Distributed Intelligence for Enhancing Security and Privacy of Decentralised and Distributed Systems (Di4SPDS)

Topic: Chist-era 2022 — Security and Privacy in Decentralised and Distributed Systems (SPiDDS)

Deliverable D.3.2 Multi-Agent and Self-Aware Collaborative Intrusion Detection System

Work Package	Multi-agent and self-aware collaborative intrusion detection system
Delivery Date	M20
Responsible Partner	NU/LUT
Authors	Kandaraj Piamrat (NU) Houssem Jmal (NU) Md Raihan Uddin (LUT)
Contributors	Prabhat Kumar (LUT) Gauri Shankar (LUT)
Distribution	PU – Public, *fully open (automatically posted online)* SEN – Sensitive, *limited under the conditions of the Grant Agreement*
Version	0.1

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	4.4.2025	Table of content	Kandraj pimrat
1.1	2.5.2025	Executive Summery and Section 1	Houssem Jmal
1.2	13.5.2025	Section 2	Houssem Jmal
1.3	11.9.2025	Section 3	Houssem Jmal
1.4	5.12.2025	First Review	Kandraj pimrat Md Raihan Uddin
1.5	22.1.2026	Correction on First Review	Houssem Jmal
1.6	22.3.2026	Final Review	Kandraj pimrat
2.0	31.3.2026	Final Version	Prabhat Kumar, Md Raihan Uddin, Gauri Shankar

List of Abbreviations and Acronyms

Abbreviation / Acronym	Meaning
IDS	Intrusion Detection System
SAFL	Semi-Asynchronous Federated Learning
IoT	Internet of Things
GDPR	General Data Protection Regulation
IID	Independently and Identically Distributed
non-IID	Not Independently and Identically Distributed
FL	Federated Learning
ML	Machine Learning
TUNE-FL	Adaptive semi-synchronous / semi-decentralized federated learning framework
FLAIR	Federated Learning with Adaptive and Intelligent Reasoning for client selection
RL	Reinforcement Learning
MDP	Markov Decision Process
DT	Decision Transformer

IoMT	Internet of Medical Things
MLP	Multi-Layer Perceptron
MSCIDS	Multi-Agent and Self-Aware Collaborative Intrusion Detection System
P2P	Peer-to-Peer
RAM	Random Access Memory
UNSW-NB15	UNSW-NB15 intrusion detection dataset
CIC-IDS2017	CIC-IDS2017 intrusion detection dataset
IoTID20	IoT anomaly-detection dataset for IoT networks
IIoTID20	Dataset label appearing in the results discussion

Executive Summary

This deliverable presents the development of a multi-agent, self-aware collaborative intrusion detection technique designed to monitor network traffic for potential security events. It builds upon the inputs from Deliverable 3.1 and incorporates a semi-asynchronous federated learning approach. The proposed algorithm first determines the optimal number of workers to potentially reduce training time during communication rounds involving non-IID data. Secondly, it aims to enhance overall accuracy by achieving a high intrusion detection rate while maintaining a low false alarm rate. Based on the collected intelligence, a self-aware collaborative intrusion detection system (IDS) has been developed to detect previously unseen attacks in distributed and decentralized smart infrastructures.

This deliverable also provides methods for the coordinate response capability of the self-aware system in case of intrusion that could affect the whole environment among different infrastructures. It will consider autonomous conceptual primaries who are responsible to protect specific parts of the system, as well as supervisor agents who coordinate with each other for collaborative response. The coordination among the agents will follow the Multi-agent Systems principles, where agents are autonomous and capable of performing specific actions in dynamic operational decentralised and distributed environments.

Table of Contents

1. Introduction

With the rapid development of decentralized and smart infrastructures, ensuring robust cybersecurity has become a pressing concern. Traditional intrusion detection systems (IDS) often fall short when confronted with the dynamic, distributed nature of modern networks. To address these challenges, this deliverable presents the development of a multi-agent, self-aware collaborative intrusion detection technique, tailored for monitoring network traffic and identifying potential security threats in distributed systems while maintaining data privacy. Building upon the foundations laid in Deliverable 3.1, this work integrates a semi-asynchronous federated learning (SAFL) framework to support scalable, privacy-preserving model training across distributed nodes with non-IID data distributions. One important contribution is the dynamic determination of the optimal number of participating workers during training, aiming to reduce communication overhead and improve convergence speed without compromising accuracy. This deliverable also advances the capability of intrusion detection by introducing self-awareness and collaborative behavior within the system. Based on the principles of multi-agent systems, we aim to improve the efficiency of the IDS.

The proliferation of Internet of Things (IoT) devices, edge computing nodes, and cloud-based services has fundamentally transformed network architectures. These distributed environments generate massive volumes of heterogeneous traffic data across geographically dispersed locations, making centralized monitoring approaches increasingly impractical. Moreover, regulatory frameworks such as GDPR and emerging data sovereignty requirements impose strict constraints on data sharing and centralization, necessitating privacy-preserving approaches to security monitoring. Conventional IDS architecture typically rely on centralized data aggregation and analysis, which creates several critical vulnerabilities: single points of failure, scalability bottlenecks, and privacy concerns related to sensitive data transmission. Furthermore, the assumption of independently and identically distributed (IID) data—common in traditional machine learning approaches—rarely holds in real-world distributed networks, where each node may observe fundamentally different traffic patterns based on its role, location, and operational context.

Beyond detection, the system includes mechanisms for coordinated response, ensuring that detected intrusions trigger appropriate mitigation actions across the entire infrastructure. This is managed through two agent roles. The first one is that the agent is responsible for protecting specific segments of the infrastructure. They take localized actions such as traffic filtering, access control, or alert generation. The second works as a supervisor, which will facilitate information exchange across domains and may initiate system-wide actions like quarantining affected nodes or triggering policy updates.

This deliverable addresses these limitations through several key innovations. The semi-asynchronous federated learning framework enables participating nodes to train local detection models on their respective data without requiring raw data exchange. Instead, only model parameters or gradients are shared, preserving data locality and privacy. The semi-asynchronous nature allows nodes to contribute updates at different rates, accommodating heterogeneous computational capabilities and network conditions while maintaining overall system convergence.

A particularly novel aspect of this work is the adaptive worker selection mechanism. Rather than requiring all available nodes to participate in every training round, the system dynamically determines the optimal subset of workers based on factors such as data quality, model staleness, communication costs, and current detection performance. This intelligent selection reduces bandwidth consumption and accelerates convergence by prioritizing contributions from nodes that can provide the most valuable updates.

The multi-agent architecture introduces autonomous, self-aware agents at each monitoring node. These agents possess introspection capabilities, allowing them to assess their own detection confidence, data quality, and contribution potential. Through collaborative protocols, agents can share threat intelligence, coordinate responses to detected anomalies, and collectively adapt their detection strategies based on emerging attack patterns. This decentralized coordination enables the system to respond rapidly to localized threats while maintaining awareness of system-wide security posture.

