case that might be harmful. But using them also comes with challenges, such as concerns about protecting people's private information and the difficulty in understanding how the system makes its decisions.

We will cite in table 1 some real-word attacks statistics in many different domains according to this article [Martin,] that was last updated on June 6, 2025 :

Metric	Value
Daily cyberattacks	600 million per day
Cybercrime victims	4.6 million victims per day (54 per second)
Malware attacks (2024)	6.54 billion (up 8% from 2023)
Ransomware attacks (2023)	317.59 million globally
Average ransomware payment	$850,700
Total cost of ransomware attack	$4.91 million (includes downtime and recovery)
Business email compromise losses	Over $6.3 billion transferred (2024)
IoT attacks growth	124% increase (YoY); average cost per incident: $330,000
Phishing	Nearly 1 million phishing websites created per month (Q4 2024)
Cryptojacking attacks (Q1 previously)	332.3 million in a single quarter
Zero-day attacks (2023)	80 exploited vulnerabilities (record high)
Data breaches (Verizon 2025)	12,195 breaches recorded
Healthcare breach victims	Over 198 million individuals affected in 2024
Cybersecurity job gap	3.4 million positions unfilled worldwide
Average data breach cost	$4.45M (avg.); insider breach cost: $4.99M
Cybersecurity spending per employee	$52.16 per employee (2024)

Table 1: Cybersecurity attacks statistics in (2024–2025)

Below, we mention one of the most known attacks that happened in real world :

WANNACRY: According to the Center for Internet Security (2018) [Omar et al., 2019], this began in May 2017 and persisted for several months. This global ransomware assault encrypted data on Microsoft Windows Operating Systems (OS) and demanded Bitcoin cryptocurrency ransom payments in exchange. When the attack was carried out, it would look for and encrypt particular filename extensions on Windows 7 and Server 2008 clients. Additionally, it would append a text file with the ransom of around $300 and a payment deadline to each encrypted file.

Thesis objectives

The goal of this project is to design a smart and secure intrusion detection model that not only detects threats accurately but also preserves users privacy .To make sure the privacy is respected, We are going to use Federated Learning (FL) which allows different devices or clients to work together without the need of sharing their private data.More specifically, this project aims to:

- Minimize the number of false alerts, so security teams can focus on real threats without being distracted by unnecessary warnings.
- Explore and compare different techniques to find the most effective way to detect

unusual or harmful behaviour in a network.

- Test and evaluate the model using existing real-world dataset (UNSW-NB15), making sure it works well in performance.

Thesis contributions

As we mentioned earlier in our objectives that we are focusing on building an intrusion detection system model by integrating federated learning while using either ML or DL techniques based on which one has the best performance, the main contributions of our work are:

- **Building a privacy-aware IDS using Federated Learning**: We developed a distributed system where several devices can together train a model without sharing any data .
- **Working with real-world data**: We used the UNSW-NB15 dataset, which includes various real attack types, to train and evaluate the models. This gave us a good idea of how the system would perform in realistic conditions.
- **Comparing different models**: We compared several algorithms (such as Random Forest, Gradient Boosting, GRU, LSTM, and MLP) to see which one performs best on a real dataset (UNSW-NB15). The Keras MLP turned out to be the most accurate.
- **Using deep learning for better detection**: We used a deep learning model which is **Keras MLP** and tested its ability to detect attacks in network traffic. The results showed that it offers strong performance that achieved around 98% for intrusion detection tasks.

Problem Statement

The performance of intrusion detection systems (IDS) has been the subject of numerous research projects .However, many of these methods still have trouble addressing fundamental issues like high false positive rates and zero-day assaults.

For instance, Kus et al. [Kus et al., 2022] evaluated industrial ML-based IDS and found that while performance on known attack patterns was strong, detection rates for zero-day attacks dropped significantly—down to 3.2% in some cases. This reveals a critical blind spot in modern IDS models.

Similarly, a comprehensive benchmark study by Wolsing et al. [Wolsing et al., 2023] highlighted the lack of generalizability in IDS. Models that performed well on one dataset often failed drastically on another, with F1-scores sometimes falling from 0.95 to as low as 0.02. This inconsistency underscores the challenge of real-world deployment.

Chergui et al. [Chergui et al., 2020] introduced a context-aware IDS (NOC-IDS) based on ontological filtering to reduce false alerts. While their system showed a notable reduction in false positives, it still required significant expert configuration and lacked scalability for large networks.

These examples show that even with major improvements, IDS solutions continue to have issues with accuracy, flexibility, and operational viability, particularly in dynamic and privacy-sensitive environments. This supports the need for an IDS paradigm that is more intelligent, federated, and resilient, as this work explores.

Thesis organization

This work is structured in 3 main chapters as follows:

- **Chapter 01: Background and Theoretical Framework**
  In this chapter, we introduce the concepts and technologies related to Intrusion Detection Systems (IDS), including the different architectures and types ,their advantages and disadvantages,in addition to that we mention the role of Machine Learning (ML) and Deep Learning (DL) in modern cybersecurity.Moreover we explain Federated Learning (FL) principles and their importance in building privacy-preserving IDS solutions.

- **Chapter 02: Proposed Approach**
  The design of the proposed solution is presented in this chapter that focuses on the development part. We describe the choice and preparation of the dataset (UNSW-NB15), the implementation of different models, and the setup of the Federated Learning framework.

- **Chapter 03: Experimentation and Results**
  In the final chapter, we present and detail the experimental process steps and evaluate each of proposed models performance. Then we compare between different algorithms using metrics in order to choose the strongest one and develop it by integrating Federated Learning.Finally, we present visualizations to support our findings.

Chapter 1

Background and Theoretical Framework

1.1 Introduction

In today's digital world, cybersecurity threats are increasing rapidly, making Intrusion Detection Systems (IDS) a crucial component of network and system security. An IDS helps monitor, analyse, and detect malicious activities, ensuring that potential threats are identified before they cause harm. We chose to work with IDS because of its importance in modern cybersecurity. With the rise of cyberattacks such as hacking, malware infections, and unauthorized access, organizations need a strong defence mechanism. IDS plays a key role in identifying suspicious behaviour and helps administrators take immediate actions. Another reason for our interest in IDS is its ability to use advanced techniques such as anomaly detection and machine learning to recognize threats, even if they are unknown. This proactive approach makes IDS a powerful tool in protecting networks and systems. Furthermore, working with IDS allows us to explore various cybersecurity concepts, gain hands-on experience, and contribute to building more secure digital environments. By studying IDS, our aim is to understand how cyber threats are detected and mitigated, making it a valuable field for our knowledge and future projects in cybersecurity.

1.2 Intrusion Detection Systems (IDS)

In this section we are going to explain how Intrusion Detection Systems (IDS) are essential tools in cybersecurity , it is necessary to first define what an IDS is and explore its core components and functions.

1.2.1 Intrusion Detection Systems Definition

An Intrusion Detection System (IDS) [Bace et al., 2001] is a software device and cybersecurity solution that continuously monitors, inspects and analyses network traffic or system activities to identify unauthorized access, security breaches, potential attacks, suspicious activities and malicious behaviour. It acts as a defence layer, helping organizations detect, log, and respond to potential cyber threats before they cause serious damage such as cyberattacks, policy violations, and malicious activities, meaning that if an IDS detects a threat, it alerts the system or network administrator to make the necessary actions.

IDS does not actively prevent attacks. It acts as an early warning system, providing detailed reports on security incidents for further investigation.

So we can summarise that IDS has 4 steps while working:

- First ,it monitors traffic on a computer network to detect suspicious activity.
- then , it analyses the data flowing through the network to look for patterns and signs of abnormal behaviour.
- After that, IDS compares the network activity with a set of predefined rules and patterns to identify any activity that might indicate an attack or intrusion.
- Finally, it alerts the administrators immediately with all the possible attacks.

1.2.2 The importance of Intrusion Detection Systems in Cybersecurity

Intrusion Detection Systems (IDS) are essential to current cybersecurity [Sundaram, 1996] because it is nearly impossible to provide comprehensive system security because of software defects, cryptography restrictions, insider threats, and usability constraints. IDSs act as reactive mechanisms that track and examine system activity to find indications of malicious activity, unauthorized access, or policy violations rather than preventing assaults. They play a crucial role in both real-time intrusion detection and post-attack forensic analysis.

1.2.3 Intrusion Detection Systems architectures

Before exploring into how intrusion detection systems identify malicious activity, it is essential to understand their basic structures [Liao et al., 2013] because they influence its capabilities, deployment strategy, and overall effectiveness. In the following section, we will explore the main architectures of IDS, their roles, advantages, and limitations in protecting different computing environments.

1.2.3.1 Distributed Intrusion Detection Systems

An innovative method for intrusion detection systems (IDS) is the DIDS architecture (Distributed Intrusion Detection Systems) [Snapp et al., 1991] , which blends centralized data analysis with distributed monitoring and data reduction. A DIDS director, one host monitor for each host, and one LAN monitor for every broadcast LAN segment in the network under observation make up this system. The DIDS director assesses the evidence of unauthorized or suspicious behaviour that is gathered by the host and LAN monitors. The DIDS director receives reports from the host and LAN monitors asynchronously and independently over a communications infrastructure.

Advantages and Limitations of DIDS: The Distributed Intrusion Detection System (DIDS) offers both notable strengths and certain limitations [Zarringhalami and Rafsanjani, 2012] based on its architectural design and operational goals [Snapp et al., 1991].

Advantages

- **Distributed Monitoring:** DIDS deploys both host-level and LAN-level monitors, allowing comprehensive and scalable intrusion detection across different points in the network.
- **Centralized Analysis:** The DIDS Director aggregates data from distributed agents, enabling the system to detect complex and coordinated intrusions that might be missed otherwise.
- **Efficient Data Reduction:** Each monitoring agent performs local filtering and summarization of audit data before transmission, reducing network overhead and improving processing efficiency.
- **Cross-System User Tracking:** DIDS can monitor user activity across multiple systems, even when multiple login names are used, enhancing accountability and traceability.

Limitations

- **Architectural Complexity:** The distributed and centralized hybrid architecture adds complexity in terms of configuration, communication, and synchronization of components.
- **Single Point of Failure:** The DIDS Director represents a central point in the system—if compromised or fails, it can degrade the entire detection capability.

- **Network Overhead:** Although local data reduction is applied, transmitting processed alerts and summaries still contributes to network load, which may impact performance in high-traffic environments.

1.2.3.2 Network-based Detection Systems architecture

We have also another architecture called Network Intrusion Detection Systems (NIDS) [Li et al., 2019], which is deployed at strategic points within a network to monitor and analyse traffic in real-time or near real-time, aiming to detect malicious activity. It inspects packets traversing the network by analysing headers and payloads across the IP, transport, and application layers, allowing detection of various attacks such as TCP SYN floods and fragmented packet attacks. NIDS uses either signature-based methods, comparing patterns to a database of known threats, or anomaly-based detection to identify unusual behaviour. However, NIDS is limited in its ability to inspect encrypted traffic or activities occurring within hosts. Tools like Snort and NetSTAT exemplify NIDS implementations. In machine learning-based NIDS approaches, intrusion detection involves three phases: pre-processing of traffic data, training to build behaviour models, and detection, where current traffic is compared against learned patterns to trigger alerts when threats are identified.

Advantages and Limitations of NIDS: The Network Intrusion Detection System (NIDS) has its advantages and limitations [Li et al., 2019] based on its architectural design .

Advantages

- **Real-time Monitoring:** NIDS provides continuous, real-time monitoring of network traffic, enabling early detection of threats and suspicious activities.
- **Non-intrusive Operation:** As passive sensors, NIDS operate without interfering with normal network functions, making them difficult for attackers to detect.
- **Wide Coverage:** NIDS can analyse traffic from multiple hosts simultaneously, offering broad visibility into network behaviour and helping to identify misconfiguration or misuse.
- **Complementary Use:** NIDS complements other security systems such as firewalls and antivirus tools by identifying threats those systems may miss.

Limitations

- **Encrypted Traffic Blindness:** NIDS generally cannot analyse the contents of encrypted network packets, limiting their effectiveness in modern secure communications.
- **High False Alarm Rate:** NIDS can produce a large number of false positives due to benign anomalies or misconfigured detection rules, which may overwhelm security analysts.
- **Dependence on Signature Updates:** Signature-based NIDS may fail to detect new or unknown attacks until updated, creating vulnerabilities during that window.
- **Limited Internal Visibility:** When deployed at network perimeters, NIDS may miss threats occurring inside the network or across segmented environments.
- **Performance Bottlenecks:** In high-traffic environments, NIDS may drop packets or slow down due to processing limitations, affecting detection accuracy.

1.2.3.3 Host- based Intrusion Detection Systems architecture

It is a security solution installed on individual devices—such as servers or workstations—to monitor the internal state and behaviour of the host. It analyses system logs, file integrity, user activities, and local network connections to detect unauthorized access or malicious actions. HIDS operates using local agents that compare observed behaviours with known attack patterns, raising alerts or logging incidents when suspicious activity is detected. It is often integrated with Security Information and Event Management (SIEM) systems like Splunk or OSSEC for centralized oversight. HIDS focuses on host-level activities, providing deeper visibility into system-level threats such as insider attacks, abnormal application behaviour, or tampering.

Advantages and limitations of HIDS: Host-based Intrusion Detection System also has advantages and limitations based on its architecture.

Advantages

- Provides *in-depth visibility* into host-level events (e.g., file changes, system calls), enabling effective detection of internal or encrypted attacks [Raj and Sharma, 2020].
- Effectively detects *insider threats* and policy violations by monitoring user actions and system behaviour [Chauhan and Chandra, 2013a].

- Supports *file integrity monitoring* (FIM) and detailed forensic analysis through local log inspections, essential for compliance and audit trails [Scarfone and Mell, 2007].
- Reduces network overhead as all analysis and event processing occurs locally on the host, minimizing additional traffic [Raj and Sharma, 2020].

Limitations

- **High Resource Consumption**: Constant monitoring of files, logs, and processes consumes CPU, memory, and disk space, which may degrade system performance—especially on resource-constrained devices [Raj and Sharma, 2020].
- **Limited Scope**: Only protects the individual host it's installed on and cannot detect threats that move laterally across the network or target other hosts [Scarfone and Mell, 2007].
- **Management Complexity**: Requires installation and configuration on each host, and maintaining updates and rule sets across many machines increases administrative overhead [Chauhan and Chandra, 2013a].
- **Vulnerability to Tampering**: If attackers compromise the host, they can disable or modify the HIDS, allowing malicious activity to go undetected [Raj and Sharma, 2020].
- **False Positives and Alert Fatigue**: Innocent anomalies often trigger alerts, causing alert fatigue and potentially obscuring true threats [Scarfone and Mell, 2007].

