Collaborative Anomaly-based Intrusion Detection Systems in Lightweight IoT

Suzan Hajj

THÈSE DE DOCTORAT

DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE-FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE FRANCHE-COMTÉ

École doctorale n°37

Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

par

SUZAN HAJJ

Collaborative Anomaly-based Intrusion Detection Systems in Lightweight IoT

Systèmes Collaboratifs de Détection d'Intrusion basés sur les Anomalies dans l'Internet des Objets

Thèse présentée et soutenue à Dijon, le 21 November 2023

Composition du Jury :

PROFESSEUR ABDALLAH MAKHOUL	Université de Franche-Comté	Président
PROFESSEUR SALIMA BENBERNOU	Université Paris Cité	Rapporteuse
PROFESSEUR PASCAL LORENZ	Université de Haute Alsace	Rapporteur
MAÎTRE DE CONFÉRENCE KINDA KHAWAM	Université de Versailles	Examinatrice
MAÎTRE DE CONFÉRENCE JACQUES BOU ABDO	University of Cincinnati	Examinateur
PROFESSEUR DOMINIQUE GINHAC	Université de Bourgogne Franche-Comté	Directeur de thèse
PROFESSEUR JACQUES DEMERJIAN	Université Libanaise	Directeur de thèse
PROFESSEUR CHRISTOPHE GUYEUX	Université de Franche-Comté	Codirecteur de thèse

N° | X | X | X |

ACKNOWLEDGEMENTS

In reaching the end of this academic journey, my deepest gratitude goes to those who made my PhD dreams a reality. Thanks to my husband, Ziad Hobeiche, for his unwavering support, guiding me through every twist. To my children, Fawzi, Angelina, and Majd, your enduring love inspired me forward.

Heartfelt appreciation to my parents, family, and friends for their unwavering belief, reinforcing my commitment to perseverance.

Special thanks to my supervisors, Dr. Jacques Demerdjian, Dr. Dominique Ghinhac, and Christoph Guyeux, for shaping my thesis and enriching my skills.

Gratitude to my colleagues, whose silent contributions were vital. To everyone who shared in the challenges and triumphs, thank you for being the driving force.

My sincere thanks to each of you for your indispensable role in making my dream a reality.

Contents

List of Abbreviations

Abbreviation	Definition
AE	Autoencoder
AIDS	Anomaly-based Intrusion Detection System
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
BS	Backing Sampling
CAIDA	Center for Applied Internet Data Analysis
CANN	Clustering-Based Anomaly Neural Network
CIDDS	Coburg Intrusion Detection Dataset
CPU	Central Processing Unit
CR	Change Ratio
CS	Chain Sampling
CS+	Chain+ Sampling
DARPA	Defense Advanced Research Project Agency
DDoS	Distributed Denial of Service
DEFSW	Deterministic Sampling over a Fixed Window
DEFTW	Deterministic Sampling over a Sliding Window
DNN	Deep Neural Networks
DoS	Denial of Service
EM	Expectation Maximization
ELNIDS	Ensemble Learning IDS
EWMA	Exponentially Weighted Moving Average
Fβ-score	F-beta score
FP	False Positive
FTP	File Transfer Protocol
FPGA	Field Programmable Gate Array
FN	False Negative
GA	Genetic Algorithm
GANs	Generative Adversarial Networks
GHz	Gigahertz
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HTTP	Hypertext Transfer Protocol

Abbreviation	Definition
IDC	International Data Corporation
IDS	Intrusion Detection System
IMAP	Internet Message Access Protocol
IoT	Internet of Things
ISCX	Information Security Centre of Excellence
ISO	International Organization for Standardization
ITU	International Telecommunication Union
KDD	Knowledge Discovery and Data Mining
KDE	Kernel Density Estimation
KNN	K Nearest Neighbors
LOF	Local Outlier Factor
MAC	Media Access Control
MCD	Minimum Covariance Determinant
MSM	Model Selection Method
MLP	Multilayer Perceptron
ML	Machine Learning
NSL-KDD	Network Security Laboratory-KDD
OS	Overall Statistic
PCA	Principal Component Analysis
PS	Priority Sampling
QoS	Quality of Service
RAM	Random Access Memory
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
RP	Random Pairing Sampling
RS	Reservoir Sampling
R2L	Remote-to-Local
SS	Stratified Sampling
SIDS	Signature-based Intrusion Detection System
SNMP	Simple Network Management Protocol
SMLC	Semi-Supervised Multi-Layered Clustering
SRSSW	Simple Random Sampling over a Sliding Window
SRSFW	Simple Random Sampling over a Fixed Window
SPC	Statistical Process Control
STA	Semi-Supervised Tri-Adaboost
SS	Stratified Sampling
SVM	Support Vector Machine
TCP	Transmission Control Protocol

Abbreviation	**Definition**
TN	True Negative
TP	True Positive
TUIDS	Traffic Usage and Intrusion Detection Scenario
UDP	User Datagram Protocol
U2R	User-to-Root
WRS	Weighted Random Sampling without Replacement
ZB	Zettabytes

1

INTRODUCTION

The Internet has become an omnipresent and indispensable part of our lives, expanding at an astonishing pace. Its global reach and influence continue to grow exponentially, fundamentally transforming the way we communicate, access information, and conduct business. Recent statistics project a mind-boggling 175 zettabytes of generated data worldwide by 2025, with an annual growth rate of approximately 61% (IDC, 2018). Moreover, the number of Internet users has surpassed 4.9 billion, accounting for nearly 63% of the world's population (ITU, 2021). These remarkable figures underscore the relentless expansion of the internet, necessitating a comprehensive exploration and analysis to comprehend its profound and far-reaching impact on society.

With the evolution of the Internet, malicious actors have also evolved their techniques to exploit vulnerabilities for financial gain, data theft, extortion, and denial of service attacks. Intrusion Detection Systems (IDSs) play a vital role in monitoring and analyzing network traffic to identify and respond to unauthorized or suspicious activities, as these attacks pose significant threats to network security and integrity.

The proliferation of Internet of Things (IoT) devices, driven by advancements in computer technology and affordability, has resulted in a vast network generating massive amounts of data. However, relying solely on cloud-based processing for IoT data presents challenges such as latency, bandwidth constraints, and privacy concerns. Processing data at the edge, closer to its source, has emerged as a solution to overcome these limitations, enabling faster response times and reduced network congestion.

IoT devices have revolutionized various industries and transformed our daily lives, but they have also introduced unprecedented security challenges. IDSs play a critical role in identifying and mitigating these threats, ensuring the security and integrity of IoT networks.

Machine learning (ML) is a fundamental aspect of artificial intelligence that enables computational systems to learn and improve from data without explicit programming. In the context of IDS, ML techniques are employed to identify and classify malicious activities

and network attacks by analyzing network traffic data. Through training on extensive datasets, ML algorithms can recognize and flag suspicious or anomalous behavior.

However, deploying ML models for IDS faces two significant challenges. Firstly, the computational demands of these models, coupled with the vastness of the datasets involved, present obstacles, especially for resource-constrained IoT devices. Lightweight and efficient ML algorithms are crucial to ensure their suitability without compromising performance.

Secondly, training ML models for IDS heavily relies on large-scale datasets comprising real network traffic. However, privacy concerns surrounding sensitive information in these datasets pose challenges for data collection and sharing. Developing novel learning algorithms that strike a balance between the necessity for extensive training data and privacy-preserving techniques is essential to generate accurate models while safeguarding privacy. Addressing the challenges of computational overhead and privacy-preserving training processes concurrently represents a critical research endeavor for the effective development and deployment of IDS within IoT environments.

1.1/ MOTIVATION

Effective IDSs are crucial in today's interconnected and digitized world. Networks face complex and sophisticated attacks, posing significant threats to their security and integrity. Traditional IDS approaches have evolved to adapt to emerging threats, utilizing techniques such as signature-based detection, anomaly-based detection, and behavior-based detection. However, these approaches struggle to keep pace with the evolving threat landscape and the scale of modern networks.

The challenges of IDS are amplified in IoT networks, where limited computational power and the sheer volume of network traffic pose additional hurdles. Novel learning-based IDS, particularly those leveraging ML algorithms, have emerged as promising solutions. ML empowers IDS to autonomously learn from vast amounts of data and adapt to the network environment. Despite the evolution of lightweight IDSs to provide security for thin nodes within IoT networks operating in untrusted environments, there is still a crucial missing component: an efficient sampling algorithm that complements the IDS and minimizes resource strain. Acknowledging the challenges imposed by limited resources and the impracticality of analyzing every packet, our research endeavors to develop a lightweight sampling algorithm specifically designed for thin IoT nodes. Our goal is to create an algorithm that operates at low sampling ratios, enabling us to extract representative packets for analysis. By addressing these challenges, our aim is to enhance the security posture of thin IoT nodes, ensuring the integrity and representativeness of the

collected data samples.

Given the resource constraints faced by IoT devices and the ever-evolving threat landscape, there is an urgent need to address the inherent challenges and enhance the effectiveness of IDSs in lightweight IoT networks. To tackle these obstacles, we acknowledge the significance of integrating sampling and intrusion detection through a cross-layer design approach. By embracing the potential of distributed (federated) processes in IDS, we seek to enable collaboration among multiple IoT devices in training models while preserving data privacy and enabling efficient knowledge sharing across the network. This approach not only addresses the limitations of individual devices but also promotes collaboration and information sharing to combat sophisticated attacks, while preserving privacy by eliminating the need for nodes to share their sensitive traffic datasets.

Motivated by the challenges encountered in anomaly-based intrusion detection, particularly the persistent issue of false positives, we aim to explore the potential of ensemble learning techniques to mitigate this problem. Previous research has shown that employing ensemble learning, such as a two-stage architecture combining unsupervised and supervised methods, can effectively reduce false positives. However, the suitability of unsupervised learning and the availability of labeled data pose limitations. Considering these factors and our research's focus on a semi-supervised novelty detection task with a limited labeled dataset, we propose a lightweight, semi-supervised, federated IDS for IoT devices. Our motivation stems from the desire to enhance the detection capabilities of IoT security systems while maintaining a low false positive rate over time, ensuring reliable and sustainable protection against malicious activities.

In conclusion, the motivations behind our research are driven by the need to overcome resource constraints, enhance the security posture of IoT networks, and address the evolving threat landscape. By developing efficient sampling algorithms, incorporating cross-layer design, and leveraging federated learning and ensemble techniques, we aim to create robust intrusion detection mechanisms for lightweight IoT environments. Our ultimate goal is to ensure the resilience, reliability, and sustainability of IoT-based systems, safeguarding them against sophisticated attacks and protecting the privacy and integrity of IoT data.

1.2/ RESEARCH OBJECTIVES

Our research aims to address the challenges posed by limited computational power, memory constraints, and the dynamic nature of IoT networks in order to enhance the effectiveness and efficiency of IDS. By leveraging advanced cluster sampling techniques, federated learning, cross-layer approaches, and ensemble learning, we seek to achieve

the following objectives

1. Conduct a comprehensive survey on existing sampling algorithms and perform a thorough comparative analysis in terms of execution speed and accuracy. The main objective of this survey is to evaluate the performance and effectiveness of various sampling techniques in capturing network traffic characteristics. Additionally, we aim to analyze the behavior and robustness of different network features under diverse sampling strategies and parameters. Moreover, we strive to identify attacks that exhibit resilience to the sampling process and determine a set of features that demonstrate greater stability. By providing valuable insights into the strengths and weaknesses of different sampling techniques, this survey will assist in identifying the most suitable approach for effective IoT intrusion detection.

2. Develop a novel Cluster-based Sampling technique specifically designed for thin IoT nodes within IoT networks. By leveraging clustering techniques, our algorithm will create representative samples from the data stream while considering the limited resources available. The objective is to overcome the limitations of existing sampling methods by ensuring high sample representativeness, low sampling error, and minimal resource utilization. Our proposed Cluster-based Sampling approach, with its focus on very small sampling ratios, provides an optimized and privacy-preserving solution that retains the essential characteristics of the traffic. This research endeavor aims to enhance the security posture of thin IoT nodes by addressing the challenges associated with sampling in resource-constrained environments and providing a tailored solution for lightweight IoT applications.

3. Investigate the application of cross-layer federated learning techniques in the context of intrusion detection in lightweight IoT. We proposed the baseline k-means, a semi-supervised novelty detection approach, based on the original K-means clustering technique. This technique leverages a small amount of labeled data to establish a baseline for learning and trains a classifier to label additional unlabeled data points. To adapt the K-means algorithm for intrusion detection, we adopt the approach of using a distance measure to a known baseline. This enables the identification of outliers or observations that significantly deviate from the distribution of the training data, even if they fall within a high-density region. By integrating these core ideas of the baseline k-means technique and distance-based anomaly detection into the cross-layer federated learning framework, we aim to develop a robust and privacy-preserving IDS for lightweight IoT environments. This integration allows for the collaborative training of models across multiple IoT devices, enhancing the overall detection capabilities and enabling efficient adaptation to evolving network conditions and emerging threats.

4. Develop cross-layer ensemble learning techniques as the latest evolution to enhance the accuracy and precision of IDSs in lightweight IoT networks, as depicted in Figure 1.1. While our previous contribution, the baseline k-means approach, effectively utilizes available labeled data to enhance intrusion detection accuracy, it prioritizes the false-negative rate over the false-positive rate. However, a high false-positive rate could undermine the trustworthiness of the IDS and compromise its utility. To mitigate the false positive rate inherent in anomaly-based IDSs, we propose a heterogeneous ensemble learning approach that incorporates different models with the baseline k-means approach. This ensemble consists of local novelty detection models assigned to individual workers, which are then integrated using weighted and voting-based strategies. By embracing this ensemble approach and pushing the sampling ratios to new lows, we not only improve the classification of normal and abnormal data points in lightweight IoT networks but also address the scarcity of labeled regular or benign traffic data. Our research objective is to present a lightweight, semi-supervised, federated IDS for IoT, specifically designed to operate following a sampling layer and employing a heterogeneous ensemble learning strategy.

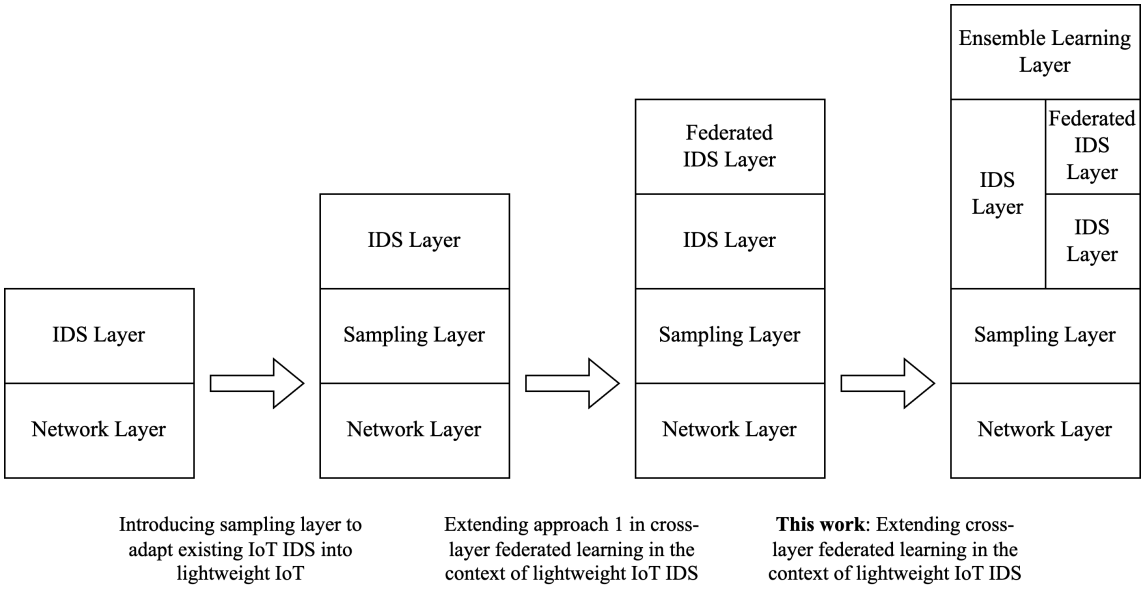


Figure 1.1: Innovative Evolution in Lightweight IoT IDS Research: Sampling, Federated, and Ensemble Learning Advancements

1.3/ THESIS ORGANIZATION

The thesis is organized into five chapters, each focusing on specific aspects of intrusion detection in IoT networks and contributing to the development of robust security mea-

sures.

- Chapter 1 serves as the introduction, providing an overview of the research topic, discussing the motivation behind the study, and outlining the research objectives. It highlights the complexity of attacks in interconnected environments, emphasizes the need for effective IDSs, and introduces novel learning-based IDS types.
- Chapter 2 focuses on cluster-based sampling, discussing its advantages over traditional sampling techniques and reviewing related work. The chapter presents the taxonomy of packet sampling policies, methodology, experiments, and discusses the implications of cluster-based sampling.
- Chapter 3 introduces the concept of cross-layer federated IDS, addressing challenges in IoT networks and exploring existing IDS and network attacks. The chapter evaluates lightweight semi-supervised intrusion detection techniques through experiments, concluding with a summary of findings and implications.
- Chapter 4 integrates ensemble learning techniques into the cross-layer federated IDS framework. It explains anomaly-based intrusion detection techniques, discusses ensemble learning methods, and presents the implementation methodology and experimental results of the cross-layer federated ensemble learning IDS.
- Chapter 5 is the conclusion, providing a summary of the key findings and contributions of the thesis. It highlights the strengths of the proposed approaches and discusses future research directions for intrusion detection techniques in securing IoT networks.

2

CLUSTER-BASED SAMPLING

2.1/ INTRODUCTION

The rapid emergence of new technologies and applications has resulted in a significant increase in Internet traffic, both in terms of speed and volume. To meet the growing demands of businesses, advanced information networks must integrate various technologies such as distributed storage systems, encryption/decryption mechanisms, and remote and wireless access. The increased reliance on those information networks opens them up to security risks from malicious attacks.

These attacks come in many different forms, such as unauthorized intruders attempting to gain access to protected resources, legitimate users attempting to escalate their privileges (Privilege escalation attacks), or malicious agents trying to disrupt the network and deny others access to a network's resources (DDoS attacks). As networks become more complex and traffic volume increases, malicious attacks become harder to detect and prevent. Hence the need for IDSs [1].

IDSs play a critical role in ensuring network security by monitoring user activity and detecting security violations based on data patterns. However, the effectiveness of an IDS is often challenged by the flaws of the network, the complexity of attacks, the diversity of traffic data sources, and the traffic volume going through the network.

This increased focus on network monitoring and analysis is essential to maintain the security and integrity of computer networks. By detecting and responding to potential threats in real-time, IDSs can help prevent data breaches, malware infections, and other cyberattacks. As technology continues to advance, it is likely that network-based security systems will become even more sophisticated and effective in protecting against a wide range of threats.

Real-time network monitoring requires rapid processing and inspection of network traffic, which is challenging when the traffic is large, and IDSs cannot inspect every incoming packet. In IoT environments, IDSs must be implemented on programmable devices such

as Field Programmable Gate Arrays (FPGAs). However, IoT devices' limited computing and energy resources can affect the IDS performance [2]. Collaborating between IoT devices and the router can shift the compute load from resource-constrained IoT devices to the resource edge router and reduce intrusion detection time. However, in the case of lightweight IoT networks where the nodes possess limited computational and energy resources, ensuring adequate security becomes a challenge. Implementing a lightweight IDS becomes crucial in elevating the security level of IoT nodes with resource constraints.

Traditional packet measuring and processing technologies are not practical for the extensive use of high-speed internet, which makes it difficult to apply IDSs to network traffic [3]. One possible solution to this issue is to distribute network packets to multiple IDSs, which can increase the storage and computing resources of the IDS, although this approach can be expensive [4, 5]. Another approach is to summarize the data on the fly and store only the relevant information by filtering the data before applying ML algorithms in IoT environments [6]. However, the application of ML in IoT environments is a challenging task due to the computing and energy constraints of IoT devices, and the high computational needs of modern ML algorithms. Thus, there is a need to filter and summarize data efficiently to enhance the performance of IDSs in IoT environments, hence the need for efficient sampling algorithms. It is worth noting that while Lightweight IDS for IoT is receiving attention from the research community, as will be elaborated in section 2, there still exists a gap in the development of efficient sampling algorithms for lightweight IDS. This work aims to address this gap by introducing the first sampling algorithm specifically designed to enable IoT nodes to implement a lightweight IDS system with minimal impact on their limited resources.

2.1.1/ DATA SAMPLING

Given the limitations of analyzing every individual incoming packet and the issues previously outlined, it is paramount to adopt a methodical selection of a statistically representative subset of network packets, as demonstrated in Figure 2.1. As a result, intrusions can be identified based on the sampled data instead of analyzing the entire traffic.

Data sampling techniques have been proposed to enhance attack detection accuracy and reduce network traffic volume [7]. However, data sampling causes intrinsic loss of information with adverse effects on IDS performance, as demonstrated in multiple studies [8, 9, 10].

Several studies and benchmarking papers have attempted to evaluate the impact of using different data sampling policies on intrusion detection accuracy [10, 11, 12]. Silva et al. [13] established a framework to characterize sampling techniques and evaluate their efficiency.

Adaptive sampling algorithms that aim to increase the representation of the samples, and preserve traffic feature distributions have also been proposed. For instance, Bartos et al. [14] proposed an adaptive flow-level sampling algorithm that adapts to traffic variations, and they compare its performance against other sampling techniques. In this chapter, an experimental comparison of existing sampling techniques is performed based on their impact on several well-known statistical measures.

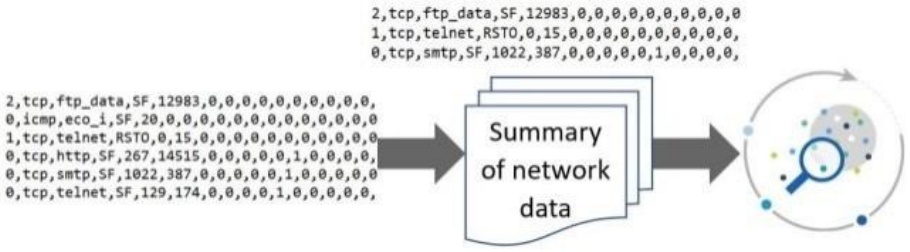


Figure 2.1: Packets sampling of network traffic.

2.1.2/ PROBLEM DEFINITION AND MOTIVATION

IDSs are critical for preventing attacks, but the limited resources make it challenging to manage and monitor the network efficiently. One possible solution to address this problem is to apply a data sampling algorithm to reduce the amount of traffic that needs to be processed [15]. Packet sampling provides a dynamic overview of the network and allows the inference of various estimates such as the number and size of packets, interarrival delays, and traffic flows. El Sibai et al. [16] provide metrics to assess the quality of a data sampling algorithm, which include single-pass processing over the data, its memory consumption, its skewing ability through packet weighting, and the algorithm's complexity.

Packet sampling is a versatile process that goes beyond merely detecting network intrusions. It can also facilitate network management by identifying faulty links and servers, providing guidance for traffic shaping, and enhancing Quality of Service (QoS).

With the ability to capture and analyze a representative subset of network traffic, packet sampling offers valuable insights into network behavior, enabling network administrators to identify and address issues proactively. Several sampling strategies can be employed to select packets for analysis. These strategies include count-based, time-based, and content-based sampling decisions. However, in the context of lightweight IoT, achieving an extremely low sampling ratio is preferable. Nonetheless, finding the delicate balance between sample representation and sampling rate proves to be a challenging task.

Within this chapter, we conduct an evaluation of various data sampling algorithms and explore the different sampling decisions available for selecting packets to be analyzed. Additionally, we propose a cluster-based sampling algorithm that demonstrates supe-

rior performance compared to other algorithms, particularly for extremely small sampling rates.

2.1.3/ OBJECTIVE AND CONTRIBUTIONS

The aim of this chapter is to assess the effectiveness of using summarized (sampled) data for detecting attacks in high-speed networks and to analyze the impact of different sampling strategies on the accuracy of intrusion detection algorithms, specifically in the context of lightweight IoT nodes. For this end, we compare statistical measures derived from the sampled data and the original traffic dataset in order to quantify the degree of degradation resulting from the sampling process. By evaluating the impact of sampling on statistical properties such as mean, variance, and distribution, we aim to gain a deeper understanding of how accurately the sampled data reflects the original traffic dataset's characteristics. We also evaluate the behavior and robustness of various network features under different sampling strategies and parameters, and identify which attacks are more resilient to the sampling process.

In this chapter, we will first present a comprehensive survey of existing data stream sampling algorithms, followed by an experimental comparison of these methods. Additionally, we will propose a cluster-based sampling algorithm as an innovative solution. Furthermore, we will highlight research challenges and offer potential solutions based on our findings, to position our work in the context of the state of the art.

Data sampling is a lossy process and will inevitably introduce distortion to the statistical distribution of the sampled data, but few studies explore how the network characteristics estimation varies according to the sampling method used, or how this affects subsequent inference processes. Pescape et al. [11] evaluated the use of 2 statistical metrics, "Hellinger" for similarities and "Fleiss Chi-Square" for classification, to assess the impact of sampling on selected feature sets. Their work concludes that the impact of data sampling algorithms on the anomaly detection process depends less on the sampling technique used, and more on the data measurement method.

Pan et al. [17] suggested a method that uses a variable sampling rate and an adaptive sampling probability for measuring packet sampling based on IP flow. They used Root Mean Square Error (RMSE) to measure volume anomalies in the size of the IP stream and the hit detection rate to measure the sequence of variance, Silva et al. [13] developed a data-sampling framework that adjusts the characteristics and sampling techniques based on the measurement task, using a sampling taxonomy that determines the granularity, selection scheme, and selection trigger.

Our contributions include a detailed survey of data sampling algorithms, a comprehensive experimental evaluation of different sampling techniques, and the identification of

research challenges and potential solutions based on our findings. We also compare our work with previously published benchmarks, as summarized in Table 2.1, where C1 shows whether the benchmark evaluated all sampling algorithms, C2 represents whether the benchmark studied the appropriate sampling policy and parameters, C3 indicates whether the benchmark presented an exhaustive study of the sampling algorithms, C4 shows if the benchmark evaluated the sampling impact on different types of attacks, C5 shows if the benchmark studied the sampling impact on feature behavior, C6 shows whether the benchmark analyzed sampling performance with respect to accuracy, and C7 shows whether the benchmark assessed sampling distortion. Furthermore, we introduce our novel proposal: a cluster-based sampling algorithm. This innovative algorithm acts as a crucial link between sampling techniques and lightweight IDSs specifically designed for IoT environments. To substantiate the effectiveness of our proposed algorithm, we delve into the simulation environment employed in our experiments and present the results we have obtained. In conclusion, we summarize our findings and engage in a discussion concerning future trends within the realm of lightweight sampling.

Table 2.1: Comparison of Our Work with Others on Benchmark Criteria.

Reference	**Year**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**
Pescape et al. [11]	2010	✓	✓	✗	✗	✓	✓	✓
Pan et al. [17]	2012	✓	✓	✗	✓	✗	✗	✓
Singh et al. [18]	2014	✓	✓	✗	✗	✓	✓	✓
Silva et al. [13]	2015	✓	✓	✗	✗	✗	✓	✓
This research	2021	✓	✓	✓	✓	✓	✓	✓

2.2/ RELATED WORK

2.2.1/ STATIC DATA SAMPLING

2.2.1.1/ TAXONOMY OF PACKET SAMPLING POLICIES

a. SIMPLE RANDOM SAMPLING OVER A SLIDING WINDOW (SRSSW)

The SRS algorithm [19] samples packets randomly such that all packets have the same probability p of being sampled. SRS can be performed with and without replacement. When applying SRS with replacement, the sample will contain redundant packets because each packet may be selected at least once. However, with SRS without replacement, each packet can be sampled only once. In this study, we only considered SRS without replacement.

b. SIMPLE RANDOM SAMPLING OVER A FIXED WINDOW (SRSFW)

The SRS without replacement algorithm can also be applied to build a sample over a fixed window. This is done by constructing a sample over each window of the traffic stream and removing the samples constructed on the former windows. To sample k packets from a window of size n, each packet is selected with a probability p equal to the sampling ratio k/n. This step must be repeated until k distinct packets are selected [16].

c. DETERMINISTIC SAMPLING OVER A FIXED/SLIDING WINDOW

The deterministic algorithm is a non-probabilistic sampling algorithm that constructs a sample without randomness. It consists of constructing a sample of size k by selecting one packet from every x packet of the traffic stream. Assuming that the traffic stream consists of packets with an always-increasing index, to construct a sample of distinct packets among the n most recent packets of the traffic, and given the sampling ratio p, each 1/x packet is sampled. The selection of one packet from every x packet depends on the packet index. If the packet index equals α × n/k, where α is a positive integer, the packet will be selected.

d. SYSTEMATIC SAMPLING

The systematic sampling algorithm partitions the traffic into x groups of size x=n/k and selects a random value j ϵ [1,x] and adds packets to the sample at the following indices: j, j+x, j+2x, j+3x, etc. [19]. One potential limitation of this method is that when the periodicity of the traffic stream and the sample size are synchronized, there may be a reduction in randomness. This can occur because the sample may not fully capture the full range of variation in the traffic stream, leading to biased or incomplete data.

e. STRATIFIED SAMPLING

The stratified sampling algorithm [19] divides the traffic into homogeneous subgroups and randomly builds a sample from each subgroup. Stratified sampling enhances sampling accuracy and ensures the representation of extreme or rare groups of packets in the sample

f. WEIGHTED RANDOM SAMPLING WITHOUT REPLACEMENT

The Weighted Random Sampling algorithm (WRS) [20, 21], samples each packet with a probability based on the packet's weight to deal with the lack of representativeness of some packets in the sample.

g. RESERVOIR SAMPLING

The reservoir sampling algorithm [22, 23] aims to randomly retain a fixed size sample (k) from a continuous stream of data by probabilistically selecting elements from

the stream based on their index. When a new element arrives, it may be included in the sample while a random element from the existing sample is removed.

h. BACKING SAMPLING

The backing sampling [24, 25] adds the first k elements of the stream to the sample and then skips a random number of elements and adds the next element to the sample with a probability equal to k/n, and repeats the process.

i. CHAIN-SAMPLING

The chain-sample algorithm [26] provides a random sample of size k selected from the last elements of the stream by constructing a sample containing one element selected from the last sliding window of the stream. The algorithm samples one element from the first window and chooses a successor's index for the ith element from the elements with indexes [i+1, i+n], and repeats the process.

j. CHAIN+ SAMPLING

the chain+ sampling algorithm [27] builds a sample of k elements by sampling each element in the first sliding window with a probability equal to min(i,n)/n, where n is the window size and i is the index of the element in the window. The algorithm repeats this process until k distinct elements are sampled.

k. PRIORITY SAMPLING

The priority sampling algorithm [26] constructs and maintains a random sample over a physical sliding window. To build a sample of one element, the algorithm assigns a random priority p ϵ [0, 1] to each element, and selects the element with the highest priority in the sliding window. To construct a sample of k elements, the process is repeated k times. In this work, we implement a newer version of the priority sampling algorithm as proposed in [28], which assigns weights to the packets based on their arrival time, contents, and impact on the sample accuracy.

l. RANDOM PAIRING SAMPLING

The Random Pairing (RP) sampling algorithm [29, 30] constructs and retains a random sample over the most recent sliding window of the stream. The algorithm tracks the number of expired elements in the sample over the window, and replaces expired items with a probability that increases as more items in the sample expire.

m. STREAMSAMP

The StreamSamp [31] is a progressive sampling technique based on the Simple Random Sampling algorithm [19]. The algorithm samples stream elements with a predefined sampling ratio, and stores the sample in an order equal to 0. When the sample size is reached, a second sample of the same size is constructed, and

both samples are merged using a simple random sampling approach. When the number of samples in order 0 exceeds a certain limit, StreamSamp merges the two old samples of order 0 into a single sample, and the process continues for order 1 and so on.

Table 2.2 provides a comparison of the presented sampling algorithms highlighting the strengths and weaknesses of each method.

2.2.1.2/ METHODOLOGY

Our methodology employs static sampling algorithms to summarize a real traffic dataset. This allows us to gain insights into the behavior of these algorithms. To estimate the degree of distortion introduced by the sampling process, we conducted a comparative evaluation of diverse statistical metrics.

The Overall Statistic (OS) that captures the overall statistical disparity between the unsampled and sampled traffic datasets is defined by equation 2.1 [32, 33]:

$$OS = |(\mu_0 - \mu)/\mu_0| + |(med_0 - med)/med_0| + |(\sigma_0 - \sigma)/\sigma_0| \tag{2.1}$$

μ_0 and μ represent the actual average values approximated before and after traffic sampling respectively.

med_0 and med represent the median values of the traffic parameter before and after sampling respectively.