2. Semi-asynchronous Federated learning

Federated Learning (FL) is a distributed Machine Learning (ML) paradigm that enables multiple clients, i.e data owners, such as mobile devices or enterprise servers, to collaboratively train a shared global model without exchanging their raw data. Instead of pushing the data to a central server, each client performs local training on its own data and shares only their models updates (gradients or weights) with a coordinator (an aggregator) [1], and this process is repeated to a certain number of FL rounds, or a certain criteria is met, such as reach a target accuracy. This approach enhances data privacy, reduces communication overhead and enables learning distributed and heterogeneous environments [2].

A key aspect of FL is its orchestration mechanism, which governs how training is coordinated across clients and aggregators. One important implication of the chosen orchestration strategy is that it can influence the convergence speed of the global model [3]. Besides, it governs how many local training epochs each client performs in each FL round. Some mechanisms may assign the same number of local epochs to all clients, aiming for simplicity and synchronization, such the first FL work FedAvg. However, this could be impractical with environments with heterogenous computing power capability and with non-IID data. Others may allow flexibility,

letting each client perform a different number of local epochs depending on factors such as computational resources, network conditions, energy constraints.

This variation introduced 4 types of orchestration mechanisms in literature as shown in Figure 1. First, synchronous FL refers to a setup where all selected clients perform local training and wait for each other to complete before aggregation. However, this could lead to delays due to the "straggler effect" (slow clients). In contrast, asynchronous FL allows clients to train and aggregate their models independently, without waiting for others. The server updates the global model as soon as it receives a new update, improving responsiveness but potentially introducing stale or inconsistent updates. Recently, researchers focused more on developing and introducing more efficient ways to FL orchestration, introducing two new categories, which are Semi-asynchronous and Semi-synchronous. The former corresponds to a hybrid mechanism between synchronous and asynchronous by allowing clients to stay asynchronous and triggering a synchronization point after the occurrence of a pre-defined event, such as very high staleness level of a client, thus controlling staleness while reducing idle time. The latter, semi-synchronous orchestration, requests clients' updates after a certain period, however it does not necessarily wait for all clients to finish their training. These strategies reflect different trade-offs between efficiency, consistency, and system robustness.

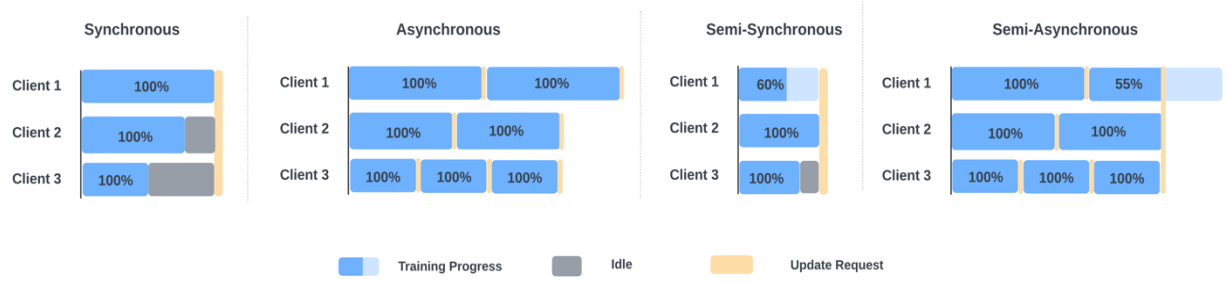
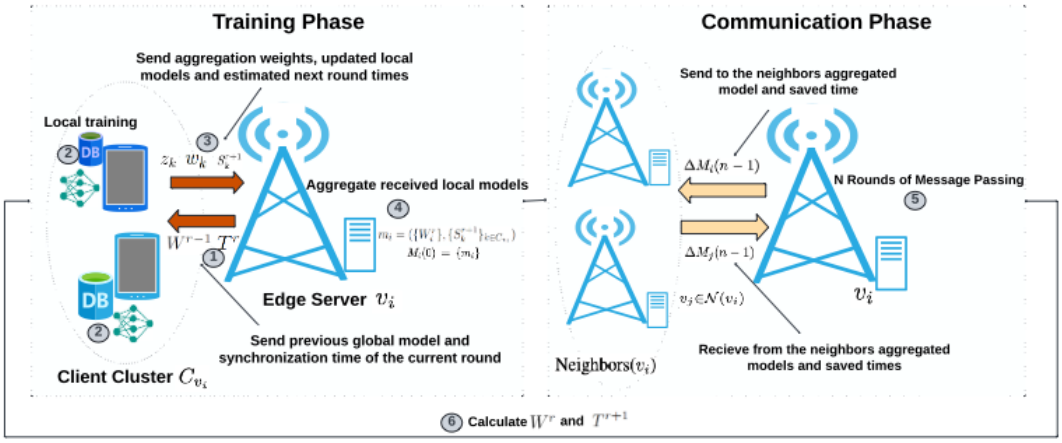


Figure 1: FL orchestration methods

2.1 TUNE-FL

This section describes the work that have been conducted to handle different challenges present in semi-decentralized FL setups such as network topology, data and system heterogeneity, and convergence speed [4]. TUNE-FL combines the benefits of both synchronous and asynchronous learning by allowing clients to train locally until either they converge or reach a calculated synchronization deadline. This deadline is dynamically estimated by each client based on its past performance, enabling the system to adjust to variations in computing power and training speed. Unlike traditional approaches, TUNE-FL does not require all clients to wait or finish training simultaneously, which significantly reduces idle time and communication overhead. Besides, edge servers coordinate using a lightweight message-passing strategy that ensures global model

consistency across the network, regardless of its topology. The system also incorporates a weighted aggregation mechanism that accounts for incomplete or delayed client contributions, ensuring robustness. Our method is composed of two main phases in each FL round as shown the figure below. The training phase corresponds to global model dissemination, local training and convergence time estimation, and sub-global model aggregation and saving elements in each edge servers mailbox. On the other hand, edge servers achieve full consensus with minimal cost, to generate one global model across the whole network and be able to estimate the next round convergence time based on the estimated times from all the clients.



T^r: Synchronization time of round r W^r: Global model of round r w_k: client k's model

S_k^{r+1}: Estimated Convergence time for round $r+1$ z_k: Participation Weight

$$\|f_k^j - f_k^{min}\| \leq \epsilon,\ \epsilon > 0 \qquad T^{r+1} = IQM(S_1^{r+1}, \cdots, S_{|C|}^{r+1}) \qquad S_k^{r+1} = \begin{cases} \beta t_k^r + (1-\beta)S_k^r & \text{if } t_k^r \leq T^r \\ (1-\beta)T^r + \beta S_k^r & \text{otherwise.} \end{cases}$$

$$z_k = \begin{cases} |\mathcal{D}_k|, & \text{if } S_k^r \leq T^r \\ \frac{T^r}{S_k^r} \cdot |\mathcal{D}_k| & \text{otherwise} \end{cases} \qquad W_{v_i}^r = \sum_{k \in C_{v_i}} \frac{z_k}{\sum_{j \in C_{v_i}} z_j} w_k$$

Figure 2: TUNE-FL Workflow

TUNE-FL was tested using two benchmark intrusion detection datasets (UNSW-NB15 [5] and CIC-IDS2017 [6]) and compared to three established FL methods. The results show that our method provides an the Accuracy is consistently higher, with improved F1-scores indicating better balance between detection and false alarms, **Training time** is reduced by up to **97×**, highlighting major efficiency gains, the approach handles non-IID data and heterogeneous client behaviour more effectively than the baselines and the approach could be applied to any network topology while maintaining the same performance.