1.2.3.4 Hybrid-based Intrusion Detection Systems

Combining multiple detection models is the aim of hybrid intrusion detection systems [Maseno et al., 2022] [Satilmiş et al., 2024] in order to improve performance. Two parts make up a hybrid intrusion detection system. The unclassified data is processed by the first part. The processed data is scanned by the second component to identify any instances of infiltration. It works by combining two learning algorithms [Tsai et al., 2009].Every learning algorithm has distinct characteristics that help to enhance the hybrid's performance. The three main types of hybrid intrusion detection systems are single hybrid, integrated-based hybrid, and cascaded hybrid.

Advantages and limitations of Hybrid IDS: Hybrid Intrusion Detection System has advantages and inconveniences [Chauhan and Chandra, 2013b] , we mention :

Advantages

- Low false alarm/positive ratio.
- Provides a tighter and broader perimeter, enabling good performance against both internal and external unauthorized access.
- Allows correlation between network events and those occurring on target hosts.
- Detects intrusive activity targeting multiple hosts and offers information about affected systems.
- Capable of analysing encrypted data that has already been decrypted on the host.
- Correlates alerts from HIDS and NIDS to improve the likelihood of detecting real intrusions.
- Enhances overall detection rate of attacks.

Limitations

- Implementation is highly complex.
- Not easily adaptable to frequently changing network environments.
- May cause significant performance degradation on hosts due to the combined resource overhead of HIDS and NIDS components.

Out of all these architectures , in our thesis we are interested about the Distributed Intrusion Detection Systems.

1.2.4 Intrusion Detection Systems Architectures Comparison

Below is table 1.1 that compares between the IDS architectures based on those two papers [Othman et al., 2018] and [Zhang et al., 2019]

Criteria	HIDS	NIDS	DIDS	Hybrid IDS
Deployment Location	Individual hosts/endpoints	Network devices (e.g., routers)	Both hosts and network	Across hosts, networks, cloud
Input Data	System logs, audit trails, file integrity, process behaviour	Network traffic (headers, payloads, protocols)	Combination of host and network data	All available data (host, network, cloud, external feeds)
Platform Support	OS-level (Windows, Linux, Mac)	Network infrastructure devices	Mixed platform	Cross-platform with scalable architecture
Attack Types Detected	Insider threats, privilege abuse, malware, rootkits	DoS/DDoS, port scans, sniffing, brute force	Combined threats (internal and external)	Sophisticated and multi-stage attacks
Detection Capability	High for internal attacks	High for external threats	Balanced internal and external	High for all types (anomaly, misuse, hybrid)
Resource Usage	High (per host)	Low to moderate	Moderate to high	Depends on components
Response Time	Fast for local events	Real-time for network threats	Moderate (coordination overhead)	Fast and coordinated (if optimized)
Scalability	Limited due to host dependence	High	Moderate	High with distributed design
Advantages	In-depth visibility, insider threat detection	Broad network monitoring, early detection	Comprehensive view, better accuracy	Adaptive, accurate, flexible
Limitations	Host tampering, scalability	Can't analyze encrypted traffic or local events	Complex management	Complex design and high resource need

Table 1.1: Comparison of Different IDS Architectures

1.2.5 Intrusion Detection Systems Classification

Intrusion detection Systems are classified into methods [Maseno et al., 2022] [Zarringhalami and Rafsanjani, 2012] or techniques [Satilmiş et al., 2024], we mention of them the next 3 methods:

- **Signature-based Intrusion Detection Systems (SIDS)**: Also known as knowledge-based [Khraisat et al., 2019] or misuse detection, identify known threats by comparing monitored network traffic or host activities against a database of predefined intrusion signatures or past log patterns. These systems, used in tools like Snort and NetSTAT, offer high accuracy for previously encountered attacks but struggle

to detect zero-day or polymorphic threats that lack existing signatures. Traditional SIDS often fail to capture complex attacks spanning multiple packets and have become less effective against the growing sophistication of modern malware. Despite improvements like using state machines and semantic pattern matching, the increasing rate of novel and targeted attacks has highlighted the limitations of SIDS and encouraged a shift toward Anomaly-based Intrusion Detection Systems (AIDS).

- Anomaly-based Intrusion Detection Systems (AIDS): [Khraisat et al., 2019]Identify abnormal or malicious behaviour by detecting significant deviations from normal system activity, using heuristics based on machine learning, statistical, and knowledge-based methods—machine learning being the most effective. These systems are designed to overcome the limitations of Signature-based IDS (SIDS), particularly in detecting zero-day and previously unknown attacks. AIDS models are trained during a learning phase using normal traffic data and tested on new data to detect anomalies. They can operate in supervised, semi-supervised, or unsupervised modes. Supervised approaches require labelled data to distinguish normal from abnormal patterns, while semi-supervised methods use mostly normal data, which can result in high false positive rates. Unsupervised techniques, which do not require labelled data, focus on learning from normal behaviour to detect previously unseen intrusions more effectively. AIDS can also detect internal malicious activities and are difficult for attackers to bypass due to their use of customized behaviour profiles. However, their sensitivity to novel behaviours may still result in false positives.

- Specification-based Intrusion Detection Systems: [Tseng et al., 2003]identifies attacks by monitoring system or network behaviour and comparing it to manually defined security specifications that describe correct and expected operations. These specifications are crafted based on security policies, system functionality, and normal usage patterns. Rather than detecting attacks directly, this method flags deviations from expected behaviour at runtime, which may indicate an intrusion. It is particularly effective at detecting previously unknown attacks, as it is not limited to known signatures. Commonly used to protect critical applications and protocols like ARP and DHCP, specification-based detection enforces rules on message structure, sequence, and content. Although it requires manual effort to define accurate specifications, combining it with other detection methods can improve accuracy and reduce false positives.

1.2.6 Intrusion Detection Systems Methods Comparison

Here is table 1.2 that compares between the 3 different classification methods [Liao et al., 2013] of intrusion detection systems:

Criteria	Signature-based (SIDS)	Anomaly-based (AIDS)	Specification-based
Detection Principle	Matches activities against known attack signatures	Detects deviations from a learned model of normal behaviour	Compares runtime behaviour against predefined specifications
Detection of Zero-day Attacks	Poor (cannot detect unknown attacks)	Good (can detect unknown and novel attacks)	Good (can detect previously unseen behaviour if it violates specifications)
False Positives	Low	High (especially in semi- and unsupervised approaches)	Moderate (depends on completeness and correctness of specifications)
Data Requirements	Requires labelled attack signatures	Requires normal (and sometimes labelled) data for training	Requires manually defined specifications based on expected behaviour
Approaches Used	Pattern matching, rule-based detection	Machine learning (supervised, semi-supervised, unsupervised), statistical, knowledge-based	Manual rule specification, protocol behaviour modelling
Examples	Snort, NetSTAT	KDD-based ML models, anomaly profilers	ARP/DHCP protocol monitors, runtime policy enforcement
Strengths	High accuracy for known attacks; fast detection	Detects novel and internal attacks; adaptable to evolving threats	Detects violations of intended behaviour; not limited to known signatures
Weaknesses	Cannot detect new or obfuscated attacks	High false alarms; needs large/clean datasets	Labor-intensive to define; limited by specification quality

Table 1.2: Comparison of Intrusion Detection Methods

1.3 Federated Learning Fundamentals

The foundation for collaborative model training that protects privacy is laid by the idea of Federated Learning and we are going to explain it in this section.

1.3.1 Federated Learning Definition

Federated Learning (FL) [Yang et al., 2019] [Gosselin et al., 2022] is a decentralized machine learning technique in which multiple data owners collaboratively train a shared model without exchanging their private data. Unlike traditional centralized methods that require aggregating all data in one location, FL allows training to occur locally on user devices, preserving both data privacy and security. This approach is particularly promising for developing privacy-preserving solutions to address emerging cybersecurity threats.

1.3.2 Federated Learning Concept For IDS

Federated Learning (FL) for IDS [Agrawal et al., 2022] is a decentralized approach where multiple devices train an IDS model locally and share only model updates instead of raw data, preserving privacy. FL enhances intrusion detection in sensitive environments while reducing data exposure risks. It improves security while addressing privacy, scalability, and communication challenges in network security.

1.3.3 Federated Learning Types

This section describes the main types of Federated Learning (FL) [Li et al., 2020b].

1.3.3.1 Horizontal Federated Learning

It is a type of FL that addresses overlap between data features across nodes and differences in sample space. Currently, FL algorithms are primarily used in smart devices and IoT devices, with horizontal FL being the most common. Google's federated model solution for Android mobile phone updates is typically horizontal FL due to similar feature dimensions. Gao et al. introduced hierarchical heterogeneous horizontal FL frames to address limited labelled entities and address data annotation issues in EEG classification. In real applications like medical care, cross-regional cooperation is difficult due to the difficulty in building data pools for sharing. FL could construct a federal network for cross-regional hospitals with similar medical information to improve joint models.

1.3.3.2 Vertical Federated Learning

It is a machine learning approach that partitions data vertically according to feature dimension, allowing for the prediction and personalization of diseases like diabetes. This approach is particularly useful for medical institutions that need to analyse homogeneous data, such as age, weight, and medical history, to predict and personalize diseases. Vertical FL can also work with companies that hold smartphone application data sets, allowing them to cooperate without raw data transmission. However, it is more challenging to apply vertical FL due to entity resolution issues and the need for correspondence between different owners. Various methods have been developed to preprocess vertical partitioned data, such as token-based entity resolution algorithms, end-to-end schemes on linear classifiers, and secure frameworks like SecureBoost. However, these methods are only applicable to simple machine learning models like logistic regression, leaving room for improvement in more complex machine learning approaches. Overall, vertical FL has potential for further development in machine learning applications.

1.3.3.3 Federated transfer learning

It is a method that generalizes deep learning (FL) to address the issue of data sharing and poor data quality. It allows knowledge from one domain to another, achieving better learning results. FTL is the first complete stack for FL based on transfer learning, including training, evaluation, and cross validation. Neural networks with additive homomorphic encryption technology can prevent privacy leakage and provide comparable accuracy with traditional methods. However, communication efficiency remains a challenge. Sharma et al. (2019) worked on improving FTL by using secret sharing technology instead of HE to reduce overhead and hinder malicious servers. Chen et al. (2019) constructed a FedHealth model that gathers data from different organizations via FL and offers personalized healthcare services through FTL. While FTL research is not mature, it is an effective way to protect data security and user privacy while breaking data islands.

1.3.4 Federated Learning Architectures

The efficiency of an Intrusion Detection System (IDS) depends on choosing the right deployment architecture. There are two main types:

1.3.4.1 Centralized Federated Learning

It is a common architecture of Federated Learning systems [Zhang et al., 2020], where a single central node handles communication, aggregation, and deployment of models. This

architecture offers smooth and elegant model transmission, quick updates, and efficiency for small systems. However, it can lead to scalability issues, as the server node may not improve performance when thousands of client nodes join, and communication bottlenecks may arise when traffic increases exponentially. Additionally, the system can easily break down due to a Denial-of-Service attack on the server.

1.3.4.2 Decentralized Federated Learning

It is another architecture [Yuan et al., 2024] that allows model weights to be shared according to broadcast, gossip, or pointing protocols in order to produce the best models for every client.Pointing is one of the easiest and most direct ways for two peers to establish a one-to-one, unidirectional, and defined communication relationship.

1.3.5 Federated learning fundamentals

The most common fundamentals [Banabilah et al., 2022] of FL are these:

- **Data Privacy:** In federated learning, the data does not need to be collected and centralized on a server. It remains on the local devices, preserving user privacy and complying with data protection regulations such as GDPR.
- **Data Distribution:** The data is distributed across a wide range of devices. These devices may vary in terms of data size, quality, and type. This diversity helps in building more robust models.
- **Client-Side Training:** Each participating device trains the model locally using its own data. This involves running training algorithms such as gradient descent on the local data, without sending the data to a central server.
- **Local Model Updates:** After training on local data, each client computes updates (such as model parameters or gradients) to improve the model. These updates are then shared with the central server.
- **Federated Averaging (FedAvg):** The server collects model updates from clients and aggregates them (usually by averaging) to form a global model. This aggregated model is then sent back to the clients for further training. This process continues iteratively.
- **Aggregation Server:** The central server coordinates the training, aggregates the local model updates, and distributes the global model back to clients.

- **Communication Overhead:** Since model updates are exchanged over a network, it is important to minimize communication overhead. Techniques like model compression, quantization, or sending only significant updates can help reduce the amount of data exchanged.
- **Asynchronous Updates:** Updates can be sent asynchronously, avoiding the need to wait for all clients to send their updates at once.
- **Device and Data Heterogeneity:** Federated learning systems need to handle differences in devices (e.g., computational power, network connectivity) and data (e.g., class imbalance, non-IID data). These challenges require careful model design and adaptive learning strategies.
- **Non-IID Data:** In federated settings, data may not be independently and identically distributed (non-IID). This can lead to challenges in convergence and generalization.
- **Differential Privacy:** To protect sensitive information, techniques like differential privacy can be applied. This ensures that the model updates do not reveal too much about any individual client's data.
- **Secure Aggregation:** Methods such as secure multi-party computation (SMPC) or homomorphic encryption can be used to ensure that the updates are encrypted and private, preventing adversaries from gaining insight into the model updates or individual data.
- **Personalized Models:** After global model aggregation, the model can be fine-tuned locally on individual devices to better fit each client's data. This can improve performance, especially in highly diverse datasets.
- **Large-Scale Deployment:** Federated learning systems are designed to scale to a large number of clients (e.g., millions of devices). Effective scheduling and optimization algorithms are required to ensure that this large-scale system operates efficiently.

1.3.6 Federated learning use case and applications

Federated learning has been useful in several real-world scenarios, below are some examples of use cases and applications also:

1.3.6.1 Use case

Below we mention some FL use cases [Brik et al., 2020]:

- **Medical Imaging**:A prominent use-case of FL in healthcare is for the early detection of Rheumatic Heart Disease (RHD). This was implemented on the ATMOSPHERE platform, which supports the development and deployment of medical imaging applications using federated cloud infrastructures. The proposed solution utilized a deep learning classifier that analyses echocardiographic video data and demographic information to classify patients into three categories: Definite RHD, Borderline, or Normal.

