Federated Learning-enabled Network Incident Anomaly Detection Optimization for Drone Swarms

Kevin Kostage
Florida Gulf Coast University
Fort Myers, United States
kskostage9457@eagle.fgcu.edu

Rohan Adepu
University of Colorado Boulder
Boulder, United States
rohan.adepu@colorado.edu

Jenaya Monroe
University of Missouri-Columbia
Columbia, United States
jimd4n@missouri.edu

Trevontae Haughton
University of Missouri-Columbia
Columbia, United States
thgtx@missouri.edu

Juan Mogollon
University of Missouri-Columbia
Columbia, United States
jdmcnw@missouri.edu

Saketh Poduvu
University of Missouri-Columbia
Columbia, United States
spf94@missouri.edu

Kannappan Palaniappan
University of Missouri
Columbia, United States
pal@missouri.edu

Chengyi Qu
Florida Gulf Coast University
Fort Myers, United States
cqu@fgcu.edu

Prasad Calyam
University of Missouri
Columbia, United States
calyamp@missouri.edu

Reshmi Mitra
Southeast Missouri State University
Cape Girardeau, United States
rmitra@semo.edu

Abstract

The increasing reliance on drone swarms for various applications necessitates robust real time anomaly detection mechanisms to ensure operational security and efficiency. Federated learning is particularly well-suited in the drone context as it enables decentralized data processing, preserving data privacy and security while enhancing detection accuracy. In this paper, we explore optimization methods for federated learning-enabled *network incident anomaly detection* in drone swarms using the NSF AERPAW platform. To achieve this, we demonstrate a defense mechanisms such as differential privacy and adversarial training, to strengthen the robustness of federated learning models against data poisoning attacks. We are collecting three sets of metrics for accuracy, system and network usage under a variety of test cases reflecting benign, mildly poisoned, highly malicious, safe situations. The experimental results reveal that adversarial training is particularly effective, achieving up to 91.1% accuracy with a 33% data poisoning volume. Additionally, we evaluate the computational overhead introduced by these defenses, finding that while they enhance security, they also increase CPU usage by up to 233% in active drone scenarios. These findings highlight the trade-offs between security and operational efficiency in FL-enabled drone swarms, offering critical insights for deploying robust, real-time anomaly detection systems in decentralized environments.

CCS Concepts

• **Security and privacy** → **Mobile and wireless security**; **Intrusion detection systems**.

Keywords

Drone Swarm, Federated Learning, Anomaly Detection.

ACM Reference Format:
Kevin Kostage, Rohan Adepu, Jenaya Monroe, Trevontae Haughton, Juan Mogollon, Saketh Poduvu, Kannappan Palaniappan, Chengyi Qu, Prasad Calyam, and Reshmi Mitra. 2025. Federated Learning-enabled Network Incident Anomaly Detection Optimization for Drone Swarms. In *26th International Conference on Distributed Computing and Networking (ICDCN 2025), January 04–07, 2025, Hyderabad, India.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3700838.3700857

1 Introduction

Drone swarm topology is highly dynamic, with frequent changes due to adhoc network connections and flight path adjustments [16, 27]. Drones combine mobility with operational flexibility for meeting highly critical mission goals such as disaster response, environmental monitoring, precision agriculture, surveillance, supply chain management [1, 16]. These applications require drones to produce and process massive volumes of real-time, multi-modal sensitive data on-site. The resource-constrained drones must continuously process data to make immediate informed decisions for a variety of mission-critical tasks such as avoiding obstacles, adjusting flight

ICDCN 2025, January 04–07, 2025, Hyderabad, India

ACM ISBN 979-8-4007-1062-9/25/01
https://doi.org/10.1145/3700838.3700857

paths, or responding to environmental changes. This real-time requirement necessitates low-latency communication and reliable on-board processing capabilities [7, 16]. Federated Learning (FL) can play a crucial role in this context by enabling decentralized data processing across drone swarms, allowing for the development of models that improve decision-making while preserving data privacy in these resource-constrained environments.