3. Optimal worker selection for efficient FL training: FLAIR

In this section, we introduce the necessary background for our client selection solution. Our proposition will be based on our previous work TUNE-FL enhanced with offline RL by using Decision transformer (DT) for client selection. Our solution is called Federated Learning with Adaptive and Intelligent Reasoning for client selection (FLAIR) [7].

3.1 Background

We consider the standard Reinforcement Learning (RL) setting where an agent interacts with an environment modeled as a Markov Decision Process (MDP), defined by the tuple (S, A, P, r). Here, S is the state space, A the action space, $\mathrm{P}\left(\mathrm{S}_{\{\mathrm{t}+1\}}\middle|\mathrm{S}_{\mathrm{t}},\mathrm{a}_{\mathrm{t}}\right)$ the transition probability, r(S_t, a_t) the reward function. The goal in RL is to learn a policy that maximizes the expected return $\mathrm{E}\left[\Sigma_{\{\mathrm{t}=1\}}^{\{\mathrm{T}\}\mathrm{r}}\right]$. Alternatively, in offline RL, the agent learns from a fixed dataset of trajectories collected using arbitrary policies, without any interaction with the environment. Following the same principles, the DT has been introduced as an offline RL method that leverages the Transformer architecture, with minimal changes. Instead of learning a value function or policy directly, DT predicts actions autoregressively by conditioning on the desired return-to-go, past states, and actions. Each trajectory is represented as a sequence of $\left(\widehat{\mathrm{R}}_{\mathrm{t}},\mathrm{S}_{\mathrm{t}},\mathrm{a}_{\mathrm{t}}\right)$ triplets, and the Transformer is trained offline to predict the next action a_t given the previous return, state, and action history. Also, the reward is incorporated through the return-to-go $\hat{\mathrm{R}}\_\mathrm{t} = \Sigma\_\{\mathrm{t}' = \mathrm{t}\}\hat{}\{\mathrm{T}\}\ \mathrm{r}\_\{\mathrm{t}'\}$, defined as the cumulative future rewards from a given time step t.

3.2 Proposed Solution

Methodology Overview:

In this work, we adopt a similar setting to TUNE-FL and extend it to a more realistic scenario by incorporating client communication capabilities as illustrated in Figure 3. Clients are characterized not only by their dynamic computational capabilities but also by a time-varying wireless transmission channel to the edge server. While we assume the downlink to be reliable and stable, the uplink can be significantly perturbed due to co-channel interference caused by other devices in the surrounding environment. Therefore, our objective is to adaptively select a subset of clients that ensures the convergence of the global model while maintaining acceptable upload times.

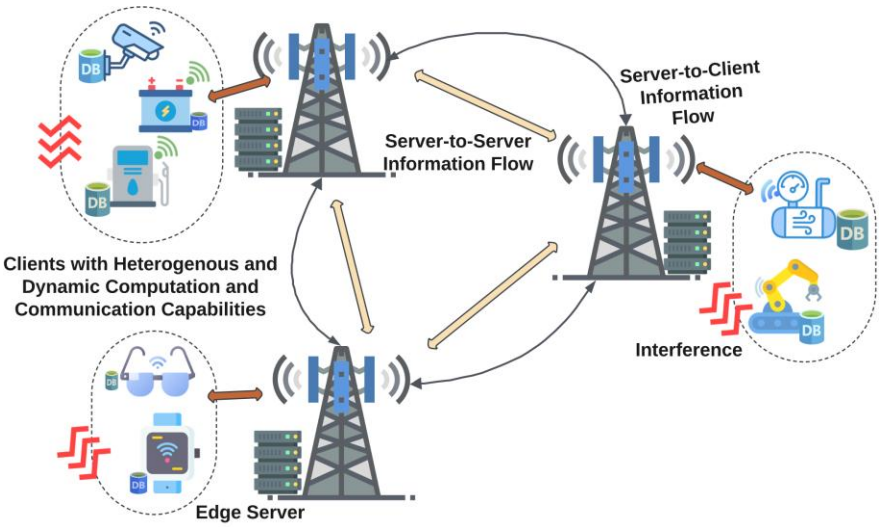


Figure 3: FL Environment

Since TUNE-FL effectively addresses the issue of computational heterogeneity, we narrow the client selection criteria to focus solely on statistical and communication efficiencies. To this end, we introduce an agent responsible for making client selection decisions. This agent operates with access to a global view of all clients independently of the number of servers, while also performing the selections based on both current and historical states and actions. In this context, we represent a sequence of FL rounds as a $\tau = \left(\hat{\mathrm{R}}1, \mathrm{S}1, \mathrm{a}1, \hat{\mathrm{R}}2, \mathrm{S}2, \mathrm{a}2, \ldots, \hat{\mathrm{R}}\mathrm{T}, \mathrm{ST}, \mathrm{aT}\right)$, where $\mathrm{S_t} \in \mathrm{R}^{\{\mathrm{N}\times\mathrm{M}\}}$ denotes the state of the FL system, capturing the status of all N clients, M is the number of features characterizing each client's state, and T is the trajectory length. The action $a_t \in \{0,1\}^N$ represents a binary client selection vector that indicates which clients will upload their local models, and the return-to-go $\hat{R}^t = \Sigma\_\{t' = t\}\hat{}\{T\}\, r\_\{t'\} \in R$ encodes the cumulative future reward. FLAIR consists of two phases: offline training and online deployment, as illustrated in Figure 2.

In the offline phase, we begin by executing a variety of selection policies (1), under different FL configurations by varying the number of clients, number of nodes, and other relevant parameters (2). For each configuration, the FL system is trained using the TUNE-FL method (3), during which we log the system state, the set of selected clients, and the corresponding reward at each FL round (4). This process results in the construction of an offline database that captures diverse FL trajectories. The collected data is then preprocessed and normalized (5) to serve as training input for the Decision Transformer, which is trained offline to learn effective client selection strategies and their outcome (6). Finally, during the online phase, FLAIR operates FL client selection with a centralized logic that can be deployed on any edge server. In each FL round t, all clients train their local models following the TUNE-FL mechanism (7). Subsequently, clients send their metadata to the corresponding edge server. Once consensus is achieved, the agent uses the necessary

information to determine which clients should upload their local models for aggregation (8) as described in Figure 4.