- **Anomaly Detection in IoT Devices**:Another use case is the DIoT system, which uses FL to detect anomalies in Internet of Things (IoT) devices without the need for labelled data or human intervention.

- **Augmented Reality**:FL has also been applied in Augmented Reality (AR) to address issues related to high data volume and latency. The proposed framework integrates FL with Mobile Edge Computing (MEC).

- **Robotics**:In robotics, Federated Learning allows multiple robots to collaboratively improve their models while keeping data local. This is particularly useful in multi-robot systems, where centralized training can be limited by communication constraints and privacy concerns. By sharing only essential model updates instead of raw data, FL enhances learning efficiency without overloading the network.

- **Transportation: Autonomous Vehicles**:Autonomous vehicles use machine learning for tasks like obstacle detection and adaptive driving. Traditional cloud-based training can cause latency and risks in fast-paced environments. Federated Learning addresses this by enabling vehicles to collaboratively train models in real time while keeping data local. This approach improves decision-making based on current road conditions without requiring constant data transfer.

- **Smart Manufacturing: Predictive Maintenance**:In Industry, predictive maintenance powered by AI helps reduce downtime and improve efficiency. However, data privacy and cross-border restrictions can limit centralized approaches. Federated Learning offers a solution by allowing each manufacturing site to train models locally. These local models contribute to a global predictive system without sharing sensitive or proprietary industrial data.

1.3.6.2 Applications

Below we mention some of FL applications [Li et al., 2020b] :

- **Google Gboard Suggestions:** FL improves query suggestions on Gboard while preserving user privacy and minimizing latency by using Android's Job Scheduler and a client-server architecture.
- **Mobile Keyboard Prediction:** A Recurrent Neural Network (CIFG, a variant of LSTM) was trained using FL to provide faster and more efficient text predictions on mobile keyboards.
- **Browser History Ranking (Firefox):** FL was used to train models for ranking browser history suggestions, resulting in faster URL bar completions with privacy maintained through client-server optimization.
- **Visual Object Detection (FedVision):** FedVision supports decentralized training of YOLOv3 object detection models, allowing companies to train models across clients for hazard detection applications.
- **Patient Clustering from EMRs:** The CBFL algorithm uses FL to predict mortality and hospital stay durations using distributed Electronic Medical Records from over 200 hospitals.
- **fMRI Analysis:** FL is used for analysing fMRI data (ABIDE dataset) to identify autism, enhancing model generalizability while maintaining privacy.
- **Brain Tumour Segmentation:** FL applied to the BraTS 2018 dataset achieves accurate tumour segmentation without centralized data sharing.
- **Distributed Medical Databases:** An FL framework enables privacy-preserving meta-analysis of subcortical brain structures across institutions.
- **FedNER (Medical Named Entity Recognition):** FedNER framework uses FL to detect medical entities in text data across clients with improved accuracy over baseline methods.

1.3.7 Federated learning benefits and challenges

Federated learning has plenty of benefits and challenges that we are going to mention in this section.

1.3.7.1 Benefits of federated learning

we will start by mentioning some of its benefits [Shen et al., 2021] that make it appropriate for distributed and privacy-sensitive environments.

- **Data Privacy and Security:** Training occurs locally on devices, and only model updates are shared, which reduces the risk of sensitive data leakage.
- **Decentralized Training:** FL removes the need for centralized data collection, reducing transmission costs and improving system resilience.
- **Real-Time and Offline Prediction:** Since models are stored on-device, predictions can be made instantly and even without an internet connection.
- **Minimal Infrastructure Requirements:** Training can take place when devices are charging, idle, or connected to Wi-Fi, reducing the need for high-performance hardware.
- **Efficient Resource Usage:** FL leverages edge devices' computational capabilities, allowing scalable and continuous learning directly at the data source.

1.3.7.2 Challenges of Federated Learning

Despite significant advancements in Intrusion Detection Systems (IDS) using Federated Learning (FL), several challenges [Li et al., 2020c] remain which we are going to site some of :

- **Increased Power and Memory Consumption:** Devices with limited resources may struggle with the local training process.
- **Bandwidth Limitations:** Communication between devices and the server can be constrained by low bandwidth, causing latency or slower convergence.
- **Device Reliability:** FL depends on consistent device participation. Devices dropping out mid-training can degrade model performance.
- **Non-IID and Unbalanced Data:** Data is not identically distributed across devices, which can hinder model accuracy and convergence.
- **Scalability Issues:** Large-scale deployment requires careful orchestration of devices, communication, and resource allocation.

1.3.8 Federated Learning for Cybersecurity Applications

Federated Learning (FL) offers significant potential in enhancing cybersecurity across various industries, some of applications [Ghimire and Rawat, 2022] of FL in cybersecurity include:

- **Threat Detection and Anomaly Identification:** FL enables real-time monitoring by pooling insights from diverse sources without exposing sensitive data, improving the detection of evolving cyber threats.
- **Malware Detection and Classification:** FL enhances malware detection by aggregating data from different organizations, resulting in robust, real-time malware classifiers that adapt to evolving threats.
- **Predictive Analysis for Cyber Attacks:** FL improves predictive models by analysing historical data from multiple entities, allowing proactive identification of potential attack vectors.
- **Collaborative Defence Strategies:** FL facilitates cooperation between organizations to share threat intelligence without exposing sensitive data, creating a stronger collective defence against common adversaries.
- **Privacy-Preserving Intrusion Detection:** FL enables privacy-preserving intrusion detection by keeping sensitive data local and sharing only anonymized insights, allowing collective identification of unauthorized access while protecting user confidentiality.

1.4 Related works

We can divide different related works based on these approaches:

1.4.1 Machine learning approaches

Machine learning [Kreuzberger et al., 2023]is a rapidly growing field of computing that uses computers to learn and improve tasks without explicit programming. It has been around for over half a century, with Alan Turing and John McCarthy being key founders. The field has gained popularity in recent decades, particularly in the medical field, where computer vision has become faster and more reliable than human labour.

To identify cyberthreats, machine learning (ML) and intrusion detection systems (IDS) employ algorithms [Halim et al., 2021] such as XGBoost, SVM, k-NN, and decision trees.

Techniques for feature selection decrease computational complexity and increase detection accuracy. By choosing pertinent features from big datasets, a study presents GbFS, an enhanced Genetic Algorithm-based feature selection technique that improves IDS accuracy. With a maximum accuracy of 99.80 % and lower computing costs than conventional methods, GbFS successfully removes redundant features without sacrificing detection precision.

According to the study [Logeswari et al., 2023] anomaly-based IDS examines deviations, but standard signature-based IDS finds it difficult to fend off zero-day attacks. Network management is improved by software-defined networking (SDN), but there are drawbacks as well. SDN security is enhanced by machine learning (ML)-based intrusion detection systems (IDS), while they have drawbacks such as computational inefficiencies and false positive rates. With an accuracy of 98.7%, HFS-LGBM IDS performs better than other ML models.

The paper [Nimbalkar and Kshirsagar, 2021] suggests a feature selection technique that minimizes IoT network traffic features while preserving high detection accuracy by utilizing Information Gain and Gain Ratio. Using the IoT-BoT and KDD Cup 1999 datasets, the system creates two feature subsets (RFS-1 and RFS-2) and assesses them using the JRip classifier. With a high accuracy rate of 99.9992%, the system surpasses current techniques by lowering the feature set to 16 and 19 features.

1.4.2 Deep learning approaches

Deep learning [Chassagnon et al., 2020] is a branch of machine learning that uses multi-layered neural networks, known as deep neural networks, to mimic the complex decision-making capabilities of the human brain. Today, it powers many of the artificial intelligence (AI) applications we use in everyday life.

Advanced neural networks [Caville et al., 2022] such as CNNs, RNNs, and Graph Neural Networks (GNNs) are used in Deep Learning (DL) for intrusion detection (IDS) in order to identify intricate patterns in cyberthreats. DL models are better at identifying complex attacks than regular ML models since they extract features without the need for human selection. In order to detect anomalies without labelled data, Anomal-E, a self-supervised GNN-based NIDS, uses edge characteristics, network graph structures, and a modified Deep Graph Infomax model. The model outperformed conventional techniques in detecting complex cyberthreats after being evaluated on benchmark datasets.

Modern vehicles are highly connected, exposing them to cyber threats. Intrusion Detection Systems (IDSs) are deployed in In-Vehicle Networks (IVNs) [Rajapaksha et al., 2023]to counter these threats. AI-driven IDSs, using Machine Learning and Deep Learn-

ing, offer improved security by analysing CAN bus traffic patterns and detecting anomalies. However, existing IDSs lack comprehensive coverage of AI techniques, recent developments, and benchmark datasets. This study presents a novel AI-based IDS taxonomy for IVNs.

1.4.3 Federated learning approaches

Intrusion Detection Systems (IDS) play a critical role in network security, with Machine Learning (ML) and Deep Learning (DL) improving their detection capabilities. However, traditional ML/DL-based IDS rely on centralized data storage, raising privacy and security concerns.

The paper [Agrawal et al., 2022] reviews existing FL-based IDS implementations, analyzing their strengths and limitations while addressing challenges such as communication overhead, security risks, and model aggregation complexities. Despite FL's potential to enhance IDS privacy and scalability, further research is needed to optimize its efficiency and robustness.

This comprehensive review [Fedorchenko et al., 2022] compares various intrusion detection systems based on Federated Learning. It highlights how FL reduces privacy risks by keeping data localized and supports collaboration between entities with private datasets. It discusses its specific challenges like non-IID data, device heterogeneity, and model aggregation issues. It also analyses system architectures and datasets used in IDS research, to compare datasets, machine learning methods, and evaluation metrics, ultimately identifying open research gaps and suggesting directions and solutions for future work.

FELIDS [Friha et al., 2022] is a federated learning-based intrusion detection system designed to secure agricultural IoT infrastructures while preserving data privacy. It employs three deep learning classifiers to detect cyber threats, enhancing privacy and security. FELIDS is evaluated using datasets like CSE-CIC-IDS2018, MQTTset, and InSDN, demonstrating superior performance compared to non-federated machine learning models. Despite challenges like data privacy, communication overhead, and power consumption, FELIDS offers a promising alternative to traditional machine learning solutions.

Federated Learning (FL) can be adapted for different types of computer networks, such as PAN, LAN, MAN, WAN, and satellite networks, based on factors like network size, traffic, and data interaction. For Satellite-Terrestrial Integrated Networks (STINs) [Li et al., 2020a], FL-based IDS has proven essential for defending against attacks like DoS, due to the resource and bandwidth gaps between satellite and terrestrial networks. An FL-based STIN algorithm outperforms traditional IDS by optimizing energy use, reducing packet loss, and lowering CPU utilization. In Local Area Networks (LAN),

where device numbers range from 3 to 60, segmented FL architectures with multiple global models improve prediction accuracy and aggregation results by addressing non-IID data. Grouping client models based on similarity further enhances performance through optimized hyper-parameters.

1.4.4 Summary Of Related Works On IDSs

Below , table 1.3 that represents all the related work sited above summarised :

Research	Technique	Method	Dataset	Key Results / Contribution
Anomal-E [Caville et al., 2022]	GNN (Self-supervised)	Graph-based anomaly detection (Deep Graph Infomax)	Custom graph-based dataset	Effective anomaly detection; high computational cost; good for complex threats
GbFS [Halim et al., 2021]	Genetic Algorithm	Feature selection to enhance ML-IDS	NSL-KDD	98.5% accuracy with only 12 features; reduced complexity
AI-based IDS for In-Vehicle Networks [Rajapaksha et al., 2023]	ML & DL	AI taxonomy for vehicular security	CAN Bus, other automotive datasets	99.99% detection rate for known attacks; F1-score > 0.95 for novel ones
FL for IDS [Agrawal et al., 2022]	Federated Learning	Distributed IDS with privacy-preserving model updates	CICIDS2017, NSL-KDD	Enhanced privacy; scalable; suffers from communication overhead
FELIDS [Friha et al., 2022]	FL + DL (DNN, CNN, RNN)	Federated IDS for Agri-IoT	Edge-IIoTset, CSE-CIC-IDS2018	Strong performance in IoT; privacy preserved; faces complexity and energy issues
HFS-LGBM IDS [Logeswari et al., 2023]	LightGBM + FS	ML for SDN anomaly detection	SDN-specific dataset	Achieved 99.9% accuracy; reduced false positives; efficient
IoT Feature Selection IDS [Nimbalkar and Kshirsagar, 2021]	Info Gain & Gain Ratio	Dimensionality reduction for IoT IDS	IoTID20, KDD Cup 1999	Accuracy up to 99.98%; only 16–19 features selected

Table 1.3: Summary of different Related Works on Intrusion Detection Systems (IDS)

1.5 Others

This section examines site other modern and novel approaches to intrusion detection systems (IDS) that go beyond conventional and federated learning techniques.

1.5.1 Hybrid Meta-heuristic-Based IDS for Enhanced Network Security

In response to the increasing complexity of cyber threats, this study [Fadhil et al., 2024]proposes a hybrid Intrusion Detection System (IDS) that combines meta-heuristic algorithms with deep learning. The system integrates the Lion Optimization Algorithm (LOA) and Grey Wolf Optimizer (GWO) to enhance detection accuracy and reduce false alarms. The key contribution is the development of a hybrid approach, referred to as LOF-SGWO, which merges Lion Optimization Feature Selection (LOFS) with GWO within a CNN-LSTM deep learning framework. This method enables the detection of unknown vulnerabilities and stealthy attacks in real time. Experiments using the NSL-KDD dataset and comparisons with the WUSTL-EOM 2020 system demonstrate over 99.26% accuracy. The results validate the proposed model's effectiveness and its potential application in real-world network security challenges.

1.5.2 Quantum-Inspired Horse Herd Optimization for Intrusion Detection

This paper [Ghanbarzadeh et al., 2023] introduces a novel intrusion detection method based on the Horse Herd Optimization Algorithm (HOA), inspired by horse herd behaviour. The algorithm is adapted into a discrete form and enhanced using quantum computing principles to create a quantum-inspired multi-objective version, called MQBHOA. The method aims to improve feature selection and classification performance for detecting network intrusions. A K-Nearest Neighbour (KNN) classifier is used to classify network packets into normal and four attack categories. Experiments conducted on the NSL-KDD and CSE-CIC-IDS2018 datasets show that MQBHOA achieves superior results, including a 6% improvement in feature selection and classification accuracy, with an overall detection accuracy of 99.8%. These results demonstrate the effectiveness of MQBHOA in addressing intrusion detection as a multi-objective optimization problem.