Another critical aspect for drone operations is ensuring the security and privacy, which has led to many recent works [14, 20, 21, 26]. Addressing these challenges is essential to maintain the reliability, integrity, and effectiveness of drone-based operations. However, there is a lack of high quality state-of-art on *drone testbed integration with federated learning threat models and defense solutions*. Developing robust security measures and integrating advanced computational models are vital to overcoming these challenges and ensuring the successful deployment of drones in various critical applications. Prior works have often focused on isolated aspects of swarm architecture, such as ad-hoc networks, AI, or security along with flight path management, without comprehensive testbed experimentation in realistic scenarios [7].

In this paper, we design and implement secure solutions for drones during full flight, ensuring they can perform mission-critical tasks effectively while managing computational efficiently. We have developed a comprehensive framework that integrates FL techniques for malicious client defense by performing network anomaly detection and adversarial training defense in a real-world drone testbed. The complete FL pipeline, including baseline and defense mechanisms, is integrated into the Aerial Experimentation and Research Platform for Advanced Wireless (AERPAW) setup [13], with flight durations ranging from 25 to 60 minutes. To assess our solution, we evaluate it using three sets of metrics in terms of: machine learning performance, network stability, and hardware usage.

Our primary contributions include developing a custom FL model trained on two different IoT attack datasets. We set up a basic FL system for drones, utilizing fixed and edge nodes to address computational limitations and privacy concerns. We present two attack scenarios for drone data poisoning: label flipping and feature noise. Our threat model involves a utility-centric attack, where client data is poisoned to compromise model accuracy. The defense technique leverages inbuilt TensorFlow library features such as robust aggregation and differential privacy. Additionally, we incorporate computationally-intensive adversarial training for severe poisoning cases. We demonstrate the effectiveness of the proposed strategies in emulated scenarios using the AERPAW infrastructure [2]. We propose a series of experiments to evaluate the performance of the system and defense strategies, considering scenarios involving simulated flight paths and simultaneous computationally intensive tasks involving image detection.

We study the various trade-offs, balancing the computational efficiency of offloading tasks to edge nodes with the need to ensure data privacy through FL. By integrating FL with advanced defense mechanisms, we provide evidence for effective anomaly detection and response strategies. Our approach mitigates the impact of adversarial attacks and data poisoning, ensuring that the FL models maintain high performance and reliability in dynamic and resource-constrained environments. This not only strengthens the security posture of drone swarms but also optimizes their computational resources, enabling more effective real-time decision-making and mission success in critical applications.

The rest of the paper is organized as follows: Section 2 presents our review of the related work. Section 3 outlines the methodology for distributed anomaly detection, featuring FL attack and defense scenarios. Section 4 details the results of our experiments in the AERPAW testbed. Finally, Section 5 concludes the paper.

2 Related Work

2.1 AERPAW Testbed

The AERPAW is a cutting-edge infrastructure [13] that is designed for advanced research in wireless communication and networked aerial systems [18]. AERPAW offers a comprehensive infrastructure for conducting experiments with drones, ground nodes, and edge computing resources. This platform is instrumental in facilitating innovative research and development in diverse critical areas, such as disaster response management, smart agriculture, and smart city [15].

The infrastructure includes a diverse array of drones equipped with state-of-the-art communication and sensing technologies. These drones can operate autonomously or in coordination with ground nodes, enabling complex experimental setups. These edge nodes can perform computationally intensive tasks on-site, reducing latency and improving response times. AERPAW supports robust wireless communication infrastructure, including Wi-Fi, 4G/5G, and specialized drone communication protocols, ensuring seamless data transfer between fixed and portable nodes [13]. The platform offers both virtual real-time emulations and approved real portable node missions. It enables researchers to test and validate new technologies in a controlled, repeatable, and cost-effective environment. We utilize AERPAW physical platform for our drone swarm experimental testbed. We also leverage the FLYPAW [8] application in AERPAW to enhance the performance monitoring capabilities in terms of the network and system metrics. This makes AERPAW an ideal platform for our efforts to develop *optimized and secure threat intelligence frameworks for dynamic drone environments*.