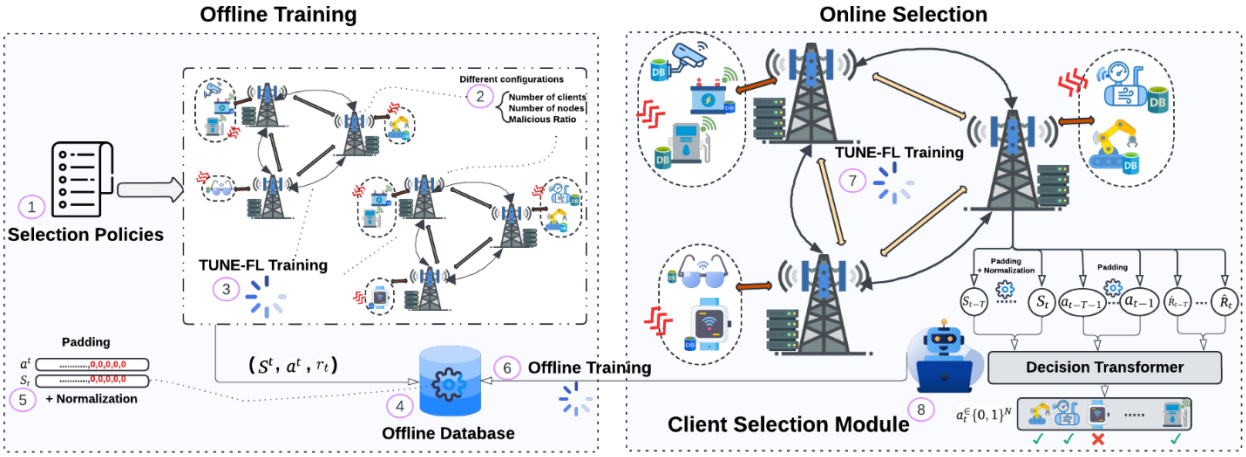


Figure 4: FLAIR Workflow

Technical Details:

Here, we describe the different steps to prepare the necessary information for the client selection process.

Clients' States

We aim to provide various indicators related to both statistical and communication utility, as well as model divergence and historical behavior for each client k, while ensuring data privacy. First, we build the statistical utility for client k as follows:

$$U_{\{stat,k\}}^{\{(t)\}} = \left|D_{\{k\}}^{\{(t)\}}\right| * sqrt\left(\frac{1}{\left|D_k^{\{(t)\}}\right|} * \Sigma^2_{\left\{j \in D_k^{\{(t)\}}\right\}} Loss(j)\right)$$

where $D_{\{k\}}^{\{(t)\}}$ denotes the enlarged seen data samples in the FL round t, as the number of processed samples depends on the synchronization time allocated for each round.

Furthermore, we characterize the communication utility in terms of the client's upload time. The achievable uplink data rate by the client k at round t is formulated as:

$$\lambda_k^{\{(t)\}} = \gamma_{\{k,i\}} B_i \ln\left(1 + \left(h_k^{\{(t)\}}\right\} p_k\right) / I\_\{k,i\}\text{^}\{(t)\}\right)$$

where $\gamma_{\{k,i\}}$ denotes the allocated fraction of the total available bandwidth B_i at server i with

$\Sigma_{\{k\in C_i\}\gamma_{\{k,i\}}} = 1$, p_k is the transmission power, $h_k^{\{(t)\}}$ represents the channel gain at round t, while $I_{\{k,i\}}^{\{(t)\}}$ captures the co-channel interference and noise. Let d be the size of the model parameters w. The upload time for sending w to the server i and the communication utility for client k are calculated as:

$$upload_{\{k:i\}}^{\{(t)\}} = \frac{d}{\lambda_k^{\{(t)\}}}$$

$$U_{\{com,k\}}^{\{(t)\}} = \frac{1}{upload_{\{k:i\}}^{\{(t)\}}}$$

Finally, we define the state $s_k^{\{(t)\}}$ of the client k at round t as:

$$s_k^{\{(t)\}} = \left[U_{\{stat,k\}}^{\{(t)\}}, Mean\left(U_{\{stat,k\}}\right\}\right), Euc\left(w_k^{\{(t)\}}\right\}, W^{\{t-1\}}), Cos(w_k^{\{(t)\}}\}, W\text{^}\{t - 1\}), U\_\{com, k\}\text{^}\{(t)\}, Mean(U\_\{com, k\}) \right]$$

where $Mean\left(U_{\{stat,k\}}\right)$ and $Mean\left(U_{\{com,k\}}\right)$ are the average values over the past rounds, and Euc(·) and Cos(·) denote Euclidean distance and cosine similarity respectively.

FL System State and Reward Function

Given N clients, each represented by a local state vector $s_k^{\{(t)\}}$, we construct the FL system state matrix $S_t \in R^{\{N\times M\}}$. To ensure invariance to client ordering, we compute the L2 norm of each vector, sort them, and flatten the result into a single vector:

$$S_t = Flatten(\, Sort_{\left(s_1^{\{(t)\}}\right\},\ldots,s_N^{\{(t)\}}\dashv)}$$

Finally, the reward function is defined as:

$$r_t = \gamma 1 * \Delta F1_t + \gamma 2 * \Delta CommTime_t$$

where $\Delta F1_t = F1\left(W^{\{(t)\}}\right) - F1\left(W^{\{(t-1)\}}\right)$, $\Delta CommTime_t = CommTime(t-1) - CommTime(t)$, and CommTime(t) is the maximum upload time among selected clients. γ1 and γ2 are tunable weights.

Offline Training and Online Inference

We assume that the system can accommodate up to N clients. If fewer clients are present, the system state and action vectors are padded with zeros, and attention masks are applied. Various FL configurations are recorded under different selection policies in a database, normalized, and used to train the decision transformer offline. Once trained, the online agent only requires appropriate inputs for efficient client selection.

3.3 Results and Analysis

Here, we present the experimental settings, compare FLAIR with baseline methods and analyze potential client selection bias and sensitivity to data heterogeneity.

Experimental Settings:

To evaluate our work, we use two reference IDS datasets. The first one (used for training the DT), IoTID20 [8] is a dataset that includes flow-based and general features tailored to wireless environments, enabling the identification of anomalous behavior across IoT networks. It comprises a total of 625,783 instances, of which 93.60% represent anomalous activity. We randomly split the dataset into 80% for training and 20% for testing. The second dataset (not seen by the agent), IoMT-TrafficData [9] , captures real-world network flow traffic from Internet of Medical Things (IoMT) devices. The training set consists of 439,307 samples, with 48.78% labeled as attack. The test set includes 146,436 samples, maintaining the same attack-to-benign ratio of 48.78%. Finally each client randomly allocates 10% of his data for validation.

Baselines:

We compare FLAIR against various client selection baselines, mentioned previously, which represent state-of-the-art approaches and demonstrate competitive performance in FL:

- PyramidFL: A fine-grained client selection method that improves FL efficiency by jointly considering both data quality and system heterogeneity.
- DivFL: Selects a representative subset of clients by leveraging submodular maximization, aiming to preserve overall diversity in the aggregated model.
- Power-of-Choice: Prioritizes clients with the highest local training loss for participation.
- TUNE-FL: Includes all available clients in each round without any selection criteria.

We note that the first three baselines require a predefined number of participating clients K and E local epochs.

Offline Training Policies:

We build the offline dataset by recording the state, action, and reward tuples generated by the following 7 selection policies:

- All: Select all existing clients.
- K-Random: Randomly select K clients from each edge server and from the entire client pool.
- Top-K Statistical Utility: Select the top-K clients based on statistical utility solely, per group, and globally.
- Top-K Joint Utility: Select the top-K clients based on a joint ranking of statistical and communication utility, both per group and globally.