1.5.3 Hybrid Federated Learning-Based IDS for IoT Using CNN and BiLSTM

The rapid growth of Internet of Things (IoT) devices has created significant challenges for maintaining system security and privacy. This paper [Bukhari et al., 2024] presents a scalable Intrusion Detection System (IDS) based on Federated Learning (FL) tailored for IoT environments. The proposed horizontal FL model combines Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) to effectively extract spatial and temporal features from distributed data. CNN identifies local patterns, while BiLSTM captures sequential dependencies. Adopting a zero-trust model, the system retains data on edge devices and shares only trained weights with a centralized FL server for global model updates. Experimental results using CICIDS2017 and Edge-IIoTset datasets show that this hybrid approach outperforms existing centralized and federated deep learning IDS models in both accuracy and scalability.

1.6 Research Gaps and Motivation

Despite significant advancements in Intrusion Detection Systems (IDS) leveraging Machine Learning (ML), Deep Learning (DL), and Federated Learning (FL), several critical challenges [Issa et al., 2024] remain unresolved. Many existing systems still struggle to detect zero-day attacks due to their reliance on predefined signatures, underscoring the need for anomaly-based detection models. Additionally, high computational costs and the presence of redundant features reduce model efficiency, highlighting the importance of effective feature selection techniques. Centralized ML/DL models also raise serious privacy concerns, which makes Federated Learning a compelling solution for secure, decentralized training. Moreover, most current IDS solutions lack real-time performance and scalability, particularly in resource-constrained environments such as IoT and Software Defined Networks (SDNs). Furthermore, the limited integration of advanced hybrid models—such as combinations of ML, DL, and Graph Neural Networks (GNNs)—represents a missed opportunity to enhance detection accuracy and robustness. Recent studies on collaborative IDS frameworks using FL have introduced taxonomies to classify existing approaches based on datasets, aggregation strategies, ML models, and architectural configurations. These works recognize advantages of FL in preserving privacy and enabling distributed learning, while also identifying persistent challenges including high false positive rates, communication overhead, and secure model aggregation.

Motivated by these gaps, our thesis objective is how can we design a distributed IDS that balances high detection performance, data privacy, and scalability across scalabil-

ity across diverse environments. Traditional IDS frameworks either compromise privacy through centralized learning or fail to generalize across diverse data distributions. Moreover, the integration of Federated Learning into IDS remains under-explored in practical, real-world settings where attacks evolve rapidly, and client data is non-IID. This thesis addresses the need for a federated IDS architecture capable of learning collaboratively without exposing sensitive data, while maintaining robustness, low latency, and minimal communication cost in detecting a wide range of cyber threats.

This thesis proposes a federated IDS architecture that learns collaboratively without exposing sensitive data, maintains robustness, low latency, and minimal communication cost.

1.7 Conclusion

Intrusion Detection Systems (IDS) play a critical role in cybersecurity, helping detect and mitigate cyber threats in various network environments. While Machine Learning (ML), Deep Learning (DL), and Federated Learning (FL) have significantly improved IDS capabilities, several challenges remain, including zero-day attack detection, high computational costs, data privacy concerns, and scalability issues. To address these challenges, advanced feature selection methods (e.g., Genetic Algorithms), self-supervised learning (e.g., Graph Neural Networks), and privacy-preserving techniques (e.g., Federated Learning) are emerging as promising solutions. However, further research is needed to optimize IDS for real-time performance, reduce false positives, and enhance model efficiency in large-scale networks such as IoT and SDN. Future work should focus on hybrid approaches that combine ML, DL, and FL, enabling more adaptive, efficient, and privacy-preserving IDS solutions. By overcoming existing limitations, IDS can continue to evolve as a robust defence mechanism against modern cyber threats.

Chapter 2

Proposed Approach

2.1 Introduction

This chapter introduces a novel Distributed Intrusion Detection System (DIDS) architecture based on Federated Learning (FL), outlining the overall system design and deployment plan. The architecture is described, then the dataset used to simulate a federated environment is introduced, the data preprocessing stage is discussed in detail, including methods for locally preparing the dataset at each client node.Finally, the chapter examines the use of the the chosen model within the Federated Learning framework, focusing on aggregation techniques and challenges. This distributed and privacy-aware design aims to deliver accurate, scalable, and secure intrusion detection suitable for modern networked environments.

2.2 Architecture of The Proposed Solution

The architecture of the proposed intrusion detection system is built on top of a Federated Learning (FL) platform that enables privacy-preserving distributed model training,it consists of multiple clients, each having a local dataset, and a central server to collect the local models without directly accessing raw data.

Each customer trains a model with its own data, allowing particular traffic behaviours or patterns of attack to be captured. Instead of sending data to a central location, clients only send their model weights or gradients to the server. The server then sums them up to create a global model that is augmented with distributed knowledge without compromising data privacy.

For our application, the architecture involves:

- **Local Nodes:** Each node (client) has a model that is trained using local samples

from the dataset.

- **Central Aggregator:** This component combines the learned parameters using either Weighted Aggregation or Difference Aggregation strategies.
- **Global Model:** The combined model is distributed once again to all the clients so that they can improve performance step by step without data leakage.

This distributed design is especially useful in cybersecurity deployments, where data sensitivity and confidentiality are critical.

the architecture of the proposed intrusion detection system consists four main phases: data preprocessing, feature selection, model training, and classification. Each phase contributes to enhancing the performance and accuracy of the detection engine.

Below is a figure 2.1 that represents a graph describing the architecture of our proposed solution and its phases.

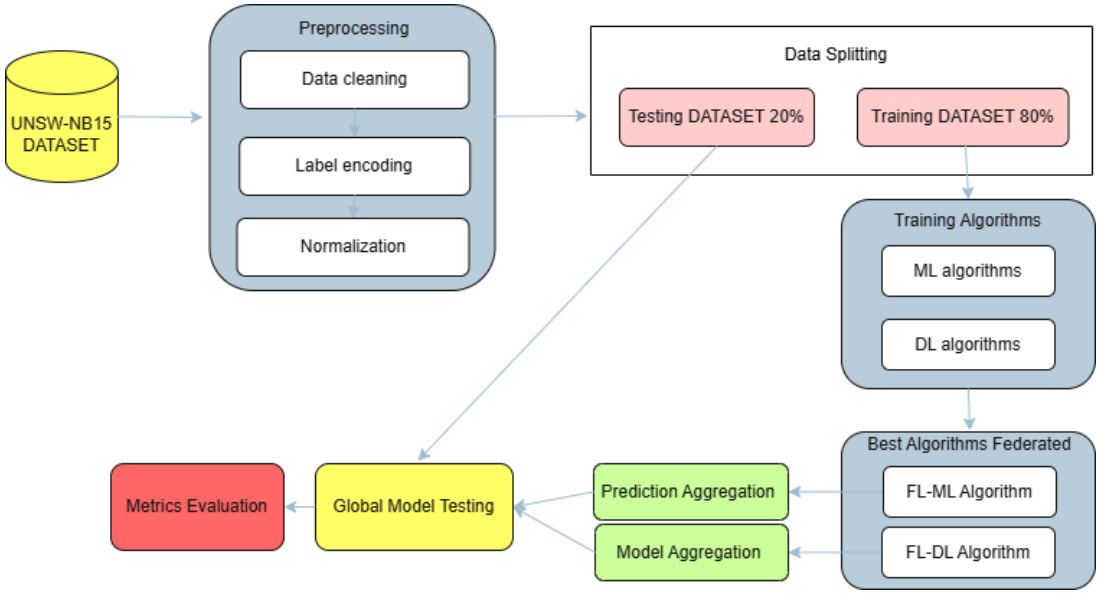


Figure 2.1: Architecture of the solution

2.3 Dataset

In this project, the UNSW-NB15 dataset [Moustafa and Slay, 2015] was used to develop, train, and evaluate the proposed Federated Learning-based Intrusion Detection System (IDS). While the dataset originates from a centralized source, it was partitioned and used in a federated simulation environment . This approach allowed the modelling of decentralized, privacy-aware IDS architectures under realistic attack scenarios.

The UNSW-NB15 dataset was selected for its realism and comprehensiveness, published in 2015, comprises circa 2.54 million labelled records (totalling 2,540,044 records), covering nine contemporary attack types and 49 features categorized into basic, content-based, temporal, general-purpose, and connection metrics. The dataset is split into 175 341 training and 82 332 testing instances, originally partitioned by the authors. Its diversity and modern attack representation make it a suitable benchmark for federated learning-based IDS research.

The UNSW-NB15 dataset includes 10 classes, comprising one normal class and nine different attack types: Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The dataset is highly imbalanced, with Normal traffic representing about 36% of the samples and minority classes like Worms and Shellcode each accounting for less than 1%. This imbalance presents challenges for effective detection and fair model evaluation.

Due to its large size and detailed labels, the dataset is useful not only for model evaluation but also for simulating non-IID conditions by splitting the data across clients.

2.3.1 Features Of The UNSW-NB15 Dataset

As we mentioned in the description the dataset is splitted into training and testing instances , each with 45 features, and they are categorized , we are going to mention them above in different tables according to their categories:

2.3.1.1 Basic Features

Table 2.1 represents basic characteristics of network flow are captured by these features, they offer a fundamental perspective of every network session.

Feature Name	Description
srcip	Source IP address
sport	Source port number
dstip	Destination IP address
dsport	Destination port number
proto	Protocol used
state	Connection state
dur	Duration of the connection
sbytes	Bytes sent from source to destination
dbytes	Bytes sent from destination to source
service	Network service on the destination
is_sm_ips_ports	Indicator for same IP addresses and ports

Table 2.1: Basic Features in UNSW-NB15

2.3.1.2 Time-based Features

These features measure the timing behaviour of network traffic, they are presented in table 2.2 :

Feature Name	Description
sttl	Source TTL value
dttl	Destination TTL value
sloss	Packets lost from source
dloss	Packets lost from destination
Sload	Source bits per second
Dload	Destination bits per second
Spkts	Number of packets from source
Dpkts	Number of packets from destination
swin	Source TCP window size
dwin	Destination TCP window size
smeansz	Mean packet size from source
dmeansz	Mean packet size from destination
sintpkt	Inter-arrival time from source
dintpkt	Inter-arrival time from destination
sjit	Source jitter
djit	Destination jitter

Table 2.2: Time-based Features in UNSW-NB15

2.3.1.3 Flow Features

Flow features describe the dynamics of communication between source and destination over a connection, they are presented in table 2.3 :

Feature Name	Description
stcpb	Source TCP base sequence number
dtcpb	Destination TCP base sequence number
tcprtt	Round-trip time of TCP connection
synack	Time between SYN and SYN-ACK
ackdat	Time between ACK and data
ct_state_ttl	State and TTL-based features
ct_srv_src	Connections from same source to same service
ct_srv_dst	Connections to same service from destination
ct_dst_ltm	Connections to same destination in last 100 connections
ct_src_ltm	Connections from same source in last 100 connections
ct_dst_src_ltm	Connections between same src/dst in last 100 connections

Table 2.3: Flow Features in UNSW-NB15

2.3.1.4 Content Features

These features are useful for identifying application-layer attacks and protocol abuse since they examine the payload content and command behaviour of network protocols, they are presented in table 2.4 :

Feature Name	Description
trans_depth	HTTP transaction depth
res_bdy_len	Response body length
is_ftp_login	Successful FTP login indicator
ct_flw_http_mthd	Count of HTTP methods
ct_ftp_cmd	Count of FTP commands

Table 2.4: Content Features in UNSW-NB15

2.3.1.5 Other Features

Table 2.5 represents extra and additional metadata:

Feature Name	Description
stime	Start time of the connection
ltime	End time of the connection

Table 2.5: Other Features in UNSW-NB15

2.3.2 Types of Attacks in the UNSW-NB15 Dataset

The UNSW-NB15 contains normal attacks and other 9 types of attacks that we are going to site and describe them in table 2.6 :.

Attack Type	Description	Training Instances	Testing Instances
Normal	Benign traffic with no attack patterns.	56,000	37,000
Analysis	Information gathering and scanning activities, such as port scanning and OS fingerprinting.	2,000	677
Backdoor	Attacks that establish unauthorized remote access to a system by bypassing security mechanisms.	1,746	583
DoS	Denial of Service attacks that aim to disrupt services by overwhelming resources.	12,264	4,089
Exploits	Attacks that take advantage of vulnerabilities in systems or applications to gain control or disrupt operations.	33,393	11,132
Fuzzers	Random data sent to targets to trigger faults and discover vulnerabilities.	18,184	6,062
Generic	Broad attacks that apply to many targets, such as brute force or payload manipulation.	40,000	18,871
Reconnaissance	Scanning and probing activities to map the network and identify potential targets.	10,491	3,496
Shellcode	Injection of binary Shellcode into a vulnerable system to gain control.	1,133	378
Worms	Self-replicating malware that spreads across networks to infect other systems.	130	44

Table 2.6: Different Attack Categories in the UNSW-NB15 Dataset

below is an example of a row in the dataset presented in table 2.7:

Feature	Value
id	1
dur	1.1e-05
proto	UDP
service	-
state	INT
Spkts	2
Dpkts	0
sbytes	496
dbytes	0
rate	90909.0902
sttl	254
dttl	0
Sload	180363632.0
Dload	0.0
sloss	0
dloss	0
sintpkt	0.011
dintpkt	0.0
sjit	0.0
djit	0.0
swin	0
stcpb	0
dtcpb	0
dwin	0
tcprtt	0.0
synack	0.0
ackdat	0.0
smean	248
dmean	0
trans_depth	0
response_body_len	0
ct_srv_src	2
ct_state_ttl	2
ct_dst_ltm	1
ct_src_dport_ltm	1
ct_dst_sport_ltm	2
ct_dst_src_ltm	0
is_ftp_login	0
ct_ftp_cmd	0
ct_flw_http_mthd	0
ct_src_ltm	1
ct_srv_dst	2
is_sm_ips_ports	0
attack_cat	Normal
label	0

Table 2.7: Single sample row from the UNSW-NB15 dataset.

The integration of the UNSW-NB15 dataset allowed for comparative testing and helped assess the adaptability of the proposed Federated Learning-based IDS under diverse attack scenarios and realistic network behaviours. When partitioned to simulate distributed environments, the dataset introduced challenges typical of real-world systems, such as class imbalance, client heterogeneity, and local overfitting critical aspects for validating the robustness of federated intrusion detection systems.