2.2 Federated Learning & Anomaly Detection

FL is an advanced machine learning paradigm introduced by Google in 2016 [9] that facilitates decentralized data processing [5]. Unlike traditional approaches that require transferring data to a central server, FL trains models locally on edge devices or clients. Each client computes model updates using its local data, which are then aggregated on a central server to create a global model. This approach enhances data privacy and security by ensuring that raw data remains on the clients, thereby reducing the risk of data breaches. Additionally, it minimizes communication overhead and latency associated with data transfer to central servers. By leveraging the computational power of edge devices, FL is scalable and efficient for large-scale applications.

Anomaly detection, which involves identifying patterns in data that deviate from expected behavior, is crucial for detecting critical issues such as security breaches, fraud, or system failures. Traditional methods rely on centralized data processing, aggregating data from multiple sources to identify these anomalies. However, FL offers significant advantages over centralized learning for real-time

detection of rare or distributed anomalies that may remain undetected in aggregated datasets [24]. Consequently, FL has become a well-explored approach for threat intelligence applications in IoT and edge computing environments. Due to the dynamic nature of network connections and the limited computational resources of individual drones, *implementing anomaly detection using FL on a drone swarm is a challenging endeavor* [19]. Our novelty is that we explore the benefits of FL for optimizing network incident anomaly detection and enabling decentralized data processing.

2.3 Adversarial Attacks on Federated Learning

Despite their privacy-preserving capabilities such as secured aggregation and differential privacy, FL is vulnerable to data poisoning attacks. The decentralized FL becomes susceptible, as the central server relies on aggregated updates from numerous clients. By corrupting the training data on even a few clients, adversaries can significantly distort the performance of the global model, leading to inaccurate, biased, or harmful outputs [10]. FL threats are categorized into utility-centric and privacy-centric threats. *Utility-centric threats* involve poisoning data or models, compromising model accuracy. *Privacy-centric threats* focus on extracting sensitive information from the training data, jeopardizing data confidentiality.

Depending on the knowledge and capabilities of the adversary, data poisoning attacks can be classified into *white, grey, and black box* settings [10, 25]. White-box attacks provide the attacker full access to model parameters and predictions, making these attacks highly sophisticated and difficult to defend against, as the adversary can precisely manipulate the model. Grey-box attacks offer partial knowledge of the system, which is a more realistic scenario as attackers can exploit specific vulnerabilities with moderate effort and information, making them practical and challenging. Black-box attacks provide the least information, requiring the attacker to infer the system design and behavior based solely on observed outputs, making these attacks less precise but potentially damaging [11].

Feature noise [4] is an adversarial technique where the attacker introduces random or maliciously crafted noise directly into the features of the training data, causing the model to learn incorrect patterns thereby degrading the performance. Techniques such as data obfuscation, where data is intentionally scrambled or distorted, and feature tampering, where specific features are corrupted. In *Label flipping attack* [12] the adversary intentionally alters the labels of the training data, causing the model to learn incorrect associations between features and labels. This manipulation degrades the model's performance by introducing systematic errors during the training process. In a typical label-flipping attack, benign labels are changed to incorrect or misleading labels, leading to biased or inaccurate predictions.

The impact of a label-flipping attack can be more severe in comparison with the feature noise attack because the model is directly misled about the correct relationship between the inputs and the outputs. However, both these attacks pose a significant threat in FL due to the decentralized nature of the training process. Each client trains on its local data and sends model updates to a central server. An attacker controlling one or more clients can inject noise into the local training data, compromising the global model performance [17]. Detection becomes difficult because the central server does not have direct access to the raw data and relies on the integrity of the client updates. *In our work, we uniquely focus on black-box utility-centric attacks as they degrade the accuracy and effectiveness of the model. We present sophisticated attack pipelines to emulate the behavior of label flipping and feature noise attacks.*

3 Framework for Experimental Configuration and Computational Analysis

3.1 AERPAW Experiment Configuration

Scope of Experimentation. Our primary experimentation objective is to evaluate distributed anomaly detection modeling of network events for drone swarms, featuring FL attack and defense scenarios. Utilizing the AERPAW platform, we conduct stress testing to discover the limits of both the drones and the platform itself. By pushing the system to its boundaries, *we can identify potential vulnerabilities in communication channels and ensure that data integrity is maintained.* This aspect of the testbed's utility is crucial for real-world applications, as it helps address unforeseen security and system utilization challenges. The ability to simulate and test these scenarios in a controlled setting allows for a more thorough understanding of potential issues, leading to the development of more resilient systems. The significance of these experiments extends beyond theoretical research; they provide practical insights that can be directly applied to enhance the safety and reliability of drone operations using industry standard hardware, and realistic field conditions.