For each policy, we vary the total number of clients among {100, 150, 200}, and experiment with both 5 and 10 edge servers. For every configuration, we simulate and record 30 FL rounds using only the IoTID20 dataset, distributed among clients following the Dirichlet distribution alpha=0.5. In some experiments, we introduce a small fraction of corrupted clients to simulate adversarial behavior or poor-quality data.

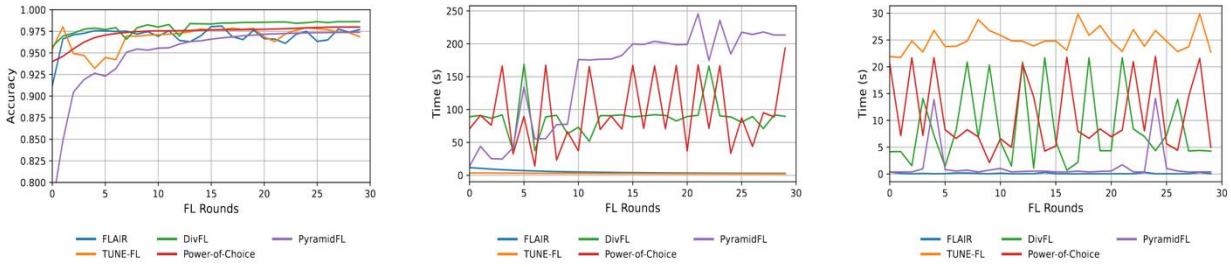
Simulation Settings:

For all experiments, we use a unified simulation setup as summarized in Table. We simulate the computational heterogeneity by introducing random training interruptions depending on each client's type as described previously in TUNE-FL. We also simulate non-IID data distributions by partitioning the training set among clients using a Dirichlet distribution Dir(alpha). Besides, clients are randomly distributed among edge servers. For the detection model, we employ a multi-layer perceptron (MLP) with three hidden layers. Since the task is binary classification, clients with adversarial data are simulated by flipping their labels. Lastly, we use GPT-2 as the DT as suggested by the original proposition. We note that all experiments have been conducted using a MacBook Pro equipped with an Apple M3 Pro chip and 36GB of RAM.

Results and Analysis:

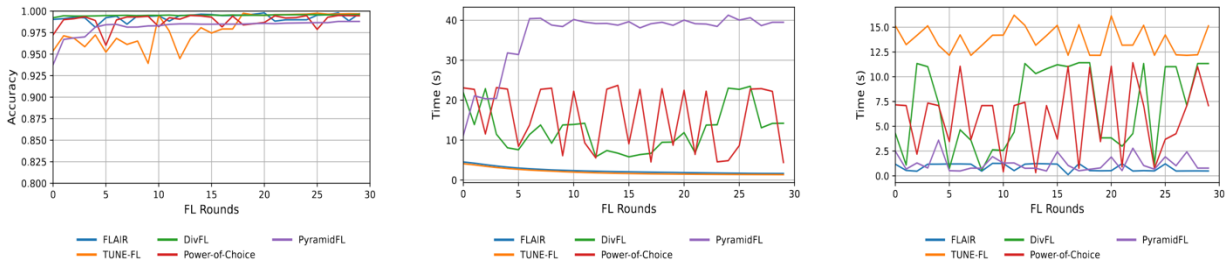
FLAIR can balance between wall clock time and model's performance. We compare in Figure 5 and Figure 6 the test accuracy, computation time, and communication cost for the IIoTID20 and IoMT datasets, respectively, using 100 clients across all selection methods. First, we observe in figs 5.a and 6.a that all selection methods achieve high accuracy over the course of FL rounds, with only marginal differences between them. Notably, DivFL and Power-of-Choice achieve the highest performance, which aligns with their design as they select clients based solely on their statistical utility. As a result, they do not consider computation or communication time, leading to noticeable fluctuations across FL rounds (figs 5.b, 6.b, 5.c and 6.c). In contrast, TUNE-FL aggregates updates from all clients and significantly reduces training time due to its adaptive synchronization mechanism that tracks clients' convergence states. However, it suffers from high communication time, as it must wait for the most interfered client to upload its model. Alternatively, PyramidFL selects clients based on a joint consideration of statistical utility and system efficiency. On one hand, we observe an increase in computation time due to the time pacer mechanism introduced by the approach. On the other hand, it effectively reduces communication time by avoiding clients that would cause prolonged upload delays. Finally, our approach, FLAIR, adopts a training strategy

similar to TUNE-FL, which explains their similar computation times. In addition, FLAIR achieves the lowest communication time. This improvement can be attributed to two key factors. First, the inherent nature of SDFL allows clients to benefit from distributed bandwidth, as communication is not restricted to a single central server. Second, the DT effectively learns from the offline database to select clients that are beneficial not only for the global model's performance but also for reducing communication time, even on new unseen datasets, as shown in Figure 6.



(a) Test Accuracy per Round (b) Computation Time per Round (c) Communication Time per Round

Figure 5: Comparison of FL approaches over accuracy and time on the IIoTID20 dataset for 100 clients.

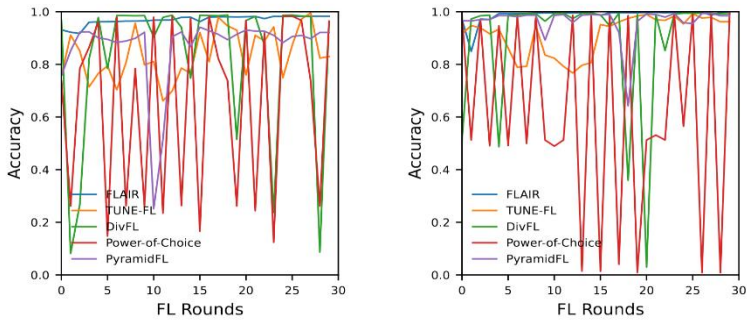


(a) Test Accuracy per Round (b) Computation Time per Round (c) Communication Time per Round

Figure 6: Comparison of FL approaches over accuracy and time on the IoMT dataset for 100 clients.