2.4 Preprocessing

Preprocessing is a crucial step in cybersecurity applications where raw network data is often noisy, inconsistent, and high-dimensional. In the context of the proposed Federated Learning-based Intrusion Detection System (FL-IDS), preprocessing plays a dual role: ensuring the quality and consistency of local data at each client node, and preserving privacy by avoiding any data centralization.

The preprocessing pipeline suggested in this project consisted of different phases, carefully designed to clean, transform, and standardize the data across all client nodes. We are going to introduce an algorithm that describes preprocessing phases below then we will explain each step of it.

Algorithm 1 Data Preprocessing for UNSW-NB15 Dataset

Input: Training file D_{train}, Testing file D_{test}
Output: Preprocessed dataset (X_{train}, X_{test}, y_{train}, y_{test})
// **Data Merging**
Load D_{train} and D_{test} files using pandas.read_csv() and concatenate them to form full dataset D
// **Data Cleaning**
Replace missing or empty values in D with 0
for each numerical column c in D **do**
    if $\max(c) > 10 \times \text{median}(c)$ and $\max(c) > 10$ **then**
        Clamp values above 95th percentile to the 95th percentile using np.where()
    end if
    if c has > 50 unique values **then**
        if $\min(c) = 0$ **then**
            Apply $\log(c + 1)$ transformation
        else
            Apply $\log(c)$ transformation
        end if
    end if
end for
for each categorical column cat in D **do**
    if nunique(cat) > 6 **then**
        Replace rare categories with a placeholder "-"
    end if
end for
Remove duplicate records using drop_duplicates()
// **Label Encoding**
Apply OneHotEncoder() to features proto, service, and state using ColumnTransformer
// **Data Normalization**
Apply StandardScaler() to numerical features starting from index 18 onward
// **Data Splitting**
Split D into 80% train and 20% test using train_test_split()
Return: X_{train}, X_{test}, y_{train}, y_{test}

2.4.1 Data Merging

Two separate files, UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv, were loaded and merged using pandas.concat() to form a unified dataset of over 2.5 million records.

2.4.2 Data Cleaning

The starting point is that the initial data set has plenty of missing values in certain feature columns, making it unsuitable for feeding into machine learning or deep learning models, this step aims to improve data quality and it is being done by 3 operations which are:

- **Missing Values:** Rows containing missing or empty cells were replaced by a default "0" value .
- **Extreme Outliers:** For each numerical feature, values exceeding 10 times the median and above the 95th percentile were clamped to the 95th percentile using np.where().
- **Logarithmic Transformation:** For numerical features with high variance and over 50 unique values, a log transformation was applied to reduce skewness. np.log() or np.log(x+1) was used depending on whether the minimum value was zero.
- **Rare Categories:** Categorical features with many rare values were grouped. Only the top frequent categories were retained; the rest were replaced with a placeholder value "-".
- **Duplicate Entries:** Duplicate records, which could bias the model, were identified and removed by using drop_duplicates(inplace=True) function from the pandas library.

2.4.3 Label Encoding

The second stage involves transforming the text that has been stored as a categorical feature in the cells into numerical values, with each category being represented by a specific and unique number value such as The features proto, service, and state were transformed using OneHotEncoder() from Scikit-learn ColumnTransformer.

2.4.4 Data Normalization

Data normalization has been particularly useful for systems where measurements are typically represented at highly disparate levels.Z-score normalization (standard scaling) was applied using StandardScaler() because it accurately preserves all data connections and therefore does not introduce any bias, it was applied to each feature,transforming all numerical values into common range.

2.4.5 Data Splitting

At this stage, the main dataset was divided into two segments: a 80% training-set and a 20% test-set.

2.5 Federated Learning (FL) Model Procedure

The model is divided into two main steps: Model training that uses the training-dataset-file and Model testing that evaluates the performance of the trained model while using the testing-dataset file.

2.5.1 Training Phase

The training phase is performed in a federated manner where multiple clients train independently the same model architecture on their local data then contribute to building a global model without sharing their private data. We present the steps of building the Machine Learning algorithm below:

Algorithm 2 Federated Learning Procedure (Classical Models)

Input: Local datasets $\{D_1, D_2, \ldots, D_N\}$, test set D_{test}
Output: Final prediction $\hat{y}_{\text{final}}$
for each client $i = 1$ to N **in parallel do**
    Train local model M_i on D_i
    Evaluate M_i on D_{test} to compute F1-score: $F1_i$
end for
Normalize client weights:

$$w_i = \frac{F1_i}{\sum_{j=1}^{N} F1_j}$$

Initialize prediction probability vector $\hat{y}_{\text{proba}} = 0$
for each client $i = 1$ to N **do**
    Predict probability $\hat{y}_i = M_i(D_{\text{test}})$
    Aggregate:

$$\hat{y}_{\text{proba}} += w_i \cdot \hat{y}_i$$

end for
Compute final predictions:

$$\hat{y}_{\text{final}} = \arg\max(\hat{y}_{\text{proba}})$$

Return: Final aggregated prediction $\hat{y}_{\text{final}}$

We present the steps of building the Deep Learning algorithm below:

Algorithm 3 Federated Learning Training Procedure (Neural Network)

Input: Initial global model w_0, client datasets $\{D_1, D_2, \ldots, D_N\}$, number of rounds T
Output: Final global model w_T
for each round $t = 1$ to T **do**
    Server broadcasts global model w_{t-1} to all clients
    for each client $i = 1$ to N **in parallel do**
        Preprocess local dataset D_i (cleaning, normalization, encoding)
        Train local model w_i^t initialized from w_{t-1}
        Send updated model w_i^t to the server
    end for
    if Aggregation = Weighted **then**
        Server updates global model using:

$$w_t = \sum_{i=1}^{N} \frac{n_i}{n_{\text{total}}} w_i^t$$

    else if Aggregation = Difference **then**
        Server updates global model using:

$$w_t = w_{t-1} + \sum_{i=1}^{N} \frac{n_i}{n_{\text{total}}} (w_i^t - w_{t-1})$$

    end if
end for
Return: Final model w_T

The federated training process is composed of two major stages: client-side local training and server-side model aggregation. This architecture is adaptable for both classical machine learning (ML) models and deep learning (DL) models.

- **Client-Side: Local Training**
  Each client performs the following steps independently, regardless of the underlying model type:
  - **Preprocessing:** Each client cleans, encodes, balances, and normalizes its local dataset.
  - **Model Initialization:** Clients receive the current global model from the central server.
  - **Local Training:** The model is trained on the client's local data.
  - **Model Update:** The updated model or predictions are sent back to the server.

- **Server-Side: Model Aggregation**
  Once updates are received from all clients, the server aggregates them using one of the following strategies, depending on whether the model is ML or DL:

  – **ML-Based: Weighted Soft Voting for Random Forest**
  In this strategy, predictions from client models are aggregated using F1-score–based weighting:

$$\text{Final Probability} = \sum_{i=1} w_i \cdot \hat{y}_i \quad \text{where} \quad w_i = \frac{F1_i}{\sum_{j=1}^{n} F1_j} \tag{2.1}$$

Weighted Soft Voting equation formula

We will explain the elements of this equation by relying every symbol to its description:

$\hat{y}_i$: Predicted probability vector from client i's local Random Forest model.

$F1_i$: F1-score of client i evaluated on the test set.

w_i : Normalized weight for client i based on its F1-score.

n : Total number of participating clients.

Final Probability : Aggregated soft prediction used for final class decision.

  – **DL-Based: Model Weight Aggregation for MLP (Keras-based)**
  The Keras-based MLP model uses weight-based aggregation strategies suitable for federated learning:

  * **Weighted Aggregation (FedAvg):**

$$w_t = \sum_{i=1}^{N} \frac{n_i}{n_{\text{total}}} w_i^t \tag{2.2}$$

Weighted Aggregation equation formula

  * **Difference Aggregation:**

$$w_t = w_{t-1} + \sum_{i=1}^{N} \frac{n_i}{n_{\text{total}}} (w_i^t - w_{t-1}) \tag{2.3}$$

Difference Aggregation equation formula

We will explain the elements of this equation by relying every symbol to its description:

w_t : Global model weights after aggregation at round t

w_{t-1} : Global model weights from the previous round

w_i^t : Local model weights from client i after round t

n_i : Number of training samples on client i

n_{total} : Total number of samples across all clients

N :Total number of clients

Training Flow:

- The server broadcasts the current global model to clients.
- Clients retrain locally and send updates (weights or predictions).
- The server aggregates the updates based on the model type and strategy.
- The global model is updated and the process repeats for multiple rounds.

2.5.2 Testing Phase

After completing the training phase , the final model is evaluated using the testing dataset to assess its generalization capability.

Algorithm 4 Model Testing and Evaluation Procedure

1: **Input:** Final global model w_T, test dataset D_{test}
2: **Output:** Evaluation metrics: Accuracy, Precision, Recall, F1-Score
3: Apply the same preprocessing pipeline to D_{test}
4: Use w_T to predict labels $\hat{y}$ for D_{test}
5: Compare $\hat{y}$ to ground-truth labels y
6: Compute the metrics.
7: **Return:** Accuracy, Precision, Recall, F1-Score

- **Global Model Distribution:** The trained global model is shared with each client or directly evaluated at the central server.
- **Preprocessing the Test Data:** The test set undergoes the same preprocessing steps as the training data, which we mentioned before.
- **Model Evaluation:** The global model is used to make predictions on the test data. These predictions are compared against the ground truth to compute key performance metrics.
- **Metrics Computation:** Since this research aims to maximize the correct predictions of instances in the test dataset by using the values of the following fundamental classification outcomes :

- **True Positive (TP):** this value represents the correct classification attack packets as attacks.
- **True Negative (TN):** this value represents the correct classification normal packets as normal.
- **False Negative (FN):**this value illustrates that an incorrectly classification process occurs. Where the attack packet classified as normal packet.
- **False Positive (FP):** this value represents incorrect classification decision where the normal packet classified as attack.

Based on the outcomes we site above We calculate certain metrics [Yacouby and Axman, 2020] to evaluate the model's effectiveness, which are:

- **Accuracy:**Measures the overall correctness of the system by calculating the proportion of correct predictions (both attacks and normal traffic) out of all predictions,it is calculated using this formula:

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

Accuracy equation formula

- **Precision:** Represents the proportion of instances predicted as attacks that are actually attacks. It estimates the reliability of positive predictions, it is calculated using this formula:

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

Precision equation formula

- **Recall:** Indicates the proportion of actual attacks that were correctly identified by the system. It reflects the model's ability to detect all relevant positive cases, it is calculated using this formula:

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

Recall equation formula

- **F1-Score:** Provides a harmonic mean of precision and recall, offering a balanced assessment, particularly in scenarios with imbalanced datasets. It is

widely used to evaluate classification performance when both false positives and false negatives are important,, it is calculated using this formula:

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.7}$$

F1-Score equation formula

2.6 Model for FL-Distributed Intrusion Detection

Since we are interested in our thesis by the Distributed Intrusion Detection System (DIDS) we trained multiple different models, evaluated each one of them ,and then compared them to find the most powerful model that achieves the best performance , we are going to mention the models we used:

2.6.1 Machine Learning Models

We classified all the models that we will evaluate according to whether they are Machine Learning-based or Deep Learning-based, we will start by siting the ML algorithms first.

2.6.1.1 Decision Trees (DT)

A decision trees are a popular approach for machine learning. It evaluates the attributes of the data in the form of a tree and outputs its results layer by layer.It is composed by these following components :

- **The root node**: the starting point of the decision tree, it symbolizes the complete dataset.
- **Internal Nodes**: each node represents a decision based on a feature.
- **Branches**:each branch represents the outcome of the decision , is it yes or no.
- **Leafs**:each leaf represents a final class label which in this case will contain either an attack or normal.

The final leaf node indicates the outcome of a classification or prediction.

2.6.1.2 Random Forest (RF)

Random forest builds many decision trees to make predictions using the output of all trees after combining them. Each tree in the "forest" is created by resampling using the

bootstrap technique and trained on a random subset of training data.The final prediction is made through majority vote.

2.6.1.3 Light Gradient Boosting Machine (LightGBM)

LightGBM is a Gradient Boosting Decision Tree algorithm used in various data mining problem ,it uses one-sided gradient analysis and exclusive features bundling techniques. It aims to find an approximation to a function that reduces the expected loss function value. LightGBM integrates multiple regression trees to approximate the final model. It employs a one-sides-sampling (GOSS) approach to detect split values in data instances.

2.6.1.4 K-Nearest Neighbors (KNN)

The purpose of the KNN algorithm is to use a database in which the training examples are expressed as data points in the problem feature space and separated into several classes. To predict the label (target class) of a new sample point which is initially projected in the considered feature space. Then the distances between that sample and the K-th nearest examples are calculated. Finally, the sample is classified by a majority vote of its K-neighbours.

2.6.2 Deep Learning Models

Now we are going to mention the DL algorithms below.

2.6.2.1 Multiple Layer Perceptron (MLP)

Multilayer Perceptron is a type of feedforward artificial neural network that has several layers of nodes in it, we define them by : an input layer that receives feature vectors , several hidden layers and the output layer that produces the prediction , it is composed from nodes also called neurons that are connected all together so that every neuron in one layer connects to every other neuron in the layer below, each node computes a weighted sum of inputs and passed the result.

2.6.3 Gated Recurrent Unit GRU

Gated Recurrent Unit (GRU) is made to process sequential data and identify temporal dependencies. It is made up of gating systems that control information flow, allowing the model to remember or forget data over time.It consist of two primary gates:

Update Gate: Regulates the amount of the prior hidden state that should be preserved.

Reset Gate: Decides How much of the prior concealed state to be forgotten.

2.6.3.1 Long Short Term Memory (LSTM)

LSTM is designed to avoid the long-term dependency issue,due to his capability of remembering data for long periods, its architecture is more complex because it is constituted of 4 hidden layers. The principal components of LSTM are the cell state which represents the principal component and other 3 gates which are:

- **Forget Gate**:specifies which historical data should be deleted.
- **Input Gate**:determines what new data should be stored.
- **Output Gate**:decides what information will be displayed in the output.

2.7 Conclusion

This chapter presented all the necessary informations and calculations that helps measuring all operations while building the federated model.

Chapter 3

Experimentation and Results

3.1 Introduction

This chapter presents a deep dive into the experimental pipeline, architecture configurations, and evaluation metrics used to validate the effectiveness of our Federated Learning (FL) approach in building a privacy-preserving Intrusion Detection System (IDS). The aim is to demonstrate that we can detect various types of network attacks with high accuracy and reliability—all while keeping sensitive data decentralized.