σ_0 and σ represent the standard deviation values of the traffic approximated before and after sampling respectively.

In our research, we carried out the following experiments for each of the studied sampling strategies:

Step 1 - Without sampling: In this scenario, we compute the mean, standard deviation, and median of the unsampled dataset to establish a baseline. We obtain the values for μ_0, med_0 and σ_0.

Step 2 - With sampling: iIn this scenario, we compute the mean, standard deviation and median of the sampled dataset. We obtain the values for μ, med and σ .

We calculate the value for our OS metric defined in equation (1) and use it to assess multiple sampling strategies with varying parameters. We also evaluated the computational resources and execution time required for each of those algorithms. Finally, we conducted a comparative analysis of all the studied strategies.

Our study contributes to the comprehension of static sampling algorithms and their influ-

Table 2.2: Advantages and weaknesses of data stream sampling algorithms

Sampling Algorithm	**Advantages**	**Weak Points**
Simple Random (SRSFW and SRSSW [16] and [19])	The sample is accurate, convenient, and representative of the entire stream	Biased sample in case of periodicity in the data stream, no skewing ability
Deterministic (DETFW and DETSW) [19], [16]	It provides a sample with an exact size, the sample is representative of the entire stream when no periodicity is displayed	Biased sample in case of periodicity in the data stream, no skewing ability
Chain-sample [26]	It provides a sample with an exact size	Redundant samples, no skewing ability
Chain+ [27]	It provides a representative sample with an exact size without duplication	No skewing ability
Stratified [19]	The use of this algorithm is beneficial when it is desired to highlight a specific subgroup within the data and ensure its presence in the sample, this algorithm is also used to represent the smallest, extreme or rare subgroups of the data in the sample	Unbounded sample size, no policy to choose the sample size, no skewing ability
Systematic [19]	The sample is easy to be built, The sampling process is fast and accurate since sampled data are spread over the entire stream	Unbounded sample size, biased sample in case of data stream periodicity, no skewing ability
Reservoir [22, 23]	This algorithm is simple and suitable for streaming environments since it is executed in one pass	Recent elements have less chance of being sampled, no skewing ability
Backing [24, 25]	This algorithm is suitable for streaming environments since it is executed in one pass	Performs several passes over the data, no skewing ability
Priority [26]	It provides a sample with an exact size	No policy for the determination and revision of the weights
Random Pairing [29, 30]	The algorithm builds and maintains a uniform sample of fixed size	No skewing ability
StreamSamp [31]	The algorithm maintains a sample of a fixed size	No policy for the determination and revision of the weights

ence on statistical metrics. We believe that our results will facilitate informed decisions regarding the selection of an appropriate sampling strategy for a particular application.

2.2.1.3/ EXPERIMENTS AND RESULTS

a. Dataset choice

The dataset used in this study is the Network Security Laboratory-KDD (NSL-KDD), which is a benchmark dataset widely used for evaluating the performance of network intrusion detection algorithms.It is an improved version of the KDD Cup 1999 dataset, originally developed for the Conference on Knowledge Discovery and Data Mining (KDD) in 1999. However, the KDD Cup 1999 dataset suffers from redundant data and duplicate records, which can lead to inaccurate intrusion detection results [34]. The obtained dataset contains approximately 150K records divided into training and testing subsets. The NSL-KDD dataset consists of 41 attributes and includes 22 attack types and contains 125.974 records with a size of 18.662 MB.

b. Comparison of computational resources

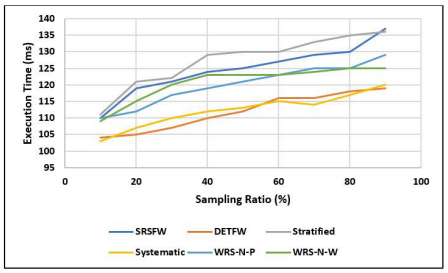
In this section, we evaluate the impact of different sampling policies and rates on the distortion introduced by the sampling process. Our goal is to identify the features that are least prone to distortion by the sampling process. The specifications of our machine include 8 GB RAM, a 450 GB system disk, and a 2.7 GHz Intel Core i7 processor.

Intuitively, computational costs are positively correlated with the sampling rate, i,e. higher sampling rates produce larger samples and thus require more processing. This can be seen in the trends shown in Figure 2.2 comparing execution times of different sampling policies presented in this study, relative to the sampling ratio. For stream sampling algorithms, there is also a trade-off between execution time and window size, with execution time becoming longer for a larger window.

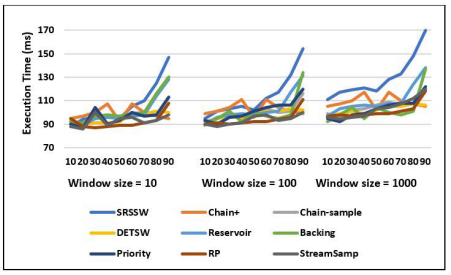
Figure 2.2 shows that the deterministic and systematic sampling algorithms have the lowest execution time, out of all non-stream sampling algorithms.

Stream sampling algorithms show higher discrepancies. For instance, in the SRSFW and stratified sampling algorithms, the sample may contain duplicate elements. Figure 2.3 plots the collision rate of the SRSFW and stratified sampling algorithms relative to the sampling ratio, against the theoretical probability of collision $P_{collision}$ which is computed as follows:

$$P_{collision} = 1 - C_k^n = 1 - (n!)/((n-k)!\, n^k) \tag{2.2}$$



(a) Non-stream sampling



(b) Stream sampling

Figure 2.2: Execution time of non-stream (a) and stream sampling (b) algorithms using different window sizes.

where k is the sample size and n is the window size.

This equation shows that the collision rate increases with k and n, as revealed in Figure 2.3. Redundancy greatly affects the accuracy of the sample, and therefore a deduplication process is necessary, which introduces additional overhead. This effect is demonstrated by comparing the stratified sampling algorithm against SRSFW in Figures 2.2 and 2.3. While the stratified sampling algorithm produces slightly fewer collisions, SRSFW is slightly faster to compute.

A notable comparison in Figure 2.2 is between the SRSSW and Chain+ sampling algorithms. The execution time for both of these algorithms increases as k/n grows. The difference, however, is notable for k/n $>$ 0.5, where the Chain+ algorithm reduces the collision rate to k/n - 0.5, thus decreasing its execution time, as demonstrated in [27]. It's worth mentioning that the priority and chain-sample algorithms exhibit nearly identical execution times.

For stream sampling algorithms, execution time is dependent on both the sampling rate and the sliding window size. Some algorithms show more variation in execution time for different values of k and n than others. For example, it is observed that the DETSW algorithm has less variation in the execution time regardless of the sampling ratio, whereas the SRSSW exhibits the highest variation as shown in Figure 2.4.

c. Comparison of traffic statistics

In this study, various sampling policies and sampling rates were evaluated to assess their effectiveness in detecting four types of cyber-attacks: DoS, Probe, R2L, and U2R attacks. Table 2.3 presents the most important numeric features of the NSL-KDD dataset that can be used to detect DoS, Probe, R2L, and U2R attacks, according to Ao et al. [35].

The methodology adopted involves sampling the NSL-KDD traffic using different sampling techniques, followed by the calculation of statistical measures to evaluate

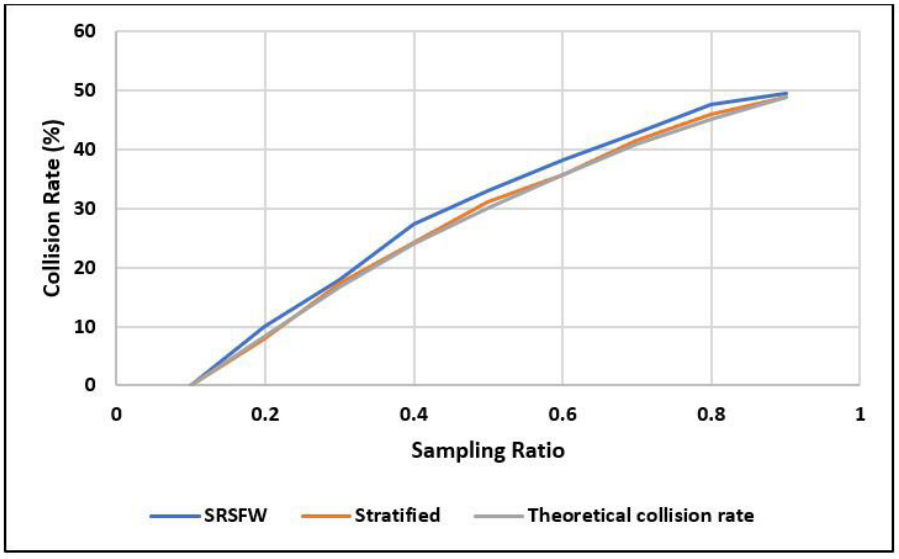


Figure 2.3: Collision rate of the SRSFW and stratified sampling algorithms.

the accuracy of the sampling estimates for the unsampled traffic. The mean, standard deviation, and median of the traffic stream were estimated before and after sampling, from which we calculated the OS Metric using equation (1) to represent the relative deviation of the sample from the original traffic, as detailed in section III.

For non-stream sampling strategies, we compare the OS metric of the studied strategies at different sampling ratios. For stream-based sampling strategies, we consider the average OS metric of 3 window sizes of 10, 100 and 1000 respectively, at different sampling ratios.

Below is a comparative study of all the different sampling strategies, regarding their efficiency in preserving traffic statistics used in each of the considered attack types.

i DoS Attack

Denial of Service (DoS) attacks are best detected using features 5, 7, and 8 as listed in Table 2.3. The results in Figure 2.5 show that regardless of the non-stream sampling policy used, the level of distortion introduced by the sampling process does not vary significantly when the sampling ratio is ϵ [70, 90]. Conversely, for a given sampling ratio of less than 50%, the estimation accuracy of all algorithms is highly variable. The results also show that the SRSFW and WRS-N-W are the worst sampling strategies because they provide the highest OS value, regardless of the sampling ratio.

Based on the above discussion, one can conclude that to estimate the values of features 5, 7, and 8 needed to detect DoS attacks, the best non-stream sampling strategy, and sampling ratio to apply in order to achieve the lowest distor-

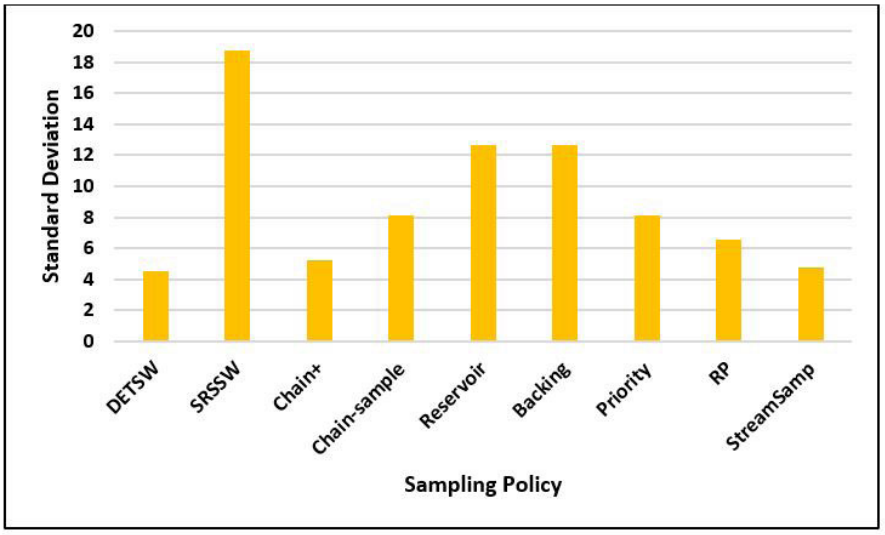


Figure 2.4: Variation of the execution time of stream sampling algorithms, using a window size equal to 10.

tion level is the deterministic/systematic sampling with a sampling ratio of 60%. Conveniently, these 2 strategies also produced the lowest execution times as shown in Figure 2.2. This makes them excellent candidates to consider when looking for sampling strategies for DoS attack detection. For stream sampling strategies, as with their non-stream counterparts, OS values drop significantly when the sampling rate reaches 90%. Our results in Table 2.4 show that the Chain sampling algorithm performs poorly regardless of sampling rate for features 5, 7 and 8. Conversely, DETSW and Priority sampling achieve optimal OS scores at a sampling rate of 60%, which is a considerable reduction in the total volume of data needed to be analyzed.

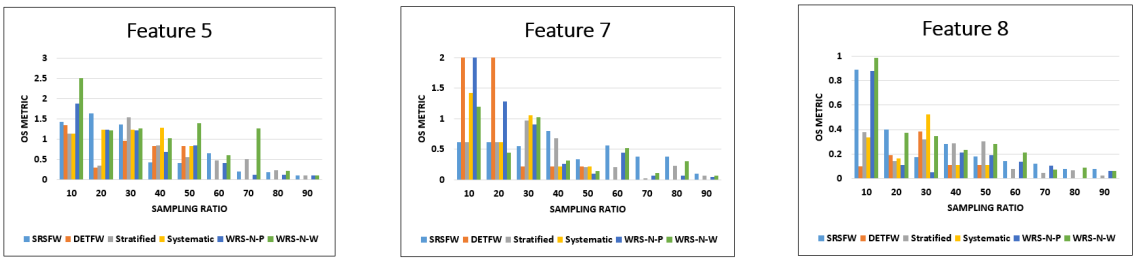
ii Probe Attack

Probe attacks are best detected using features 5, 28, 30, and 36 as listed in Table 2.3. The results in Figure 2.6 show that regardless of the non-stream sampling policy used, the level of distortion introduced by the sampling process does not vary significantly when the sampling ratio is ϵ [70, 90]. For stream sampling algorithms, the level of distortion is stable when the sampling ratio is ϵ [60%, 90%]. The results in Table 2.5 show that the variation in the OS value at a given sampling ratio is dependent on the sampling policy. DETSW and Priority sampling strategies achieve OS values close to zero at a sampling rate of 60%. However, for certain features, other algorithms reach acceptable distortion levels at much lower sampling rates.

iii R2L Attack

Table 2.3: Relevant features for each attack type in the NSI-KDD dataset

Attack	feature names	feature numbers
DoS	source bytes, land, wrong fragment	5, 7, 8
Probe	source bytes, srv error rate, diff srv rate, src port rate	5, 28, 30, 36
R2L	destination bytes, failed logins, count, dst host error rate	6, 11, 23, 39
U2R	root shell, srv count, src port rate	14, 24, 36



(a) Feature 5 (b) Feature 7 (c) Feature 8

Figure 2.5: Comparison of OS metric for non-stream sampling policies at different sampling ratios for features 5, 7, and 8.

Remote-to-Local (R2L) attacks are best detected using features 6, 11, 23 and 39 as listed in Table 2.3. The results in Figure 2.7 show that regardless of the non-stream sampling policy used, the level of distortion introduced by the sampling process does not vary significantly when the sampling ratio is ϵ [70, 90]. For all stream sampling algorithms, the OS value reaches its minimum when the sampling rate is equal to 90%. At lower sampling rates, the variation of the OS value at a given sampling ratio is dependent on the sampling policy. The results in Table 2.6 show that for the SRSSW, Chain+, and StreamSamp sampling algorithms, the minimum OS value is achieved when the sampling rate is equal to 70%. For the reservoir and RP sampling algorithms, it was achieved for a sampling rate equal to 80%. DETSW and Priority sampling achieve an OS value close to zero at a sampling rate of 60%.

iv U2R Attack

User-to-Root (U2R) attacks are best detected using features 14, 24, and 36 as listed in Table 2.3. The results in Figure 2.8 show that regardless of the non-stream sampling policy used, the level of distortion introduced by the sampling process does not vary significantly when the sampling ratio is ϵ [50, 90]. For all stream sampling algorithms, the OS value reaches almost its minimum when the sampling rate is equal to 90%. At lower sampling rates, the variation of the OS value at a given sampling ratio is dependent on the sampling policy. For instance, Table 2.7 shows that for the SRSSW, Chain+, and RP sampling algorithms, the minimum OS value is achieved when the sampling rate is equal

Table 2.4: Lowest achieved OS values for stream-sampling policies and the corresponding sampling rates, for features 5, 7 and 8

Feature	SRSSW	Chain	Chain+	DETSW	Reservoir	Backing	Priority	RP	StreamSamp
5	80% 0.137	10% 1.245	80% 0.067	60% 0.0004	80% 0.171	60% 0.318	60% 0	80% 0.421	70% 0.104
7	80% 0.197	10% 2.06	80% 0.161	60% 0.0002	60% 0.264	20% 0.574	60% 0	80% 0.446662	30% 0.070885
8	70% 0.047	10% 0.455	70% 0.028	60% 0.0006	80% 0.076	30% 0.101	60% 0	60% 0.068	80% 0.048

Table 2.5: Lowest achieved OS values for stream-sampling policies and the corresponding sampling rates, for features 5, 28, 30, and 36

Feature	SRSSW	Chain-sample	Chain+	DETSW	Reservoir	Backing	Priority	RP	StreamSamp
5	80% 0.137	10% 1.245	80% 0.067	60% 0.0004	80% 0.171	60% 0.318	60% 0	80% 0.421	70% 0.003
28	70% 0.002	10% 0.037	70% 0.002	60% 0.0001	80% 0.001	40% 0.002	60% 0	80% 0.008	40% 0.002
30	80% 0.01043	10% 2.935	80% 0.014	60% 0.0001	40% 0.007	60% 0.02127	60% 0	70% 0.009	80% 0.003
36	70% 0.007	10% 2.463	70% 0.014	60% 0.00004	80% 0.003	50% 0.0193	60% 0	40% 0.003	50% 0.008

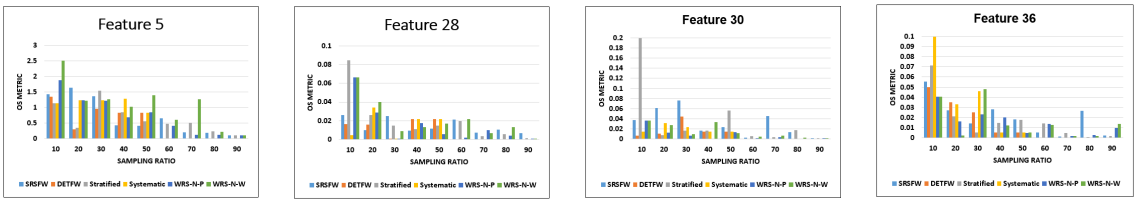
to 70%. For the reservoir and StreamSamp algorithm, it was achieved for a sampling rate equal to 80%. For the Backing algorithm, it was achieved for a sampling rate equal to 50%. DETSW and Priority sampling achieve an OS value close to zero at a sampling rate of 60%.

2.3/ CLUSTER-BASED SAMPLING

2.3.1/ CLUSTER-BASED SAMPLING ALGORITHM

Stream sampling algorithms and non-stream sampling algorithms demonstrated exceptional performance at high sampling ratios, but their effectiveness diminishes significantly at low sampling ratios, rendering them unsuitable for resource-constrained or lightweight scenarios, such as IOT environments. Therefore, there is a pressing need to identify a sampling algorithm that is specifically tailored for low sampling ratios. In this section, we introduce a novel Cluster-based sampling algorithm that enhances sample representation, even when confronted with challenging low sampling rates. Cluster-based sampling involves dividing a large dataset into clusters or groups and selecting a representative subset of clusters for analysis. A sample must be truly representative of the entire data stream. This category of algorithms is related to other types of static data sampling, such as stratified sampling and systematic sampling, but is distinct in that it employs a clustering technique to categorize data into distinct clusters and maintain a representative sample for each cluster. Clustering algorithms aim to uncover new emerging patterns in the data and detect any changes in the patterns and data distribution.

The proposed algorithm classifies the data into different clusters and builds and maintains a sample for each cluster. Data is inserted in the samples in a way that ensures that



(a) Feature 5 (b) Feature 28 (c) Feature 30 (d) Feature 36

Figure 2.6: Comparison of OS metric for non-stream sampling policies at different sampling ratios for features 5, 28, 30, and 36.

Table 2.6: Lowest achieved OS values for stream-sampling policies and the corresponding sampling rates, for features 6, 11, 23, and 39

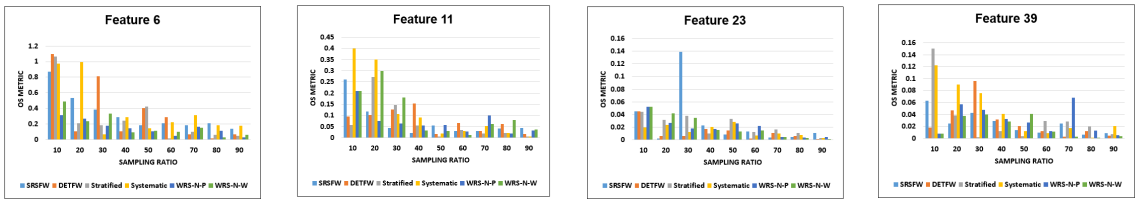
Feature	SRSSW	Chain-sample	Chain+	DETSW	Reservoir	Backing	Priority	RP	StreamSamp
6	70% 0.066	10% 3.092	70% 0.097	60% 0.0003	60% 0.099	60% 0.058	60% 0	80% 0.105	80% 0.109
11	60% 0.014	10% 2.198	60% 0.014	60% 0.0006	80% 0.0199	60% 0.055	60% 0	70% 0.039	70% 0.007
23	80% 0.006	10% 2.719	80% 0.002	60% 0.0003	80% 0.002	50% 0.009	60% 0	60% 0.004	70% 0.005
39	70% 0.002	10% 2.525	70% 0.0015	60% 0.0006	80% 0.008	30% 0.021	60% 0	80% 0.0111	70% 0.009

the difference between the sample and the corresponding cluster is minimal. Thus, the cluster tends to be homogeneous so that a small sample from this cluster contains a large amount of information about all of the items in the cluster. This ensures that all the subgroups within the data are present in the final sample, and minimizes the sampling error due to high data variance.

The proposed method uses the k-means algorithm to divide the data stream into separate groups or clusters, as shown in Figure 2.9. Then, a sample is built and maintained for each cluster. To ensure proper representation of all subgroups, the size of each sample is kept proportional to the size of the cluster. As more stream items arrive, the size of each cluster sample is adjusted accordingly, as shown in Figure 2.10. The final sample is constructed by combining all the clusters' samples.

The first k items of the stream are classified using the k-means algorithm and serve as a learning base. From the data in the first window of the stream, i initial clusters are constructed. Then, for each cluster j, a sample is built and maintained by randomly selecting k_j items from the cluster.

When new items arrive, they will be processed in sequential order. Each new item will be added to one of the clusters based on the Euclidean distance between the item and the center of each cluster. After fitting the item to one of the clusters, a decision will be made on whether to add it to the corresponding cluster sample. This is done by comparing the sampling error of the cluster sample with and without the item: If adding the item to the cluster sample leads to a higher sampling error, it will be rejected, otherwise, it will be



(a) Feature 6 (b) Feature 11 (c) Feature 23 (d) Feature 39

Figure 2.7: Comparison of OS metric for non-stream sampling policies at different sampling ratios for features 6, 11, 23, and 39.

Table 2.7: Lowest achieved OS values for stream-sampling policies and the corresponding sampling rates, for features 14, 24, and 36

Feature	SRSSW	Chain-sample	Chain+	DETSW	Reservoir	Backing	Priority	RP	StreamSamp
2*14	80% 0.075	10% 2.112	80% 0.049	60% 0.0006	70% 0.071	70% 0.093	60% 0	80% 0.069	50% 0.088
2*24	70% 0.006	10% 2.104	70% 0.001	60% 0.0004	80% 0.011714	50% 0.059	60% 0	70% 0.010	80% 0.005
2*36	70% 0.007	10% 2.463	70% 0.014	60% 0.00004	80% 0.003	50% 0.0193	60% 0	40% 0.0030	50% 0.008

sampled with a probability equal to

$$\left| 1 - \frac{e_i}{center(C_j)} \right|. \tag{2.3}$$

The sampling error with/without a given item e_i is calculated as follows:

$$error = \left| \frac{m - \underline{X}}{m} \right| \tag{2.4}$$

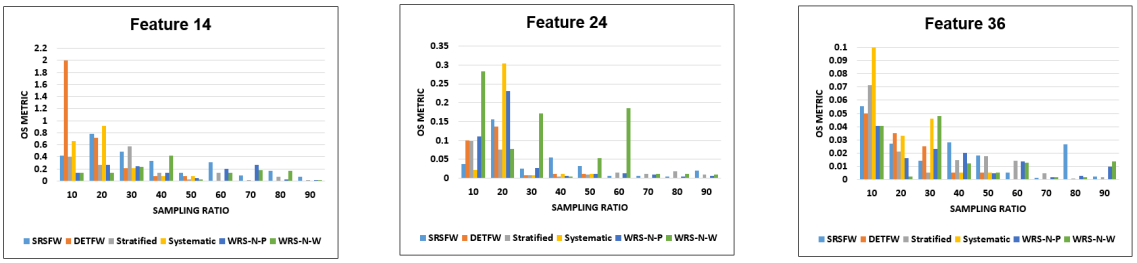
Where m is the cluster data's mean and $\underline{X}$ is the sample's empirical mean, an unbiased estimator of m calculated as follows:

$$\underline{X} = \frac{1}{k} \sum e_i \tag{2.5}$$

where k is the sample size, and e_i is the value of the item at index i in the sample.

To keep the cluster sample size fixed, an item must be removed from the sample when the featured cluster size exceeds k_j. To find out which item to replace, we propose a ranking procedure that follows the "remove-from-sample" principle, in which we assign a rank for each item in the sample based on the item's sampling error contribution, i.e. lower ranks are assigned to items that lead to higher sampling errors. The items with the lowest rank are the first candidates to be replaced.

Ideally, all items in the current cluster sample should be re-ranked after a new item is added to the sample. This is done by recalculating the sampling error using the "remove-from-sample" principle for all items in the current cluster sample. To keep our computational load low, a trade-off between the sampling precision and needed computing re-



(a) Feature 14 (b) Feature 24 (c) Feature 36

Figure 2.8: Comparison of OS metric for non-stream sampling policies at different sampling ratios for features 14, 24, and 36.

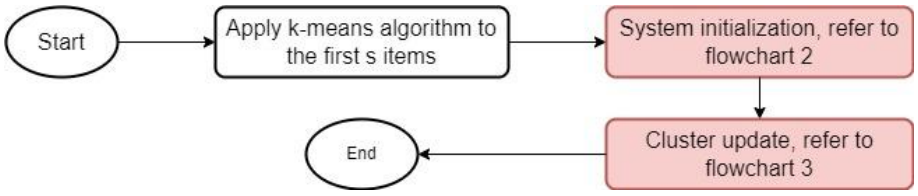


Figure 2.9: Proposed algorithm.

sources is needed here. Thus, re-ranking is only done after x% of a cluster's items are changed, where

$$x = (number\ of\ replaced\ items)/(cluster\ size) * 100 \tag{2.6}$$

Table 2.8 shows the conceptual idea of ranking. For instance, item 10 is ranked highest because its removal produces the minimum sampling error. Similarly, item 17 is ranked lowest because its removal produces the highest sampling error. Thus, it is the best candidate to be replaced by future incoming items.

Figure 2.11 describes the proposed algorithm which is also explained in the pseudo-code procedure CLUSTER_BASED_SAMPLING.

Three parameters impact the sampling accuracy: (1) the sampling ratio k/n, (2) change percentage. It is possible to sample the data with a sampling ratio equal to k/n while using different k and n values, such that:

$$p = k/n = (x.k')/(x.n') \tag{2.7}$$

where x is a positive integer.

Thin IoT nodes have three main concerns that should be the focus of any lightweight sampling algorithm. The concerns are:

- Limited memory: Dynamic sampling algorithms have increasing sample size [36] and this exceeds thin nodes' capabilities. Static sampling algorithms have fixed

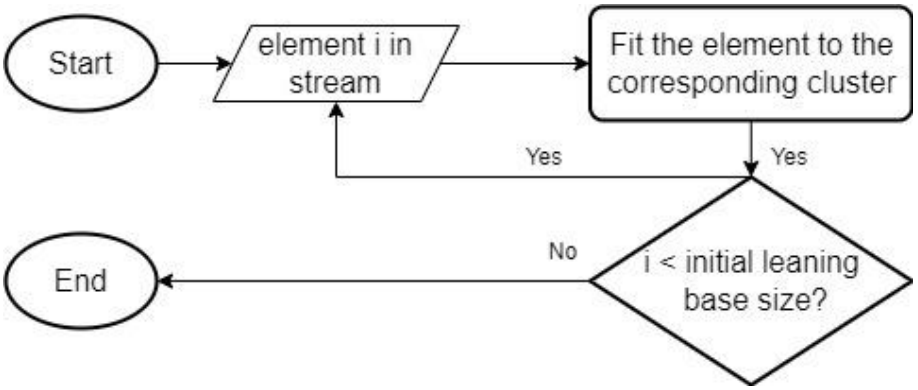


Figure 2.10: Item admission.

Table 2.8: Items ranking within a cluster

Item Index	Sampling error	Rank
10	3.61	1
11	...	2
12	...	3
17	9.62	4

sample size. In the current version of the proposed algorithm, a fixed number of elements are stored in clusters making the stored sample size similar to static sampling algorithms. As will be discussed later in this article, it is possible to eliminate all the stored elements and only keep the cluster definitions. That will significantly reduce the memory requirements of the algorithm to less than 1% of the current standard.

- Limited computation/energy: The proposed algorithm can be configured for running with significantly less computation using the Change Ratio (CR). Lower CR values result in lower computational requirements. Computational resources vs. sample distortedness should be optimized.

- Privacy-preserving collective learning: The ability for IoT nodes to share their knowledge about the network traffic is a significant enhancement to the security of the whole system and allows for a significant reduction in computational and energy requirements. By keeping and sharing only cluster definitions among different nodes, our method can guarantee privacy preservation, on top of reducing bandwidth requirements and network load.

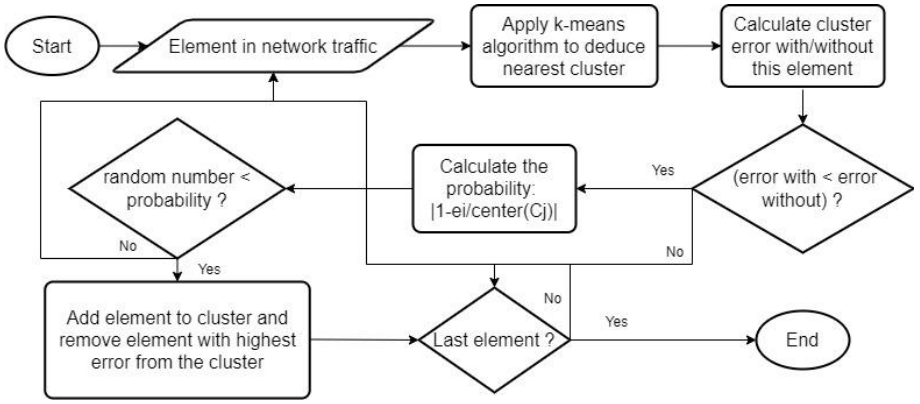


Figure 2.11: Cluster management.

2.3.2/ Simulation

2.3.3/ Simulation environment

Our algorithm is implemented using Python. The code is accessible through GitHub [37]. In this simulation parameters are defined in Table 2.9. The algorithm has been tested against the NSL-KDD test dataset [38]. Clusters' compactness is measured using the silhouette score [39]. The specifications of our machine are RAM: 12GB, System disk: 108GB, and processor: 2.20GHz Intel Xeon CPU.

Table 2.9: Simulation Parameters

Parameter	Range of values
Change rate	0.5, 0.75, 0.9
Learning base size	0.1, 0.3, 0.5, 0.7, 0.9 (× Dataset size)
Number of clusters	2, 5 (3, 4 and 6 tested but not reported)

2.3.3.1/ Simulation results

We've experimented with cluster sizes 2, 3, 4, 5 and 6. Here we report the results of our algorithm with cluster sizes 2 and 5, as these configurations performed consistently better on mean, median and standard deviation. With our 2 cluster configuration, we group packets into normal or attack, whereas with the 5 cluster configuration, we group packets into normal or one of the four attack types listed in Table 2.3. To validate our observations, we measured the clustering performance using silhouette score for each of the configurations presented in Table 2.9. The silhouette measurements are shown in Figure 2.12, where the x-axis is (Learning base, Change rate). As shown in Figure 2.12, the cluster fitness is increasing with the increase in the number of clusters, but this is not always correlated with the performance in maintaining the stream's mean, median and

standard deviation within the sample. For this reason, cluster fitness is not a good metric for selecting the appropriate configuration of the IDS.