FLAIR can intelligently avoids selecting clients that would degrade the performance of the global model. Here, we assess the robustness of the selection methods against clients with corrupted data or adversarial behavior. To simulate this, we randomly select 20\% of the 100 clients and flip their labels. We then monitor the evolution of test accuracy over FL rounds to evaluate the impact of such corrupted participation, as shown in Figure 7. We observe that both Power-of-Choice and DivFL exhibit significant instability and fail to converge in the presence of corrupted clients. Power-of-Choice prioritizes clients with high loss, which leads to the frequent selection of corrupted clients across FL rounds, as these clients consistently possess higher losses. Similarly, DivFL aims to diversify the selected client set; however, corrupted clients often deviate from the behavior of normal clients, producing divergent local models. As a result, DivFL tends to prioritize these outliers, which explains the performance degradation and instability observed in both approaches. PyramidFL occasionally exhibits drops in accuracy, which can be attributed to its exploration mechanism that may select corrupted clients during the search process. In contrast, TUNE-FL, which selects all clients for aggregation, also shows some fluctuations, though with less intensity. This improved stability is due to its two-level aggregation strategy, which helps absorb and mitigate the negative impact of corrupted clients. Finally, our approach, FLAIR, demonstrates consistent accuracy improvement without noticeable fluctuations. This indicates that the decision transformer

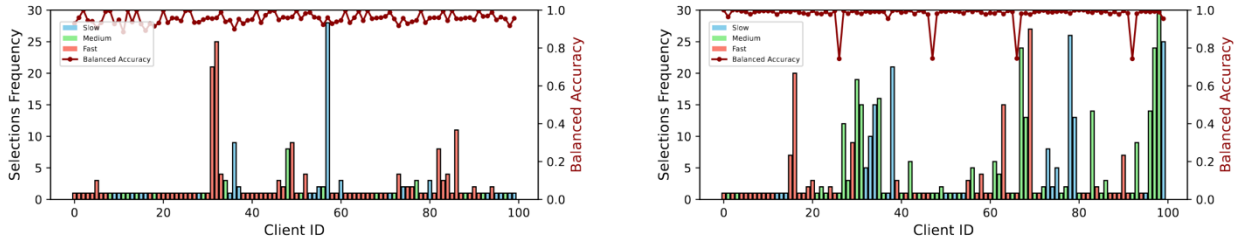
effectively avoids selecting corrupted clients that will lead to performance degradation. These observations hold true across both datasets, highlighting the robustness of FLAIR.



(a) IIoTD20 Dataset (b) IoMT Dataset

Figure 7: Corrupted clients effect 20% from 100 clients

Client Selection Decision Analysis. We assess in Figure 8 the performance of individual clients on the generated global model to evaluate whether FLAIR introduces bias toward frequently selected clients. We show the balanced accuracy of all clients' validation sets across the two datasets. Balanced accuracy corresponds to the average of recall obtained on each class, providing a more reliable metric in the presence of class imbalance resulting from the non-IID distribution. Overall, we observe that the final global model performs well across all clients, achieving validation performance comparable to that on the test set. We also visualize the selection frequency of clients and their types over 30 FL rounds. First, we observe that the selection is independent of the clients' computation types, which is expected as this information is not included in the selection mechanism. In some cases, fast clients appear more frequently in the selection, which can be attributed to the fact that they represent 60% of the total clients. Second, we notice that some clients are selected only once, yet the global model still performs well on their data. This suggests that the decision transformer effectively identifies clients that contribute meaningfully to the global model. Clients whose data is likely already captured by the global model or that may introduce prolonged upload time are ignored, supporting efficient and targeted participation. This indicates that FLAIR does not introduce significant bias and that the global model generalizes well across clients.



(a) IIoTID20 with 100 clients (b) IoMT with 100 clients

Figure 8: Performance of the global model on the clients' validation sets and selection frequency during the 30 FL rounds.

4. Self-Aware and Collaborative System Design

This section presents the architectural design and operational principles of the Multi-Agent and Self-Aware Collaborative Intrusion Detection System (MSCIDS). The system addresses the challenge of detecting security threats in distributed and decentralized smart infrastructures. It uses a combination of autonomous agent-based coordination, adaptive federated learning, and blockchain-enabled trust mechanisms. The design builds upon the inputs from Deliverable 3.1 and incorporates the semi-asynchronous federated learning approach described in Sections 2 and 3, optimizing both training efficiency and detection accuracy across heterogeneous network environments. This architecture is the integration of self-awareness capabilities that enable agents to assess their own detection confidence, data quality, and contribution potential, thereby facilitating the detection of previously unseen attacks while maintaining data privacy throughout the collaborative detection process.

4.1 Proposed Architecture

MSCIDS introduces autonomous, self-aware agents at each monitoring node within the distributed infrastructure. This architectural approach enables decentralized threat detection while maintaining coordinated defense capabilities across domain boundaries. The multi-agent design addresses the inherent challenges of monitoring distributed, heterogeneous smart infrastructure components where centralized approaches would introduce unacceptable latency and single points of failure.

The architecture defines two distinct agent roles that operate in a hierarchical yet collaborative manner, following the principles of multi-agent systems where agents are autonomous and capable of performing specific actions in dynamic operational decentralized and distributed environments . Protection agents are deployed at individual monitoring nodes and are responsible for protecting specific segments of the infrastructure. These agents continuously monitor network traffic within their assigned domains, applying trained intrusion detection models to identify potential security events. Upon detection, they execute localized actions such as traffic filtering, access control, or alert generation.

Supervisor agents operate at a higher coordination level, facilitating information exchange across domains and aggregating threat intelligence from multiple protection agents. When the severity or scope of a detected threat exceeds local response capabilities, supervisor agents may initiate system-wide actions like quarantining affected nodes or triggering policy updates. This dual-role structure enables both rapid local response to immediate threats and coordinated system-wide defense against distributed attack patterns that may target multiple infrastructure domains simultaneously [10-16].

A key characteristic of this architecture is that agents share threat intelligence and coordinate responses to detected anomalies through collaborative protocols. This collaborative capability allows the system to detect previously unseen attacks in distributed and decentralized smart infrastructures by correlating observations across multiple monitoring points. The autonomous nature of the agents ensures that local detection and response can proceed even when communication with other domains is temporarily disrupted, providing resilience against network partitions and targeted attacks on coordination infrastructure. Simultaneously, the self-aware properties enable each agent to assess its own detection confidence, data quality, and contribution potential to the collective defense posture, facilitating intelligent participation in the federated learning process and collaborative threat response mechanisms.

4.2 Parallel Modules: Semi-Asynchronous Federated Learning and Blockchain

The MSCIDS architecture incorporates two parallel operational modules that address complementary aspects of collaborative intrusion detection: a semi-asynchronous federated learning framework for distributed model training, and a dual-layer blockchain infrastructure for secure parameter exchange and device authentication. These modules operate concurrently, with the federated learning framework enabling privacy-preserving collaborative model training while the blockchain infrastructure provides tamper-resistant mechanisms for parameter exchange and trust establishment across domain boundaries [

4.2.1 Semi-Asynchronous Federated Learning: TUNE-FL Framework

The federated learning component builds upon the TUNE-FL framework described in Section 2, which combines the benefits of both synchronous and asynchronous learning paradigms. TUNE-FL allows clients to train locally until either they converge or reach a calculated synchronization deadline, which is dynamically estimated by each client based on its past performance. This approach enables the system to adjust to variations in computing power and training speed across heterogeneous agents without requiring all clients to wait or finish training simultaneously, thereby significantly reducing idle time and communication overhead. The framework incorporates a weighted aggregation mechanism that accounts for incomplete or delayed client contributions, ensuring robustness in the presence of network heterogeneity and varying computational capabilities.

TUNE-FL operates through two distinct phases in each federated learning round, as illustrated in Figure 2. The training phase encompasses global model dissemination, local training with convergence time estimation, and sub-global model aggregation. The whole process has been explained in detail in section 2.