We start by detailing the technical environment and software stack employed. Next, we describe the dataset UNSW-NB15 and the necessary preprocessing steps to standardize and balance it. Then, we introduce the models tested, emphasizing the improvements made in the MLP architecture. Finally, we break down our FL simulation setup, discuss the results of various experiments, and conclude with visual analytics, a discussion of the findings, and the rationale behind the final model's performance.

Ultimately, our iterative experimentation process led to the development of an enhanced federated MLP model that surpassed 99% in key metrics including accuracy, precision, recall, and F1-score—surpassing not only previous FL-based IDS frameworks but also centralized baselines.

3.2 Experimental Setup

All experiments were conducted in a controlled yet realistic setup that we are going to cite in this section.

3.2.1 Hardware and Software Environment

In this section we will present all the simulation environment and tools in table 3.1 :

Component	Specification
Processor	Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz, 2496 MHz, 2 Core(s), 4 Logical Processor(s)
RAM	8.00 GB
GPU	Intel(R) HD Graphics 520
Operating System	Microsoft Windows 10 Pro
Platform	Kaggle Cloud Notebooks (GPU-accelerated)
Programming Language	Python 3.10

Table 3.1: System software and hardware environment

3.2.2 Model Simulation Setup

Each client trained its own local model and returned weights or deltas to the server. The server computed the global update and sent it back. Below are the main parameters of the simulation.

- **Simulated Clients:** 10 nodes representing separate data owners
- **Rounds:** 20 communication rounds per experiment
- **Aggregation Techniques:** Mentioned and explained in chapter 2
  - Weighted Aggregation (FedAvg)
  - Difference Aggregation (based on model delta)
- **Local Training:** 5 epochs per round, batch size of 128 , with learning rate 0.001
- **Evaluation Metrics:** Accuracy, Precision, Recall, and F1-score measured globally.
- **Visual Analytics (VA):**Used after the calculations of metrics,it represents the results in a visual way for better understanding, in the context of our work, visual analytics that were employed are:
  - **Confusion Matrix**:A table used to determine a classification model's performance is called a confusion matrix. It helps identify the areas in which the model is flawed by offering a thorough analysis of the differences between the predictions and the actual outcomes.

- **Learning Curves**:It is a graphical representation that shows how proficiency improves with increasing experience or practice over time. Simply put, it visually demonstrates how long it takes to acquire new skills or knowledge.
- **Bar Chart**:It is a chart type for graphing categorical data. It is made up of several rectangles that are all aligned to the same baseline, and each rectangle's length corresponds to the value it represents.

3.3 Model Implementation and Building

The objective is to design a system capable of detecting intrusions using distributed training across several clients, so in this section we are going to cite all the steps we gone through while building it :

3.3.1 Data Preparation

We explained this step before , it is the first step involved preprocessing the dataset we worked with which is UNSW-NB15 dataset , it included operations which are all mentioned in chapter 2.

Important Note :During the generating of our model we dropped the id column because it's a unique identifier with no predictive value not a feature, if we kept it that will introduce noise and ending negatively affecting the model's learning process. Also, we removed the category of the attacks column because it introduces redundancy with the binary target label (either 1 or 0) since it is a binary classification (either attack or normal) and will lead to inflate model performance , so it is not required.

3.3.2 Model Selection

Several models were implemented and evaluated under a federated learning setup, each model was wrapped with appropriate training and evaluation functions, then they were compared to choose the most powerful model for the simulation.

After evaluating each model and getting results we will develop the model with high results using federated learning setup and get the final results.

3.4 Results and Evaluation

This section will display all the results and metrics values.

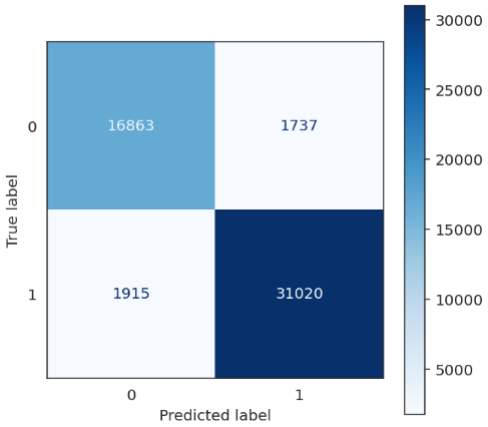
3.4.1 Centralized Training Baselines

Before applying federated learning, we trained all models in a centralized manner to establish benchmarks, and we are going to show results :

3.4.1.1 k-Nearest Neighbors (KNN)

```
Accuracy: 92.91%
Recall: 92.91%
Precision: 92.93%
F1-Score: 92.92%
time to train: 0.02 s
time to predict: 164.55 s
total: 164.57 s
```

(a) Metrics Values of KNN Model



(b) Confusion Matrix for KNN Model

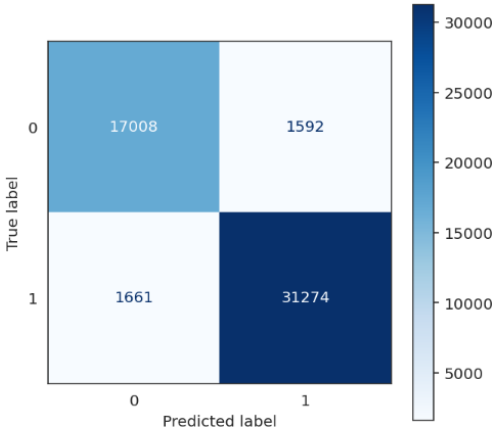
Figure 3.1: Results for K-Nearest Neighbors Model

The kNN model results shown in figure 3.1 achieved a relatively high accuracy of 92.91%. While its training phase was extremely fast, the prediction time was exceptionally high (164.5 seconds), making it computationally expensive for real-time IDS applications. This is due to its lazy learning nature, which requires comparison against the entire dataset at prediction time.

3.4.1.2 Decision Tree (DT)

```
Accuracy: 93.69%
Recall: 93.69%
Precision: 93.69%
F1-Score: 93.69%
time to train: 3.97 s
time to predict: 0.01 s
total: 3.99 s
```

(a) Metrics Values of DT Model



(b) Confusion Matrix for DT Model

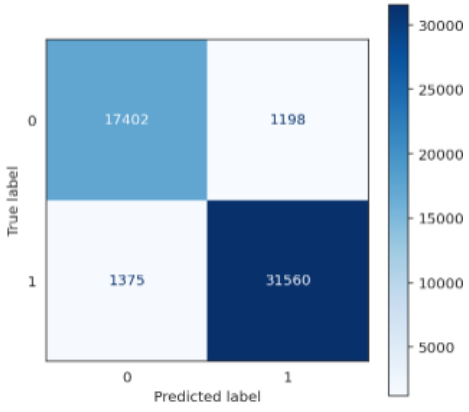
Figure 3.2: Results for Decision Tree Model

The Decision Tree model results shown in figure 3.2 performed with 93.69% accuracy. It provided quick training and prediction, and its ability to model non-linear decision boundaries was a clear advantage. However, Decision Trees tend to overfit the data, which could limit generalizability on unseen attacks.

3.4.1.3 Random Forest (RF)

```
Accuracy: 95.01%
Recall: 95.01%
Precision: 95.02%
F1-Score: 95.01%
time to train: 12.92 s
time to predict: 0.31 s
total: 13.23 s
```

(a) Metrics Values of RF Model



(b) Confusion Matrix for RF Model

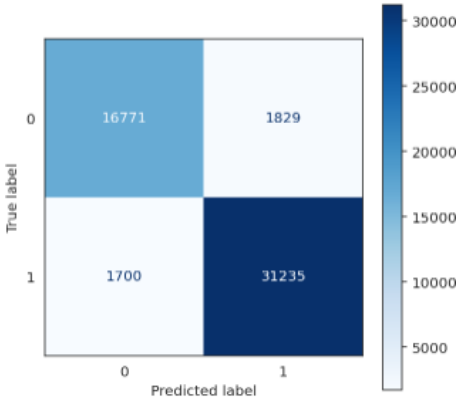
Figure 3.3: Results for Random Forest Model

Random Forest results shown in figure 3.3 achieved the best result among classical ensemble models with 95.01% accuracy. Its ability to combine multiple decision trees reduced overfitting and improved generalization. Despite its higher training time (12.92 seconds), it offers a solid trade-off between accuracy and efficiency.

3.4.1.4 Light Gradient Boosting Machine (LightGBM)

```
Accuracy: 93.15%
Recall: 93.15%
Precision: 93.14%
F1-Score: 93.15%
time to train: 126.47 s
time to predict: 0.09 s
total: 126.56 s
```

(a) Metrics Values of LightGBM Model



(b) Confusion Matrix for LightGBM Model

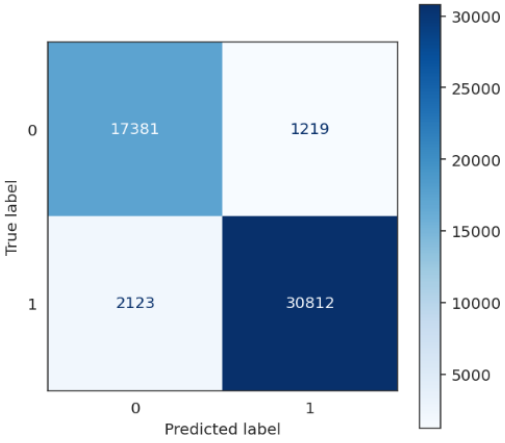
Figure 3.4: Results for Light Gradient Boosting Machine Model

LightGBM results shown in figure 3.4 reached 93.15% accuracy and similar recall and precision values, but with an F1-score slightly lower at 92.15%. It is generally known for its speed and efficiency in handling large datasets, though in this case, its training time was relatively long (126.47 s), likely due to extensive hyper-parameter space. However, it maintained a fast prediction time (0.09 s), making it useful for deployment if training time is not a concern.

3.4.1.5 Multiple Layer Perceptron (MLP)

```
Accuracy: 93.52%
Recall: 93.52%
Precision: 93.64%
F1-Score: 93.55%
time to train: 93.70 s
time to predict: 0.04 s
total: 93.74 s
```

(a) Metrics Values of MLP Model

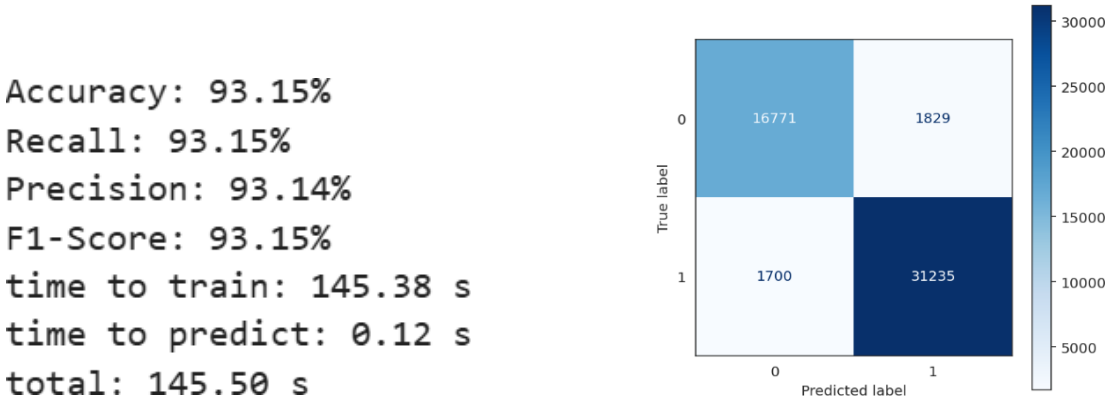


(b) Confusion Matrix for MLP Model

Figure 3.5: Results for Multiple Layer Perceptron Model

Results shown in figure 3.5 The Keras-based MLP offered deeper architectures with flexibility in tuning and regularization. Despite similar accuracy (93.52%), it required significantly more training time. However, its integration into the federated learning framework led to a highly optimized performance, reaching up to 98.94% accuracy after aggregation—demonstrating the advantage of deep learning in a collaborative setup.

3.4.1.6 Gated Recurrent Unit (GRU)

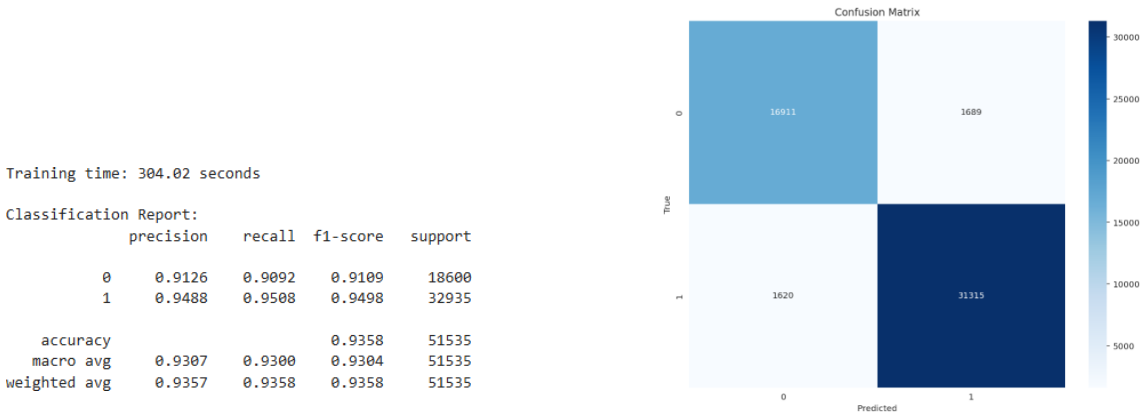


(a) Metrics Values of GRU Model **(b)** Confusion Matrix for GRU Model

Figure 3.6: Results for Gated Recurrent Unit Model

GRU results shown in figure 3.6 achieved 93.15% accuracy. Designed for sequential data. However, for tabular data like UNSW-NB15, they did not outperform the MLP models, possibly due to limited sequential relationships in features.

3.4.1.7 Long Short Term Memory (LSTM)



(a) Metrics Values of LSTM Model **(b)** Confusion Matrix for LSTM Model

Figure 3.7: Results for Long Short Term Memory Model

The LSTM model results shown in figure 3.7 achieved 93.58% across all metrics, slightly outperforming MLP in accuracy and F1-score. As a sequential deep learning model, it is well-suited for capturing temporal dependencies in network traffic data. However, its training and prediction times were significantly higher (304.02 s and 5.00 s, respectively), which may limit its applicability in time-sensitive environments unless further optimized.