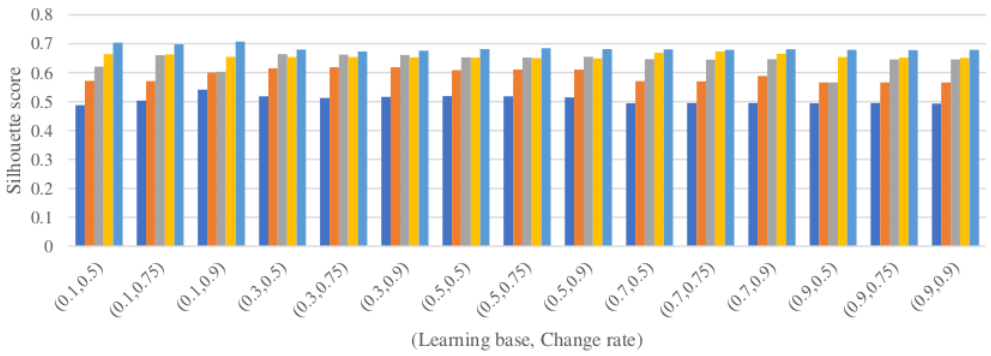


Figure 2.12: Silhouette score for cluster fitness.

The results shown in this work are consistent with the format used in [40], including mean, median, standard deviation, and OS value of the selected dataset features that are most representative of network attacks shown in Table 2.3. In the following, we compare our proposed lightweight sampling algorithm to other sampling algorithms, surveyed in the previous section.

Figure 2.13 shows the mean, stddev and OS value of feature 5 in the sample generated by each sampling algorithm. The red dotted line represents the dataset's original metrics for that feature.

It can be seen in Figure 2.13 that the most accurate algorithms at high sampling ratios/learning base sizes, higher or equal than 0.7, are systematic and deterministic sampling algorithms, as shown in the survey in the previous section. For lower sampling ratios/learning base sizes, those same algorithms are among the least accurate. Those observations are consistent with [40].

For low sampling rates, SRS offline, stratified and the proposed cluster-based sampling algorithm (with 2 clusters and 90% change ratio) are among the best performers.

The results of feature 6, shown in Figure 2.14 are consistent with those of feature 5. feature 6 shows even better performance for the proposed cluster-based sampling algorithm (with 2 clusters and 90% change ratio) for lower sampling ratios than SRS offline and stratified. The proposed cluster-based sampling algorithm (with 2 clusters and 90% change ratio) resulted in the least distortion for two sampling ratios/learning base sizes 0.3 and 0.5.

The results for feature 14, shown in Figure 2.15, are not as consistent with the other features. The proposed cluster-based sampling algorithm, all configurations, resulted in the least distortion for sampling ratio/learning base sizes = 0.3 only. The performance of the proposed algorithm is average for all other sampling ratios/learning base sizes.

The results for feature 28 are consistent with that of features 5 and 6. The proposed cluster-based sampling algorithm (with 2 clusters and 90% change ratio) achieved the lowest distortion for sampling ratios/learning base sizes 0.1, 0.3, and 0.5 as shown in Figure 2.16.

The results of the remaining features are consistent with the ones presented here. They show that the proposed algorithm performs consistently well at very low sampling rates, which is a feature that thin IoT nodes with limited resources can greatly benefit from.

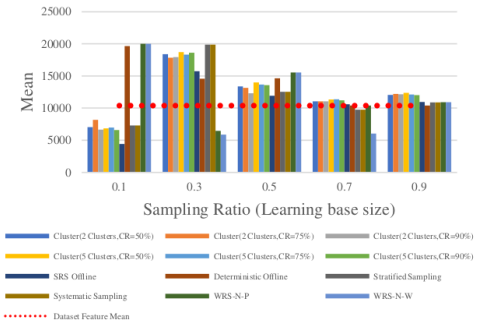
2.3.4/ Discussion and conclusion

The literature covers sampling algorithms for traditional IDSs and lightweight IDSs for IoT, but lightweight sampling algorithms for IDSs operation in IoT has not been explored fully. This work is the first, to the best of our knowledge, to try to cover this gap and develop a sampling algorithm that requires minimal resources and achieves good results for low sampling ratios, which is the type of configuration suitable for thin IoT nodes.

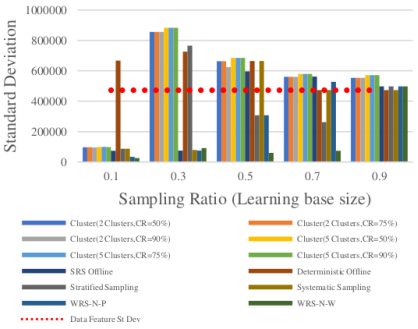
The highlight of our proposed method, on top of it being lightweight and performing well on very low sampling rates, is it being a stepping stone for more interesting privacy-preserving learning-based anomaly and IDSs. Such a system could be designed to collect and share summarized cluster statistics, such as cluster centroids and sizes, among multiple nodes to create a distributed lightweight IDS, without the need to share sensitive packet meta-data or traces of actual network traffic.

Algorithm 1 procedure CLUSTER_BASED_SAMPLING {n: initial learning base size}

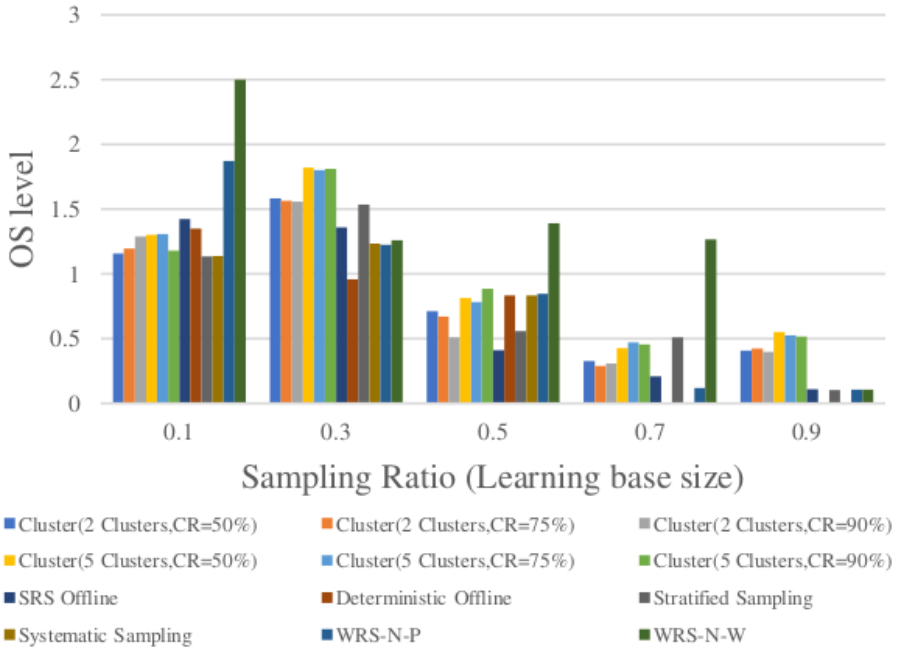
```
Initialization:
Swj is the initial sample of cluster Cj,
at the beginning S1 = {},
k is the sample or learning base size
x is the change ratio
N is the number of clusters

//starting pseudo-code presented in Figure 2.10
for i in [1, k] do
     for each item ei in the learning base of size k do
          Add ei to the corresponding cluster Cj
     end for
end for
//ending pseudo-code presented in Figure 2.10

//starting pseudo-code presented in Figure 2.11
While a new item ei is received, (i<n) do
     i <- i + 1 {i is the index of e}
     Add ei to the corresponding cluster Cj
     CalculateError(ei, Sj) {Compute sampling error with and without ei added to Sj}
     if samplingErrorWithSampling < samplingErrorWithoutSampling then
          Add ei to Sj with a probability |1- ei/center(Cj)|
          Cj.change_count <- Cj.change_count + 1
          if(Sj.size() > k/n) then
                Remove the item having the highest rank from Sj
          end if
     end if
     if(Sj.size()* x < Cj.change_count) then
          UpdateRankTable();
     end if
end while
return Sj
end procedure=0
//ending pseudo-code presented in Figure 2.11
```



(a) Mean

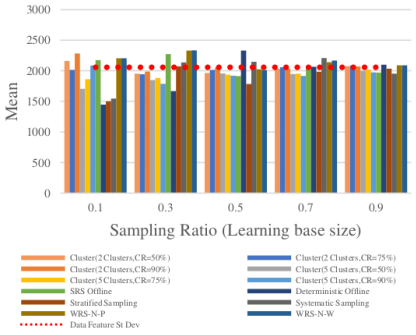


(b) Stddev

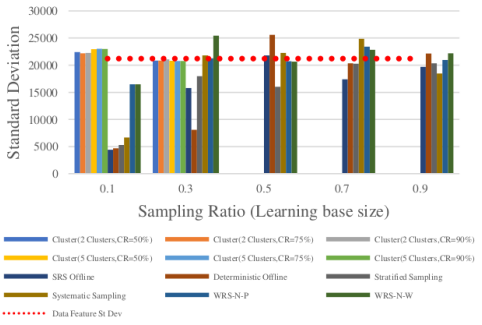


(c) OS values

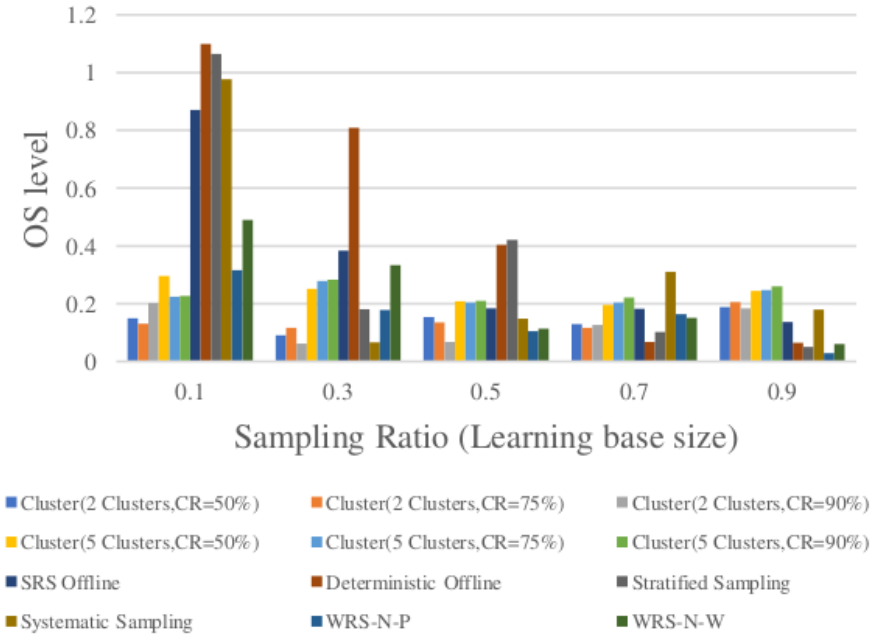
Figure 2.13: Mean, Stddev, and OS values for feature 5.



(a) Mean

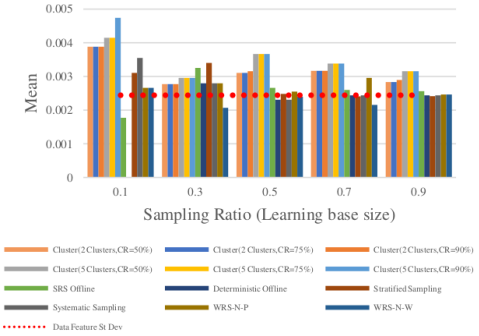


(b) Stddev

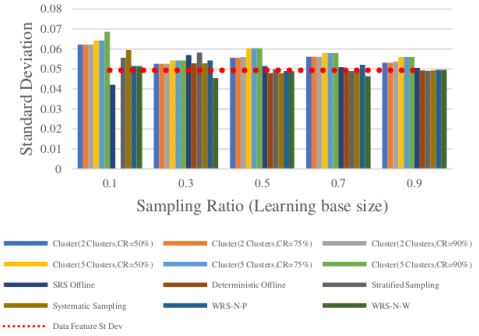


(c) OS values

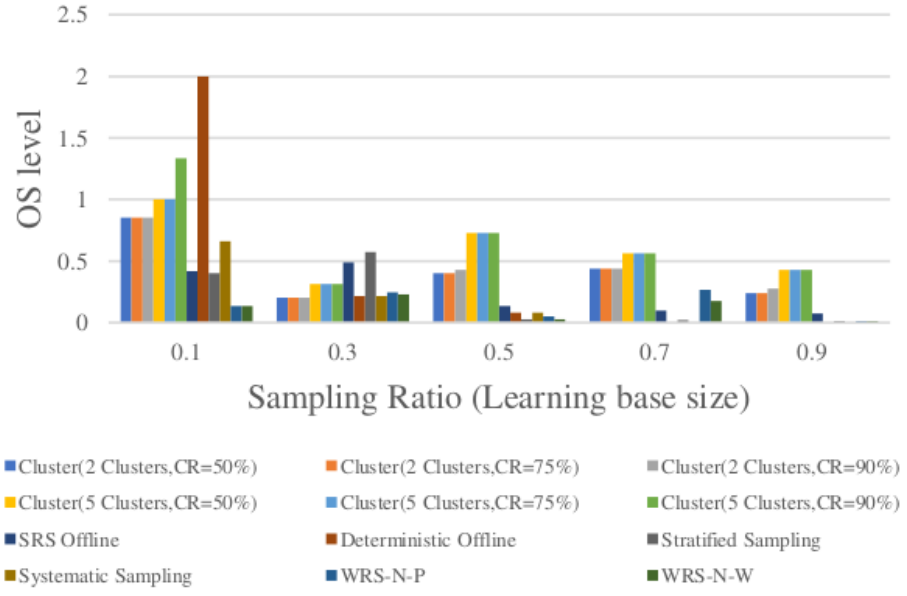
Figure 2.14: Mean, Stddev, and OS values for feature 6.



(a) Mean

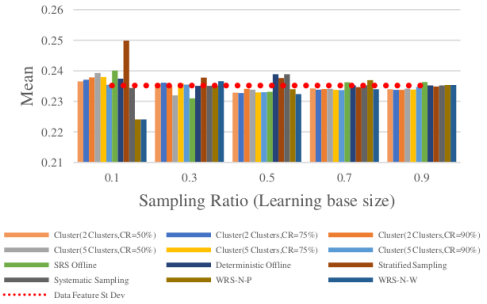


(b) Stddev

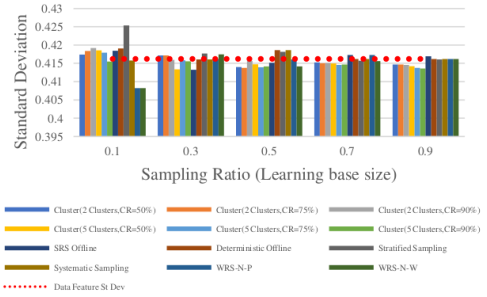


(c) OS values

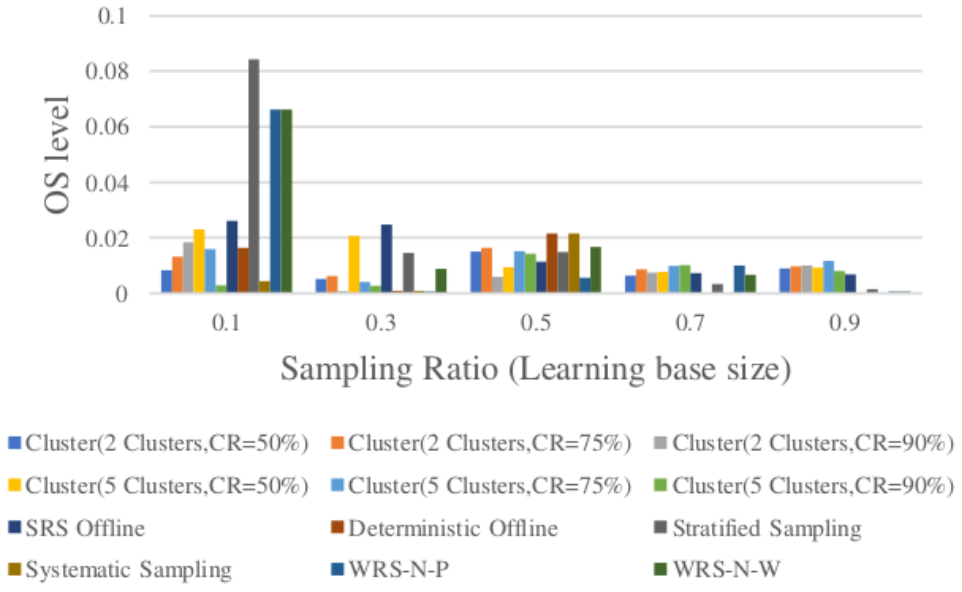
Figure 2.15: Mean, Stddev, and OS values for feature 14.



(a) Mean



(b) Stddev



(c) OS values

Figure 2.16: Mean, Stddev, and OS values for feature 28.

3

Cross-layer Federated IDS

3.1/ Introduction

Intrusion detection plays a pivotal role in securing IoT systems. As the number of interconnected devices continues to grow and data transmission exponentially increases, the potential for malicious actors to exploit vulnerabilities has become a significant concern [41]. Developing effective IDSs for IoT applications presents a fundamental challenge of efficiently handling large volumes of data while ensuring data privacy and minimizing energy consumption [42]. In this chapter, we will explore the importance of intrusion detection in IoT systems and delve into the challenges associated with managing data, privacy, and energy efficiency. By leveraging IDSs, we can effectively monitor network activities, detect intrusion attempts, and protect IoT systems from potential threats.

To address this challenge, we propose a lightweight federated sampling and intrusion detection approach based on an adaptation of the K-means clustering algorithm, known as the baseline k-means algorithm.

The baseline k-means algorithm serves as a semi-supervised novelty detection technique that trains a classifier using limited labeled benign data. It then utilizes this classifier to label additional unlabeled data points. This approach proves advantageous in scenarios where there is an abundance of unlabeled data but a scarcity of benign data. The proposed model employs two centroids, one for the baseline distribution and another for the anomalous distribution, and utilizes the Mahalanobis distance as a metric to account for the covariance of the distributions. Compared to the standard K-means clustering that relies on Euclidean distance, this approach enables a more robust and accurate identification of anomalous behavior.

3.1.1/ LIGHTWEIGHT IDS FOR IOT

Cybersecurity has become increasingly challenging with the rapid growth of the IoT and the massive volume of data generated by IoT devices, which is projected to reach 73.1 ZB (zettabytes) by 2025 [43]. Due to the limited computational capabilities of IoT devices, researchers have focused on designing lightweight IDSs that can provide the necessary security while operating efficiently on resource-constrained devices. In the literature, two tracks of lightweight IDSs have emerged: signature-based and anomaly-based. This work focuses on the anomaly-based track, which aims to detect abnormal behaviors and attacks in IoT systems. Various approaches have been proposed to address specific challenges in lightweight IDS design. Researchers such as Lee et al. [44] and Le et al. [45] have tackled energy consumption attacks by developing IDSs that limit sensing operations to cluster heads, allowing other nodes to operate normally. Jan et al. [46] have concentrated on creating computationally lightweight IDSs using supervised ML techniques like support vector machines, which can handle multiple attack types and scale to a large number of nodes.

Feature selection has also been explored to reduce computational overhead. Soe et al. [47] proposed a lightweight anomaly-based IDS strategy that selects features with the highest gain ratio while discarding others, reducing computation requirements. Davahli et al. [48] proposed a hybrid feature selection method combining genetic algorithms and the Grey Wolf Optimizer.

To enhance communication security in lightweight IoT devices, Khater et al. [49] combined feature reduction and supervised deep learning. They used Modified Vector Space Representation (MVSR) N-gram for feature extraction and a Multilayer Perceptron (MLP) model to classify network traffic. Selective IDS activation based on time was proposed by Sedjelmaci et al. [50]. They applied a game-theoretic approach to identify periods when attacks are most likely to occur and enabled IDS functionality accordingly. Deep Neural Networks (DNN) were employed to improve detection accuracy.

A recent development in lightweight IDSs is ”Realguard” by Nguyen et al. [51], which utilizes a DNN-based approach. Realguard implements a simple MLP with five hidden layers and achieves high attack detection accuracy while running on low-end IoT devices.

3.1.2/ FEDERATED LEARNING FOR IOT IDS

ML-based IDSs for IoT devices can be trained using cloud services, edge computing, or on-device ML solutions [42]. Cloud-based training has the advantage of unlimited resources and storage but can result in security and privacy concerns, high data transmission and energy consumption. Edge computing-based training processes data locally,

reducing storage and network bandwidth requirements and latency, but lacking highly configurable hardware opportunities and unlimited resources. On-device ML solutions offer real-time learning and training on the IoT end devices, with the added benefit of data privacy, but require significant computational power and data storage. The choice of training approach depends on the specific trade-offs be- tween resources, latency, and security and privacy concerns. Federated learning is a distributed ML technique that allows multiple IoT nodes to collaboratively train a global model without sharing their raw data with a centralized server. In the context of intrusion detection, federated learning can be used to detect attacks on IoT devices without compromising the privacy of the individual devices. Pei et al. [52] developed a data aggregation method for network traffic anomaly detection based on self-coding of long- and short-term memory networks. Attota et al. [53] employed multi-view classification and multi-view ensemble learning to enhance prediction accuracy while preserving privacy. Mothukuri et al. [54] used Gated Recurrent Units in a federated learning process.

To address data poisoning attacks, Nguyen et al. [55] incorporated Generative Adversarial Networks (GANs) into federated learning. GANs help mitigate data poisoning attacks, as demonstrated in [56] and [57]. Saadat et al. [58] compared hierarchical federated learning with federated learning in the IoT context and showed that hierarchical federated learning outperforms federated learning in terms of training loss, testing accuracy, and convergence speed. This aligns with the work of Sarhan et al. [59], where a hierarchical federated learning framework was designed using blockchain technology. Federated learning for IoT IDS has been extensively studied and benchmarked in the literature, as evidenced by works such as [60, 61, 62, 63, 64, 65].

3.1.3/ OBJECTIVE AND CONTRIBUTION

Our primary objective is to present a federated, lightweight, and privacy-preserving IDS tailored for lightweight IoT environments. We introduce a novel approach for real-time intrusion detection in IoT applications by combining a federated version of the baseline k-means algorithm with a cluster-based sampling method, referred to as the ”cross-layer” federated IDS. This approach addresses the challenge at two distinct layers: sampling and anomaly detection. By shifting the clustering process to the sampling layer, we achieve a co-optimized performance of both layers. The proposed solution adopts a cross-layer design, significantly reducing memory, power, and computational resources while enhancing detection capabilities without compromising individual nodes' privacy.

The cluster-based sampling technique serves as a crucial foundation for our proposed federated baseline k-means algorithm. It ensures a high level of data representation, including rare subgroups, effectively minimizing sampling errors caused by data variance.

Initially, the cluster-based sampling algorithm applies feature reduction to the available packets, decreasing the overhead based on a study that identifies essential features for detecting various attacks. The algorithm offers several advantages, including significantly improved efficiency compared to other sampling algorithms, particularly for very low sampling rates resembling a lightweight IoT environment. Additionally, it provides extended traffic visibility, vital in countering slow-rate attacks. Subsequently, the K-means clustering algorithm is employed to partition the data stream into homogeneous clusters, which are then sampled proportionally according to a chosen sampling rate, ensuring consistent representation of the cluster data even after sampling. In our approach, multiple IoT devices actively participate in the training process by computing their local statistics, such as means and distances to the benign distribution. These statistics are transmitted to a central coordinator for aggregation, allowing each IoT node's model to be updated with recent and representative statistics without compromising data privacy. Moreover, by employing cluster-based sampling prior to the intrusion detection process, we reduce processing energy consumption and the volume of data to be transmitted, thereby extending the lifetime of IoT nodes. The federated baseline k-means algorithm offers a practical, scalable, and privacy-preserving solution for intrusion detection in IoT applications, effectively addressing energy efficiency concerns. We provide a comprehensive explanation of our proposed algorithm and its integration with the cluster-based sampling technique. Furthermore, we present experimental results validating the effectiveness and efficiency of our approach in detecting intrusions in IoT applications, showcasing the progressive enhancement of IoT nodes' performance over time through the merging and sharing of statistics between the coordinator and the workers.

The remaining sections of this chapter are structured as follows: Section 3.2 presents a comprehensive taxonomy of IDSs based on four primary characteristics. It provides a systematic classification that helps understand the different types of IDSs and their features. Additionally, it outlines well-known computer network threats, highlighting the specific risks that organizations face in today's digital landscape. Furthermore, this section includes a taxonomy of available datasets specifically curated for benchmarking IDSs, enabling researchers and practitioners to evaluate the performance and effectiveness of different intrusion detection approaches. Then, in section 3.3, we present our proposal of a cross-layer federated learning framework for intrusion detection. We explore the integration of federated learning techniques with cross-layer analysis to enhance the accuracy and efficiency of IDSs. This novel approach leverages the collaborative power of federated learning and sampling to improve threat detection and response. We discuss the benefits, challenges, and potential applications of cross-layer federated learning in the context of intrusion detection. Finally, section 3.4 concludes the work.

3.2/ Related Work

3.2.1/ Intrusion Detection Systems

In the realm of network security, intrusion detection plays a pivotal role in upholding the confidentiality, integrity, and availability of data exchanged over networks. As illustrated in 3.1, intrusion detection entails the vigilant monitoring of network traffic from diverse sources to identify abnormal patterns or anomalies, swiftly notifying the network administrator upon detecting any potential attacks. Unlike firewalls, IDSs do not impose strict restrictions on end-users to adhere to predefined network security policies. Instead, IDSs employ a variety of detection techniques and algorithms to monitor network traffic and identify anomalies or specific patterns [66]. Furthermore, IDSs maintain a comprehensive record of previous attacks, facilitating incident response and aiding security teams in recognizing emerging threats.

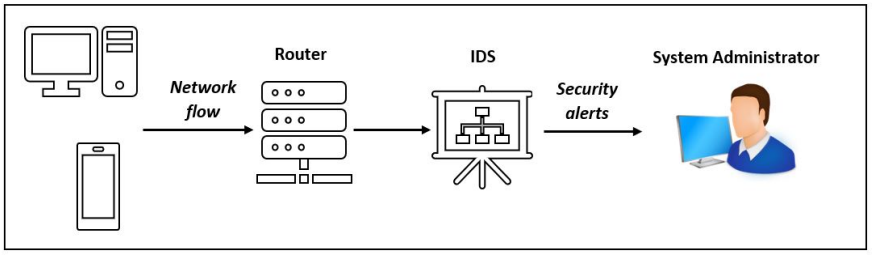


Figure 3.1: Intrusion Detection System.

In recent years, the research community has extensively explored the field of intrusion detection, leading to the publication of numerous studies and works. While Table 3.1 provides citations for recently published surveys focusing on anomaly-based intrusion detection techniques, datasets, or both, none of these surveys offer a comprehensive review encompassing intrusion detection approaches, IDS types, IDS requirements, available datasets, attack forms, categories, tools, and relevant features for collectively detecting attacks.

One contribution in our work is to present an updated survey of IDSs that addresses these research gaps. Our work distinguishes itself from other surveys in several key aspects. Firstly, we thoroughly discuss the requirements of IDSs, providing a comprehensive and well-structured survey of Network Intrusion Detection Systems (NIDSs), with a particular emphasis on those employing anomaly detection techniques. We meticulously analyze their advantages and disadvantages, introducing a taxonomy of IDSs based on four essential criteria. For each criterion, we explore various approaches and assess their performance.

Moreover, unlike the majority of existing surveys that focus solely on four network attacks

(Dos, Probe, R2U, and U2R), we present a taxonomy encompassing a wide range of network attack forms, and we delve into the appropriate features required for detecting these attacks. Additionally, we summarize the most commonly utilized datasets for IDS benchmarking, along with the associated challenges they present. This includes a taxonomy and comparison of these datasets based on their properties. Furthermore, we discuss the performance criteria employed for evaluating IDSs.

Lastly, we address recent research challenges within the field and propose potential solutions to overcome them. A summary of this comprehensive discussion can be found in Table 3.1, which includes a comparison of various aspects such as Signature-based IDS as **C1**, Anomaly-based IDS as C2, Hybrid IDS as **C3**, IDSs taxonomy based on several criteria as **C4**, IDS requirements as **C5**, Datasets taxonomy and issues as **C6**, Data types in datasets as **C7**, Attacks taxonomy as **C8**, Attacks types and tools as **C9**, and Attacks features as **C10**. By providing a holistic view of IDSs, their characteristics, and their performance, our survey aims to offer valuable insights to both researchers and practitioners, contributing to the advancement of intrusion detection techniques.

Table 3.1: Comparison of this survey and similar recent works.

Reference	Year	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Liao et al. [67]	2013	✓	✓	x	x	x	x	x	x	x	x
agrawal et al. [68]	2015	x	✓	✓	x	x	x	x	x	x	x
Buczak et al. [69]	2015	✓	✓	✓	x	x	x	x	x	x	x
Ahmed et al. [70]	2016	x	✓	x	✓	x	✓	x	x	x	x
Hodo et al. [71]	2017	✓	✓	✓	✓	x	x	x	x	x	x
Khraisat et al. [72]	2019	✓	✓	✓	x	x	✓	x	✓	x	x
Fernandes et al. [73]	2019	✓	✓	✓	✓	x	x	x	✓	x	x
This survey	2019	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

In the existing literature, IDSs are classified based on various criteria. In this thesis, we classify IDSs based on four key criteria: data sources, detection strategy, detection mode, and architecture, as illustrated in Figure 3.2.

- Data Sources: The first criterion is data sources, distinguishing between Host Intrusion Detection Systems (HIDSs) and Network Intrusion Detection Systems (NIDSs). HIDS monitors traffic generated by specific hosts, enabling the tracking of individual user behavior and aiding in the detection of insider threats. Conversely, NIDS analyzes information from all network traffic, monitoring every packet transmitted across the network to identify attacks and malicious behavior [67]. Table 3.2 provides a summary of the advantages and disadvantages of Host-based and Network-based IDSs.
- Detection Strategy: The detection strategy in IDSs can be categorized into two types: misuse detection techniques (signature-based techniques) and anomaly-

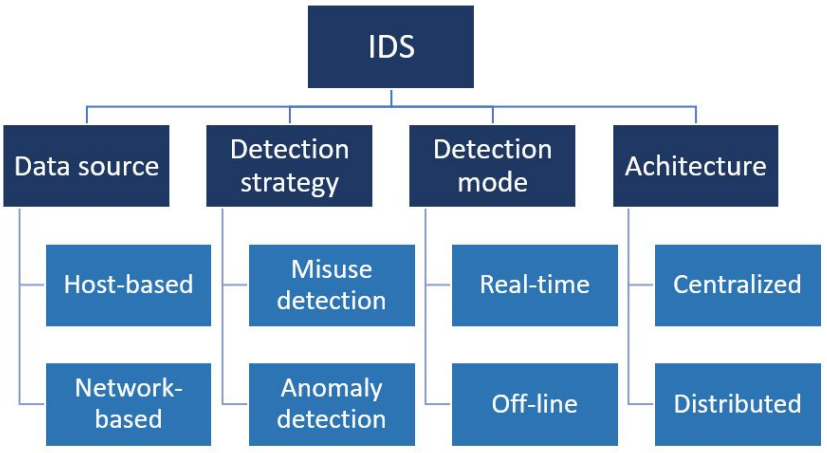


Figure 3.2: Taxonomy of Intrusion Detection Systems (IDSs).

Table 3.2: Comparison of Host-based and Network-based IDSs.

	Advantages	**Drawbacks**	**Data sources**	**Examples**
Host	Ability to detect malicious or improper activities of all the users Very effective in detecting suspicious user behaviors Ability to operate in encrypted environments	They can get compromised as soon as the host server is compromised by an attack Cannot monitor network traffic High system resources consumption High setting-up and management costs Portability issue	Audits records Security audit information System log files System accounting System calls System configuration	PortSentry [74] Lhotsky [75] Kim [76]
Network	Extremely portable Independent of the installed operating systems Real-time detection Low deployment and management costs	Scalability issue Cannot operate in encrypted environments	Simple Network Management Protocol (SNMP) Network connections Network traffic packets	RealSecure [**?**] SecureNet [77] Roesch [78]

based detection techniques, as depicted in Figure 3.3. Signature-based techniques compare actual traffic data with predefined patterns stored in an attack signatures database, providing high detection accuracy for known attacks. However, they struggle to detect rare and unknown attacks and require regular updates to the attack signatures database [68]. On the other hand, anomaly-based IDSs identify anomalies by comparing network traffic with normal behavior, making them effective in detecting unknown attacks. However, they face challenges in determining normal network traffic patterns, resulting in a higher false alarm rate [71]. Figure 3.4 illustrates the three main phases of anomaly-based IDSs: data collection, training phase, and testing/detection phase.