4.2.2 FLAIR: Adaptive Client Selection

Building upon TUNE-FL, Federated Learning with Adaptive and Intelligent Reasoning for client selection (FLAIR) mechanism enhances the system with intelligent client selection capabilities. While TUNE-FL effectively addresses computational heterogeneity through adaptive synchronization, FLAIR narrows the client selection criteria to focus on statistical and communication efficiencies. The objective is to adaptively select a subset of clients that ensures the convergence of the global model while maintaining acceptable upload times, considering both client communication capabilities and dynamic computational capabilities.

FLAIR employs an agent responsible for making client selection decisions based on offline reinforcement learning using a Decision Transformer (DT). This agent operates with access to a global view of all clients independently of the number of edge servers, while performing selections based on both current and historical states and actions. The agent dynamically determines the optimal subset of workers based on factors such as data quality, model staleness, communication costs, and current detection performance. The whole process has been explained in detail in section 3.

4.2.3 Blockchain Architecture for Secure Parameter Exchange

The second parallel module implements a dual-layer blockchain architecture that provides secure, tamper-resistant infrastructure for model parameter exchange and device authentication. This architecture addresses the trust and security challenges inherent in collaborative intrusion detection across multiple administrative domains. At the edge server level, a private blockchain handles local device registration and mutual authentication, ensuring that only authorized agents can participate in the collaborative detection system. This private blockchain maintains a registry of legitimate devices within each domain, enabling rapid local authentication without requiring communication with external authorities.

At the P2P Cloud Server Network level, a consortium blockchain handles the upload and download of model parameters, enabling secure cross-domain model sharing without requiring a centralized trusted authority. This consortium blockchain is maintained by the participating edge servers, which collectively validate and record parameter exchange transactions. Smart contracts deployed within this blockchain infrastructure facilitate the upload of parameters to the consortium blockchain and enable cross-domain registration processes. These smart contracts enforce access control policies, verify the integrity of uploaded parameters, and maintain audit trails of all parameter exchanges.

This dual-layer structure combines a private blockchain at the edge with a consortium blockchain at the P2P cloud level. It ensures sensitive local device information remains within domain boundaries while enabling secure, verifiable parameter exchange across the broader federated learning network. The blockchain infrastructure provides tamper-resistance through cryptographic hashing and distributed consensus, ensuring that malicious actors cannot inject corrupted model parameters or impersonate legitimate agents. This security layer is essential for maintaining the integrity of the collaborative intrusion detection system in adversarial environments where attackers may attempt to poison the federated learning process or compromise the trust relationships between domains [11, 12, 17].

4.3 Self-Awareness for Unseen Attack Detection

The self-aware capabilities of MSCIDS enable the system to detect previously unseen attacks while maintaining data privacy throughout the detection process. These agents have introspection capabilities, allowing them to assess their own detection confidence, data quality, and contribution potential. This self-awareness is operationalized through the state vector components employed in the FLAIR client selection mechanism, which provide each agent with quantitative measures of its current operational status and historical performance. Unlike traditional intrusion detection systems that rely solely on predefined attack signatures or supervised learning on known attack patterns, the self-aware agents in MSCIDS can recognize when their local observations diverge from the global model in ways that suggest novel threat patterns rather than data heterogeneity.

The self-awareness state vector components enable multiple forms of introspection. The statistical utility and mean statistical utility enable agents to evaluate whether their local data distributions are representative and informative for the global detection task. An agent observing high statistical utility indicates that its local data contains valuable information for improving the global model, potentially including novel attack patterns not yet captured by the collective knowledge . Conversely, low statistical utility may indicate that the agent's observations are already well-represented in the global model or that its local data quality is insufficient for meaningful contribution.

The communication utility and mean communication utility enable agents to understand their connectivity constraints and adjust their participation strategies accordingly. An agent experiencing degraded communication conditions can recognize this limitation and potentially defer participation until conditions improve, or alternatively, prioritize the transmission of critical threat intelligence over routine model updates. This awareness of communication constraints ensures that the collaborative detection system remains efficient even in the presence of network heterogeneity and dynamic channel conditions.

This introspective capability is particularly valuable for detecting previously unseen attacks, as agents can recognize when their local observations diverge significantly from the global model, suggesting novel threat patterns rather than mere data heterogeneity. By combining self-assessment with collaborative intelligence sharing, the system can distinguish between benign variations in network behavior and genuine security anomalies that warrant investigation and response. When multiple agents simultaneously report significant divergence in their local models, this convergent evidence strengthens the hypothesis of a novel, distributed attack campaign. The federated learning framework enables this collaborative detection without requiring agents to share raw traffic data, thereby preserving data privacy while facilitating collective threat intelligence [18-22].

4.4 Coordinated Response and Collaborative Defense

MSCIDS architecture implements a response strategy that enables both rapid local action and coordinated system-wide defense. This multi-level response capability is essential for addressing the diverse threat landscape in distributed smart infrastructures, where attacks may range from isolated intrusion attempts targeting individual nodes to sophisticated, coordinated campaigns spanning multiple domains. The response architecture leverages the dual-role agent structure described in Section 4.1, with protection agents providing immediate local response and supervisor agents orchestrating system-wide coordination.

When a protection agent detects a potential security event through its trained intrusion detection model, it first executes localized actions, including traffic filtering, access control, or alert generation. These immediate responses contain potential threats within the affected domain, while the agent shares threat intelligence with other agents through the collaborative framework. The localized response capability ensures that time-critical threats can be addressed with minimal latency, as the protection agent does not need to wait for coordination with other domains before taking defensive action. This autonomy is particularly important for mitigating fast-spreading attacks such as worm propagation or denial-of-service attempts, where delays of even seconds can result in significant damage.

Simultaneously with local response, the protection agent transmits threat intelligence to its associated supervisor agent, including details of the detected anomaly, the confidence level of the detection, and contextual information about the affected network segment. Supervisor agents facilitate information exchange across domains, aggregating threat intelligence from multiple protection agents and identifying patterns that may indicate distributed or coordinated attacks. This aggregation process leverages the self-awareness capabilities described in Section 4.3, as supervisor agents can assess the reliability and significance of reports from different protection agents based on their historical performance and current operational status.

When the severity or scope of a detected threat exceeds local response capabilities, supervisor agents may initiate system-wide actions like quarantining affected nodes or triggering policy updates. This escalation mechanism ensures that threats that could propagate across different infrastructures are

addressed through coordinated defensive measures. The decision to escalate is based on multiple factors, including the number of domains reporting similar anomalies, the confidence levels of individual detections, the potential impact of the threat, and the effectiveness of local containment measures. Supervisor agents employ collaborative protocols to reach consensus on system-wide response actions, ensuring that defensive measures are coordinated and do not inadvertently disrupt legitimate operations.

The collaborative defense capability is enabled by the combination of the federated learning framework, which ensures all agents benefit from collective training on diverse attack patterns. Blockchain infrastructure provides secure, verifiable channels for sharing threat intelligence. Agents coordinate responses to detected anomalies by sharing both detection alerts and contextual information about the nature of observed threats, enabling the system to mount effective defenses against sophisticated attacks that target multiple infrastructure domains simultaneously. The federated learning process continuously updates the global detection model based on newly observed attack patterns, ensuring that all agents benefit from the collective experience even if they have not directly encountered specific attack types.