3.4.2 Comparison of evaluated models

Here is table 3.2 that includes all informations about the evaluated models .

Model	Accuracy	Recall	Precision	F1-Score	Train Time (s)	Predict Time (s)	Total Time (s)
kNN	92.91%	92.91%	92.93%	92.92%	0.02	164.55	164.57
Decision Tree	93.69%	93.69%	93.69%	93.69%	3.97	0.01	3.99
Random Forest	**95.01%**	**95.01%**	**95.02%**	**95.01%**	**12.92**	**0.31**	**13.23**
LightGBM	93.15%	93.15%	93.14%	92.15%	126.47	0.09	126.56
GRU (Keras)	93.15%	93.15%	93.14%	93.15%	145.38	0.12	145.5
LSTM (Keras)	93.58%	93.58%	93.57%	93.58%	304.02	5.00	309.02
MLP (Keras)	**93.52%**	**93.52%**	**93.64%**	**93.55%**	**93.7**	**0.04**	**93.74**

Table 3.2: Performance Comparison of Different Models

In this study, a broad range of machine learning (ML) and deep learning (DL) models were evaluated using the UNSW-NB15 dataset to identify the most promising candidates for building a Federated Learning-based Intrusion Detection System (FL-IDS). Among the ML models tested, Random Forest achieved the highest overall performance with an accuracy of 95.01%, outperforming other models in terms of both speed and prediction quality. On the DL side, the Keras-based MLP (Multilayer Perceptron) emerged as the top-performing model, delivering competitive accuracy and robustness compared to other deep models like GRU and LSTM.

Given these results, two models were selected for further exploration in a federated learning setting: Random Forest as the strongest ML model and MLP as the most effective DL model. This selection aims to fairly evaluate how both paradigms perform when deployed in a distributed and privacy-preserving environment like federated learning.

The motivation behind choosing the MLP model in particular also stems from its natural compatibility with federated architectures. Deep learning models such as MLP are well-suited for distributed training using weight or gradient aggregation, which are essential in federated learning frameworks. Furthermore, MLP allows for advanced techniques like dropout, batch normalization, and fine-tuning, which make them more adaptable and

scalable in dynamic cybersecurity environments.

So, we are going to develop both of them in a federated-based and recompare them to decide which one is working best with IDS.

3.4.3 Federated Performance

We chose to implement our model with both RF and Keras MLP separated not hybrid model as we mentioned earlier, we applied the following steps:

3.4.3.1 Federated Random Forest (RF) Model

- **Model Type:** We used the Random Forest classifier, a robust ensemble learning method based on multiple decision trees, well-suited for classification tasks in intrusion detection.
- **Federated Learning Integration:** The training was decentralized by partitioning the dataset across 10 clients. Each client trained an independent Random Forest model locally on its own data.
- **Aggregation Method:** Rather than aggregating model weights (which is not applicable for ensemble trees), we implemented a prediction-level aggregation strategy using weighted soft voting. Each client's predicted probabilities on the test set were weighted based on its local F1-score before being combined into a global prediction.
- **Hyper-parameter Optimization:** After multiple iterations, we set the number of trees to 400, with a maximum depth of 35. We also fine-tuned 'min-samples-split', 'min-samples-leaf', and 'max-features' to improve generalization and accuracy.

  The final global model, formed after aggregation, achieved competitive results with an accuracy of **94.93%**, precision of **95.04%**, recall of **94.93%**, and F1-score of **94.87%**, demonstrating the model's strong detection capability in a federated setting.

The final results are shown in figure 3.8 below :

```
 Final Global Model Performance:
Accuracy:  0.9493
Precision: 0.9504
Recall:    0.9493
F1 Score:  0.9487
```

Figure 3.8: Final global model performance of Federated RF model

3.4.3.2 Federated-based RF Model Discussion

the Random Forest (RF) model was chosen as a representative of traditional ensemble-based machine learning algorithms,due to its well-known resilience, interpretability, and excellent high performance in a variety of supervised learning tasks. In this study, we investigated whether RF deployment is feasible in a federated learning (FL) context, in which models are trained separately on several clients and then combined centrally using weighted soft voting determined by the F1-score of each client. 10 clients participated in the federated RF implementation, and they were all trained locally using a stratified split of the UNSW-NB15 dataset. Each client generated its own probability predictions on the global test set following local training. Client F1-scores were utilized to calculate each fore-cast's contribution to the final global prediction, which was then combined using a weighted soft voting technique. These results provide dependable performance across all important criteria, confirming the effectiveness of RF in intrusion detection. Several factors contributed to this performance:

- Weight-based federated aggregation techniques aren't advantageous for RF. The ensemble's trees are individually taught and difficult to break down or combine among clients. This restricts the model's capacity for truly federated collaborative learning.
- With more trees and depth, RF models get more sophisticated and have larger memory capacities. In environments with limitations or edge devices, this can become difficult, particularly when implemented in federated configurations.
- Random Forest does not naturally support continued training across rounds, which is a key component of federated iterative optimization.

3.4.3.3 Federated MLP (Keras) Model

- **Architecture Optimization:** We implemented a deep feed-forward neural network with three fully connected (dense) layers using the ReLU activation function, followed by a final linear output layer with two legits, suitable for binary classification using cross-entropy loss.
- **Optimizer and Learning Rate:** The model was trained using the Adam optimizer with a learning rate of 0.001, which provided adaptive and stable convergence.
- **Federated Learning Integration:** The MLP model was integrated into a federated learning framework using both **Weighted Aggregation (FedAvg)** and

Weight Difference Aggregation across 10 clients, preserving data locality and privacy.

- **Hyper-parameter Tuning:** Several values for batch size, epochs (50 per round), number of rounds (20), and hidden layer sizes (e.g., 20-20) were tested to achieve optimal performance.
- **Feature Normalization:** The input data from the UNSW-NB15 dataset underwent log transformation, chi-squared feature selection, one-hot encoding, and standardization to ensure consistent training and reduce noise.

These improvements enabled the Federated MLP model to achieve high performance, with accuracy and F1-scores exceeding 98%, demonstrating its robustness and scalability in a distributed intrusion detection context.

The final results are shown for both weighted aggregation in figure 3.9 and difference aggregation in figure 3.10 :

```
Final Test Metrics with Weighted Average Aggregation:
Accuracy: 0.9858, Precision: 0.9864, Recall: 0.9919, F1 Score: 0.9891
```

Figure 3.9: Performance metrics using Weighted Aggregation for Federated MLP model

```
Final Test Metrics with Weight Difference Aggregation:
Accuracy: 0.9862, Precision: 0.9870, Recall: 0.9918, F1 Score: 0.9894
```

Figure 3.10: Performance metrics using Difference Aggregation for Federated MLP model

3.4.3.4 Federated-based MLP Model Discussion

This study's deep learning accurate, the Multilayer Perceptron (MLP) model, was chosen because of its capacity to represent intricate, non-linear patterns in data. The MLP architecture was implemented using PyTorch and included a final classification layer after several dense layers that were activated by ReLU algorithms. Weighted Aggregation (FedAvg) and Difference Aggregation are two distinct aggregation algorithms that were used to train the MLP model across ten dispersed clients in the federated configuration. Over several epochs and communication rounds, each client separately trained its local model; the global model was updated using the combined client parameters at the end of each round. It achieved high results (beyond 98% in all metrics) which demonstrates how well the MLP adapts to dispersed learning settings and how well it works with the

iterative weight-based updates of federated learning. Several factors contributed to this performance:

- Deep neural networks like MLP are suitable for federated learning, as their weights can be easily aggregated and updated across clients using standard optimization strategies.
- The Model architecture works well for a variety of deployment scenarios, ranging from edge devices to cloud infrastructure, because it can be readily scaled up or down based on the computing power of the client devices.
- The MLP model demonstrated smooth and continuous improvement across communication rounds,this allowed the global model to benefit from local patterns learned by each client.

3.4.4 Comparison Between MLP and RF Models

In the table 3.3 we are going to compare both of the federated models and decide which one is the best for our final result:

Aspect	Federated MLP (Keras)	Federated Random Forest (Sklearn)
Algorithm Type	Deep Learning (Neural Network)	Machine Learning (Ensemble Trees)
Architecture	3 Dense Layers (ReLU) with Dropout, Sigmoid output	400 Decision Trees, Depth=35, Bootstrap, Balanced Weights
Federated Setup	Model weight aggregation (FedAvg and Weight Difference)	Prediction-level aggregation via weighted soft voting
Client Count	10	
Accuracy (%)	**98.62**	94.93
Precision (%)	**98.70**	95.04
Recall (%)	**99.18**	94.93
F1-Score (%)	**98.94**	94.87
Training Time	Higher (DL training loops per round)	Faster (Single pass tree training per client)
Flexibility for FL	High (supports dropout, fine-tuning, regularization)	Limited (tree-based models cannot be aggregated via weights)
Scalability	Highly scalable for deep FL systems	Moderate scalability
Best Use Case	Real-time federated learning with adaptive models	Fast local training in low-resource or ensemble-focused FL

Table 3.3: Comparison between Federated MLP and Federated Random Forest Models

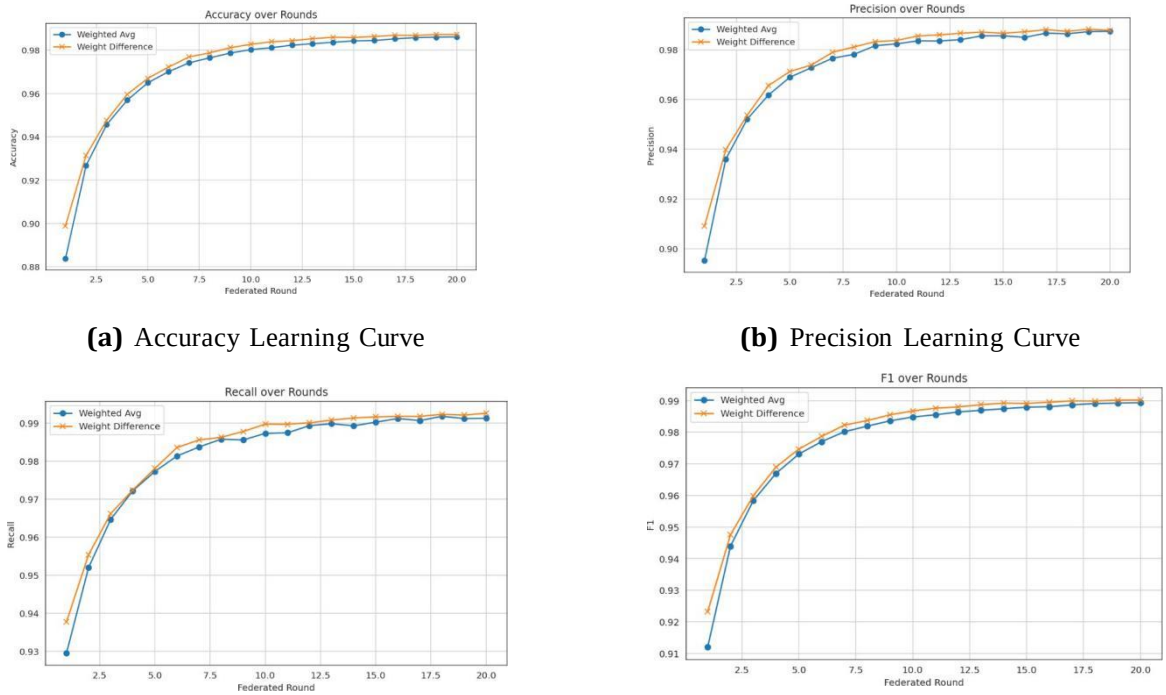
The MLP model was selected over the Random Forest model for a number of important reasons based on the comparison of the two models' federated implementations. The federated MLP outperformed the federated Random Forest, which performed well (94.93% accuracy, 94.87% F1-score), by achieving a much greater performance (about 98% across all important criteria). The choice of MLP as the main model for our suggested federated intrusion detection system is justified by the significant improvement in accuracy, precision, recall, and F1-score.

Furthermore, because of its increased scalability in dispersed contexts and compatibility with weight-based aggregation, MLP which is a deep learning model, is more naturally suited to federated learning systems. As a result, using MLP supports the objectives of federated learning systems for scalability and architectural compatibility in addition to performance supremacy.

3.4.5 Visual Analysis

Since the best model for our Federated-based Intrusion Detection System is MLP ,we are going to present its final results we had in visual analysis in this section.

Below in figure 3.11 we present learning curves for both aggregation methods on the same axes for all the metrics : accuracy,precision,recall and F1-Score:



(a) Accuracy Learning Curve

(b) Precision Learning Curve

(c) Recall Learning Curve

(d) F1-Score Learning Curve

Figure 3.11: Final Learning Curves for both aggregation method

Below in figure 3.12 we present a bar chart that compares each metric result for both aggregation methods utilized :

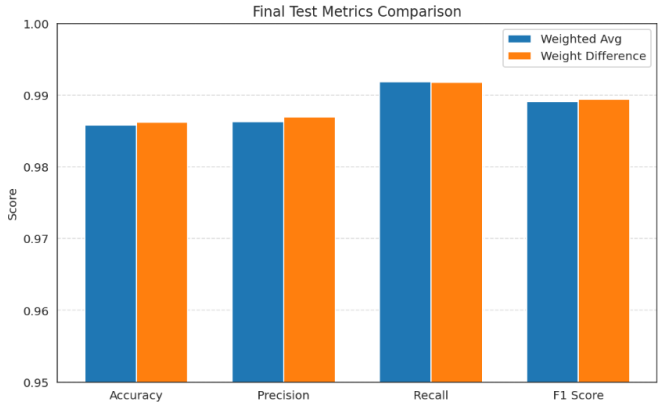
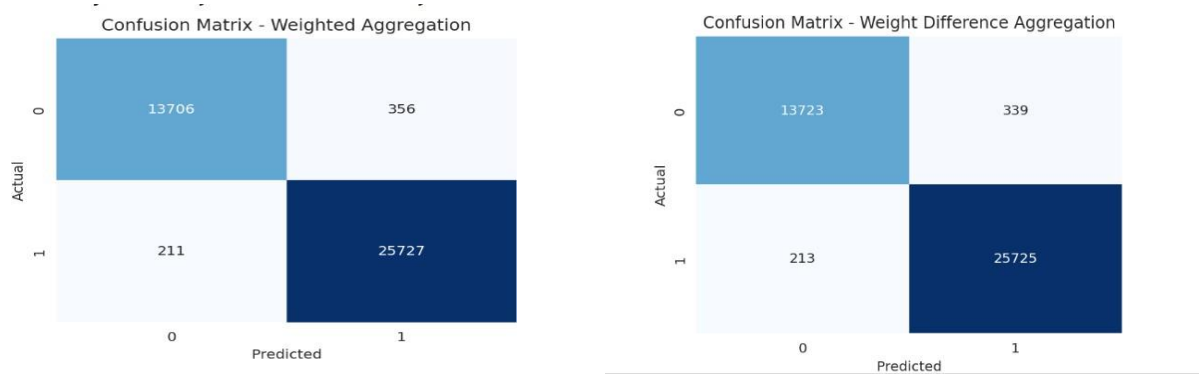


Figure 3.12: Final Performance Metrics Comparison Bar Chart

We present in figure 3.13 below confusion matrix for each aggregation method :



(a) Confusion Matrix for Weighted Aggregation **(b)** Confusion Matrix for Difference Aggregation

Figure 3.13: Final classification results of the global model using both aggregation methods.