The introduction of several complex scenarios in the AERPAW testbed, such as combining FL with anomaly detection networks, highlights the timeliness and significance of our approach. FL allows for the development of machine learning models using decentralized data, which is particularly useful in scenarios where data privacy is a concern. Anomaly detection at the network-level, on the other hand, helps in identifying unusual patterns or behaviors that could indicate security threats or system malfunctions. Integrating these technologies within AERPAW experiments in our work not only demonstrates the platform's versatility but also highlights its potential for advancing the state-of-the art in drone technology experimentation in advanced wireless environments, and network security related studies.

Overall Workflow. The experimental setup within the AERPAW network, as depicted in Figure 1, illustrates a holistic network configuration designed to assess the resilience of FL models against malicious threats. The setup encompasses 2 to 5 portable nodes, each furnished with identical computational resources, among which one node is deliberately designated as malicious. This node is tasked with simulating attacks to evaluate their potential impact on the performance of the FL model. Essential to this configuration is a fixed node that manages the crucial FL average aggregation process, a core component of FL architecture, facilitated by the AERPAW network infrastructure.

AERPAW provides the essential network backbone, virtual machines, and SSH access, which supports the exploration of the platform's capabilities and its constraints for diverse experimentation. Each portable node is equipped with datasets that may be either normal or poisoned to simulate data integrity attacks, alongside

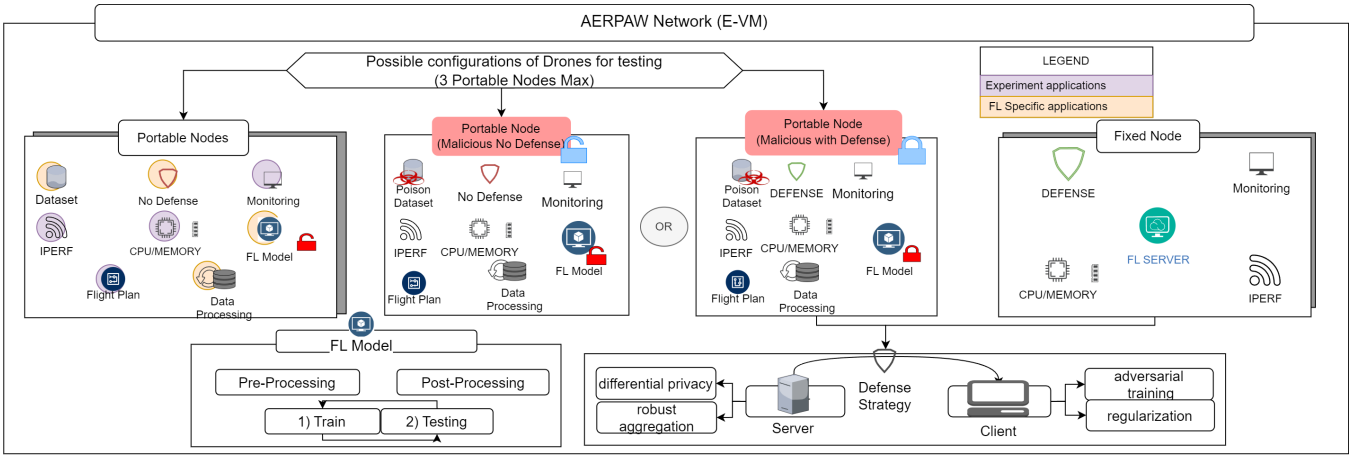

Figure 1: **Overall Experimental Setup of Federated Learning Network with Defense Mechanisms in AERPAW**

monitoring tools that include both hardware and network monitoring, and FL models that incorporate defensive mechanisms to guard against these attacks.