- Detection Mode: IDSs are categorized into online and offline modes based on how network data is analyzed. Online IDSs inspect network packets in real-time, providing immediate detection and response to malicious activity. Offline IDSs perform

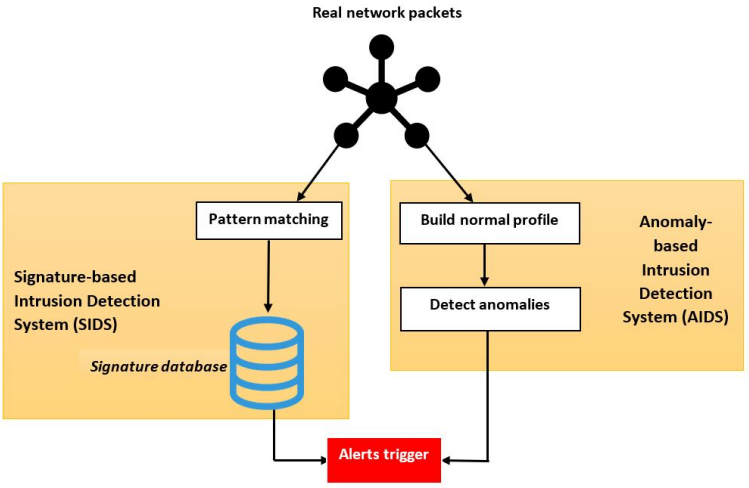


Figure 3.3: SIDSs vs AIDSs detection main phases.

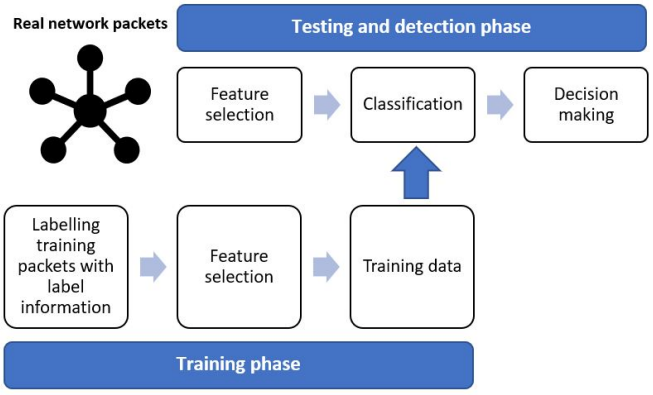


Figure 3.4: Anomaly-based intrusion detection phases.

post-analysis of recorded data, offering the advantage of lower computational resource requirements. However, they cannot provide real-time responses to prevent attacks or mitigate their damage [72].

- Architecture: The IDS architecture determines how traffic is monitored and analyzed. Centralized IDSs involve collecting data from individual hosts and analyzing it in a central entity. While easier to manage, centralized IDSs face scalability and single point of failure limitations [73, 79, 80, 81]. Distributed IDSs consist of multiple IDSs monitoring individual systems and cooperating to detect intrusions, offering fault-tolerance and scalability advantages [82, 74, 75, 76]. Table 3.3 summarizes the advantages and limitations of centralized and distributed IDS architectures.

By classifying IDSs based on these criteria, this thesis aims to provide a comprehensive understanding of IDSs, their capabilities, and their applicability in different network environments.

Table 3.3: Comparison of IDS architectures.

	Advantages	Limitations	Examples
Centralized	More secure All the monitoring and detection activities of the IDS are managed by a central unit Require low maintenance and administration costs	Cannot detect attacks occurring at different locations at the same time Prone to single point of failure Large computational and storage resources are required	Lunt [83] Hochberg [84] Mantur [85] Wang [86] Denning [87] Hidoussi [88] Abhishek [89] Zkik [90]
Distributed	Flexibility Scalability Reliability Low computational cost Attack anticipation is possible No single point of failure Fast processing	Less secure High deployment costs High network traffic overheads	Abraham [91] Maglaras [92] Folino [93] Riyad [94] Kannadiga [95] Zhang [96] Idhammad [97] Li [98]

3.2.2/ NETWORK ATTACKS

As networks grow in complexity, they also become more susceptible to attacks originating from two types of intruders: external intruders seeking network access from the internet and legitimate users aiming to misuse their authorized privileges for unauthorized gains. According to [99], attackers typically follow a three-step process to launch their attacks: (1) Information gathering, where the attacker collects sensitive information about the target network; (2) Vulnerability assessment, where the attacker attempts to compromise hosts within the network; and (3) Attack launch, where the attacker initiates the attack on the target network using the compromised hosts. [77] classifies these attacks into two main categories: information gathering and attack launching, as depicted in Figure 3.5.

- Information Gathering: Information-gathering tools can be categorized into two classes: sniffing tools and scanning tools [77]. Sniffing tools intercept, capture, and analyze data transmitted across the network, enabling diagnosis of network issues, monitoring of network usage and activity, identification of network vulnerabilities, filtering of network traffic, and identification of network configuration issues. Scanning tools are used to gather information about active hosts in the network and identify their vulnerabilities to potential attacks. Table 3.4 provides a summary of existing sniffing and scanning tools.

- Attack Launching: Attack launching tools are utilized by intruders to generate attacks against victim hosts within the network. These attacks may involve disabling certain computers, stealing sensitive data, blocking or deleting resources, or launching further attacks through compromised hosts. The following attack tools are noteworthy

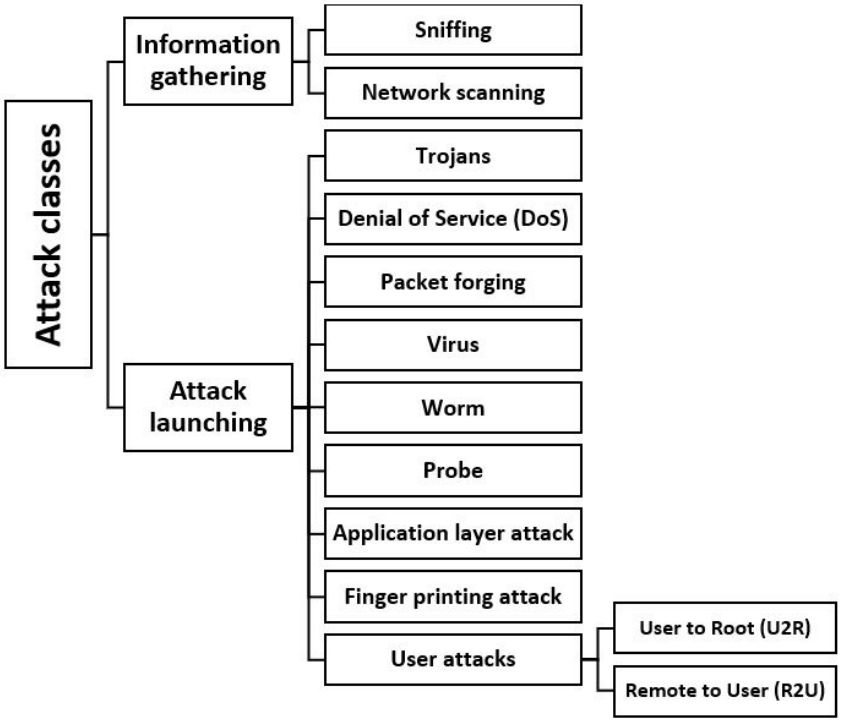


Figure 3.5: Taxonomy of computer network attacks. [Adapted from [100]]

- Trojan: Malicious software employed by attackers to breach system security and gain unauthorized access.
- DoS: DoS attacks involve overwhelming the server with an excessive number of connection requests, rendering the targeted website inaccessible to legitimate users. Table 3.5 provides an overview of DoS attack variations.
- Packet forging: Manipulating and generating traffic packets, which are then injected into the network connection.
- Virus: Malicious code designed to disrupt the normal functioning of a computer system, capable of self-replication by spreading across multiple computers.
- Worm: A malicious program that replicates and spreads itself from one host to another.
- Probe: An attack enabling intruders to gather information about the network or a specific host, such as identifying applications in use or the number of active machines on the network. Table 3.6 outlines different forms of probe attacks.
- Application layer attack: This tool targets web servers using Hyper Text Tranfer Protocol (HTTP) requests originating from legitimate hosts.
- Fingerprinting attack: A traffic analysis tool used by intruders to sniff and identify specific features and characteristics of an encrypted network.

Table 3.4: Summary of network information gathering tools.

Tool name	Category	Purpose	Explanation	Reference
Snoop	Sniffing	Packet capturing	It is a Linux packet analyzer software. It allows the attacker to filter, capture, read, interpret, and visualize packets data.	www.softpanorama.org
Tcptrace	Sniffing	Traffic analysis	It is used by the attacker to analyze tcpdump files and to produce various types of outputs including information about the connections, such as the elapsed time, round trip times, throughput, etc.	www.tcptrace.org
Nmap	Scanning	Scanning	It is used to scan large networks in a fast manner. It can identify different parameters and configurations about the network such as the available hosts and the services offered by them, the running operating system, and the used firewalls.	www.insecure.or
Ike-scan	Scanning	Host discovery	This tool uses IKE protocol to discover IPSec VPN servers.	www.stearns.org
Tcpinject	Attack launching	Packet generator	This tool is used by the attacker to transmit TCP/IP packets to a specific host in the network by specifying several parameters, such as the source IP address, destination IP address, source port, destination port, TCP window size, payload, etc.	www.packetstormsecurity.org
Dsniff	Sniffing	Password sniffing	It is password sniffer tool that permits the attacker to intercept the network and perform man-in-the-middle attack against SSH and HTTPS sessions.	www.naughty.monkey.org
FSMax	Attack launching	DoS	It is DoS attack that tests a server to find buffer overflows that may be exploited during an attack.	www.brothersoft.com
Nast	Sniffing	Traffic analysis	It captures the header and payload information of network packets, and saves them in files.	www.nast.berlios.de
RefRef	Attack launching	DDoS	It is a DoS attack that exploits MySQL vulnerabilities using features included in MySql to perform SQL injection.	www.hackingalert.net
Libnet	Attack launching	Packet injection	It is a popular attack tool used by intruders to construct and inject new packets in the network through an interface.	www.packetfactory.net/libnet

- Remote to User (R2U): During this attack, the intruder illicitly gains access to an administrative account. Table 3.7 illustrates various forms of R2U attacks.
- User to Root (U2R): The intruder attempts to access the victim host as a legitimate user to send packets over the network. Table 3.8 presents different forms of U2R attacks.

3.2.3/ AVAILABLE DATASETS

Datasets play a critical role in evaluating and comparing IDSs and are considered the backbone of state-of-the-art research in this field. These datasets enable researchers to assess the performance of their systems, including their effectiveness in quickly and accurately detecting various types of malicious activities. In this section, we present a

Table 3.5: DoS attack forms

Attack name	Description
Neptune	The attacker generates a SYN Flood attack by sending session establishment packets using a fake source address. The receiving host will be waiting for the session to be confirmed which will make it unavailable.
Smurf	It is a resource consumption attack where the attacker uses all the available bandwidth by flooding the target server with Internet Control Message Protocol (ICMP) packets, thus making the bandwidth unavailable for legitimate users.
Land	The attacker accesses the server by sending a TCP SYN packet where the IP address and port number are the same. When the target machine tries to reply, it will send the replies to itself, entering thus in an infinite loop, causing the server to crash.
Back	It is an attack on the Apache web server where the intruder sends requests with URLs containing many front slashes. This will slow down the server which becomes unable to process these requests and the upcoming ones.
Ping Of Death (POD)	The attacker tries to crash and destabilize the targeted server by sending malformed packets or packets larger than the maximum allowable size.
Teardrop	The attacker sends fragmented packets to the target server, making the server unable to reassemble these packets due to TCP/IP vulnerabilities. The packets will overlap one another, crashing the target server.

Table 3.6: Porbe attack forms

Attack name	Description
Ipsweep	This tool identifies the hosts that are listening on the network. Such information is useful to the attacker to help him launch other attacks and find vulnerable machines.
Nmap	It is used to scan large networks in a fast manner. It can identify different parameters and configurations about the network, such as the available hosts and the services offered by them, the running operating system, and the used firewalls.
Portsweep	This attack aims to determine the hosts and open ports available in the network. This information helps the attacker find vulnerable machines.
Satan	It collects data from remote hosts and networks by examining multiple network services such as NFS, NIS, FTP, statd, etc.

comprehensive summary of several cutting-edge datasets and discuss their characteristics, limitations, and relevance in the context of state-of-the-art IDS evaluation.

- DARPA Dataset: The Defence Advanced Research Project Agency (DARPA) dataset is created in 1998 at the MIT Lincoln Lab, stands as one of the foundational datasets for IDS evaluation. It consists of seven weeks of real network traffic data captured in packet-based format, along with host log files. This dataset, with its 5 million labeled records and 41 attributes, continues to serve as a benchmark for

Table 3.7: Remote to User (R2U) attack forms

Attack name	Description
FTP write	It occurs when the intruder takes advantage of a bad FTP configuration. The attacker can thus create and add files, and gain illegitimate access to the system.
Guess password	It is an attack on guest accounts without a password or with an easy-to-guess password.
Imap	It exploits the vulnerability in the IMAP server to allow the attacker to execute arbitrary commands on the server with root privileges.
Phf	The attacker abuses a badly written CGI script to perform an attack against the Web Server running the CGI script.
Warezclient	This attack is usually initiated after the warezmaster attack.
Warezmaster	This attack exploits a bug in the FTP server and occurs when the FTP server gives users write permission by mistake.

Table 3.8: User to Root (U2R) attack forms

Attack name	Description
Buffer overflow	Buffer overflow attack gives the attacker control over a privileged program that accepts input data from the user. It occurs when a program attempts to write excess data to a fixed-length buffer.
Load module	It is an intrusion on SunOS 4.1 systems that use the xnews window system. It loads two dynamically loadable kernel drivers into the currently running system and creates special devices in the directory to use those modules.
Rootkit	A rootkit attack allows the attacker to gain privileged access to a computer. It is installed through different malicious tools such as keyloggers, password stealers, antivirus disablers, etc., or by exploiting system vulnerabilities.
Perl	The Perl attack exploits a bug in some Perl implementations.

assessing IDS performance. However, it is essential to consider its limitations, such as the presence of artificially injected attacks and a significant number of duplicate records, which impact the accuracy of experimental results.

- KDD CUP 99 Dataset: The Knowledge Discovery and Data Mining (KDD) CUP 99 dataset, based on the DARPA dataset, was used for the competition held in 1999 as part of the KDD conference. it includes various types of attacks and comprises 5 million records, making it a valuable resource for studying network intrusion detection. However, one challenge associated with the dataset is the presence of a significant number of duplicate records. Addressing this limitation is crucial for developing detection algorithms that effectively handle both frequent and infrequent attack patterns.

- NSL-KDD Dataset: The Network Security Laboratory-KDD (NSL-KDD) dataset emerged as an enhanced version of the KDD CUP 99 Dataset, specifically de-

signed to overcome its limitations. By removing redundant records from the training and testing subsets, this dataset enhances the accuracy of intrusion detection results. With approximately 150K records encompassing 41 attributes and 22 attack types, the NSL-KDD Dataset serves as a vital resource for evaluating the performance of modern IDSs.

- CAIDA Dataset: The Center for Applied Internet Data Analysis (CAIDA) dataset is recorded in 2007, provides valuable insights into network traffic with Distributed Denial of Service attacks. However, it is important to note that this Dataset has certain limitations when used for evaluating IDSs. Its specificity to a particular type of attack and the absence of labels for distinguishing normal and anomalous traffic restricts its applicability.
- ISCX 2012 Dataset: The Information Security Centre of Excellence (ISCX) 2012 dataset offers a comprehensive view of real network traffic, encompassing diverse protocols such as HTTP, Internet Message Access Protocol (IMAP), and File Transfer protocol (FTP). With labeled and non-anonymized records collected over one week, it presents a wide range of network attack types, including Denial of Service and DDoS attacks. However, the absence of HTTPS records, which constitute a significant portion of current network traffic, hinders its realism and requires careful consideration during IDS evaluation.
- TUIDS Dataset: The Traffic Usage and Intrusion Detection Scenario (TUIDS) provides researchers with realistic network traffic captured in both packet-based and bidirectional flow-based formats. With its three subsets covering a period of seven days and containing approximately 250K flows, this dataset offers valuable insights for assessing IDS performance in complex network environments.
- Booters Dataset: The Booters dataset, recorded in a packet-based format, offers a vast amount of realistic, anonymized network traffic, making it an attractive resource for IDS evaluation. With its inclusion of ten different DDoS attacks executed against a null-routed IP address, this Dataset presents researchers with challenging scenarios to test and improve their detection algorithms.
- CIDDS-001 Dataset: The Coburg Intrusion Detection Dataset (CIDDS-001), recorded in 2017 from the OpenStack cloud server and external servers, provides a recent and extensive collection of labeled and anonymized traffic data in a flow-based format. Spanning four weeks with 32 million records and 14 attributes, this dataset serves as a valuable resource for evaluating IDS performance in modern network infrastructures [101].

We summarize in Table 3.9 the general information reflecting the properties of the datasets presented above.

Table 3.9: Summary of the datasets used for IDSs evaluation.

Dataset	Public	Year	Data volume	Duration	Labeled	Attack category	Traffic type	Data type	Anonymity
DARPA [102]	✓	1998	5M points	7 weeks	✓	DoS, Probe, R2U, U2R	Realistic	Packet-based, Log files	×
KDD CUP 99 [103]	✓	1998	5M points	7 weeks	✓	DoS, Probe, R2U, U2R	Realistic	Other	×
NSL-KDD [103]	✓	1998	148517 points	7 weeks	✓	DoS, Probe, R2U, U2R	Realistic	Other	×
CAIDA [104]	✓	2007	Not specified	1 hour	×	Denial of Service	Realistic	Packet-based	✓
ISCX 2012 [105]	✓	2012	2M points	1 week	✓	DoS, DDoS, SSH brute force	Realistic	Packet-based, Flow-based	×
TUIDS [106]	✓	2012	250000 points	21 days	✓	IRC, DDoS, port scans, SSH brute force	Realistic	Packet-based, Flow-based	×
Booters [107]	✓	2013	250 GB packets	2 days	×	DDoS attacks	Realistic	Packet-based	✓
CIDDS-001 [108]	✓	2017	32M points	4 weeks	✓	DoS, port scans, SSH brute force	Realistic	Flow-based	✓

3.3/ CROSS-LAYER FEDERATED IDS

3.3.1/ LIGHTWEIGHT SEMI-SUPERVISED INTRUSION DETECTION

IoT networks have experienced significant advancements, facilitated by technologies like Arduino and ESP32, enabling a wide range of applications. However, these progressions have given rise to new challenges, particularly concerning the resource limitations of IoT devices. To overcome these challenges, there is an increasing demand for lightweight and efficient algorithms. In this section, we delve into the realm of IoT intrusion detection and propose the baseline k-means approach to address this need.

K-means clustering, renowned for its efficiency and low resource requirements, emerges as a promising solution for IoT intrusion detection. By optimizing computations and memory usage based on the data points, features, and clusters, K-means is well-suited for handling small datasets and a restricted number of clusters. Consequently, it proves to be an excellent fit for resource-constrained IoT devices.

Moreover, we explore the federated implementation of the proposed approach, facilitating collaborative learning and ensuring data privacy. By adopting a federated framework, multiple IoT devices can collectively contribute to the intrusion detection process, sharing insights and knowledge while preserving the confidentiality of their individual data. In summary, the baseline k-means approach presents an efficient and lightweight solution to tackle the resource constraints faced by IoT devices. Its compatibility with small datasets and limited clusters, along with the federated implementation for collaborative learning

and data privacy, positions it as a promising technique for IoT intrusion detection.

3.3.1.1/ CROSS-LAYER

In this section, we present a lightweight federated implementation of the baseline k-means algorithm specifically designed for IoT environments. Our model, known as "Cross-Layer Federated IDS," derives its name from the two layers it operates within: sampling and anomaly detection. We utilize our efficient cluster-based sampling algorithm, as introduced in Chapter 2, along with the novel federated baseline k-means algorithm, which we will describe here. By capitalizing on the fact that our sampling algorithm already establishes and maintains clusters in the first layer, we are able to alleviate a significant portion of the workload in the second layer. This leads to a noteworthy reduction in memory, processing, and power requirements for the entire process.

The term "cross-layer" in the literature is often used in the context of layers of the network stack. Amouri et al. [109] proposed a cross-layer IoT IDS that spans MAC and Network layers. Likewise, Canbalaban and Sen [110] proposed a cross-layer IoT IDS that spans link and routing layers. Long et al. [111] proposed cross-layer industrial IoT IDS that spans its three layers which are: the application layer, the network layer, and the perception layer. This work is aligned with Malik et al. [112] and Kore and Patil [113]. A more thorough analysis was conducted by Parween et al. [114]. But none of the research surveyed here incorporates both sampling and intrusion detection.

3.3.1.2/ BASELINE K-MEANS

The baseline k-means algorithm serves as a semi-supervised novelty detection technique. It leverages a small set of labeled benign data to train a classifier, which is then utilized to assign labels to additional unlabeled data points. This approach proves particularly advantageous in situations where there is an abundance of unlabeled data but a scarcity of labeled benign data. One notable aspect of our model is the utilization of the Mahalanobis distance function, instead of the commonly used Euclidean distance employed with k-means. The Mahalanobis distance takes into account the covariance of the distribution and provides a measure of the distance between a point and a distribution. It calculates the Mahalanobis distance, denoted as d, between a point $\mathbf{x}$ and a distribution characterized by its mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, using the following equation:

$$D_M(\mathbf{x}, \mu, \Sigma) = \sqrt{(\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu)} \tag{3.1}$$

In the equation, T represents the transpose of a matrix, $\boldsymbol{\Sigma}^{-1}$ denotes the inverse of the covariance matrix $\boldsymbol{\Sigma}$, and '$(\mathbf{x} - \mu)$' represents the difference between the point $\mathbf{x}$ and the

mean μ. The inclusion of the Mahalanobis distance and covariance matrix allows the algorithm to consider correlations between features and adapt the distance metric to the specific data distribution.

Our model employs two clusters, one for the baseline (benign) data and another for the anomalous distributions. The training process, illustrated in Algorithm 2, follows these steps: for each incoming point, we calculate its Mahalanobis distance from the centroid of the anomalous cluster. If the point falls within a predefined threshold distance from the cluster, it is added to the cluster. Once a sufficient number of anomalous points are present, the anomalous centroid is updated by selecting the farthest point from the baseline centroid. This process is repeated for a certain number of iterations or until convergence. During prediction, each point is assigned to one of the two clusters based on the Mahalanobis distance and the same distance threshold, as outlined in Algorithm 3. Algorithm 4 outlines the baseline k-means class, which is constructed with four input arguments:

- baseline_data: an array representing the baseline (benign) data used for training the model.
- max_iterations: an optional integer argument specifying the maximum number of iterations to be performed during training.
- percentile: an optional integer argument indicating the percentile of the Mahalanobis distances between each baseline data point and the baseline centroid used as the threshold.
- update_after_iteration: an optional integer argument determining the number of iterations after which the baseline and anomalous statistics are updated.

Figure 3.6 demonstrates how the percentile value influences the clustering of points around the benign centroid. A lower percentile value results in a lower threshold, potentially labeling more benign points as anomalous and leading to a higher false positive rate. This parameter allows for tuning the algorithm's sensitivity.

Algorithm 2 baseline k-means fit function.

```
Procedure fit(data)
    n ← number of data points in data
    d ← number of features in data
    num_iter ← 0
    while num_iter < max_iterations or not converged do
        num_iter ← num_iter + 1
        fit_clusters ← empty list for baseline and anomalous clusters
        if num_iter is a multiple of update_after_iteration then
            update()  // update baseline and anomalous statistics
        end if
        for i in range(n) do
            point ← i-th data point in data
            compute Mahalanobis distance to baseline centroid
            if anomalous centroid exists then
                compute Mahalanobis distance to anomalous centroid
                if distance to anomalous centroid is smaller than to baseline centroid
                times a threshold then
                    classify point as anomalous and add to anomalous cluster
                    continue to next point
                end if
            end if
            if distance to baseline centroid is below threshold then
                classify point as baseline and add to baseline cluster
            else
                classify point as anomalous and add to anomalous cluster
            end if
        end for
        update()  // update baseline and anomalous statistics
        if converged then
            break loop
        end if
    end while
    return clusters and centroids
end Procedure
```

3.3.1.3/ Federated Sampling And Intrusion Detection

The training process involves updating the cluster statistics, including centroids and distances to the benign centroid, which serve as crucial information for guiding the intrusion detection process. In this work, we propose a federated intrusion detection approach where multiple IoT devices actively participate in the training process. Each node independently computes its local statistics, such as means and distances, and shares this

Algorithm 3 baseline k-means predict function.

```
Procedure predict(new_data)
    baseline_distances ← empty list
    anomalous_distances ← empty list
    for i ← 1 to len(new_data) do
        point ← i-th data point in new_data
        compute Mahalanobis distance to baseline centroid
        if distance to baseline centroid is below threshold then
            // used to label additional unlabeled data points
            classify point as baseline and append it to baseline data
            append baseline distance to baseline_distances list
        end if
        if anomalous centroid exists then
            compute Mahalanobis distance to anomalous centroid
            append anomalous distance to anomalous_distances list
        end if
    end for
    create binary vector indicating whether each point is baseline or anomalous
    assign each new data point to the nearest centroid (0 for baseline, 1 for anomalous
    return the assigned clusters as a numpy array (0 for baseline, 1 for anomalous)
end Procedure
```

information with a central coordinator for aggregation. The coordinator then updates the baseline and anomalous centroids based on the merged statistics from the IoT devices and calculates a new threshold, as illustrated in Figure 3.7.

The algorithm consists of four steps:

1. Initialization (Algorithm 5): The coordinator initializes the k-means clustering model with a fixed number of clusters and shares the cluster statistics among the participating workers.

2. Local clustering: Each node utilizes its own local k-means clustering model on its sampled data to perform clustering.

3. Sharing cluster statistics: Each node shares its cluster statistics, including cluster centroids and distances to the benign centroid, with other nodes through the coordinator.

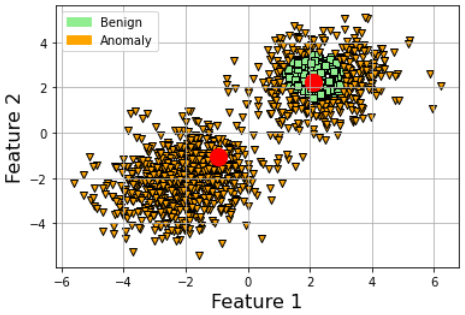
4. Merging statistics (Algorithm 6): The coordinator merges the cluster statistics received from each node to create a comprehensive global representation of the benign and anomalous clusters. The updated global model can then be shared with the worker nodes for further local training.

Algorithm 4 baseline k-means class shown in a Python-like pseudocode.

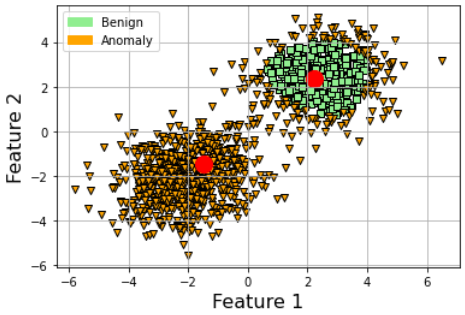
```
Input : Baseline data B, Anomalous data A, max iterations I, percentile P,
        update after iteration U
Output: Trained model with updated centroids and threshold
Class BaselineKMeans:

Function __init__(self, baseline_data, max_iterations,
                    percentile, update_af ter_iteration)
/* Initialize the model with baseline data representing the expected normal
behavior or performance of the system. */
end

Function update(self, baseline data, anomalous data, percentile)
/* Update the baseline and anomalous statistics */
/* The centroids, baseline_mean, baseline_cov, threshold, anomalous_mean, and
anomalous_cov values are returned as output from the function. */
end

Function append_baseline_anomalous(self, baseline_distance, point)
/* Append new data to either the baseline or anomalous data based on distance
to baseline centroid */
end

Function fit(self, data)
/* The function iterates until convergence or the maximum number of
iterations is reached. */
/* For each iteration, the function assigns each data point to the nearest
centroid based on Mahalanobis distance. */
/* The centroids are updated after a certain number of iterations, and the
function checks if the distance to the anomalous distribution is smaller
than to the baseline to determine if a point is an anomaly. */
/* The function stops iterating when the clusters do not change between
iterations or the maximum number of iterations is reached. */
end

Function predict(self, new_data)
/* Classify new data as baseline or anomalous based on distances and baseline
distance threshold */
end
```

3.3.2/ EXPERIMENTS

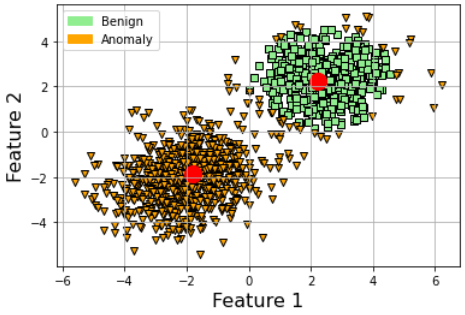
In this section, we present a comprehensive comparison of the proposed method with other semi-supervised novelty detection methods. We conduct simulations to evaluate the performance of our federated IDS approach, utilizing the cluster-based sampling method



BaselineKmeans with percentile value of 90%



BaselineKmeans with percentile value of 95%



BaselineKmeans with percentile value of 98%

Figure 3.6: This figure depicts how adjusting percentiles (90, 95, 98) for anomaly threshold calculation during data clustering impacts the trade-off between accurate anomaly detection and fewer false positives.

Algorithm 5 The export statistics function shown in a Python-like pseudocode

```
Output: Threshold, baseline mean, inverse baseline covariance, anomalous mean,
        inverse anomalous covariance

Function export_stats(self)
    linalgInv_baseline ← pinv(baseline_cov + 0.001 * identity(dimension));
    linalgInv_anomalous ← pinv(anomalous_cov + 0.001 * identity(dimension));
    return threshold, baseline mean, linalgInv_baseline,
            anomalous mean,linalgInv_anomalous;
end
```

described in the previous chapter. Through these simulations, we demonstrate how collaboration between workers and the coordinator enhances the overall effectiveness of the IoT IDS. Furthermore, we delve into the simulations focused on the cross-layer IDS, where intrusion detection is applied following the cluster-based sampling. We thoroughly analyze the impact of the sampling rate on IDS performance to gain valuable insights. To ensure reproducibility, we have made the Python code for the Baseline-Kmeans and subsequent simulations publicly available on GitHub [115]. It is important to note that

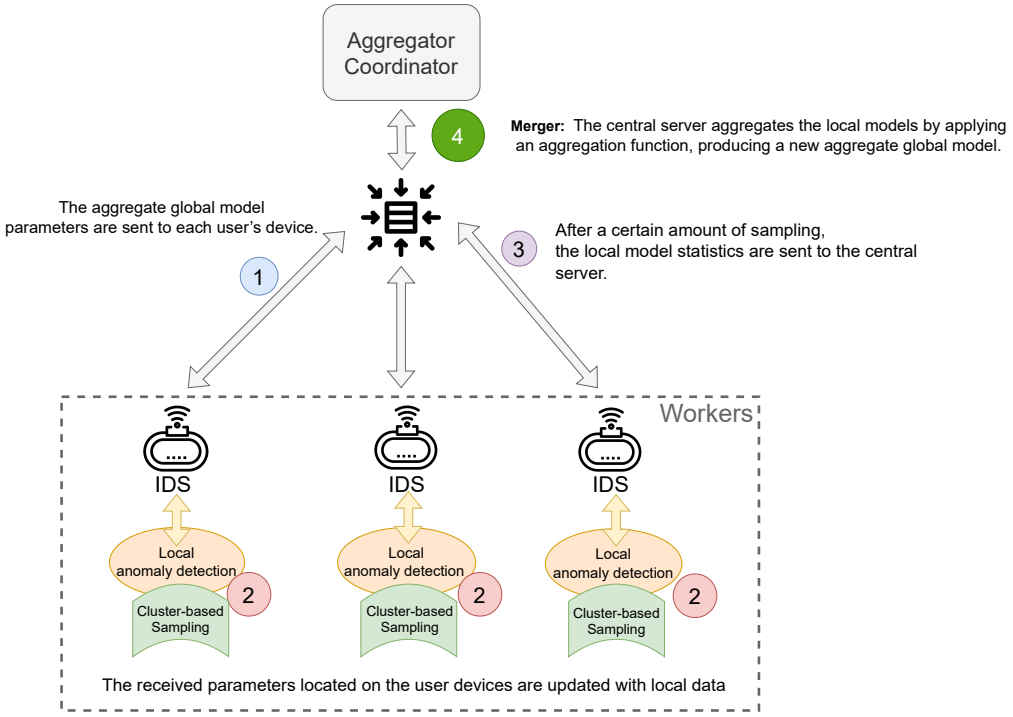


Figure 3.7: This figure presents a federated approach for intrusion detection in IoT networks. A BaselineKMeans class acts as the coordinator. Each worker utilizes a cluster-based sampling algorithm.