This coordinated response architecture balances the need for autonomous local action with the benefits of system-wide coordination. It ensures MSCIDS can respond effectively to both isolated security events and complex, distributed attack campaigns while maintaining the decentralized operational model required for resilient smart infrastructure protection. The self-aware capabilities of individual agents, combined with collaborative intelligence-sharing mechanisms, enable the system to adapt to evolving threat landscapes and detect previously unseen attacks without compromising the privacy of local network data. The blockchain-enabled trust infrastructure ensures that coordination and intelligence sharing can proceed securely even in adversarial environments where attackers may attempt to compromise the collaborative detection system itself.

References

[1]. L. Li, Y. Fan and K. -Y. Lin, "A Survey on federated learning," 2020 IEEE 16th International Conference on Control & Automation (ICCA), Singapore, 2020, pp. 791-796, doi: 10.1109/ICCA51439.2020.9264412. keywords: {Data models;Servers;Training;Optimization;Data privacy;Machine learning;Computational modeling;Federated learning;Literature survey;Citation analysis;Research front}

[2]. A. B. Mansour, G. Carenini, A. Duplessis and D. Naccache, "Federated Learning Aggregation: New Robust Algorithms with Guarantees," 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA), Nassau, Bahamas, 2022, pp. 721-726, doi: 10.1109/ICMLA55696.2022.00120. keywords: {Training;Machine learning algorithms;Federated learning;Aggregates;Mathematical models;Classification algorithms;Task analysis;Federated Learning;Aggregation Strategies},

[3]. N. Lang, A. Cohen and N. Shlezinger, "Stragglers-Aware Low-Latency Synchronous Federated Learning via Layer-Wise Model Updates," in IEEE Transactions on Communications, vol. 73, no. 5, pp. 3333-3346, May 2025, doi: 10.1109/TCOMM.2024.3486979. keywords: {Computational modeling;Training;Servers;Analytical models;Low latency communication;Federated learning;Convergence;Stochastic processes;Numerical models;Limiting;Federated learning (FL);low latency communication},

[4]. H. Jmal, K. Piamrat and O. Aouedi, "TUNE-FL: Adaptive Semi-Synchronous Semi-Decentralized Federated Learning," 2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2025, pp. 1-6, doi: 10.1109/CCNC54725.2025.10975902. keywords: {Training;Adaptive systems;Federated learning;Network topology;Image edge detection;Intrusion detection;Heterogeneous networks;Servers;Synchronization;Distributed computing;Semi-Decentralized federated learning;edge device heterogeneity;adaptive synchronization},

[5]. N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 2015, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942. keywords: {Telecommunication traffic;Feature extraction;Servers;Training;Data models;IP networks;Benchmark testing;UNSW-NB15 data set;NIDS;low footprint attacks;pcap files;testbed},

[6]. Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection," in IEEE Access, vol. 8, pp. 132911-132921, 2020, doi: 10.1109/ACCESS.2020.3009843. keywords: {Feature extraction;Classification algorithms;Anomaly detection;Information filters;Support vector machines;Filtering algorithms;Feature selection;anomaly detection;information gain;CICIDS-2017 dataset;classifier algorithm},

[7]. H. Jmal, K. Piamrat, O. Aouedi and Y. Ji, "FLAIR: Federated Learning with Adaptive and Intelligent Reasoning for Client Selection," 2025 International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Barcelona, Spain, 2025, pp. 178-182, doi:

10.1109/MSWiM67937.2025.11308736. keywords:{Wireless communication; Training; Adaptation models; Adaptive systems; Federated learning; Computational modeling; Intrusion detection; Distributed databases; Transformers; Internet of Things;Federated Learning (FL);Intrusion Detection System (IDS);Decision Transformer (DT);Client Selection},

[8]. I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in iot networks," in Advances in Artificial Intelligence, C. Goutte and X. Zhu, Eds. Cham: Springer International Publishing, 2020, pp. 508–520.

[9]. J. Areia, I. A. Bispo, L. Santos, and R. L. Costa, "Iomt-trafficdata: A dataset for benchmarking intrusion detection in iomt," Jul. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8116338

[10]. Soltani, M., Khajavi, K., Jafari Siavoshani, M. and Jahangir, A.H., 2024. A multi-agent adaptive deep learning framework for online intrusion detection. *Cybersecurity*, *7*(1), p.9.

[11]. Kumar, M., Kaushik, A., Fischione, C. and Sheng, V.S., 2026. Blockchain and Federated Learning for Decentralized Edge Intelligence in Next Generation IoT Security. *IEEE Communications Standards Magazine*, *10*(1), pp.7-9.

[12]. Abou El Houda, Z., Moudoud, H., Brik, B. and Khoukhi, L., 2024. Blockchain-enabled federated learning for enhanced collaborative intrusion detection in vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, *25*(7), pp.7661-7672.

[13]. Mansouri, F., Tarhouni, M., Alaya, B. and Zidi, S., 2025. A distributed intrusion detection framework for vehicular ad hoc networks via federated learning and blockchain. *Ad Hoc Networks*, *167*, p.103677.

[14]. Połap, D., Srivastava, G. and Yu, K., 2021. Agent architecture of an intelligent medical system based on federated learning and blockchain technology. *Journal of Information Security and Applications*, *58*, p.102748.

[15]. Xu, M., Peng, J., Gupta, B.B., Kang, J., Xiong, Z., Li, Z. and Abd El-Latif, A.A., 2021. Multiagent federated reinforcement learning for secure incentive mechanism in intelligent cyber–physical systems. *IEEE Internet of Things Journal*, *9*(22), pp.22095-22108.

[16]. Tan, K. and Zhu, C., 2025. Multi-agent deep reinforcement learning for vehicular resource allocation: A comparison study of different agent cooperation levels. *IEEE Transactions on Vehicular Technology*.

[17]. Atia, O.B., Al Samara, M., Bennis, I., Abouaissa, A., Gaber, J. and Lorenz, P., 2025. M3D-FL: Multi-layer malicious model detection for federated learning in IoT networks. *Computers & Security*, *154*, p.104444.

[18]. de Witt, C.S., 2025. Open challenges in multi-agent security: Towards secure systems of interacting ai agents. *arXiv preprint arXiv:2505.02077*.

[19]. Moudoud, H., Abou El Houda, Z. and Brik, B., 2024. Advancing security and trust in wsns: A federated multi-agent deep reinforcement learning approach. *IEEE Transactions on Consumer Electronics*, *70*(4), pp.6909-6918.

[20]. Louati, F., Ktata, F.B. and Amous, I., 2024. Big-IDS: a decentralized multi agent reinforcement learning approach for distributed intrusion detection in big data networks. *Cluster Computing*, *27*(5), pp.6823-6841.

[21]. Buyuktanir, B., Altinkaya, Ş., Karatas Baydogmus, G. and Yildiz, K., 2025. Federated learning in intrusion detection: advancements, applications, and future directions. *Cluster Computing*, *28*(7), p.473.

[22]. Belenguer, A., Pascual, J.A. and Navaridas, J., 2025. A review of federated learning applications in intrusion detection systems. *Computer Networks*, *258*, p.111023.