Additionally, each node operates under a predefined flight plan. The fixed node is instrumental in aggregating weights from the portable nodes, thereby orchestrating the overall FL process. The system is further bolstered by integrated defense mechanisms within both the portable and fixed nodes to protect against data poisoning and other forms of cybersecurity threats. The setup also includes custom monitoring and metric logging capabilities to enhance operational oversight. However, limitations exist within the AERPAW setup such as the inability to log certain metrics directly, the absence of battery life monitoring during simulations, and network restrictions that impede communication between containerized applications across different nodes. These constraints necessitate consideration when deploying and testing models that leverage constrained resources in this environment.

Experimentation Workflow on AERPAW.. The experimentation workflow within the AERPAW platform involves several key steps to ensure a comprehensive experimentation with our approach. Figure 2 details the development process on the AERPAW platform. The process begins with *Initiate Development*, which includes submitting paperwork to AERPAW, selecting fixed and portable nodes, and obtaining necessary files such as manifests and OpenVPN files. Next, in the *Local Computer Setup* phase, essential applications such as OpenVPN, QGroundControl, and a terminal for accessing the OEO Console are installed, with Mac users utilizing TunnelBlick instead of OpenVPN for connectivity[2].

Following setup, the *Add Experiment Files* step (third row) involves incorporating datasets, FL models, and data collection scripts onto the nodes. The Start Experiment file is modified to include parameters for traffic, radio, vehicle, and FL client operations, along with poison and defense files. The *Test* phase (fourth row) then verifies the Start Experiment file for correct file paths and ensures relevant scripts are uncommented. Nodes are restarted through the OEO Console, and experiments are visualized using QGroundControl. In the *Finding and Collecting Info* phase (end row), data such as QGroundControl files, CPU logs, and iPerf output files are gathered during the experiment. Finally, in the *Post Processing and Visualizing* phase, collected data is processed and visualized for final analysis to determine the success of the experiments. This structured workflow ensures that all aspects of the experiment are thoroughly examined, from setup to post-processing, allowing for accurate and reliable results.

3.2 Federated Learning pipeline

3.2.1 FL Preliminaries. Our FL pipeline is built using two primary frameworks: Flower[1] and TensorFlow[2]. Flower orchestrates the distributed training process by coordinating communication between a central server and multiple edge devices. The server collects partial model updates from each device, combines them to create a global model, and then distributes this updated model back to the devices for further training. Flower seamlessly integrates with TensorFlow, which we use to develop the deep neural network (DNN) model. It is a multilayer perceptron (MLP) specifically designed to achieve high accuracy in detecting drone network events from specialized datasets. The detailed DNN model architecture is described in Section 4 of this paper.

Our setup is deployed on the AERPAW platform, with the fixed node serving as the aggregator server and the portable nodes functioning as edge nodes. Initially, we explored deploying the FL system using Docker containers to create a portable and modular environment. However, we encountered limitations related to AERPAW's network security, which restricts inter-container communication across nodes. Although it might be possible to overcome these issues with additional configuration, we chose to simplify the experimentation process by not relying on containerization as a requirement for our deployment.

3.2.2 Adversary Goals and Capabilities. The primary goal is to execute a data poisoning attack as described in Section 2.3, where the integrity of the training data is compromised to degrade the global model performance. Figure 3 shows a detailed step-by-step attack pipeline for the process. We assume that the attacker has