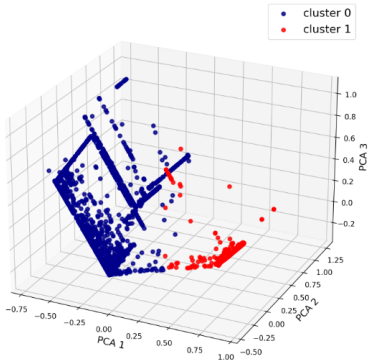
Algorithm 6 The merge statistics function shown in a Python-like pseudocode.

```
Input: Worker baseline means, worker anomalous means, distances from baseline
       points to new baseline centroid, distances from anomalous points to
       anomalous centroid
Output: Updated threshold and centroids

//Merge statistics function:
Function merge(self, worker_baseline_means, worker_anomalous_means,
               distances_inline, distances_outline)
    update centroids based on the mean of worker, baseline means,
    and worker anomalous means;
    linalgInv ← pinv(baseline_cov + 0.001 * identity(dimension));
    distances ← distances from baseline points to new baseline centroid;
    distances ← vstack((distances, distances_inline));
    min_outline ← min distance between outline points and benign centroid;
    remove the distances that are greater than min_outline;
    compute threshold based on the percentile of distances;
    return updated threshold and centroids;
end
```

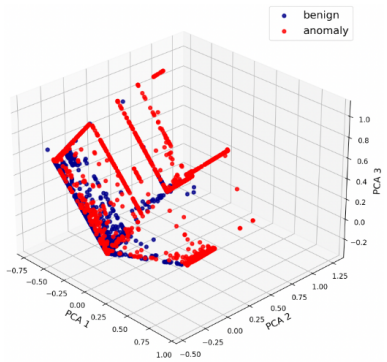
we employ the same NSL-KDD dataset used in our previous experiments for consistency and continuity [116, 103].

3.3.2.1/ SEMI-SUPERVISED LEARNING

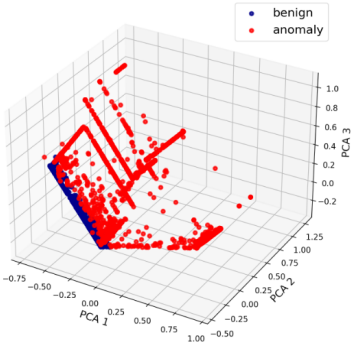
Supervised learning methods pose a significant challenge as they necessitate large volumes of labeled data that are difficult to obtain. On the other hand, designing and implementing unsupervised models proves challenging due to the requirement for well-defined boundaries between classes. This is precisely why a semi-supervised novelty detection approach utilizing K-means offers numerous advantages. To effectively demonstrate the rationale behind employing semi-supervised learning, we visualized the NSL-KDD data transformed with Principal Component Analysis (PCA) and compared the outcomes of unsupervised K-means clustering, the ground truth, and our proposed baseline k-means, as depicted in Figure 3.8.



(a) Scatter plot of 10000 data points with the unsupervised k-means (k=2) predicted classes (transformed with PCA)



(b) Scatter plot of 10000 data points with ground truth labels (transformed with PCA)



(c) Scatter plot of 10000 data points with the semi-supervised baseline k-means (percentile = 95) predicted classes (transformed with PCA)

Figure 3.8: Clustering the NSL-KDD data points using the standard unsupervised k-means and the proposed semi-supervised baseline k-means with k=2.

3.3.2.2/ SEMI-SUPERVISED NOVELTY DETECTION FOR INTRUSION

Novelty detection techniques in semi-supervised learning offer the advantage of utilizing a subset of benign data to learn normal traffic patterns, enabling the identification of anomalous traffic. In order to demonstrate the effectiveness and suitability of our pro-

posed baseline k-means algorithm for IoT intrusion detection, we compare it with alternative novelty detection techniques:

- One-Class Support Vector Machine (SVM): This technique maximizes the margin between benign samples and the origin, allowing for the detection of novel data points that deviate from the learned benign pattern.

- One-Class Gaussian Mixture Model (GMM): A generative probabilistic model that assumes benign data is generated from a mixture of Gaussian distributions. It estimates the parameters of these distributions using the Expectation Maximization (EM) algorithm, establishing the benign distribution for classification.

- Local Outlier Factor (LOF): A density-based algorithm that identifies outliers by comparing the local density of a data point with the densities of its neighbors. Data points with significantly lower densities than their neighbors are considered anomalies.

- Isolation Forest: This algorithm isolates data points by randomly selecting features and splitting them with random thresholds, creating an ensemble of trees. Anomalous data points require fewer splits and result in shorter path lengths, this property is exploited in the classification phase.

- Minimum Covariance Determinant (MCD): A statistical method for detecting outliers based on the Mahalanobis distance, considering feature correlations. MCD estimates the mean and covariance matrix of benign data by finding the subset with the smallest covariance determinant, and identifying anomalies exceeding a predefined threshold.

To evaluate the performance of each semi-supervised novelty detection algorithm, we provide 500 benign data points from our dataset for learning normal traffic patterns. Subsequently, we apply the techniques to classify the data and report precision, recall, and F1-score for each method within sliding windows of 1,000 data points. By calculating the average metrics across all windows, we enable a comprehensive comparison of their performance in IoT intrusion detection, as depicted in Figure 3.9.

The results demonstrate that our proposed method achieves the highest recall at 0.97 and a competitive F1-score at 0.85 compared to the other considered methods. This is particularly significant in intrusion detection as it emphasizes the capability to detect a greater number of real intrusions. As explained earlier, the sensitivity of our proposed method can be tuned to increase precision if desired.

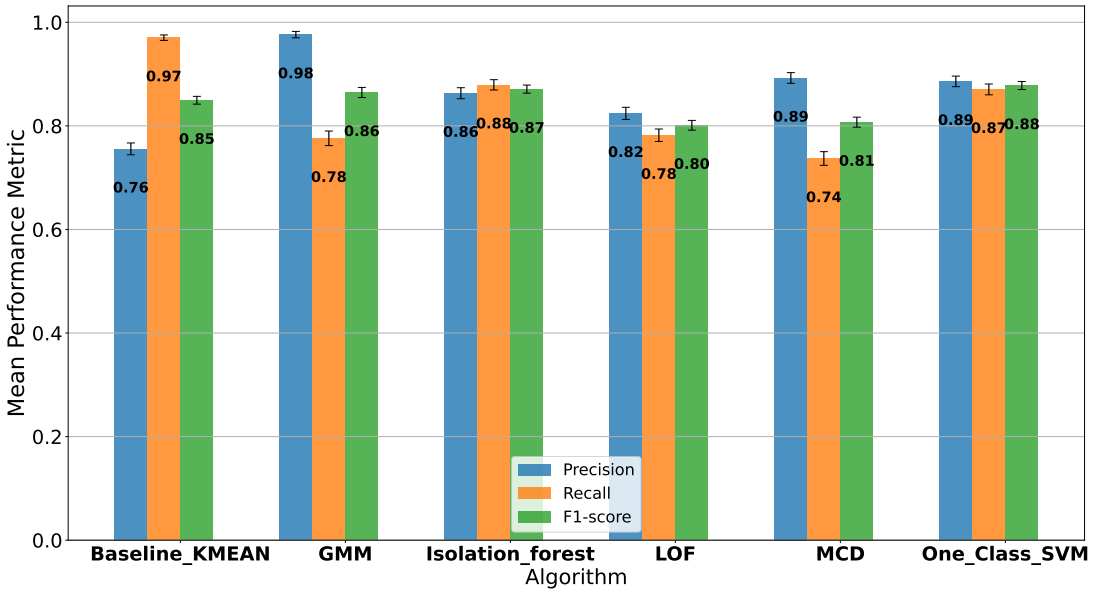


Figure 3.9: This bar plot compares semi-supervised novelty detection algorithms based on precision, recall, and F1-score. The Baseline K-means algorithm stands out with high recall and a competitive F1-score.

3.3.2.3/ FEDERATED LIGHTWEIGHT IDS

In this section, we present the results of the simulation conducted on our federated IDS, comprising one coordinator and three workers. The initial training involves the coordinator being trained with a dataset of N=100 benign data points. The coordinator then shares its statistics with the workers, and in each epoch, both the coordinator and the workers process 1000 data points. Performance metrics such as precision, recall, and F1 score are calculated for each window and aggregated for each node involved.

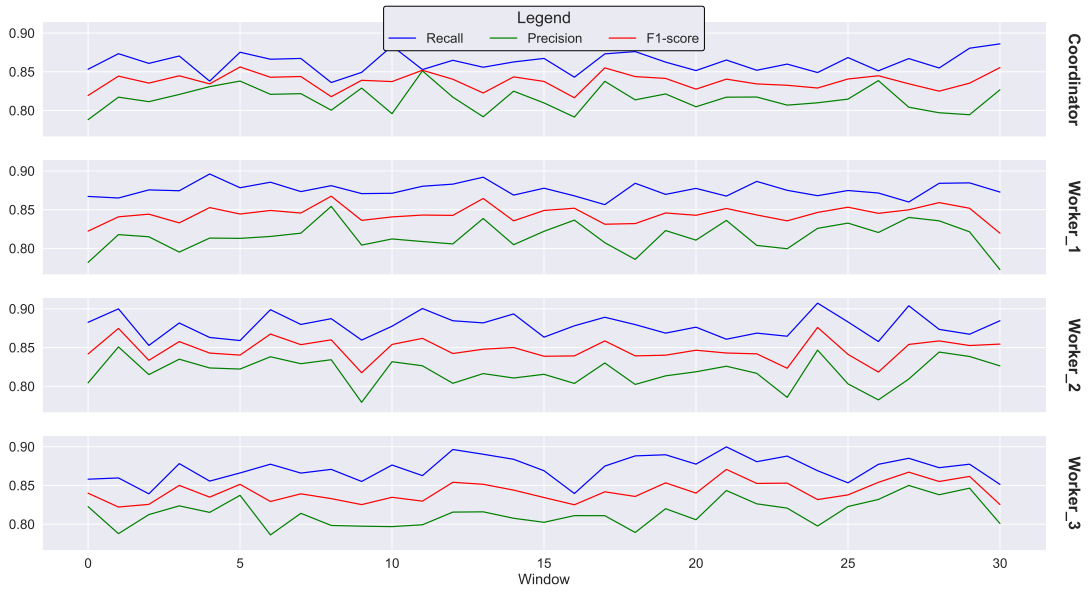
We consider three scenarios to analyze the impact of merging statistics on the system's performance and adaptability to evolving traffic. The first scenario involves no merging, where the coordinator shares its initial training statistics with the workers (Figure 3.10). The second and third scenarios incorporate three and four merging steps, respectively, enabling us to evaluate the contribution of merging operations.

The results of the first scenario, depicted in Figure 3.10-b, show that the workers' performance is comparable to, and occasionally slightly better than, that of the coordinator, with an average overall F1 score of approximately 0.84. As shown in Figure 3.10-a, there is no significant variation in performance metrics throughout the epochs since no sharing of new data occurs after the initial training phase.

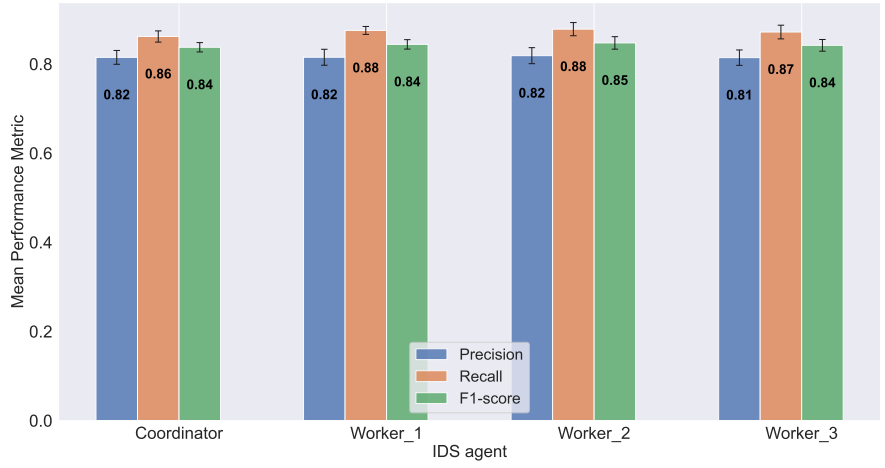
Figure 3.11 illustrates the results of the second scenario, where merging operations take place at epochs 7, 14, and 21. It is evident in Figure 3.11-a that recall improves after each merge, further validating the effectiveness of our method. The third scenario's results, as

depicted in Figure 3.12, exhibit even more pronounced improvement. By the end of the 30 epochs, the workers achieve a recall of 0.97, as shown in 3.12-a. 3.12-b reveals an average recall of approximately 0.96 for all workers and an increased F1 score of 0.86. However, similar to the second scenario, the precision decreases to 0.78.

It is worth noting that in both scenarios, as recall increases, precision decreases. Hence, it is crucial to consider the trade-off between precision and recall and determine appropriate intervals for merging operations.

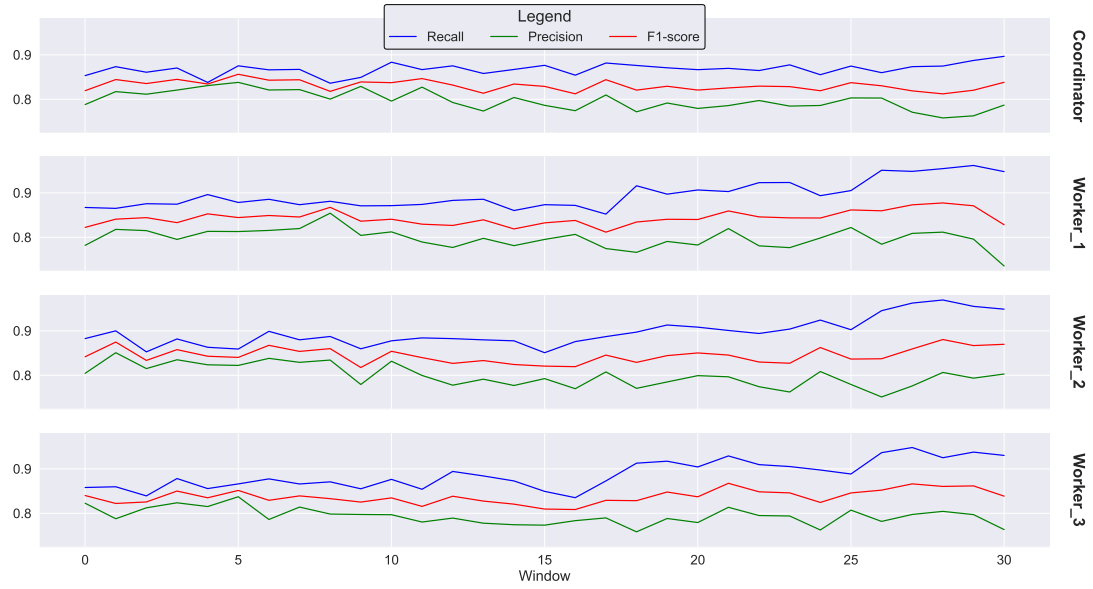


(a) Evolution of performance metrics for the coordinator and three workers over 30 epochs, each comprising a window of 1,000 data points.

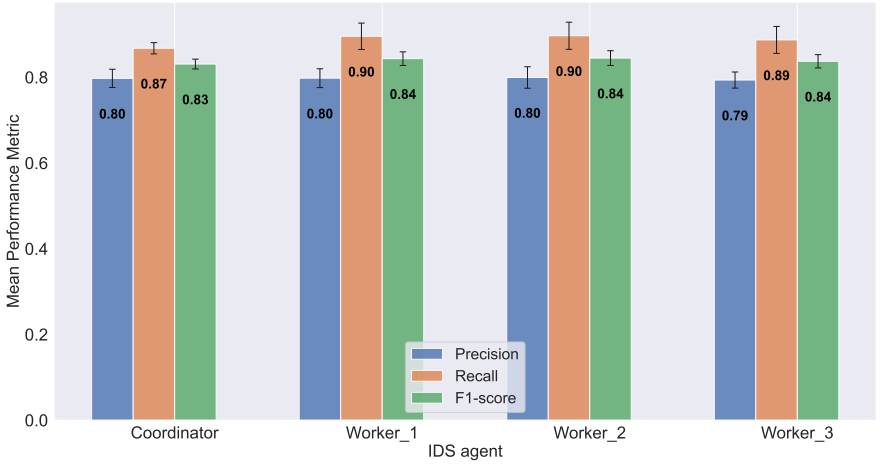


(b) Average performance metrics for the coordinator and three workers after 30 epochs.

Figure 3.10: Federated IDS without merging operations.



(a) Evolution of performance metrics for the coordinator and three workers over 30 epochs, each comprising a window of 1,000 data points.



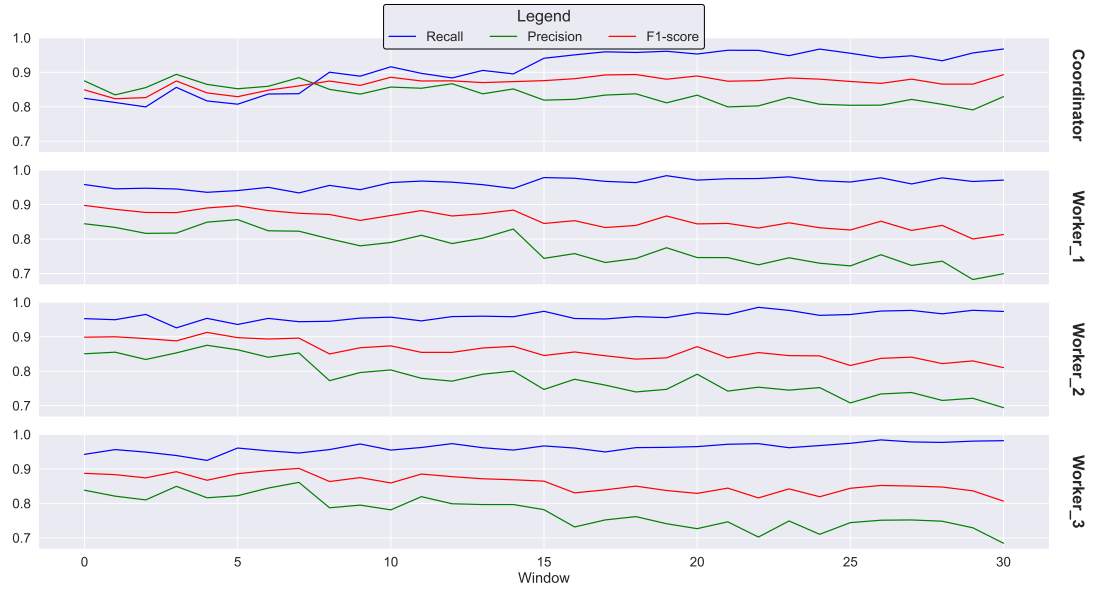
(b) Average performance metrics for the coordinator and three workers after 30 epochs.

Figure 3.11: Federated IDS with three merging operations.

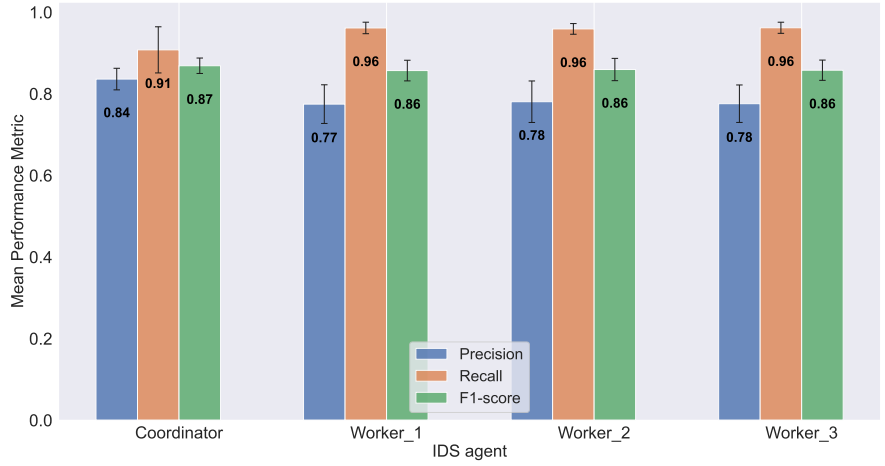
3.3.2.4/ CROSS-LAYER FEDERATED LEARNING

This section delves into the simulation of the cross-layer IDS, where intrusion detection is applied following the cluster-based sampling technique proposed in our previous work [117]. It sheds light on the impact of sampling on IDS performance metrics and emphasizes the significance of the sampling operation when implementing the IDS on a microcontroller in terms of processing time.

1. IDS Performance Metrics: We explore two scenarios, tracking performance metrics at three different sampling rates: 0.75, 0.5, and 0.25. In the first scenario, consisting



(a) Evolution of performance metrics for the coordinator and three workers over 30 epochs, each comprising a window of 1,000 data points.



(b) Average performance metrics for the coordinator and three workers after 30 epochs.

Figure 3.12: Federated IDS with four merging operations.

of one coordinator and one worker without merging statistics, we process 5,000 data points per epoch over 11 epochs. Figure 3.13 illustrates the results, indicating a decline in performance over time when utilizing sampled data through cluster-based sampling. This decline is more pronounced with lower sampling rates. In the second scenario, recall decreases from 0.88 to 0.87 at epoch 3, with the 0.25 sampling rate dropping to 0.83. However, after the first merging operation, recall increases to 0.90 for sampling rates 0.75 and 0.5, and 0.89 for the sampling rate 0.25 by epoch 7. By epoch 11, recall reaches 0.95 for sampling rates 0.75 and 0.5, and 0.93 for the sampling rate 0.25. These results, depicted in Figure 3.14,

emphasize the importance of the merging operation.

2. Processing and Implementation: To showcase the low processing requirements of our cross-layer federated IDS solution, we implement it on an Arduino Nano 33 microcontroller. We utilize an efficient C++ implementation of the Mahalanobis distance function, which plays a crucial role in our method. The coordinator exports various values such as the threshold, baseline mean, baseline covariance inverse, anomalous mean, and anomalous covariance inverse to each worker. The tests are conducted using a window of 100 data points, each containing six features at different sampling rates. Figure 3.15 demonstrates that the processing time decreases from approximately 32ms at a 100% sampling rate to around 9ms at a sampling rate of 0.25.

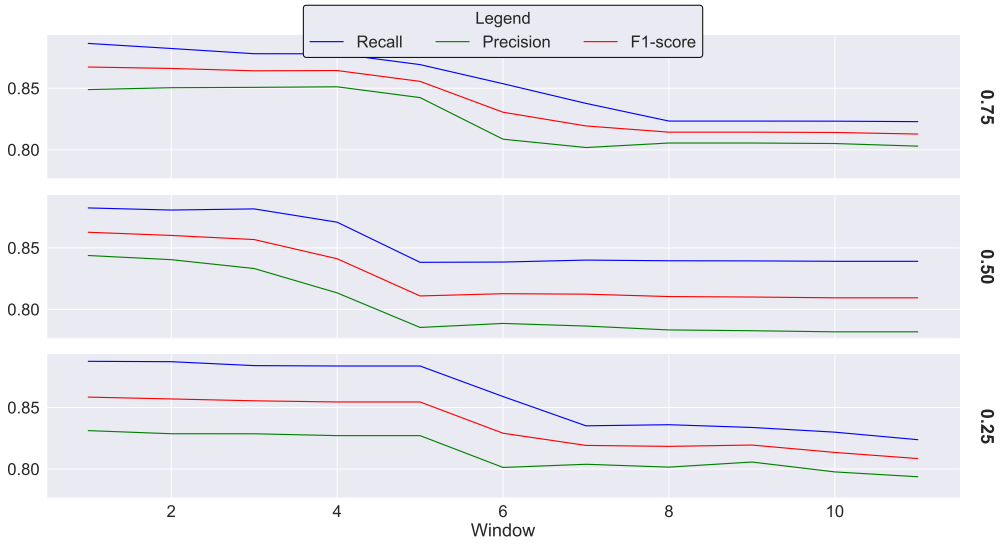


Figure 3.13: IDS performance metrics when applying cluster-based sampling with sampling rates of 0.75, 0.5, and 0.25, a change ratio of 0.5, and without merging operations.

3.3.3/ DISCUSSION

In this section, we delve into the details and explanations of the key observations derived from our experiments.

1. The performance of the coordinator does not improve over time. This can be attributed to the fact that the workers do not share their inverse covariance matrix with the coordinator. This design choice was made considering the computational burden of calculating and transmitting the covariance matrix on resource-constrained devices like microcontrollers. While utilizing the coordinator's covariance matrix is an alternative, it can result in diminished performance when there are significant

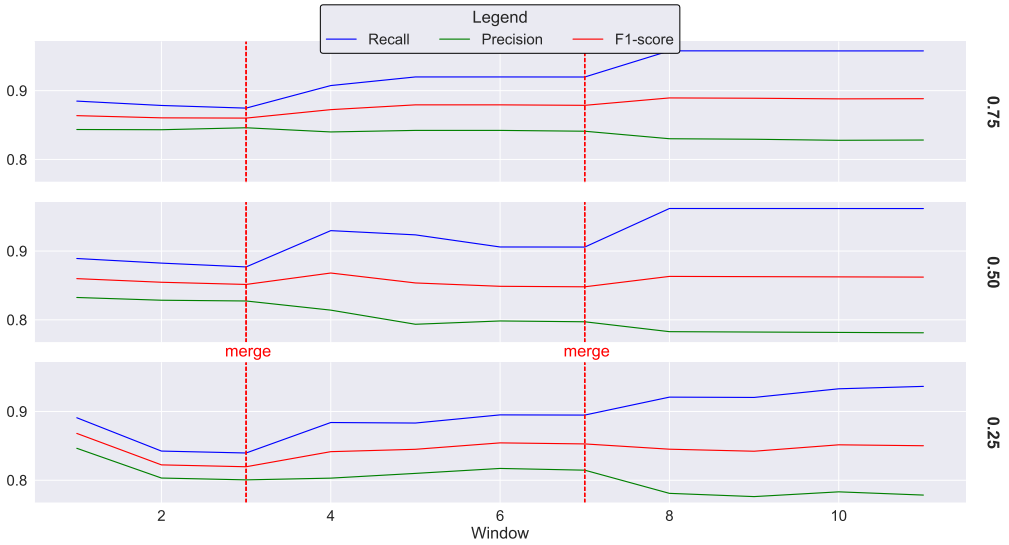


Figure 3.14: IDS performance metrics when applying cluster-based sampling with sampling rates of 0.75, 0.5, and 0.25, a change ratio of 0.5, and two merging operations.

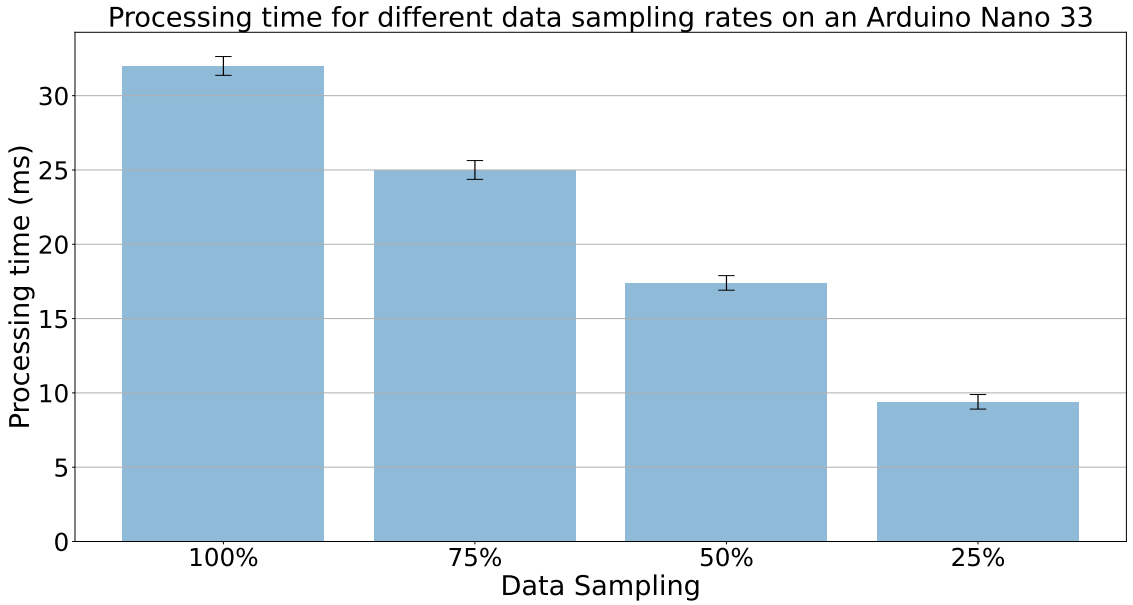


Figure 3.15: Processing time vs sampling rate on an Arduino Nano 33.

disparities in data point distribution across workers. A potential compromise could involve workers sharing a compressed or subset of their local covariance matrix, which will be addressed in future work.

2. After each merge operation, as recall increases, the precision decreases. The merge process entails calculating new means for baseline and anomalous data by averaging the workers' and coordinators' respective means, along with the threshold based on the percentile parameter described in Algorithm 3. As the merge oper-

ation progresses, the coordinator's calculated threshold becomes more conservative, considering the specified percentile and the minimum distance of a worker's anomalous data points from the baseline mean. Consequently, an increased number of data points are rejected, leading to an increase in false positives and a drop in precision. Addressing this issue will be a focus of future work as we revisit the implementation of the merge operation.

3. The performance metrics of the IDS decrease when the sampling rate is reduced over time. This phenomenon can be attributed to the selection of data points for the cluster sample by the cluster-based sampling technique, which favors points that decrease sampling error. If a chosen data point happens to be at the periphery of the benign distribution, the IDS may perceive it as a dubious data point due to its conservative nature, potentially leading to its rejection as an anomaly. However, this effect is effectively mitigated by the merging operation.

3.4/ CONCLUSION

This chapter presents and evaluates a lightweight IDS specifically tailored for IoT applications, exemplified by its implementation on a microcontroller. Through comprehensive experiments and simulations, the proposed IDS showcases both effectiveness and efficiency when integrated into a cross-layer federated learning framework, encompassing a cluster-based sampling process and an intrusion detection process. This study emphasizes the impact of merging operations on the performance of both the coordinator and workers, the inherent trade-offs between precision and recall when exchanging statistics between workers and the coordinator, and the factors influencing performance trends when adjusting the sampling rate. The conducted experiments and simulations consistently demonstrate the suitability of our approach for IoT applications, offering efficient intrusion detection capabilities within resource-constrained environments.

4

Cross-layer Federated Heterogeneous Ensemble Learning IDS

4.1/ Introduction

Distributed and federated IDSs play a crucial role in ensuring the security of IoT devices. However, there is a pressing need for more robust mechanisms to address the existing gaps in these systems [60]. One major challenge lies in extending these systems to lightweight IoT devices.

In Chapter 3, we introduced a novel cross-layer IDS that leverages a cluster-based sampling technique. This approach ensures comprehensive data representation while minimizing sampling error caused by data variance. By utilizing IDS on sampled data, we can effectively reduce memory usage and energy consumption. To further enhance this approach, we extended the proposed sampling algorithm to incorporate a federated IDS, employing a semi-supervised ML model [118]. However, this distributed approach encountered a significant increase in the false positive rate, posing a persistent challenge in anomaly detection, particularly in anomaly-based intrusion detection [119]. The model struggles to account for all normal traffic patterns, leading to misclassification of untrained or abnormal benign data as attacks.

Although a system with a high rate of false positives may successfully identify a large number of real threats, the high number of false alarms can lead to 'alarm fatigue'. This term refers to the phenomenon where, due to an overwhelming number of false alarms, system administrators begin to ignore or downplay alerts, potentially missing genuine threats amidst the noise. In the context of IoT, the problem is even more pronounced due to the large-scale and high-density nature of these networks. Frequent false positives can result in unnecessary consumption of computational resources, network bandwidth,

and human attention. Therefore, an effective IDS should aim for a balance, achieving a high recall rate while keeping false positives to a minimum. Our proposed baseline k-means approach in Chapter 3 is designed with a focus on recall, aiming to capture as many threats as possible. This design choice reflects the severity of the potential consequences of missing a true positive threat. However, this focus comes with the trade-off of a potentially higher false positive rate, a limitation that will be addressed in this chapter to enhance the overall performance of the IDS.