3.5 Global Discussion

The project's experimental results demonstrate Federated Learning's (FL) enormous promise in cybersecurity, especially for intrusion detection tasks. The improved Keras-based Multilayer Perceptron (MLP) continuously showed the best performance out of all tested models. It has a clear edge over traditional machine learning models such as Random Forest because of its capacity to learn intricate, non-linear patterns in a decentralized context. While careful architecture and training parameter adjustment allowed for effective and stable convergence across multiple rounds. The comparison of the two aggregation strategies(Weighted Aggregation and Difference Aggregation) produced a particularly interesting findings. After 20 communication rounds, both performed well. FL proved its capacity to deliver both privacy preservation and top-tier speed with the right architecture, optimized hyper-parameters, and a successful aggregation the technique. The MLP model demonstrated that FL is a workable solution—rather than merely a theoretical concept—for actual security applications by achieving approximately 98% in all significant performance criteria. Furthermore, the MLP model's success supports the feasibility of implementing deep learning in distributed, resource-constrained contexts. The architecture was robust enough to identify a variety of assaults while being small enough to be deployed on edge or Internet of Things devices. This opens up a viable approach for privacy-preserving, real-time intrusion detection across dynamic infrastructures. These findings confirm that Federated Learning can provide a solid basis for next-generation intrusion detection systems by providing a balance between scalability, privacy, and accuracy when combined with an optimized deep learning architecture.

3.6 Conclusion

Through thoughtful experimentation and architecture refinement, we built a federated IDS model that reaches beyond 98% accuracy, precision, recall, and F1-score while maintaining data privacy. These results validate the power of federated learning as a secure and scalable alternative to centralized IDS models.

General Conclusion

In this thesis, we explored the integration of Federated Learning (FL) with Intrusion Detection Systems (IDS) to address the growing challenges in cybersecurity. As cyber threats become increasingly complex and user data becomes extremely sensitive, traditional IDS models that rely on centralized data collection are no longer adequate and sufficient. These problems motivated us to design a distributed, privacy-preserving solution capable of detecting modern threats while ensuring data confidentiality.

To this end, we implemented and compared several machine learning and deep learning models, including Random Forest, LightGBM, GRU, LSTM, and particularly Multi-Layer Perceptron (MLP). After evaluating these models on the UNSW-NB15 dataset, our experiments showed that the federated MLP model achieved remarkable results, surpassing 98% in all metrics we evaluated which are accuracy, precision, recall, and F1-score. These findings confirmed the effectiveness of deep learning in detecting complex attacks and the feasibility of using federated architectures in cybersecurity.

Despite these encouraging results, some challenges remain, most notably non-IID data, communication costs, and resource constraints on client devices. These are important issues that must be addressed to extend federated systems to real-world environments. Furthermore, implementing FL requires more careful design to prevent potential attacks on the assembly process and maintain model robustness across different client devices.

Looking forward, several promising directions can be considered. Future research may focus on integrating differential privacy and secure aggregation to strengthen confidentiality,also exploring personalized federated learning for adaptive intrusion detection,and without forgetting the test of this system in real-time environments with multiple organizations and heterogeneous devices.

In conclusion, this work contributes to the ongoing efforts to design intelligent, distributed, and privacy-preserving intrusion detection systems. It opens up new possibilities for collaborative cybersecurity models and sets a foundation for future developments in secure federated intelligence.

Bibliography

[Agrawal et al., 2022] Agrawal, S., Sarkar, S., Aouedi, O., Yenduri, G., Piamrat, K., Alazab, M., Bhattacharya, S., Maddikunta, P. K. R., and Gadekallu, T. R. (2022). Federated learning for intrusion detection system: Concepts, challenges and future directions. *Computer Communications*, 195:346–361.

[Bace et al., 2001] Bace, R. G., Mell, P., et al. (2001). Intrusion detection systems.

[Banabilah et al., 2022] Banabilah, S., Aloqaily, M., Alsayed, E., Malik, N., and Jararweh, Y. (2022). Federated learning review: Fundamentals, enabling technologies, and future applications. *Information processing & management*, 59(6):103061.

[Brik et al., 2020] Brik, B., Ksentini, A., and Bouaziz, M. (2020). Federated learning for uavs-enabled wireless networks: Use cases, challenges, and open problems. *IEEE Access*, 8:53841–53849.

[Bukhari et al., 2024] Bukhari, S. M. S., Zafar, M. H., Abou Houran, M., Moosavi, S. K. R., Mansoor, M., Muaaz, M., and Sanfilippo, F. (2024). Secure and privacy-preserving intrusion detection in wireless sensor networks: Federated learning with scnn-bi-lstm for enhanced reliability. *Ad Hoc Networks*, 155:103407.

[Caville et al., 2022] Caville, E., Lo, W. W., Layeghy, S., and Portmann, M. (2022). Anomal-e: A self-supervised network intrusion detection system based on graph neural networks. *Knowledge-based systems*, 258:110030.

[Chassagnon et al., 2020] Chassagnon, G., Vakalopolou, M., Paragios, N., and Revel, M.-P. (2020). Deep learning: definition and perspectives for thoracic imaging. *European radiology*, 30:2021–2030.

[Chauhan and Chandra, 2013a] Chauhan, P. and Chandra, N. (2013a). A review on hybrid intrusion detection system using artificial immune system approaches. *International Journal of Computer Applications*, 68(20).

[Chauhan and Chandra, 2013b] Chauhan, P. and Chandra, N. (2013b). A review on hybrid intrusion detection systems. *International Journal of Computer Applications*, 68(20).

[Chergui et al., 2020] Chergui, H. et al. (2020). Noc-ids: Network ontology context-based intrusion detection system. *Expert Systems with Applications*, 140:112881.

[Fadhil et al., 2024] Fadhil, H. M., Dawood, Z. O., and Al Mhdawi, A. (2024). Enhancing intrusion detection systems using metaheuristic algorithms. *Diyala Journal of Engineering Sciences*, pages 15–31.

[Fedorchenko et al., 2022] Fedorchenko, E., Novikova, E., and Shulepov, A. (2022). Comparative review of the intrusion detection systems based on federated learning: Advantages and open challenges. *Algorithms*, 15(7):247.

[Friha et al., 2022] Friha, O., Ferrag, M. A., Shu, L., Maglaras, L., Choo, K.-K. R., and Nafaa, M. (2022). Felids: Federated learning-based intrusion detection system for agricultural internet of things. *Journal of Parallel and Distributed Computing*, 165:17–31.

[Ghanbarzadeh et al., 2023] Ghanbarzadeh, R., Hosseinalipour, A., and Ghaffari, A. (2023). A novel network intrusion detection method based on metaheuristic optimisation algorithms. *Journal of ambient intelligence and humanized computing*, 14(6):7575–7592.

[Ghimire and Rawat, 2022] Ghimire, B. and Rawat, D. B. (2022). Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things. *IEEE Internet of Things Journal*, 9(11):8229–8249.

[Gosselin et al., 2022] Gosselin, R., Vieu, L., Loukil, F., and Benoit, A. (2022). Privacy and security in federated learning: A survey. *Applied Sciences*, 12(19):9901.

[Halim et al., 2021] Halim, Z., Yousaf, M. N., Waqas, M., Sulaiman, M., Abbas, G., Hussain, M., Ahmad, I., and Hanif, M. (2021). An effective genetic algorithm-based feature selection method for intrusion detection systems. *Computers & Security*, 110:102448.

[Issa et al., 2024] Issa, M. M., Aljanabi, M., and Muhialdeen, H. M. (2024). Systematic literature review on intrusion detection systems: Research trends, algorithms, methods, datasets, and limitations. *Journal of Intelligent Systems*, 33(1):20230248.

[Khraisat et al., 2019] Khraisat, A., Gondal, I., Vamplew, P., and Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22.

[Kreuzberger et al., 2023] Kreuzberger, D., Kühl, N., and Hirschl, S. (2023). Machine learning operations (mlops): Overview, definition, and architecture. *IEEE access*, 11:31866–31879.

[Kus et al., 2022] Kus, B. et al. (2022). A false sense of security? revisiting the effectiveness of industrial intrusion detection systems on real network traffic. *Computers & Security*, 114:102580.

[Li et al., 2019] Li, J., Qu, Y., Chao, F., Shum, H. P., Ho, E. S., and Yang, L. (2019). Machine learning algorithms for network intrusion detection. *AI in Cybersecurity*, pages 151–179.

[Li et al., 2020a] Li, K., Zhou, H., Tu, Z., Wang, W., and Zhang, H. (2020a). Distributed network intrusion detection system in satellite-terrestrial integrated networks using federated learning. *IEEE Access*, 8:214852–214865.

[Li et al., 2020b] Li, L., Fan, Y., Tse, M., and Lin, K.-Y. (2020b). A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854.

[Li et al., 2020c] Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2020c). Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60.

[Liao et al., 2013] Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., and Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of network and computer applications*, 36(1):16–24.

[Logeswari et al., 2023] Logeswari, G., Bose, S., and Anitha, T. (2023). An intrusion detection system for sdn using machine learning. *Intelligent Automation & Soft Computing*, 35(1):867–880.

[Martin,] Martin, J. How many cyber attacks occur each day?(2024).

[Maseno et al., 2022] Maseno, E. M., Wang, Z., and Xing, H. (2022). A systematic review on hybrid intrusion detection system. *Security and Communication Networks*, 2022(1):9663052.

[Moustafa and Slay, 2015] Moustafa, N. and Slay, J. (2015). Unsw-nb15: a comprehensive data set for network intrusion detection systems. In *2015 Military Communications and Information Systems Conference (MilCIS)*.

[Nimbalkar and Kshirsagar, 2021] Nimbalkar, P. and Kshirsagar, D. (2021). Feature selection for intrusion detection system in internet-of-things (iot). *ICT Express*, 7(2):177–181.

[Omar et al., 2019] Omar, M. et al. (2019). A world of cyber attacks (a survey).

[Othman et al., 2018] Othman, S. M., Alsohybe, N. T., Ba-Alwi, F. M., and Zahary, A. T. (2018). Survey on intrusion detection system types. *International Journal of Advanced Computer Science and Applications*, 9(6):241–252.

[Raj and Sharma, 2020] Raj, Y. and Sharma, K. S. (2020). Comparison of host-based and network-based intrusion detection system: A review. *International Journal of Computer Sciences and Engineering*, 8(2):115–119.

[Rajapaksha et al., 2023] Rajapaksha, S., Kalutarage, H., Al-Kadri, M. O., Petrovski, A., Madzudzo, G., and Cheah, M. (2023). Ai-based intrusion detection systems for in-vehicle networks: A survey. *ACM Computing Surveys*, 55(11):1–40.

[Satilmiş et al., 2024] Satilmiş, H., Akleylek, S., and Tok, Z. Y. (2024). A systematic literature review on host-based intrusion detection systems. *Ieee Access*, 12:27237–27266.

[Scarfone and Mell, 2007] Scarfone, K. and Mell, P. (2007). Guide to intrusion detection and prevention systems (idps). Technical Report SP 800-94, National Institute of Standards and Technology.

[Shen et al., 2021] Shen, C., Xu, J., Zheng, S., and Chen, X. (2021). Resource rationing for wireless federated learning: Concept, benefits, and challenges. *IEEE Communications Magazine*, 59(5):82–87.

[Snapp et al., 1991] Snapp, S. R., Brentano, J., Dias, G. V., Goan, T. L., Heberlein, L. T., Ho, C.-L., Levitt, K. N., Mukherjee, B., Smaha, S. E., Grance, T., et al. (1991). Dids (distributed intrusion detection system)-motivation, architecture, and an early prototype. In *Proceedings of the 14th national computer security conference*, volume 1, pages 167–176. Washington, DC.

[Sundaram, 1996] Sundaram, A. (1996). An introduction to intrusion detection. *Crossroads*, 2(4):3–7.

[Tsai et al., 2009] Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., and Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *expert systems with applications*, 36(10):11994–12000.

[Tseng et al., 2003] Tseng, C.-Y., Balasubramanyam, P., Ko, C., Limprasittiporn, R., Rowe, J., and Levitt, K. (2003). A specification-based intrusion detection system for aodv. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 125–134.

[Wolsing et al., 2023] Wolsing, K. et al. (2023). Expectations vs reality: A comparative study of intrusion detection datasets and their effect on ml models. In *IEEE Symposium on Security and Privacy*.

[Yacouby and Axman, 2020] Yacouby, R. and Axman, D. (2020). Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In *Proceedings of the first workshop on evaluation and comparison of NLP systems*, pages 79–91.

[Yang et al., 2019] Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19.

[Yuan et al., 2024] Yuan, L., Wang, Z., Sun, L., Yu, P. S., and Brinton, C. G. (2024). Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 11(21):34617–34638.

[Zarringhalami and Rafsanjani, 2012] Zarringhalami, Z. and Rafsanjani, M. K. (2012). A survey on intrusion detection systems in computer networks. *Journal of applied mathematics & informatics*, 30(5_6):847–864.

[Zhang et al., 2020] Zhang, H., Bosch, J., and Olsson, H. H. (2020). Federated learning systems: Architecture alternatives. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, pages 385–394. IEEE.

[Zhang et al., 2019] Zhang, J., Li, F., Zhang, H., Li, R., and Li, Y. (2019). Intrusion detection system using deep learning for in-vehicle security. *Ad Hoc Networks*, 95:101974.