[1]https://flower.ai/
[2]https://www.tensorflow.org/federated

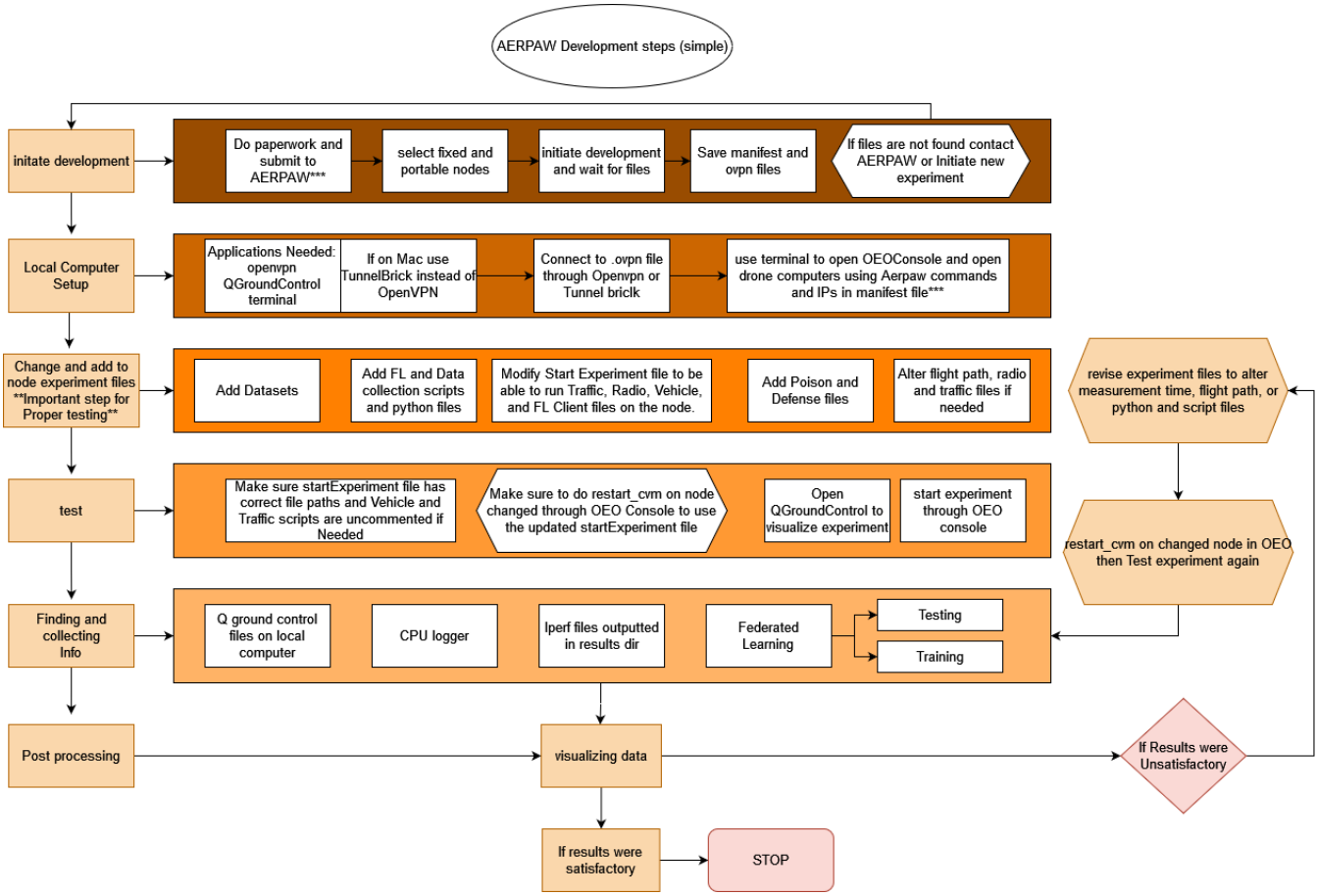

Figure 2: **A Simplified Step-by-Step Process for AERPAW Development Experiment Setup and Execution**

gained *limited access* to the file system of at least one drone in the network during flight. The attacker is aware of the existence of a FL model, but lacks in-depth knowledge of the DNN architecture, training process, secure transfer or the global aggregation algorithm. However, by using command-line tools, the attacker can inspect sub-processes running on the compromised node, potentially gaining insight into the dataset content, targets, and variables once the relevant files are located.