The literature provides several solutions to mitigate false positives in IDS, with ensemble approaches being particularly promising. Recent research suggests that employing ensemble learning techniques can be a promising strategy to mitigate false positives in anomaly detection systems [120, 121]. Ensemble techniques involve the combination of multiple models to achieve better performance than could be obtained from any of the constituent models alone. This can lead to improved prediction performance and robustness against overfitting. An ensemble IDS could utilize the strengths of various detection techniques, including our baseline k-means, to form a collective decision about the intrusion status of a data point. By integrating different models with varying strengths and weaknesses, an ensemble can potentially achieve a higher true positive rate and a lower false positive rate simultaneously. One example is a two-stage architecture proposed in [119], where unsupervised methods are used for attack detection, while supervised learning is employed for classification. However, it is important to note that unsupervised learning may not be suitable for all problem types, and supervised techniques may not be feasible without appropriately labeled data [122].

Considering our research's limited labeled dataset, we approach the problem as a semi-supervised novelty detection task. In this Chapter, we present a lightweight, semi-supervised, federated IDS for IoT devices. By leveraging ensemble learning techniques, we combine the strengths of various detection models, including our baseline k-means, to form a collective decision about the intrusion status of data points. This integration allows us to achieve a higher true positive rate and a lower false positive rate simultaneously, improving the precision of our IDS.

In conclusion, our design incorporates a sampling layer and employs a heterogeneous ensemble learning strategy. This approach effectively detects malicious packets while maintaining a low false positive rate over time, providing a more reliable and sustainable solution for IoT security.

4.2/ ANOMALY-BASED INTRUSION DETECTION TECHNIQUES

Anomaly-based techniques are broadly used in IDSs due to their effectiveness in detecting unknown attacks. Unlike signature-based systems, these techniques establish patterns of normal network behavior and utilize them to identify abnormal patterns. This section explores several such techniques illustrated in Figure 4.1, along with a discussion of their limitations.

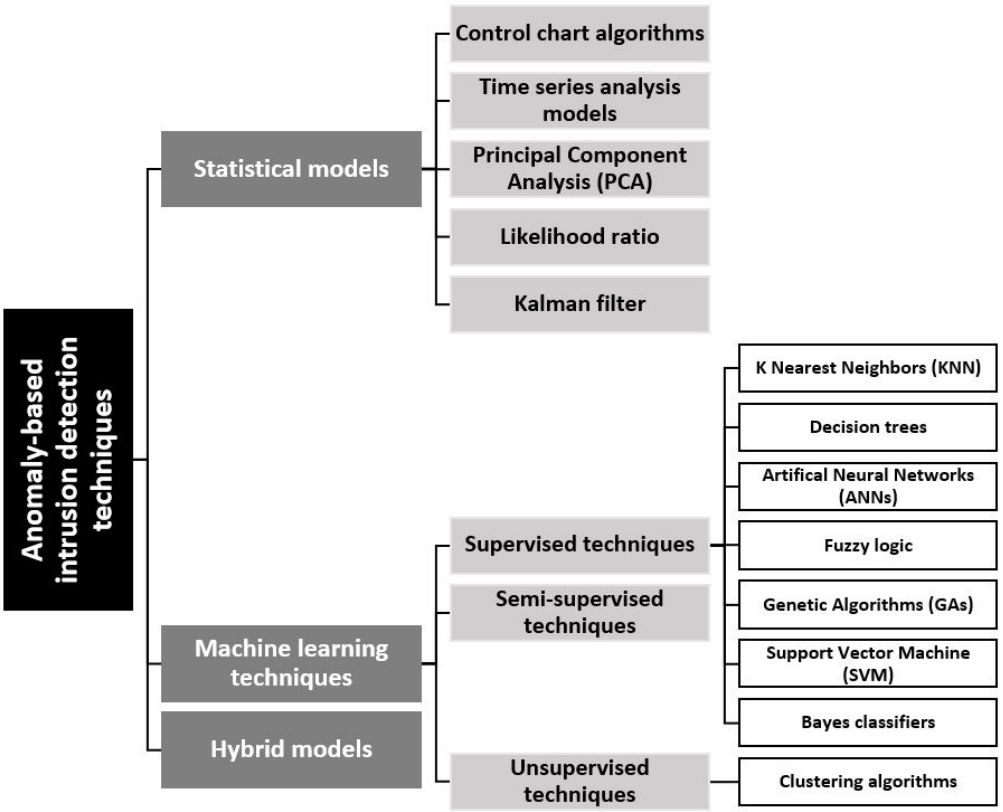


Figure 4.1: Taxonomy of anomaly-based intrusion detection techniques.

4.2.1/ STATISTICAL MODELS

Statistical models play a crucial role in anomaly-based intrusion detection techniques, assuming that anomalous data follows distinct statistical distributions from typical data. These models are built using historical traffic data, and similarity measures are employed to classify data points as normal or anomalous. Univariate detection techniques focus on analyzing individual parameters such as mean, standard deviation, and median. Control chart methods, widely used in Statistical Process Control (SPC), are prominent in this category [123, 124]. Intrusion detection models incorporating the EWMA and CUSUM control charts have shown promising results in detecting R2L intrusions in TCP packets

[124, 125, 126]. On the other hand, multivariate detection techniques allow for the examination of complex changes involving multiple variables. Time-series and forecasting models such as the ARIMA model, Principal Component Analysis, and likelihood ratio are commonly used in this category [127]. Soule et al. 61 proposed a method for the detection of traffic anomalies based on the Kalman Filter.

Statistical models offer several advantages, including tunability, adaptability to network conditions, and the ability to provide quantitative measures for anomaly detection without prior knowledge of network attacks or normal system behavior [128]. However, they also have limitations, such as the challenge of parameter tuning and the assumptions of data stationarity and linearity in real-world scenarios.

4.2.2/ MACHINE LEARNING TECHNIQUES

ML techniques play a pivotal role in the realm of IDSs, offering powerful methodologies for detecting network attacks. These techniques can be broadly classified into three main categories: supervised, semi-supervised, and unsupervised. In the following sections, we provide an overview of each category, delving into their fundamental concepts and highlighting notable research contributions that underscore their significance:

1. Supervised Techniques

   Supervised learning techniques rely on labeled data to classify network traffic as normal or attack. While popular due to their simplicity and accuracy, these techniques struggle to detect new and unknown attacks. Notable examples include:

   - K Nearest Neighbors (KNN): The KNN algorithm identifies the nearest neighbors in the training dataset to classify new data points. Variations of KNN, such as weighted KNN and feature selection with genetic algorithms, improve accuracy [129, 130].
   - Decision Trees: Decision trees are employed for network traffic classification. Combining decision tree algorithms with techniques like genetic algorithms enhances detection, especially for DDoS attacks [131, 132, 133].
   - Artificial Neural Networks (ANNs): ANNs model complex patterns in network data and can detect non-linear relationships. They consist of interconnected layers of neurons with weights defining their connections.Various training techniques, such as recurrent neural networks and feed-forward neural networks, show promise in intrusion detection [134, 135, 136].
   - Fuzzy Logic: Fuzzy logic techniques handle uncertainty in data. A fuzzy logic IDS performs on par with decision tree-based systems [137].

- Genetic Algorithms (GAs): GAs optimize solutions inspired by natural selection. GA-based systems have been effective in detecting DDoS attacks, weighted feature extraction, rule generation, and detecting unknown attacks [138, 139, 140, 141].
- Support Vector Machine (SVM): SVM finds optimal hyperplanes for classification. SVM models trained with malicious data show effectiveness in detecting specific attacks [142]. Combining SVM with genetic algorithms improves feature selection [143].
- Bayes Classifiers: Bayesian classifiers use Bayes' theorem for classification. Feature filtering followed by a Bayes classifier offers an IDS for intrusion detection [144]. Non-parametric IDS based on the Bayesian model detects both known and unknown network attacks [145].

2. Semi-supervised Techniques

Semi-supervised techniques require only normal data for training and identify anomalies based on deviations from normal behavior. Noteworthy contributions include:

- Semi-supervised Multi-Layered Clustering (SMLC): This model employs k-means clustering for network attack detection [146].
- Semi-supervised Tri-Adaboost (STA): STA combines three Adaboost algorithms and the chi-square technique for feature selection and reduced computational requirements [147].

These techniques demonstrate high detection accuracy while operating with limited labeled data, making them suitable for scenarios where obtaining labeled attack data is challenging.

3. Unsupervised Techniques

Unsupervised techniques automatically discover patterns and anomalies in data without relying on labeled examples. Remarkable illustrations include:

- K-means Clustering: This technique divides traffic data into clusters and detects outliers based on the distance from the cluster centroid [148].
- Hybrid Intrusion Detection System: Combining K-means and KNN algorithms enhances the detection of anomalous traffic [149].
- CANN: CANN utilizes distances to nearest neighbors and cluster centroids to identify anomalous traffic [150].
- EIDS-ACC-OD: This intrusion detection system combines K-means and KNN for traffic classification and attack detection [151].

Unsupervised techniques excel at fast anomaly detection and handle large datasets. However, challenges include defining stable normal behavior and differentiating intrusions from other anomalies in dynamic network environments. Researchers and practitioners can leverage these techniques to enhance the security of network systems by identifying and mitigating threats in real-time.

4.2.3/ HYBRID MODEL

HIDS have been proposed to address the limitations of Anomaly-based Intrusion Detection Systems (AIDSs) and Signature-based Intrusion Detection Systems (SIDSs). HIDS combines the strengths of both approaches, leveraging signature-based detection for known attacks while maintaining the ability to detect unknown or less-frequent attacks with a low false alarm rate. These hybrid models can be either sequence-based or parallel-based, applying different detection techniques in a specific order or simultaneously for accurate classification.

One significant contribution in this area is the hybrid IDS proposed by Elvis et al. [152]. Their approach first utilizes an anomaly detector to identify malicious instances and then employs a signature-based detector to classify suspicious instances into false alarms, intrusions, and unknown intrusions. Another notable contribution is the work of Zhang et al. [153], who designed a HIDS using the random forest algorithm for both known and unknown attack detection in the signature-based detector. Additionally, researchers in [154] developed a hybrid decision technology that combines data filtering and multiple classifiers to enhance the performance of the IDS. These contributions highlight the significance of hybrid models in improving detection accuracy and reducing false alarms in intrusion detection.

The above discussion is shown in Table 4.1.

4.2.4/ PERFORMANCE METRICS OF AIDS

The evaluation of AIDS is crucial for assessing their effectiveness and efficiency in detecting network attacks. This section focuses on key performance metrics used to measure the capabilities of AIDS. The performance evaluation of AIDS aims to achieve timely attack detection, accurate intrusion identification, and minimizing false positives.

A widely used tool in performance evaluation is the confusion matrix (Table 4.2), which provides a comprehensive overview of the classification algorithm's performance. Metrics such as precision, recall, specificity, accuracy, and false positive rate are derived from the confusion matrix, offering insights into the system's accuracy, ability to detect true attacks, capability to identify normal behavior, and efficiency in real-time scenarios. Furthermore,

Table 4.1: Comparison of anomaly-based intrusion detection techniques.

	Advantages	Drawbacks	Examples
K Nearest Neighbors	Do not require any data distribution Straight-forward technique It is a non-parametric algorithm that has no assumption regarding the data No need for a learning phase	High computing complexity making the algorithm slow Failure possibility Results are depending on the choice of the number of neighbors k Sensitive to the distance metric Cannot deal with missing values	Su [155] Adetunmbi [156] Malhotra [129] Serpen [130] Lin [150]
Clustering	Flexibility Low computational complexity Fast detection Extends the concept of anomaly from single objects to groups of objects	Require thresholds to calculate the distances Sensitive to the chosen number of clusters Anomalies may affect the initial formation of clusters	Munz [148] Aung [149] Lin [150] Peng [157] Kumar [158] Benaddi [159]
Neural Networks	Ability to detect complex and non-linear relationships between different features Ability to build the model from limited information	Require long training time High computational complexity Does not require any prior knowledge about the output	Vesely Cannady [160] Dao [161] Jadidi [162] Xu [134] Iqbal [135] Vesely [163]
Statistical models	Do not require a priori knowledge of network attacks Effective in detecting the attacks manifested by abrupt changes	Small and gradual attacks are hard to be detected	Umer [123] Sklavounos [126] Sklavounos [164] Fouladi [165] Nezhad [127]
Decision trees	Can process numerical and categorical data Capability to handle high-dimensional data Fast to train in small datasets	Require long time to train the model for large datasets	Bloedorn [131] Sinclair [132] Kevric [166]
Genetic algorithms	They do not assume any prior knowledge about system behavior	Require long time to train the model for large datasets	Chaudhary [138] Pawar [167] Middlemiss [139] Gong [140] Zhao [141] Hoque [168]
Support Vector Machine	Effective for multi-dimensional data Fast detection with high accuracy	Cannot extend the classification to more than two classes Requires long time to train the model for large datasets	Winter [142] Kuang [169] Tao [143] Al-Qatf [170]
Bayes classifiers	Ability to break complex problems into different smaller models	Slow in classifying data with multiple attributes	Kevric [166] Mukherjee [144] Zhang [171]
Fuzzy logic	Ability to deal with uncertain and/or incomplete data Does not need a large training dataset Interpretability, Simplicity No need to re-train the system when adding new rules	Can be used only if some knowledge about the output is available in the form of linguistic rules	Bridges [172] Danane [173] Shanmugavadivu [174] Mkuzangwe [137]

computational resource metrics such as CPU and memory usage are considered. These performance metrics provide valuable insights into the capabilities and limitations of AIDS, enabling researchers and network administrators to evaluate IDSs and make informed decisions regarding their deployment and optimization.

Table 4.2: Confusion matrix table.

Detected as / Actual	**Normal**	**Anomalous**
Normal	True Negative (TN)	False Positive (FP)
Anomalous	False Negative (FN)	True Positive (TP)

4.2.5/ PERFORMANCE EVALUATION

This section discusses the performance evaluation of various anomaly-based intrusion detection algorithms presented in this chapter. Table 14 presents a comparison of selected intrusion detection algorithms based on their detection accuracy. It considers the number of features used, the types of attacks detected, and the datasets employed for

testing. The primary evaluation metric used is detection accuracy.

From the results, two main conclusions can be drawn. Firstly, different algorithms show varying effectiveness in detecting specific types of attacks, with some algorithms performing better than others for certain attack types. Secondly, the number of selected features impacts the detection accuracy, with a lower number of features generally leading to higher accuracy. The evaluation also includes the accuracy of correctly classifying normal traffic data.

By examining the performance evaluation results, researchers can gain insights into the strengths and weaknesses of different intrusion detection algorithms, aiding in the selection and optimization of algorithms for specific attack scenarios.

4.3/ Ensemble learning

4.3.1/ Ensemble Learning and Data Streams

Traditional ML methods typically assign labels to data points by identifying the best single hypothesis that can explain them. In contrast, ensemble learning constructs a collection of combined hypotheses that are then voted on [175]. Each hypothesis within the voting set is generated using a ML method, and this collective approach often leads to superior accuracy compared to individual ML methods [176]. Ensemble learning, as emphasized by Dong et al. [177], is particularly well-suited for handling complex, imbalanced, high-dimensional, and noisy datasets.

ML algorithms often struggle with constrained sliding windows in data streams, resulting in sub-optimal models [178]. Polikar [179] defines ensemble learning as a means to reduce the likelihood of poor selection by reducing the likelihood of poor outcomes. Sun et al. [180] proposed a class-based ensemble method to detect emerging or disappearing data classes. Van Rijn et al. [181] proposed a heterogeneous ensemble learning framework that outperforms state-of-the-art techniques across diverse data streams. This framework relies on multiple classifiers, unlike most dynamic data stream ensembles, which rely on a single base-level classifier. Zhang and Jin [182] proposed an automatic ensemble learning algorithm that adaptively distinguishes sensible classifiers, outperforming static configurations. Individual ensemble learning techniques have been proposed to address specific data stream problems such as concept drift [183, 184, 180, 178], imbalance [185, 186], and noise [187, 178].

4.3.2/ Ensemble Learning in IoT IDS

Lama and Tim [188] conducted a systematic survey on the application of ensemble learning in IDSs, with a particular focus on the IoT domain. Their findings highlighted the widespread use of random forest within ensemble learning for IDS. They also provided a detailed breakdown, noting that homogeneous ensembles utilized random forest, bagging, and boosting techniques, while heterogeneous ensembles employed majority voting and stacking architectures. Illy et al.[189] proposed an ensemble learning mechanism for the fog-to-things environment, aiming to reduce latency and improve accuracy. Additionally, Verma and Ranga [190] introduced ELNIDS, an ensemble learning IDS specifically designed for "Low-Power and Lossy" IoT networks. ELNIDS utilized a four-model ensemble consisting of bagged trees, boosted trees, subspace discriminant, and RUSBoosted trees. Their work aligns with Abu Al-Haija and Al Badawi [191] and Mohy-Eddine et al. [192].

Alhowaide, Alsmadi, and Tang [193] employed an automatic Model Selection Method (MSM) to configure a heterogeneous set of classifiers, similar to Zhang's approach [182]. Abu Alghanam et al. [194] utilized ensemble learning to introduce LS-PIO, an enhanced feature selection algorithm for IoT IDS. K-Means was employed to reduce processing time for IoT, aligning with Gopalakrishnan and Purusothaman [195]. Abu Al-Haija and Al-Dala'ien [196] proposed ELBA-IoT, an ensemble learning model for botnet attack detection in IoT networks using AdaBoosted, RUSBoosted, and bagged trees. Hazman et al. [197] proposed IIDS-SIoEL, an intrusion detection framework for IoT-based smart environments based on Ensemble Learning using AdaBoost, Boruta, mutual information, and correlation.

The methods surveyed in this section are summarized in Table 4.3. For more comprehensive insights into ensemble learning in IoT IDS, refer to [198, 199, 200].

4.4/ Cross-Layer Federated Ensemble Learning

In this work, we propose a novel federated semi-supervised ensemble novelty detection technique for IDSs in IoT networks. Our approach addresses the challenge of limited labeled regular or benign traffic data in real-world applications. As we have previously detailed in Chapter 3, the proposed method referred to as baseline k-means [117, 201] leverages unsupervised IDS methods and transforms them into semi-supervised methods by learning a boundary that approximates the baseline observations' distribution. Remembering from the previous chapter, the baseline k-means was designed to maximize recall. However, this approach can lead to a higher rate of false positives, which is an issue that needs to be carefully managed to preserve the utility and credibility of an

Reference	Ensemble learning set
Illy et al. [189]	Random Forest, Bagging Classifier, AdaBoost and Voting
Verma and Ranga [190]	bagged trees, boosted trees, subspace discriminant and RUSBoosted trees
Abu Al-Haija and Al Badawi [191]	bagged trees, ensemble subspace kNN (ESK), RUSBoosted trees, shallow neural network (SNN), bilayered neural network (BNN and logistic regression kernel (LRK)
Mohy-Eddine et al. [192]	isolation forest (IF) and pearson's correlation coefficient (PCC)
Alhowaide, Alsmadi and Tang [193]	logistic regression, random forest, decision tree, gradient boosting, bagged tree, gaussian naive bayes, adaboosted, knn, bernoulli naive bayes, multi-layer perceptron, stochastic gradient descent and support vector machines
Abu Alghanam et al. [194]	support vector machines, isolation forest, local outlier factore and K-means
Gopalakrishnan and Purusothaman[195]	deep neural network, random forest, and AdaBoost
Abu Al-Haija and Al-Dala'ien [196]	AdaBoosted, RUSBoosted, and bagged trees
Hazman et al. [197]	AdaBoost, Boruta, mutual information and correlation

Table 4.3: List of IoT IDS using ensemble learning and the set of used ML algorithms

IDS. Let $X = \{x_1, x_2, ..., x_n\}$ be the initial baseline observations, where each $x_i \epsilon R_d$ represents a data point in the d-dimensional feature space. We apply the baseline k-means algorithm to obtain a centroid $C \epsilon R_d$ for the baseline data. The threshold τ is computed as follows:

$$\tau = \text{percentile}(D_M(x_1, C), \ldots, D_M(x_n, C), p) \tag{4.1}$$

For any new observation x', the baseline k-means classifies it as benign or anomalous based on the following decision rule:

$$\begin{cases} \text{benign}, & \text{if } D_M(x', C) \leq \tau \\ \text{anomalous}, & \text{if } D_M(x', C) > \tau \end{cases}$$

In this classification process, benign traffic is classified within a subspace based on Mahalanobis distances, while abnormal observations outside the boundary indicate an attack. The threshold is determined by the percentile of distances between data points.

4.4.1/ MOTIVATION

Our approach, based on KMeans, utilizes a distance metric to determine whether an observation originates from the baseline population or is an outlier. This method achieves a high true positive rate by minimizing the boundary to the baseline centroid, which is crucial in security contexts where attacks may be intolerable. However, it also exhibits a high false-positive rate. The main motivation for this method stems from the observation that during merging operations, worker recall increases over time while precision decreases due to IDS rejecting more packets or data points.

Consider the following equations that summarize the merge operation and threshold recalculation:

$$
\begin{aligned}
M_{\text{baseline}} &= \frac{1}{2}(M_{\text{coordinator,baseline}} + M_{\text{worker,baseline}}) \\
M_{\text{anomalous}} &= \frac{1}{2}(M_{\text{coordinator,anomalous}} + M_{\text{worker,anomalous}}) \\
S^{-1} &= \text{pinv}(S_{\text{coordinator,baseline}} + 0.001 \cdot I) \\
D(x_i, M_{\text{baseline}}) &= \sqrt{\sum_{i=1}^{n}(x_i - M_{\text{baseline}})^T S^{-1}(x_i - M_{\text{baseline}})} \\
D_{\text{min}} &= \min(D(x_i, M_{\text{worker,anomalous}})) \\
D' &= \{D(x_i, M_{\text{baseline}}) \mid \text{where } D(x_i) < D_{\text{min}}\} \\
\tau' &= \text{percentile}(D', p)
\end{aligned}
$$

Here, M_{baseline} and $M_{\text{anomalous}}$ represent the updated baseline and anomalous means, respectively, after the merge operation. S^{-1} is the pseudo-inverse of the sum of the baseline covariance matrix and a small regularization term. $D(x_i, M_{\text{coordinator,baseline}})$ denotes the Mahalanobis distance of point x_i from the coordinator's baseline mean, while D_{min} represents the minimum distance of the worker's anomalous data points to the baseline mean. D' is the set of distances that are less than D_{min}, and τ' is the updated threshold calculated based on the specified percentile of distances in D'.

During merging operations, the coordinator's threshold becomes more conservative, leading to a higher number of data points being rejected. This occurs because the threshold considers the percentile and minimum distance between worker anomalous data points and the baseline mean. To reduce false-positive rates, we propose an ensemble-based approach as shown in Figure 4.2 that utilizes cross-layer federated ensemble learning in lightweight IoT IDS.This figure illustrates a federated intrusion detection approach for IoT networks. It demonstrates how a baseline k-means coordinator is initialized with baseline data and shares statistics with worker nodes. Each worker employs a cluster-based sam-

pling algorithm to minimize data processing and labels data points with the coordinator's statistics. Additionally, local independent models assist Worker Kmeans IDS in the classification process. Workers periodically transmit their statistics to the coordinator, which, in turn, updates the global model and shares the updated statistics.

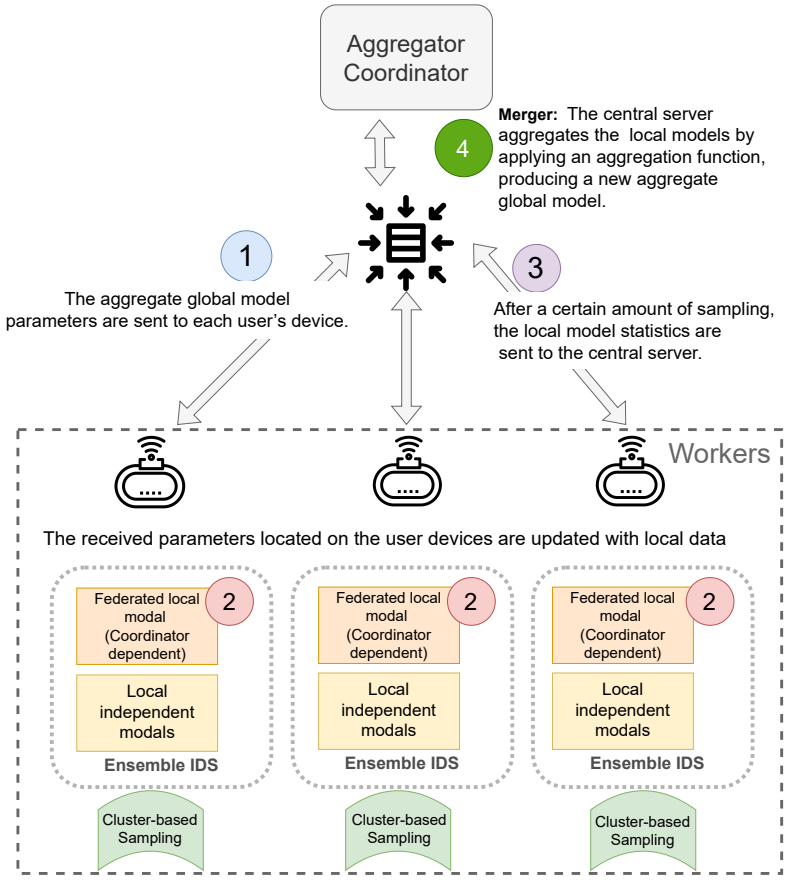


Figure 4.2: The Figure Illustrates the Proposed Cross-Layer Federated Ensemble Learning for Lightweight IoT IDS

4.4.2/ ENSEMBLE IDS

In this research, we introduced an ensemble learning technique to improve the precision of the baseline k-means model, specifically when identifying anomalous packets. The proposed approach incorporates two local semi-supervised novelty detection models, each unique to a worker, which are trained on a small batch of benign data. By leveraging the conservative nature of the baseline k-means model, which is already proficient at detecting benign packets. Let y_{bkmeans}, y_{pred1}, and y_{pred2} denote the predictions of the baseline k-means model, local model 1, and local model 2, respectively. Then, the ensemble learning output, denoted as y_{el}, is computed as shown in Algorithm 7.

Here, the function majority_vote returns 0 (benign) if both local models predict that the

Algorithm 7 Ensemble Learning IDS

```
function MAJORITY_VOTE(vote1, vote2, vote3)
    if (vote1 + vote2 + vote3) >= 2 then
        return 1
    else
        return 0
    end if
end function

for i ← 1 to n do
    if ybkmeans(i) = 0 then
        y_el(i) ← 0
    else
        y_el(i) ← MAJORITY_VOTE(ybkmeans(i), ypred1(i), ypred2(i))
    end if
end for
```

packet is benign, and 1 (anomalous) otherwise. The ensemble learning output considers a packet anomalous if at least one model predicts it to be so, while if both local models predict it as benign, the baseline k-means decision is overturned, considering the packet benign. The ensemble learning strategy improves the baseline k-means method's accuracy by incorporating voting-based and weight-based techniques. When classifying benign data points, the ensemble prioritizes the baseline k-means prediction, exhibiting weighted characteristics. The method employs a hybrid ensemble learning approach that combines voting-based and weight-based techniques to enhance the accuracy of the baseline k-means model. The final label is determined by the majority vote of the baseline k-means and two other local models (Algorithm 7). Ensemble learning effectively integrates voting-based and weight-based techniques for precise, robust classification of normal and abnormal data points. This hybrid approach is expected to enhance IDS performance by combining the benefits of both voting-based and weight-based ensemble learning techniques to improve the precision of the baseline k-means model.

4.5/ IMPLEMENTATION METHODOLOGY

The techniques described in this study were implemented in Python and made publicly available on GitHub [201]. The evaluation was conducted using the NSL-KDD dataset, an enhanced version of the original KDD Cup 1999 dataset [source: Tavallaee et al., "A Detailed Analysis of the KDD CUP 99 Data Set"].

In our experimental analysis, we assessed the performance of the baseline k-means model against several semi-supervised novelty detection techniques, including Local Out-

lier Factor, Gaussian Mixture Models, One-Class Support Vector Machines (One-Class SVM), Isolation Forests, Minimum Covariance Determinant, K Nearest Neighbor detector, Kernel Density Estimation (KDE), and Shallow Autoencoder. Our goal was to develop an ensemble learning approach where each worker is assigned two local models of semi-supervised novelty detection to assist the baseline k-means model in accurately identifying true attacks and reducing false positives. To this end, we tested various combinations of these methods, as presented in Table 4.4.

Table 4.4: Comparison of baseline k-means and various Ensemble combinations: Average performance metrics and standard deviations.

	Precision	Recall	F1-score	F2-score
baseline k-means	0.76 ± 0.02	0.96 ± 0.01	0.85 ± 0.01	0.92 ± 0.01
Bkmeans-AE-IF	0.85 ± 0.02	0.89 ± 0.01	0.87 ± 0.01	0.89 ± 0.01
Bkmeans-AE-KDE	**0.86 ± 0.02**	0.94 ± 0.01	**0.90 ± 0.01**	0.92 ± 0.01
Bkmeans-AE-KNN	0.85 ± 0.02	0.90 ± 0.01	0.87 ± 0.01	0.89 ± 0.01
Bkmeans-AE-SVM	**0.86 ± 0.02**	0.89 ± 0.01	0.88 ± 0.01	0.88 ± 0.01
Bkmeans-IF-KDE	0.83 ± 0.02	0.95 ± 0.01	0.89 ± 0.01	0.92 ± 0.01
Bkmeans-IF-KNN	0.83 ± 0.02	0.90 ± 0.01	0.87 ± 0.01	0.89 ± 0.01
Bkmeans-IF-LOF	0.82 ± 0.02	0.94 ± 0.01	0.88 ± 0.01	0.91 ± 0.01
Bkmeans-KNN-KDE	0.85 ± 0.02	**0.96 ± 0.01**	**0.90 ± 0.01**	**0.93 ± 0.01**
Bkmeans-LOF-SVM	0.85 ± 0.02	0.94 ± 0.01	0.89 ± 0.01	0.92 ± 0.01

Among the ensembles, the combination of baseline k-means, KNN detector, and KDE yielded the highest F1-score (0.90 ± 0.01) and F2-score (0.93 ± 0.01) compared to other ensembles. Additionally, the ensemble comprising baseline k-means, a shallow autoencoder, and KDE demonstrated notable effectiveness in detecting anomalous data points, achieving an F1-score of 0.90. These results indicate that combining baseline k-means with KNN and KDE, or incorporating a one-class SVM, can enhance the overall performance of the model in anomaly detection. However, Gaussian Mixture Models and Minimum Covariance Determinant were not considered in our analysis, as shown in Table 4.4, due to their high computational complexity, rendering them unsuitable for deployment on resource-constrained devices like microcontrollers or IoT devices. GMM requires substantial memory for covariance matrix calculations and storage, while MCD involves computationally intensive searches for covariance matrix computations based on subsets of data points.

4.5.1/ Semi-supervised novelty detection for intrusion detection

In this study, we conducted a comparative analysis of our proposed ensemble IDS method with several well-known novelty detection techniques. All methods utilized a subset of benign data to uncover normal traffic patterns. Specifically, each method was trained using

4,000 benign data points to capture the characteristics of normal traffic patterns. Subsequently, the novelty detection techniques were employed on sliding windows of 1,000 data points for categorization and classification purposes. Figure 4.3 clearly illustrates the significant performance advantage achieved by our ensemble learning approach, particularly when employing the combination of Federated baseline k-means, K-Nearest Neighbors, and Kernel Density Estimation. In comparison to the standalone application of Federated baseline k-means and other novelty detection techniques, the ensemble approach demonstrates superior performance in terms of accurate classification and detection of anomalies.

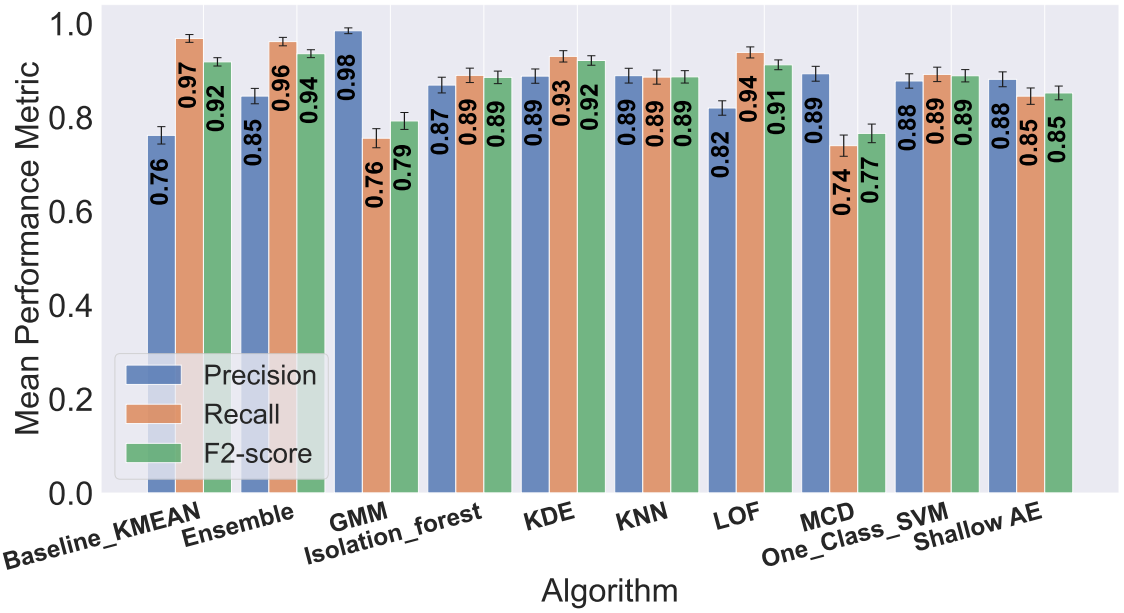


Figure 4.3: Performance Comparison of Semi-Supervised Novelty Detection Algorithms: Precision, Recall, and F-Score (beta = 2). The proposed Ensemble achieves the highest recall while maintaining competitive precision

This figure presents a comprehensive comparison of various semi-supervised novelty detection algorithms for intrusion detection, with a specific focus on precision, recall, and the $F\beta - score$ (where β = 2). It highlights the ensemble model's remarkable ability to prioritize true positives with the highest recall while maintaining competitive precision when compared to the standalone baseline k-means model.

Notably, the ensemble model exhibits a substantial improvement in precision, achieving a score of 0.85 compared to the baseline k-means model's precision score of 0.76. This improvement signifies a reduction in false positives and underscores the ensemble's superior capability to accurately identify genuine attacks. In practical applications, where false positive rates can incur unnecessary costs and resource wastage, this enhanced precision is of utmost importance.

Furthermore, the ensemble learning method maintains a high recall value of 0.96, which

is nearly equal to that of the baseline k-means model. This high recall score indicates the ensemble's effectiveness in correctly identifying positive cases, which is crucial in anomaly detection to ensure system security. By identifying as many actual attacks as possible, a high recall score contributes significantly to maintaining system integrity. Compared to other novelty detection techniques, the ensemble approach outperforms them in terms of precision, recall, and the F2 score. Gaussian Mixture Models and Minimum Covariance Determinant (CMD), while exhibiting higher precision, demonstrate lower recall scores, indicating a higher rate of false negatives. On the other hand, techniques such as Isolation Forest, KNN, Local Outlier Factor, One-Class SVM, and Shallow Autoencoder (AE) yield lower F2 scores. The F2 score, which assigns more weight to recall, is particularly crucial in intrusion detection as it minimizes potential harm and ensures system security.

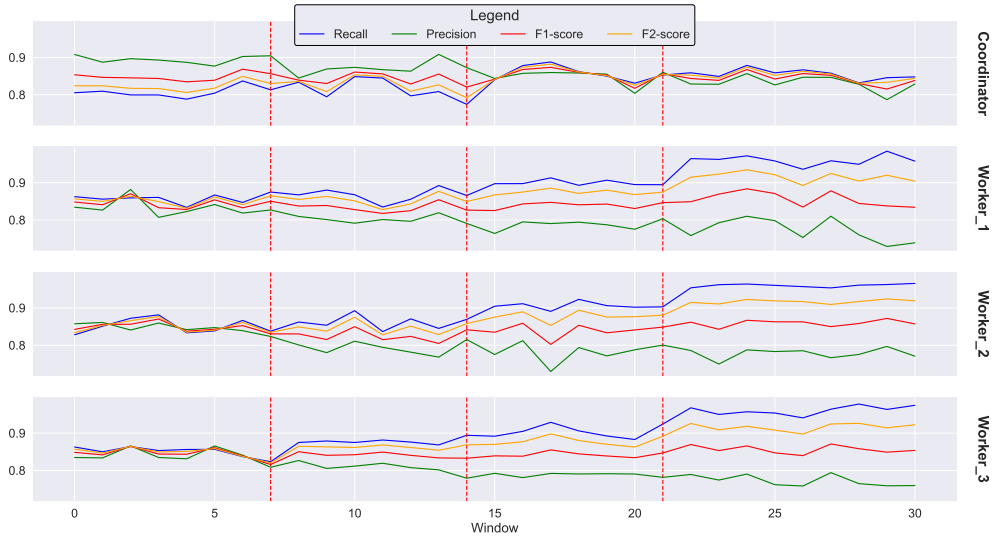
Overall, the results clearly demonstrate the ensemble model's superior performance, with higher precision, recall, and F2 scores compared to both the standalone baseline k-means model and other evaluated novelty detection techniques.

4.5.2/ FEDERATED ENSEMBLE IDS

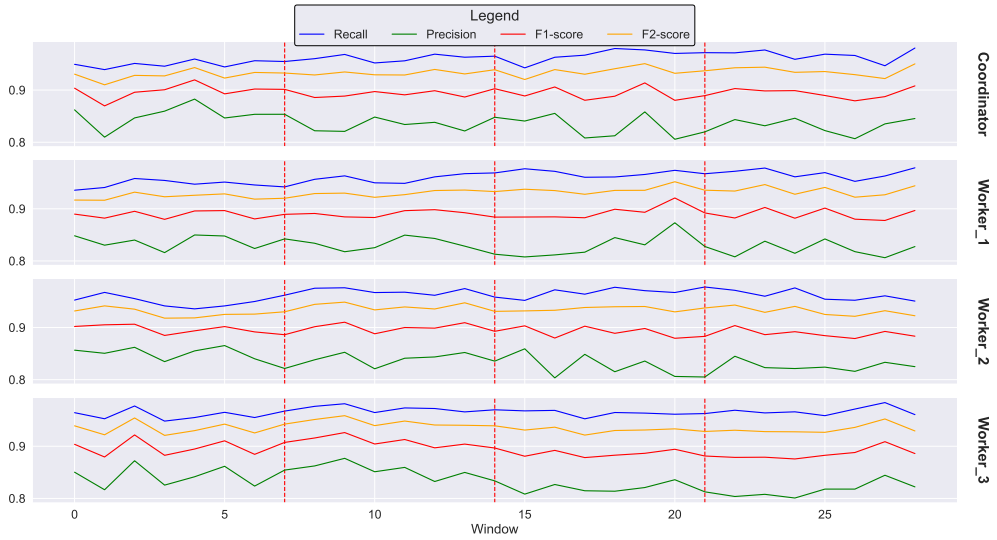
In this study, we present a simulated federated IDS composed of a coordinator and three workers. The coordinator initially learns a baseline model using 100 benign data points and shares its statistics with the workers. Our experimentation utilizes the NSL-KDD dataset, consisting of approximately 120,000 rows, and spans 30 epochs to process network traffic data. Each epoch involves the processing of 1,000 data points per entity, allowing us to evaluate the effectiveness of the baseline k-means algorithm in a distributed environment with dynamic network traffic data. Performance metrics are calculated and recorded in a Python dataframe after each epoch. Our simulation study investigates two scenarios: one employing the federated baseline k-means algorithm without integrating an ensemble, and another utilizing the proposed federated ensemble method. Both scenarios entail three merging operations.

The results from the first scenario (Figure 4.4a) demonstrate that the federated baseline k-means approach enhances performance in both the coordinator and workers through distributed data processing. Initial performance reveals high precision and recall, indicating accurate classification of true positives and actual positives from predicted positives. This accuracy is also reflected in the F1 and F2 scores. As the experiment progresses, performance generally improves for both the coordinator and workers. The F2 score, which places emphasis on recall, exhibits a significant increase in most iterations, indicating improved identification of true positives. However, fluctuations in performance are observed across precision and F1 score metrics after each merge operation. Preci-

sion shows a negative trend throughout the windows/epochs, suggesting an increasing false-positive rate with each merge. The baseline k-means algorithm prioritizes the true-positive rate over the false-positive rate, causing the workers to become more stringent in rejecting distances greater than the dynamically evolving threshold after each merging operation. The results highlight the potential of federated learning in intrusion detection, showcasing improved performance over time while mentioning the possibility of increasing false positives in the baseline k-means approach.



(a) Federated baseline k-means without local ensemble.

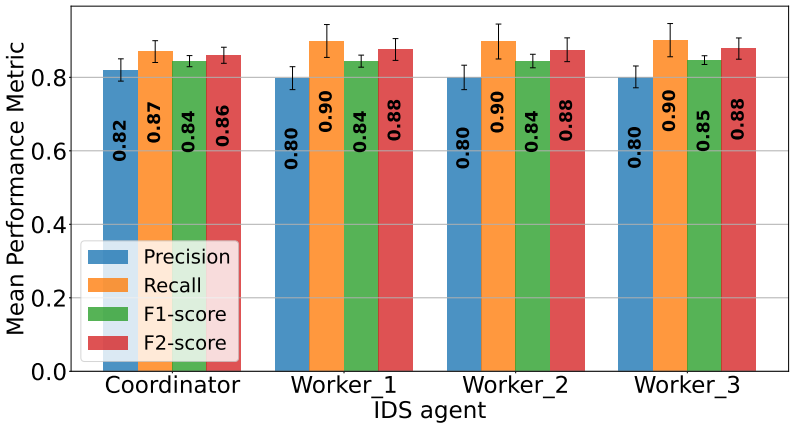


(b) Federated baseline k-means with local ensemble.

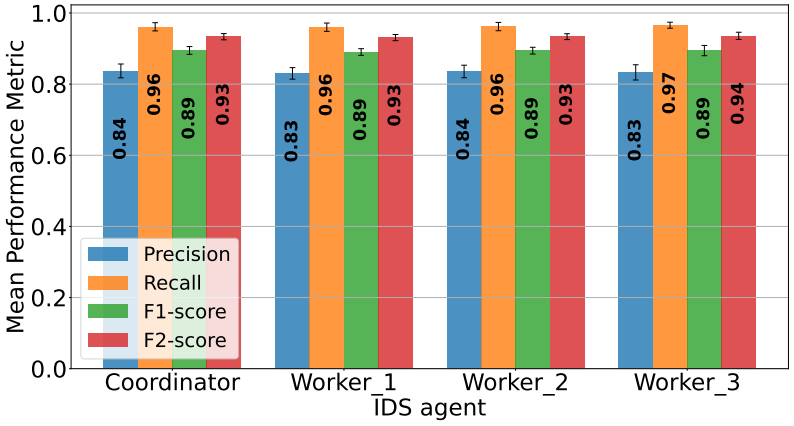
Figure 4.4: Evolution of performance metrics for the coordinator and three workers over 30 epochs, each comprising a window of 1,000 data points. The vertical dashed red lines denote points where merge operations were conducted

An ensemble approach combining federated baseline k-means, K-nearest neighbors, and kernel density estimation enhances precision for both workers and the coordinator, over-

coming limitations of standalone baseline k-means techniques. As illustrated in the second scenario (Figure 4.4b), this strategy significantly improves outcomes compared to non-ensemble strategies. Particularly noteworthy is that each worker achieves a precision score above 0.8, in contrast to the decline observed in non-ensemble approaches. The ensemble approach exhibits a stable trend, with temporal fluctuations and a noticeable increase in recall, particularly in the coordinator's results. The ensemble approach also demonstrates slightly improved F1 scores and consistently higher F2 scores, indicating its effectiveness compared to the non-ensemble method. Figure 4.5 illustrates the average performance metric differences between the ensemble and non-ensemble approaches.



(a) Average performance metrics for the federated baseline k-means without local ensemble after 30 epochs.



(b) Average performance metrics for the federated baseline k-means with local ensemble after 30 epochs.

Figure 4.5: Average performance metrics for the coordinator and three workers with three merging operations.

4.5.3/ Cross-layer Federated Learning

In this study, we explore cross-layer federated IDS utilizing our cluster-based sampling technique [117]. We consider a scenario involving a coordinator and a worker with two sampling rates: 0.60 and 0.20. The coordinator is trained on baseline data and shares its statistics with the worker, which then utilizes the same statistics for 10 epochs, processing 5,000 data points per epoch. Two merging operations are performed at epochs 2 and 6. In the context of IDSs, the sampling rate refers to the amount of data retained in memory for analysis. A high sampling rate, such as 0.60, indicates a larger data retention, while a lower rate, as in our case, 0.20, preserves a smaller portion. This sampling technique aids in detecting and preventing cyber threats like replay attacks, where attackers intercept, delay, or retransmit valid data to deceive the recipient system into unauthorized operations. The analysis of IDS performance metrics (Precision, Recall, and F1-score) at different sampling rates (Figure 4.6) reveals a slight decline in performance at the lower sampling rate (0.20) compared to the higher rate (0.60). This decline is expected as a reduced volume of data leads to less information available for accurate decision-making. However, the IDS retains its ability to identify attack packets with comparable efficiency at higher sampling rates. In contrast, the ensemble approach consistently outperforms the non-ensemble methods across all sampling rates and windows, leveraging the strength of combining multiple models for more robust and accurate results.The performance slightly degrades with the reduction in sampling rate, especially for the non-ensemble condition, further highlighting the robustness of the ensemble approach. For instance, at the first window with a sampling rate of 0.60, the precision using the ensemble approach is 0.869, higher than the precision of 0.803 achieved without ensemble. This trend of enhanced performance is consistently observed across all windows and both sampling rates.

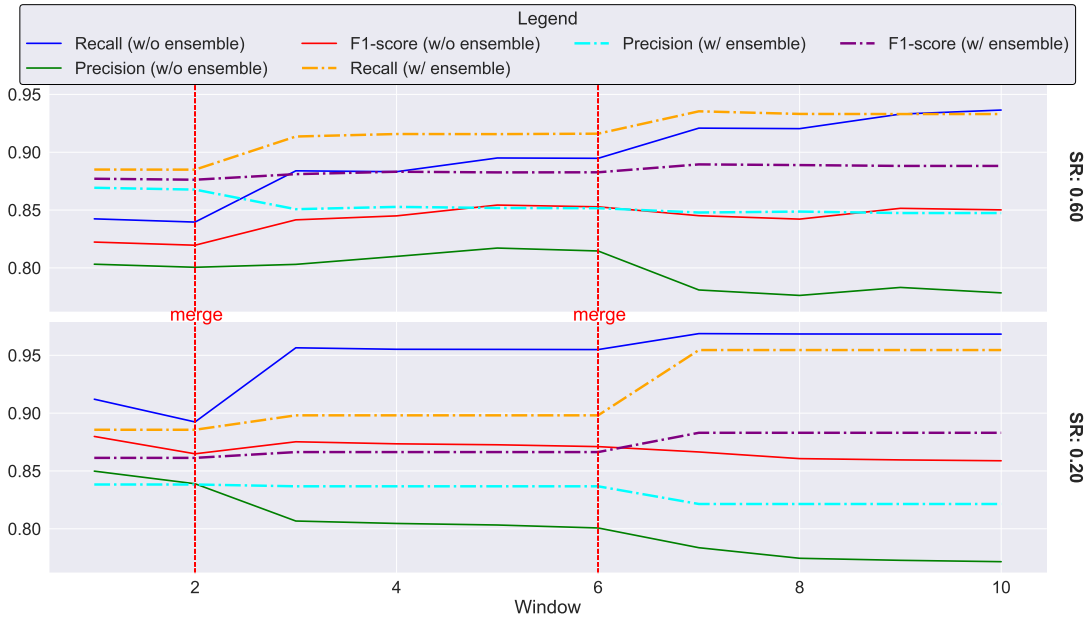


Figure 4.6: Comparative IDS Performance with and without Ensemble Learning at Different Sampling Rates across Ten Windows. Solid lines: no ensemble learning. Dashed lines: ensemble learning. Vertical dashed red lines: merge operation points

4.6/ CONCLUSION

This Chapter introduces a novel IDS that combines a heterogeneous federated ensemble approach, utilizing weighted and voting-based strategies to enhance anomaly detection. By incorporating local models into the decision-making process, the system achieves higher accuracy compared to standalone IDS models under different sampling rates (0.6 and 0.2). The integration of weighted and voting-based strategies enhances the system's ability to accurately identify anomalous packets. The federated design enables periodic merging and sharing of statistics and local models, thereby increasing the detection capacity of each worker without compromising false positives. Overall, this research demonstrates the effectiveness of heterogeneous federated ensembles in IDS, offering promising advancements for network security in IoT applications.

5

Conclusion and Perspectives

5.1/ Conclusion

Intrusion detection in networks is a highly researched field, given its critical role in today's interconnected world. This work explored the landscape of IDS systems in research, with a specific focus on lightweight IDS for IoT applications. The contributions and experiments conducted in this research have advanced the field of lightweight intrusion detection in IoT environments.

Contribution 1: Comprehensive Survey and Comparative Analysis of Existing Sampling Algorithms for Network Traffic Characterization

- The key to lightweight IDS is an efficient sampling algorithm that reduces the amount of processed data. To address this, we have developed a taxonomy for existing sampling algorithms for both stream and non-stream data. Additionally, we introduced a statistical metric (OS) that measures the degree of distortion introduced by each sampling algorithm.
- Experiments and Performance Analysis: We conducted experiments to analyze the performance of the sampling algorithms in terms of execution speed and their ability to produce highly representative samples at different window sizes and sampling ratios. The results revealed that while many of the surveyed algorithms perform well at high sampling ratios, they fall short at low sampling rates, which are crucial for a lightweight IDS solution in resource-constrained environments like IoT.

Contribution 2: Cluster-based Sampling Algorithm for Lightweight IoT Intrusion Detection System

- We introduce our Cluster-based sampling strategy, which leverages the prior distribution of data by pre-clustering similar data together. This approach ensures that the final sample captures the characteristics of all subgroups within the data.

The strategy involves feature reduction and the utilization of the K-means clustering algorithm to divide the data stream into clusters, effectively identifying patterns, particularly rare ones, and detecting changes in the data distribution.

- Experiments and Results: Our experiments demonstrate that the sampling accuracy is influenced by the sampling ratio and the change percentage. The proposed algorithm, utilizing 2 clusters and a 90% change ratio, yielded the least distortion for two sampling ratios/learning base sizes: 0.3 and 0.5. Our sampling method categorizes the sampled data into separate clusters, providing an intriguing feature that we exploit in our next contribution, the cross-layer IDS.

Contribution 3: Cross-layer Federated Learning for Lightweight IoT Intrusion Detection System

- The proposed cross-layer IDS utilizes a federated version of the baseline k-means algorithm for lightweight IoT IDS. It leverages a small amount of labeled data as the baseline for learning and identifies anomalies using the Mahalanobis distance, improving the performance of the classifier. This algorithm allows multiple IoT devices to participate in the training process while preserving data privacy. By utilizing a federated solution, the workload is divided among multiple workers, reducing the burden on individual devices and enhancing scalability. One of the key benefits of our approach is its privacy-preserving nature, where nodes only need to transmit cluster statistics to the central coordinator. This ensures that sensitive data remains on the devices and is not exposed during the training and detection process. By focusing on the transmission of cluster statistics rather than raw data, our method provides a higher level of privacy protection, which is crucial in IoT environments where data security is a top priority.

- Experimental Validation: The method is experimentally validated by testing its execution speed on an Arduino Nano. Experiments and simulations demonstrate the effectiveness and efficiency of the proposed IDS when implemented in a cross-layer federated learning framework, where a sampling operation precedes intrusion detection. The impact of cluster-based sampling on the IDS performance is assessed. The primary findings of this study highlight the influence of merging operations on the performance of the coordinator and workers, the trade-offs between precision and recall when sharing statistics between workers and the coordinator, and the factors affecting the performance trends when adjusting the sampling rate. Our method succeeds in detecting most attacks at the cost of a slightly higher false positive rate. Despite demonstrating a promising ability to detect malicious packets, this distributed approach encountered a notable increase in the false positive rate following each aggregation and parameter redistribution step by the centralized

node.

The high false positive rate represents a persistent challenge within the domain of anomaly detection, as the model faces inherent difficulties in covering all possible benign behaviors in the learning dataset. Consequently, benign data that deviates from the learned distribution or was not present during the training phase may be misclassified as an attack.

Contribution 4: Cross-layer Federated Heterogeneous Ensemble Learning for Lightweight IoT Intrusion Detection System

To counterbalance the conservative nature of our previous algorithm and improve precision, we proposed a heterogeneous federated ensemble approach.

- This research presents a novel IDs that employs a heterogeneous federated ensemble approach, combining weighted and voting-based strategies to enhance anomaly detection. We leverage the baseline k-means algorithm, supported by assisting local models, to achieve superior accuracy compared to standalone IDS under varying sampling rates (0.60 and 0.20). Integrating local models into the decision-making process represents a significant advancement, enhancing the system's capability to accurately identify anomalous packets. When the baseline k-means predicts a benign class, the system relies solely on this prediction, emphasizing its weighted aspect. However, when a potential anomaly is detected, a voting process among local models is initiated, thus implementing a voting-based strategy. This fusion of strategies enhances the system's reliability and precision in identifying cyber threats. Moreover, the federated design of our approach allows for the periodic merging and sharing of local model statistics, effectively increasing the detection capacity of each worker. Importantly, this does not lead to a higher false positive rate due to the ensemble's ability to validate and consolidate predictions.
- Experimental Validation: We experimented with a weight-based and vote-based ensemble learning approach where we pair our workers with different semi-supervised methods like KNN and KDE. Our experiments show that the ensemble approach succeeded in improving precision without affecting recall. The ensemble approach consistently outperforms the non-ensemble condition across all metrics and both sampling rates (0.2, 0.6). The performance slightly degrades with the reduction in sampling rate, especially for the non-ensemble condition, further highlighting the robustness of the ensemble approach.

Our research underlines the effectiveness of heterogeneous federated ensembles in IDS, offering promising advancements for network security in IoT applications. In conclusion, our work succeeded at producing a viable lightweight intrusion detection solution for IOT

environments capable of operating at very low memory, processing and bandwidth usage and still producing acceptable results.

5.2/ PERSPECTIVES

There are still plenty of areas we would like to cover in future research.

First, while the OS measure introduced in Chapter 2 effectively measures distortion introduced by a sampling algorithm, there are no metrics that show how well a sample represents rare packet subgroups, which are important in the context of IDS. This is the subject of future work to develop a metric that measures both properties of a sampling algorithm for better comparative analysis.

The limitation of the federated IDS proposed in Chapter 3 is the reliance on the coordinator's covariance matrix, which may not accurately represent the global dataset if there are significant disparities in the data point distribution across workers. To address this limitation, future research can explore a hybrid approach where workers send a compressed version of the covariance matrix, allowing for adjustment of the threshold value based on the local distribution of data points in each worker's dataset. This would improve the system's accuracy in detecting anomalies in scenarios with varying data point distributions.

Future work can focus on improving the system's reliability and precision in identifying cyber threats. This can involve developing advanced algorithms and techniques for weighting and voting strategies, considering the specific characteristics of different types of anomalies. Integrating multiple data sources, leveraging contextual information, and incorporating domain-specific knowledge can further enhance the precision and accuracy of the ensemble federated learning approach. This research direction aims to reduce false alarms and improve the overall effectiveness of IDSs in IoT applications.

PUBLICATIONS

JOURNAL PAPERS

- Suzan Hajj, Joseph Azar, Jacques Bou Abdo, Jacques Demerjian, Christophe Guyeux, Abdallah Makhoul and Dominique Ginhac, “Cross-layer Federated Learning for Lightweight IoT Intrusion Detection System”, Sensors, Volume 23, Issue 16, 2023. DOI: https://doi.org/10.3390/s23167038.
- Suzan Hajj, Rayane El Sibai, Jacques Bou Abdo, Jacques Demerjian, Christophe Guyeux, Abdallah Makhoul and Dominique Ginhac, “A critical review on the implementation of static data sampling techniques to detect network attacks”, IEEE Access, Volume 9, 2021. DOI: https://doi.org/10.1109/ACCESS.2021.3118605.
- Suzan Hajj, Rayane El Sibai, Jacques Bou Abdo, Jacques Demerjian, Abdallah Makhoul and Christophe Guyeux, “Anomaly based Intrusion Detection Systems: The requirements, methods, measurements and datasets”, Transactions on Emerging Telecommunications Technologies, Volume 32, Number 4, 2021. DOI: https://doi.org/10.1002/ett.4240.

CONFERENCE PAPERS

- Suzan Hajj, Joseph Azar, Jacques Bou Abdo, Jacques Demerjian, Abdallah Makhoul and Dominique Ginhac, “Cross-layer Federated Heterogeneous Ensemble Learning for Lightweight IoT Intrusion Detection System”, 10th IEEE International Conference on Data Science and Advance Analytics (DSAA’23), Thessaloniki, Greece, October 9 - 13, 2023 (Accepted, To Be Published).
- Suzan Hajj, Rayane El Sibai, Adam Barada, Jacques Bou Abdo, Jacques Demerjian, Christophe Guyeux, Abdallah Makhoul and Dominique Ginhac, “Cluster-based Sampling Algorithm for Lightweight IoT Intrusion Detection System”, 20th International Conference on Security and Management (SAM’22), Las Vegas, USA, July 25 - 28, 2022.

BIBLIOGRAPHY

[1] M. F. Elrawy, A. I. Awad, and H. F. A. Hamed, "Intrusion detection systems for iot-based smart environments: A survey," *J. Cloud Comput.*, vol. 7, pp. 1–20, 12 2018.

[2] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "A hierarchical hybrid intrusion detection approach in iot scenarios," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, pp. 1–7, 12 2020.

[3] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: Problems and solutions," *IEEE Netw.*, vol. 12, pp. 64–79, 11/12 1998.

[4] M. Colajanni and M. Marchetti, "A parallel architecture for stateful intrusion detection in high traffic networks," in *Proc. IEEE/IST Workshop Monitor., Attack Detection Mitigation (MonAM)*, (Tuebingen, Germany), pp. 1–7, 9 2006.

[5] M. Vallentin, R. Sommer, J.-M. Lee, C. Leres, V. Paxson, and B. Tierney, "The nids cluster: Scalable, stateful network intrusion detection on commodity hardware," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, pp. 107–126, 2007.

[6] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manag.*, vol. 16, pp. 445–458, 2 2019.

[7] P. D. Amer and L. N. Cassel, "Management of sampled real-time network measurements," in *Proc. 14th Conf. Local Comput. Netw.*, pp. 62–63, 1 1989.

[8] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *Proc. ACM SIGCOMM Workshop Large-Scale Attack Defense*, (Pisa, Italy), pp. 49–56, 9 2006.

[9] J. Mai, A. Sridharan, C.-N. Chuah, H. Zang, and T. Ye, "Impact of packet sampling on portscan detection," *IEEE J. Sel. Areas Commun.*, vol. 24, pp. 2285–2298, 12 2006.

[10] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?," in *Proc. 6th ACM SIGCOMM Internet Meas. (IMC)*, pp. 165–176, 2006.

[11] A. Pescape, D. Rossi, D. Tammaro, and S. Valenti, “On the impact of sampling on traffic monitoring and analysis,” in *Proc. 22nd Int. Teletraffic Congr. (ITC 22)*, pp. 1–8, 9 2010.

[12] H. Zhang, J. Liu, W. Zhou, and S. Zhang, “Sampling method in traffic logs analyzing,” in *Proc. 8th Int. Conf. Intell. Hum.-Mach. Syst. Cybern. (IHMSC)*, pp. 554–558, 8 2016.

[13] J. M. C. Silva, P. Carvalho, and S. R. Lima, “A modular sampling framework for flexible traffic analysis,” in *Proc. 23rd Int. Conf. Softw., Telecommun. Comput. Netw. (SoftCOM)*, pp. 200–204, 9 2015.

[14] K. Bartos, M. Rehak, and V. Krmicek, “Optimizing flow sampling for network anomaly detection,” in *Proc. 7th Int. Wireless Commun. Mobile Comput. Conf.*, pp. 1304–1309, 7 2011.

[15] R. E. Sibai, *Sampling, qualification and analysis of data streams*. PhD thesis, Dept. LISITE, Sorbonne Université, Université Libanaise, unknown.

[16] R. E. Sibai, Y. Chabchoub, J. Demerjian, R. Chiky, and K. Barbar, “A performance evaluation of data streams sampling algorithms over a sliding window,” in *Proc. IEEE Middle East North Afr. Commun. Conf. (MENACOMM)*, pp. 1–6, 4 2018.

[17] Q. Pan, H. Yong-Feng, and Z. Pei-Feng, “Reduction of traffic sampling impact on anomaly detection,” in *Proc. 7th Int. Conf. Comput. Sci. Educ. (ICCSE)*, pp. 438–443, 7 2012.

[18] R. Singh, H. Kumar, and R. Singla, “Analyzing statistical effect of sampling on network traffic dataset,” in *Proc. ICT Crit. Infrastruct. 48th Annu. Conv. Comput. Soc. India*, vol. 1, pp. 401–408, Springer, 2014.

[19] W. Cochran, *Sampling Techniques*. Hoboken, NJ, USA: Wiley, 1977.

[20] P. S. Efraimidis and P. G. Spirakis, “Weighted random sampling with a reservoir,” *Inf. Process. Lett.*, vol. 97, pp. 181–185, 3 2006.

[21] P. S. Efraimidis, “Weighted random sampling over data streams,” in *Algorithms, Probability, Networks, and Games*, pp. 183–195, Springer, 2015.

[22] A. I. McLeod and D. R. Bellhouse, “A convenient algorithm for drawing a simple random sample,” *J. Roy. Stat. Soc., Ser. C Appl. Statist.*, vol. 32, no. 2, pp. 182–184, 1983.

[23] J. S. Vitter, “Random sampling with a reservoir,” *ACM Trans. Math. Softw.*, vol. 11, pp. 37–57, 3 1985.

[24] P. B. Gibbons, Y. Matias, and V. Poosala, “Fast incremental maintenance of approximate histograms,” in *Proc. VLDB*, pp. 466–475, 1997.

[25] P. B. Gibbons, Y. Matias, and V. Poosala, “Fast incremental maintenance of approximate histograms,” *ACM Trans. Database Syst.*, vol. 27, pp. 261–298, 9 2002.

[26] B. Babcock, M. Datar, and R. Motwani, “Sampling from a moving window over streaming data,” in *Proc. 13th Annu. ACM-SIAM Symp. Discrete Algorithms*, pp. 633–634, 2002.

[27] R. El Sibai, Y. Chabchoub, J. Demerjian, Z. Kazi-Aoul, and K. Barbar, “A performance study of the chain sampling algorithm,” in *Proc. IEEE 7th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, pp. 487–494, 12 2015.

[28] R. E. Sibai, J. B. Abdo, and J. Demerjian, “A new priority sampling algorithm for the internet of things,” tech. rep., Tech. Rep., 2021.

[29] R. Gemulla, W. Lehner, and P. J. Haas, “A dip in the reservoir: Maintaining sample synopses of evolving datasets,” in *Proc. 32nd Int. Conf. Very Large Data Bases*, pp. 595–606, 2006.

[30] R. Gemulla, *Sampling algorithms for evolving datasets*. PhD thesis, Dept. Informatique, Technischen Universitat Dresden Fakultat Informatik, Dresden, Germany, 2008.

[31] B. Csernel, F. Clerot, and G. Hébrail, “Datastream clustering over tilted windows through sampling,” in *Proc. Workshop Knowl. Discovery Data Streams (ECML PKDD)*. To be published.

[32] A. Dogman, R. Saatchi, and S. Al-Khayatt, “An adaptive statistical sampling technique for computer network traffic,” in *Proc. 7th Int. Symp. Commun. Syst., Netw. Digit. Signal Process. (CSNDSP)*, pp. 479–483, 7 2010.

[33] A. Dogman and R. Saatchi, “Multimedia traffic quality of service management using statistical and artificial intelligence techniques,” *IET Circuits, Devices Syst.*, vol. 8, no. 5, pp. 367–377, 2014.

[34] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the kdd cup 99 data set,” in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, pp. 1–6, 7 2009.

[35] S.-I. Ao, M. Amouzegar, and B. B. Rieger, *Intelligent Automation and Systems Engineering*, vol. 103. Springer, 2011.

[36] S. Hajj, R. El Sibai, J. Bou Abdo, J. Demerjian, A. Makhoul, and C. Guyeux, "Anomaly based intrusion detection systems: The requirements, methods, measurements, and datasets," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 4, p. e4240, 2021.

[37] A. Barada, "Cluster-based-sampling." https://github.com/AdamBarada/Cluster-based-sampling, 2022.

[38] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, 2009.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, and M. Blondel, "Scikit-learn: Machine learning in python," *the Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[40] S. Hajj, R. El Sibai, J. Bou Abdo, J. Demerjian, C. Guyeux, A. Makhoul, and D. Ginhac, "A critical review on the implementation of static data sampling techniques to detect network attacks," *IEEE Access*, 2021.

[41] A. Huč, J. Šalej, and M. Trebar, "Analysis of machine learning algorithms for anomaly detection on edge devices," *Sensors*, vol. 21, no. 14, p. 4946, 2021.

[42] N. Tekin, A. Acar, A. Aris, A. S. Uluagac, and V. C. Gungor, "Energy consumption of on-device machine learning models for iot intrusion detection," *Internet of Things*, vol. 21, p. 100670, 2023.

[43] "Internet of things statistics for 2023 - taking things apart." https://dataprot.net/statistics/iot-statistics/. Accessed: 2023-03-21.

[44] T.-H. Lee, C.-H. Wen, L.-H. Chang, H.-S. Chiang, and M.-C. Hsieh, "A lightweight intrusion detection scheme based on energy consumption analysis in 6lowpan," in *Advanced technologies, embedded and multimedia for human-centric computing*, pp. 1205–1213, Springer, 2014.

[45] A. Le, J. Loo, K. K. Chai, and M. Aiash, "A specification-based ids for detecting attacks on rpl-based network topology," *Information*, vol. 7, no. 2, p. 25, 2016.

[46] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a lightweight intrusion detection system for the internet of things," *IEEE Access*, vol. 7, pp. 42450–42471, 2019.

[47] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Towards a lightweight detection system for cyber attacks in the iot environment using corresponding features," *Electronics*, vol. 9, no. 1, p. 144, 2020.

[48] A. Davahli, M. Shamsi, and G. Abaei, "Hybridizing genetic algorithm and grey wolf optimizer to advance an intelligent and lightweight intrusion detection system for iot wireless networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 11, pp. 5581–5609, 2020.

[49] B. S. Khater, A. W. Abdul Wahab, M. Y. I. Idris, M. A. Hussain, A. A. Ibrahim, M. A. Amin, and H. A. Shehadeh, "Classifier performance evaluation for lightweight ids using fog computing in iot security," *Electronics*, vol. 10, no. 14, p. 1633, 2021.

[50] H. Sedjelmaci, S. M. Senouci, and M. Al-Bahri, "A lightweight anomaly detection technique for low-resource iot devices: A game-theoretic methodology," in *2016 IEEE international conference on communications (ICC)*, pp. 1–6, IEEE, 2016.

[51] X.-H. Nguyen, X.-D. Nguyen, H.-H. Huynh, and K.-H. Le, "Realguard: A lightweight network intrusion detection system for iot gateways," *Sensors*, vol. 22, no. 2, p. 432, 2022.

[52] J. Pei, K. Zhong, M. A. Jan, and J. Li, "Personalized federated learning framework for network traffic anomaly detection," *Computer Networks*, vol. 209, p. 108906, 2022.

[53] D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh, "An ensemble multi-view federated learning intrusion detection for iot," *IEEE Access*, vol. 9, pp. 117734–117745, 2021.

[54] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-learning-based anomaly detection for iot security attacks," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2545–2554, 2021.

[55] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning attacks on federated learning-based iot intrusion detection system," in *Proc. Workshop Decentralized IoT Syst. Secur.(DISS)*, pp. 1–7, 2020.

[56] A. Tabassum, A. Erbad, W. Lebda, A. Mohamed, and M. Guizani, "Fedgan-ids: Privacy-preserving ids using gan and federated learning," *Computer Communications*, vol. 192, pp. 299–310, 2022.

[57] Y. Zhao, J. Chen, J. Zhang, D. Wu, J. Teng, and S. Yu, "Pdgan: A novel poisoning defense method in federated learning using generative adversarial network," in *Algorithms and Architectures for Parallel Processing: 19th International Conference, ICA3PP 2019, Melbourne, VIC, Australia, December 9–11, 2019, Proceedings, Part I 19*, pp. 595–609, Springer, 2020.

[58] H. Saadat, A. Aboumadi, A. Mohamed, A. Erbad, and M. Guizani, "Hierarchical federated learning for collaborative ids in iot applications," in *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–6, IEEE, 2021.

[59] M. Sarhan, W. W. Lo, S. Layeghy, and M. Portmann, "Hbfl: A hierarchical blockchain-based federated learning framework for collaborative iot intrusion detection," *Computers and Electrical Engineering*, vol. 103, p. 108379, 2022.

[60] E. M. Campos, P. F. Saura, A. González-Vidal, J. L. Hernández-Ramos, J. B. Bernabé, G. Baldini, and A. Skarmeta, "Evaluating federated learning for intrusion detection in internet of things: Review and challenges," *Computer Networks*, vol. 203, p. 108661, 2022.

[61] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of things intrusion detection: Centralized, on-device, or federated learning?," *IEEE Network*, vol. 34, no. 6, pp. 310–317, 2020.

[62] A. Belenguer, J. Navaridas, and J. A. Pascual, "A review of federated learning in intrusion detection systems for iot," *arXiv preprint arXiv:2204.12443*, 2022.

[63] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *Computer Communications*, 2022.

[64] B. Ghimire and D. B. Rawat, "Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things," *IEEE Internet of Things Journal*, 2022.

[65] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and M. Guizani, "A survey on iot intrusion detection: Federated learning, game theory, social psychology and explainable ai as future directions," *IEEE Internet of Things Journal*, 2022.

[66] N. Tuck, T. Sherwood, B. Calder, and G. Varghese, "Deterministic memory-efficient string matching algorithms for intrusion detection," in *IEEE INFOCOM 2004*, vol. 4, pp. 2628–2639, IEEE, 2004.

[67] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[68] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[69] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[70] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[71] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," *arXiv preprint arXiv:1701.02145*, 2017.

[72] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.

[73] G. Fernandes, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommunication Systems*, vol. 70, no. 3, pp. 447–489, 2019.

[74] "Psionic PortSentry Support FAQ." https://www.cisco.com/c/en/us/support/docs/security/ips-4200-series-sensors/62957-psionic-portsentry.html, 2002.

[75] B. Lhotsky, *Instant OSSEC host-based intrusion detection system*. Packt Publishing Ltd, 2013.

[76] G. H. Kim and E. H. Spafford, "The design and implementation of tripwire: A file system integrity checker," in *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pp. 18–29, ACM, 1994.

[77] "SecureNet." https://securenet.id/about/, 2017.

[78] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks.," in *Lisa*, vol. 99, pp. 229–238, 1999.

[79] V. Marinova-Boncheva, "A short survey of intrusion detection systems," *problems of Engineering Cybernetics and Robotics*, vol. 58, pp. 23–30, 2007.

[80] H. Kozushko, "Intrusion detection: Host-based and network-based intrusion detection systems," *Independent study*, 2003.

[81] B. Subba, S. Biswas, and S. Karmakar, "Host based intrusion detection system using frequency analysis of n-gram terms," in *TENCON 2017-2017 IEEE Region 10 Conference*, pp. 2006–2011, IEEE, 2017.

[82] G. Shipley, “Intrusion detection, take two,” *Network Computing*, vol. 10, pp. 44–48, Nov. 1999.

[83] T. F. Lunt, A. Tamaru, and F. Gillham, *A real-time intrusion-detection expert system (IDES)*. SRI International. Computer Science Laboratory, 1992.

[84] J. Hochberg, K. Jackson, C. Stallings, J. McClary, D. DuBois, and J. Ford, “Nadir: An automated system for detecting network intrusion and misuse,” *computers & security*, vol. 12, no. 3, pp. 235–248, 1993.

[85] B. Mantur, A. Desai, and K. Nagegowda, “Centralized control signature-based firewall and statistical-based network intrusion detection system (nids) in software defined networks (sdn),” in *Emerging Research in Computing, Information, Communication and Applications*, pp. 497–506, Springer, 2015.

[86] Z. Wang and Y. Zhu, “A centralized hids framework for private cloud,” in *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 115–120, IEEE, 2017.

[87] D. E. Denning, D. Edwards, R. Jagannathan, T. Lunt, and P. Neumann, “A prototype ides: A real-time intrusiondetection expert system,” *Computer Science Laboratory, SRI International*, 1987.

[88] F. Hidoussi, H. Toral-Cruz, D. E. Boubiche, K. Lakhtaria, A. Mihovska, and M. Voznak, “Centralized ids based on misuse detection for cluster-based wireless sensors networks,” *Wireless Personal Communications*, vol. 85, no. 1, pp. 207–224, 2015.

[89] N. V. Abhishek, T. J. Lim, B. Sikdar, and A. Tandon, “An intrusion detection system for detecting compromised gateways in clustered iot networks,” in *2018 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pp. 1–6, IEEE, 2018.

[90] K. Zkik, S. E. Hajji, and G. Orhanou, “A centralized secure plan for detecting and mitigation incidents in hybrid sdn,” in *MATEC Web of Conferences*, vol. 189, p. 10015, EDP Sciences, 2018.

[91] A. Abraham, R. Jain, J. Thomas, and S. Y. Han, “D-scids: Distributed soft computing intrusion detection system,” *Journal of Network and Computer Applications*, vol. 30, no. 1, pp. 81–98, 2007.

[92] L. A. Maglaras, “A novel distributed intrusion detection system for vehicular ad hoc networks,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 6, no. 4, pp. 101–106, 2015.

[93] G. Folino, F. S. Pisani, and P. Sabatino, "A distributed intrusion detection framework based on evolved specialized ensembles of classifiers," in *European Conference on the Applications of Evolutionary Computation*, pp. 315–331, Springer, 2016.

[94] A. Riyad, M. Ahmed, and R. Khan, "An adaptive distributed intrusion detection system architecture using multi agents.," *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 9, 2019.

[95] P. Kannadiga and M. Zulkernine, "Didma: A distributed intrusion detection system using mobile agents," in *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, pp. 238–245, IEEE, 2005.

[96] Y.-F. Zhang, Z.-Y. Xiong, and X.-Q. Wang, "Distributed intrusion detection based on clustering," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 4, pp. 2379–2383, Citeseer, 2005.

[97] M. Idhammad, K. Afdel, and M. Belouch, "Distributed intrusion detection system for cloud environments based on data mining techniques," *Procedia Computer Science*, vol. 127, pp. 35–41, 2018.

[98] Y. Li, M. Du, and J. Xu, "A new distributed intrusion detection method based on immune mobile agent," in *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*, pp. 215–219, IEEE, 2018.

[99] "RealSecure." https://www.itprotoday.com/compute-engines/realsecure-30, 1997.

[100] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita, "Network attacks: Taxonomy, tools and systems," *Journal of Network and Computer Applications*, vol. 40, pp. 307–324, 2014.

[101] H. Wu, S. Schwab, and R. L. Peckham, "Signature based network intrusion detection system and method," Sept. 9 2008. US Patent 7,424,744.

[102] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.

[103] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, IEEE, 2009.

[104] P. Hick, E. Aben, K. Claffy, and J. Polterock, "The caida ddos attack 2007 dataset," 2007.

[105] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.

[106] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Packet and flow based network intrusion dataset," in *International Conference on Contemporary Computing*, pp. 322–334, Springer, 2012.

[107] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville, and A. Pras, "Booters—an analysis of ddos-as-a-service attacks," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 243–251, IEEE, 2015.

[108] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proceedings of the 16th European Conference on Cyber Warfare and Security. ACPI*, pp. 361–369, 2017.

[109] A. Amouri, V. T. Alaparthy, and S. D. Morgera, "Cross layer-based intrusion detection based on network behavior for iot," in *2018 IEEE 19th Wireless and Microwave Technology Conference (WAMICON)*, pp. 1–4, IEEE, 2018.

[110] E. Canbalaban and S. Sen, "A cross-layer intrusion detection system for rpl-based internet of things," in *Ad-Hoc, Mobile, and Wireless Networks: 19th International Conference on Ad-Hoc Networks and Wireless, ADHOC-NOW 2020, Bari, Italy, October 19–21, 2020, Proceedings 19*, pp. 214–227, Springer, 2020.

[111] J. Long, W. Liang, K.-C. Li, Y. Wei, and M. D. Marino, "A regularized cross-layer ladder network for intrusion detection in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1747–1755, 2022.

[112] M. Malik, M. Dutta, J. Granjal, *et al.*, "Iot-sentry: A cross-layer-based intrusion detection system in standardized internet of things," *IEEE Sensors Journal*, vol. 21, no. 24, pp. 28066–28076, 2021.

[113] A. Kore and S. Patil, "Ic-mads: Iot enabled cross layer man-in-middle attack detection system for smart healthcare application," *Wireless Personal Communications*, vol. 113, pp. 727–746, 2020.

[114] S. Parween, S. Z. Hussain, M. A. Hussain, and A. Pradesh, "A survey on issues and possible solutions of cross-layer design in internet of things," *Int. J. Comput. Networks Appl*, vol. 8, no. 4, p. 311, 2021.

[115] "Baseline k-means open-source code." https://github.com/josephazar/baselineKmeans. Accessed: 2023-03-21.

[116] S. Revathi and A. Malathi, "A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 12, pp. 1848–1853, 2013.

[117] S. Hajj, R. El Sibai, A. Barada, J. Bou Abdo, J. Demerjian, C. Guyeux, A. Makhoul, and D. Ginhac, "Cluster-based sampling algorithm for lightweight iot intrusion detection system," in *2022 20th International Conference on Security and Management*, Springer, 2022.

[118] S. Hajj, J. Azar, J. Bou Abdo, J. Demerjian, C. Guyeux, A. Makhoul, and D. Ginhac, "Cross-layer federated learning for lightweight iot intrusion detection system," *IEEE Access*, 2023 (Under review).

[119] N. Kaja, A. Shaout, and D. Ma, "An intelligent intrusion detection system," *Applied Intelligence*, vol. 49, pp. 3235–3247, 2019.

[120] A. Teramoto, H. Fujita, O. Yamamuro, and T. Tamaki, "Automated detection of pulmonary nodules in pet/ct images: Ensemble false-positive reduction using a convolutional neural network technique," *Medical physics*, vol. 43, no. 6Part1, pp. 2821–2827, 2016.

[121] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A forest fire detection system based on ensemble learning," *Forests*, vol. 12, no. 2, p. 217, 2021.

[122] C. Chio and D. Freeman, *Machine learning and security: Protecting systems with data and algorithms*. " O'Reilly Media, Inc.", 2018.

[123] M. F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Computers & Security*, vol. 70, pp. 238–254, 2017.

[124] D. Sklavounos, A. Edoh, and M. Plytas, "A statistical approach based on ewma and cusum control charts for r2l intrusion detection," in *Proceedings of the Conference on Cybersecurity and Cyberforensics (CCC)*, pp. 25–30, IEEE, 2017.

[125] P. Čisar, S. Bošnjak, and S. M. Čisar, "Ewma based threshold algorithm for intrusion detection," *Computing and Informatics*, vol. 29, no. 6+, pp. 1089–1101, 2012.

[126] D. Sklavounos, A. Leondakianakos, and A. Edoh, "Statistical process control method for cyber intrusion detection (ddos, u2r, r2l, probe)," *International Journal of Cyber-Security and Digital Forensics*, vol. 8, no. 1, pp. 82–89, 2019.

[127] S. M. T. Nezhad, M. Nazari, and E. A. Gharavol, "A novel dos and ddos attacks detection algorithm using arima time series model and chaotic system in computer networks," *IEEE Communications Letters*, vol. 20, no. 4, pp. 700–703, 2016.

[128] R. El Sibai, Y. Chabchoub, R. Chiky, J. Demerjian, and K. Barbar, "An in-depth analysis of cusum algorithm for the detection of mean and variability deviation in time series," in *International Symposium on Web and Wireless Geographical Information Systems*, pp. 25–40, Springer, 2018.

[129] S. Malhotra, V. Bali, and K. Paliwal, "Genetic programming and k-nearest neighbour classifier based intrusion detection model," in *2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence*, pp. 42–46, IEEE, 2017.

[130] G. Serpen and E. Aghaei, "Host-based misuse intrusion detection using pca feature extraction and knn classification algorithms," *Intelligent Data Analysis*, vol. 22, no. 5, pp. 1101–1114, 2018.

[131] E. Bloedorn, A. D. Christiansen, W. Hill, C. Skorupka, L. M. Talbot, and J. Tivel, "Data mining for network intrusion detection: How to get started," tech. rep., Citeseer, 2001.

[132] C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," in *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*, pp. 371–377, IEEE, 1999.

[133] P. Preamthaisong, A. Auyporntrakool, P. Aimtongkham, T. Sriwuttisap, and C. So-In, "Enhanced ddos detection using hybrid genetic algorithm and decision tree for sdn," in *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 152–157, IEEE, 2019.

[134] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018.

[135] A. Iqbal and S. Aftab, "A feed-forward and pattern recognition ann model for network intrusion detection," *Int. J. Comput. Netw. Inf. Secur*, vol. 11, no. 4, pp. 19–25, 2019.

[136] W. Ma, "Analysis of anomaly detection method for internet of things based on deep learning," *Transactions on Emerging Telecommunications Technologies*, p. e3893.

[137] N. N. P. Mkuzangwe and F. V. Nelwamondo, "A fuzzy logic based network intrusion detection system for predicting the tcp syn flooding attack," in *Asian conference on intelligent information and database systems*, pp. 14–22, Springer, 2017.

[138] A. Chaudhary and G. Shrimal, "Intrusion detection system based on genetic algorithm for detection of distribution denial of service attacks in manets," *Available at SSRN 3351807*, 2019.

[139] M. J. Middlemiss and G. Dick, "Weighted feature extraction using a genetic algorithm for intrusion detection," in *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, vol. 3, pp. 1669–1675, IEEE, 2003.

[140] R. H. Gong, M. Zulkernine, and P. Abolmaesumi, "A software implementation of a genetic algorithm based approach to network intrusion detection," in *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, pp. 246–253, IEEE, 2005.

[141] J.-L. Zhao, J.-f. Zhao, and J.-J. Li, "Intrusion detection based on clustering genetic algorithm," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 6, pp. 3911–3914, IEEE, 2005.

[142] P. Winter, E. Hermann, and M. Zeilinger, "Inductive intrusion detection in flow-based network data using one-class support vector machines," in *2011 4th IFIP international conference on new technologies, mobility and security*, pp. 1–5, IEEE, 2011.

[143] P. Tao, Z. Sun, and Z. Sun, "An improved intrusion detection algorithm based on ga and svm," *IEEE Access*, vol. 6, pp. 13624–13631, 2018.

[144] S. Mukherjee and N. Sharma, "Intrusion detection using naive bayes classifier with feature reduction," *Procedia Technology*, vol. 4, pp. 119–128, 2012.

[145] W. Alhakami, A. ALharbi, S. Bourouis, R. Alroobaea, and N. Bouguila, "Network anomaly intrusion detection using a nonparametric bayesian approach and feature selection," *IEEE Access*, vol. 7, pp. 52181–52190, 2019.

[146] O. Y. Al-Jarrah, Y. Al-Hammdi, P. D. Yoo, S. Muhaidat, and M. Al-Qutayri, "Semi-supervised multi-layered clustering model for intrusion detection," *Digital Communications and Networks*, vol. 4, no. 4, pp. 277–286, 2018.

[147] Y. Yuan, L. Huo, Y. Yuan, and Z. Wang, "Semi-supervised tri-adaboost algorithm for network intrusion detection," *International Journal of Distributed Sensor Networks*, vol. 15, no. 6, p. 1550147719846052, 2019.

[148] G. Münz, S. Li, and G. Carle, "Traffic anomaly detection using k-means clustering," in *GI/ITG Workshop MMBnet*, pp. 13–14, 2007.

[149] Y. Y. Aung and M. M. Min, "Hybrid intrusion detection system using k-means and k-nearest neighbors algorithms," in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, pp. 34–38, IEEE, 2018.

[150] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "Cann: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-based systems*, vol. 78, pp. 13–21, 2015.

[151] S. Sandosh, V. Govindasamy, and G. Akila, "Enhanced intrusion detection system via agent clustering and classification based on outlier detection," *Peer-to-Peer Networking and Applications*, pp. 1–8, 2020.

[152] E. Tombini, H. Debar, L. Mé, and M. Ducassé, "A serial combination of anomaly and misuse idses applied to http traffic," in *20th Annual Computer Security Applications Conference*, pp. 428–437, IEEE, 2004.

[153] J. Zhang and M. Zulkernine, "A hybrid network intrusion detection technique using random forests," in *First International Conference on Availability, Reliability and Security (ARES'06)*, pp. 8–pp, IEEE, 2006.

[154] M. Panda, A. Abraham, and M. R. Patra, "A hybrid intelligent approach for network intrusion detection," *Procedia Engineering*, vol. 30, pp. 1–9, 2012.

[155] M.-Y. Su, "Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3492–3498, 2011.

[156] A. O. Adetunmbi, S. O. Falaki, O. S. Adewale, and B. K. Alese, "Network intrusion detection based on rough set and k-nearest neighbour," *International Journal of Computing and ICT Research*, vol. 2, no. 1, pp. 60–66, 2008.

[157] K. Peng, V. C. Leung, and Q. Huang, "Clustering approach based on mini batch kmeans for intrusion detection system over big data," *IEEE Access*, vol. 6, pp. 11897–11906, 2018.

[158] G. R. Kumar, N. Mangathayaru, and G. Narasimha, "An improved k-means clustering algorithm for intrusion detection using gaussian function," in *Proceedings of the The International Conference on Engineering & MIS 2015*, p. 69, ACM, 2015.

[159] H. Benaddi, K. Ibrahimi, and A. Benslimane, "Improving the intrusion detection system for nsl-kdd dataset based on pca-fuzzy clustering-knn," in *2018 6th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1–6, IEEE, 2018.

[160] J. Cannady, "Artificial neural networks for misuse detection," in *National information systems security conference*, vol. 26, Baltimore, 1998.

[161] V. Dao and V. R. Vemuri, "Computer network intrusion detection: A comparison of neural networks methods," *Differential Equations and Dynamical Systems*, vol. 10, no. 1, p. 2, 2002.

[162] Z. Jadidi, V. Muthukkumarasamy, and E. Sithirasenan, "Metaheuristic algorithms based flow anomaly detector," in *2013 19th Asia-Pacific Conference on Communications (APCC)*, pp. 717–722, IEEE, 2013.

[163] A. Veselỳ and D. Brechlerova, "Neural networks in intrusion detection systems," *Agriculturejournals. cz*, pp. 156–165, 2009.

[164] D. Sklavounos, G. Paraskevopoulos, and A. Edoh, "A cumulative sum technique for network cyber intrusion detection," in *The Third International Conference on Information Security and Digital Forensics (ISDF2017)*, p. 7, 2017.

[165] R. Fouladi, C. Kayatas, and E. Anarim, "Statistical measures: Promising features for time series based ddos attack detection," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, p. 96, 2018.

[166] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Computing and Applications*, vol. 28, no. 1, pp. 1051–1058, 2017.

[167] S. N. Pawar and R. S. Bichkar, "Genetic algorithm with variable length chromosomes for network intrusion detection," *International Journal of Automation and Computing*, vol. 12, no. 3, pp. 337–342, 2015.

[168] M. S. Hoque, M. Mukit, M. Bikas, A. Naser, *et al.*, "An implementation of intrusion detection system using genetic algorithm," *arXiv preprint arXiv:1204.1336*, 2012.

[169] F.-J. Kuang and S.-Y. Zhang, "A novel network intrusion detection based on support vector machine and tent chaos artificial bee colony algorithm," *J. Netw. Intell*, vol. 2, no. 2, pp. 195–204, 2017.

[170] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[171] B. Zhang, Z. Liu, Y. Jia, J. Ren, and X. Zhao, "Network intrusion detection method based on pca and bayes algorithm," *Security and Communication Networks*, vol. 2018, 2018.

[172] S. M. Bridges, R. B. Vaughn, *et al.*, "Fuzzy data mining and genetic algorithms applied to intrusion detection," in *Proceedings of 12th Annual Canadian Information Technology Security Symposium*, pp. 109–122, 2000.

[173] Y. Danane and T. Parvat, "Intrusion detection system using fuzzy genetic algorithm," in *2015 International Conference on Pervasive Computing (ICPC)*, pp. 1–5, IEEE, 2015.

[174] R. Shanmugavadivu and N. Nagarajan, "Network intrusion detection system using fuzzy logic," *Indian Journal of Computer Science and Engineering (IJCSE)*, vol. 2, no. 1, pp. 101–111, 2011.

[175] T. G. Dietterich *et al.*, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, no. 1, pp. 110–125, 2002.

[176] Y. Freund, R. E. Schapire, *et al.*, "Experiments with a new boosting algorithm," in *icml*, vol. 96, pp. 148–156, Citeseer, 1996.

[177] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020.

[178] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[179] R. Polikar, "Ensemble learning," *Ensemble machine learning: Methods and applications*, pp. 1–34, 2012.

[180] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao, "Online ensemble learning of data streams with gradually evolved classes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1532–1545, 2016.

[181] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "The online performance estimation framework: heterogeneous ensemble learning for data streams," *Machine Learning*, vol. 107, pp. 149–176, 2018.

[182] Y. Zhang and X. Jin, "An automatic construction and organization strategy for ensemble learning on data streams," *ACM SIGMOD Record*, vol. 35, no. 3, pp. 28–33, 2006.

[183] Z. Ahmadi and H. Beigy, "Semi-supervised ensemble learning of data streams in the presence of concept drift," in *Hybrid Artificial Intelligent Systems: 7th International Conference, HAIS 2012, Salamanca, Spain, March 28-30th, 2012. Proceedings, Part II 7*, pp. 526–537, Springer, 2012.

[184] A. Abbasi, A. R. Javed, C. Chakraborty, J. Nebhen, W. Zehra, and Z. Jalil, "Elstream: An ensemble learning approach for concept drift detection in dynamic social big data stream learning," *IEEE Access*, vol. 9, pp. 66408–66419, 2021.

[185] H. Du, Y. Zhang, K. Gang, L. Zhang, and Y.-C. Chen, "Online ensemble learning algorithm for imbalanced data stream," *Applied Soft Computing*, vol. 107, p. 107378, 2021.

[186] H. Li, Y. Wang, H. Wang, and B. Zhou, "Multi-window based ensemble learning for classification of imbalanced streaming data," *World Wide Web*, vol. 20, pp. 1507–1525, 2017.

[187] P. Zhang, X. Zhu, Y. Shi, L. Guo, and X. Wu, "Robust ensemble learning for mining noisy data streams," *Decision Support Systems*, vol. 50, no. 2, pp. 469–479, 2011.

[188] B. A. Tama and S. Lim, "Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation," *Computer Science Review*, vol. 39, p. 100357, 2021.

[189] P. Illy, G. Kaddoum, C. M. Moreira, K. Kaur, and S. Garg, "Securing fog-to-things environment using intrusion detection system based on ensemble learning," in *2019 IEEE wireless communications and networking conference (WCNC)*, pp. 1–7, IEEE, 2019.

[190] A. Verma and V. Ranga, "Elnids: Ensemble learning based network intrusion detection system for rpl based internet of things," in *2019 4th International conference on Internet of Things: Smart innovation and usages (IoT-SIU)*, pp. 1–6, IEEE, 2019.

[191] Q. Abu Al-Haija and A. Al-Badawi, "Attack-aware iot network traffic routing leveraging ensemble learning," *Sensors*, vol. 22, no. 1, p. 241, 2021.

[192] M. Mohy-Eddine, A. Guezzaz, S. Benkirane, M. Azrour, and Y. Farhaoui, "An ensemble learning based intrusion detection model for industrial iot security," *Big Data Mining and Analytics*, vol. 6, no. 3, pp. 273–287, 2023.

[193] A. Alhowaide, I. Alsmadi, and J. Tang, "Ensemble detection model for iot ids," *Internet of Things*, vol. 16, p. 100435, 2021.

[194] O. Abu Alghanam, W. Almobaideen, M. Saadeh, and O. Adwan, "An improved pio feature selection algorithm for iot network intrusion detection system based on ensemble learning," *Expert Systems with Applications*, vol. 213, p. 118745, 2023.

[195] B. Gopalakrishnan and P. Purusothaman, "A new design of intrusion detection in iot sector using optimal feature selection and high ranking-based ensemble learning model," *Peer-to-Peer Networking and Applications*, vol. 15, no. 5, pp. 2199–2226, 2022.

[196] Q. Abu Al-Haija and M. Al-Dala'ien, "Elba-iot: an ensemble learning model for botnet attack detection in iot networks," *Journal of Sensor and Actuator Networks*, vol. 11, no. 1, p. 18, 2022.

[197] C. Hazman, A. Guezzaz, S. Benkirane, and M. Azrour, "lids-sioel: intrusion detection framework for iot-based smart environments security using ensemble learning," *Cluster Computing*, pp. 1–15, 2022.

[198] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of ids for iot: recent updates, security issues, and challenges," *Archives of Computational Methods in Engineering*, vol. 28, pp. 3211–3243, 2021.

[199] K. A. Da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. de Albuquerque, "Internet of things: A survey on machine learning-based intrusion detection approaches," *Computer Networks*, vol. 151, pp. 147–157, 2019.

[200] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. A. Khan, "Performance analysis of machine learning algorithms in intrusion detection system: A review," *Procedia Computer Science*, vol. 171, pp. 1251–1260, 2020.

[201] "Baseline k-means github project."

List of Figures

List of Tables

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — https://github.com/gallandarakhneorg/tex-templates
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/

SPIM
école doctorale **sciences pour l'ingénieur et microtechniques**

Title: Collaborative Anomaly-based Intrusion Detection Systems in Lightweight IoT

Keywords: Machine Learning, Internet of Things, Lightweight Data Sampling, Federated Learning, Lightweight Intrusion Detection, Ensemble Learning

Abstract:

Lightweight Internet-of-Things (IoT) devices are susceptible to network attacks due to their operation in hostile environments and limited resources, which pose challenges for hosting sophisticated defenses. This thesis focuses on developing an Intrusion Detection System (IDS) that is specifically designed for lightweight IoT's constrained resources. Initially, a cluster-based sampling algorithm was leveraged to optimize performance for extremely low sampling ratios. This algorithm was then utilized to construct a cross-layer solution, enabling the IDS model to be trained using the clustered data. The model parameters are shared with a centralized aggregator, which leverages them to train a consolidated model, and subsequently shares the results with the individual IDS nodes. This federated cross-layer solution exhibits high efficiency, performance, and privacy. Furthermore, ensemble learning was employed to extend the federated cross-layer solution, leveraging both supervised and unsupervised IDS models collocated within the IoT devices. The benefits of this research are multifold. Firstly, it contributes to the body of knowledge in sampling by proposing a cluster-based sampling algorithm that can be applied beyond IDS to any field requiring very low sampling ratios. Secondly, the clustered data at the sampling layer provide processed grouping, eliminating the need for additional clustering in the IDS model. Lastly, ensemble learning enables the deployment of heterogeneous lightweight IoT nodes, each equipped with their proprietary IDS model, while collaboratively learning using a common federated IDS model.

Titre : Systèmes Collaboratifs de Détection d'Intrusion basés sur les Anomalies dans l'Internet des Objets

Mots-clés : Apprentissage automatique, Internet des Objets, Échantillonnage de Données Léger, Apprentissage Fédéré, Détection d'Intrusion Léger, Apprentissage ensembliste

Résumé :

Les dispositifs Internet des Objets (IoT) légers sont vulnérables aux attaques réseau en raison de leur fonctionnement dans des environnements hostiles et de leurs ressources limitées, posant ainsi des défis pour héberger des défenses sophistiquées. Cette thèse se concentre sur le développement d'un Système de Détection d'Intrusion (IDS) spécialement conçu pour les ressources limitées de l'IoT léger. Initialement, un algorithme d'échantillonnage basé sur des clusters a été utilisé pour optimiser les performances pour des ratios d'échantillonnage extrêmement bas. Cet algorithme a ensuite été utilisé pour construire une solution inter-couches, permettant au modèle IDS d'être formé à l'aide des données regroupées. Les paramètres du modèle sont partagés avec un agrégateur centralisé, qui les utilise pour former un modèle consolidé, puis partage les résultats avec les nœuds IDS individuels. Cette solution inter-couches fédérée présente une haute efficacité, performance et confidentialité. De plus, l'apprentissage ensembliste a été employé pour étendre la solution inter-couches fédérée, en utilisant à la fois des modèles IDS supervisés et non supervisés situés au sein des dispositifs IoT. Les avantages de cette recherche sont multiples. Premièrement, elle contribue au corpus de connaissances en échantillonnage en proposant un algorithme d'échantillonnage basé sur des clusters qui peut être appliqué au-delà de l'IDS à tout domaine nécessitant des ratios d'échantillonnage très bas. Deuxièmement, les données groupées à la couche d'échantillonnage fournissent un regroupement traité, éliminant le besoin d'un regroupement supplémentaire dans le modèle IDS. Enfin, l'apprentissage ensembliste permet le déploiement de nœuds IoT légers hétérogènes, chacun équipé de son propre modèle IDS, tout en apprenant de manière collaborative à l'aide d'un modèle IDS fédéré commun.

UBFC UNIVERSITÉ BOURGOGNE-FRANCHE-COMTÉ
Université Bourgogne-Franche-Comté
32, avenue de l'Observatoire
25000 Besançon, France