To implement this attack considering the attacker's perspective, the first step involves locating all files with a .csv extension within the identified folder. These files are then strategically corrupted with minor modifications to escape detection. Once the files are identified, the simulation proceeds to identify a "Label" column within each dataset, representing the target for a *label-flipping* attack. In this step, the simulation assumes that the attacker can recognize a generic label column. The attack involves replacing a certain percentage of the labels, simulating the introduction of errors. For binary classification tasks, labels such as 1 and 0 are flipped, while in non-binary cases, the least frequent label is identified and flipped, simulating a more sophisticated attack vector.

For the *feature noise* attack, the simulation considers each column in the dataset, identifying those that contain numeric data. For each identified numeric column, the simulation adds random noise to the values, considering the distribution of the data. By introducing small but impactful changes, the attacker subtly degrades the quality of the dataset. Finally, the adversary replaces the original training data with the poisoned data in the designated folder, representing the completion of the attack.

3.2.3 Defense strategies. To counter these adversarial threats, we incorporate two primary defense mechanisms into our FL pipeline: *Differential Privacy* and *Adversarial Training*.

Differential Privacy is employed to protect the contributions of individual data points by introducing noise into the gradients during training. This method ensures that even if an attacker gains access to the model updates, they cannot accurately infer sensitive information from any single data point. The gradients are first clipped to a predefined norm and then perturbed with Gaussian noise, as represented by the equation: $g = \frac{g}{\max(1, \frac{||g||_2}{C})} + \mathcal{N}(0, \sigma^2 C^2)$, where g represents the gradient, and $\mathcal{N}(0, \sigma^2 C^2)$ is the Gaussian noise added to ensure privacy. This technique effectively masks the impact of individual data points on the overall model, thereby preserving privacy and increasing the system resilience to inference attacks.

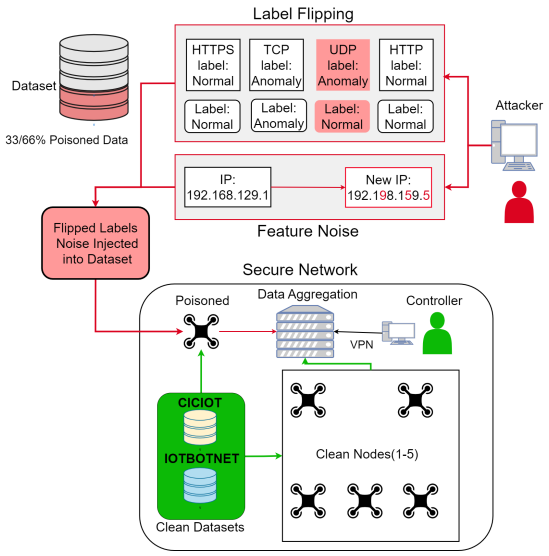

Figure 3: **Attack Pipeline Setup showcase with the label flipping attacks and feature noise attacks.**

Adversarial Training involves incorporating adversarial examples—data points intentionally perturbed to mislead the model—into the training process. By doing so, the model is trained to resist these perturbations, thereby enhancing its robustness against potential adversarial attacks. The process of generating adversarial examples can be expressed as: $x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\Delta_x \mathcal{L}(M(x), y))$, where x_{adv} is the adversarial example, ϵ is the perturbation magnitude, and $\Delta_x \mathcal{L}(M(x), y)$ is the gradient of the loss function with respect to the input x. These defense mechanisms are integrated into the Federated Averaging algorithm, where each client performs local updates with the specified defense strategies in place. The central server then aggregates these updates to refine the global model, ensuring that it remains robust against adversarial manipulation.

3.2.4 Monitoring. To ensure the integrity and stability of the FL model, we embedded monitoring and evaluation mechanisms within the FL process. These mechanisms are crucial for promptly detecting any deviations due to attacks, although they introduce additional complexity and potential latency, particularly in edge environments with limited computational resources. Despite these challenges, the chosen defense strategies are designed to provide a robust defense mechanism, balancing the trade-offs between security and efficiency. The overarching goal is to maintain the model's integrity without compromising the operational stability of the drones, ensuring they can perform their tasks with minimal disruption, even in the face of adversarial threats.

The detailed evaluation of the FL pipeline's performance, including its resilience to attacks, is discussed in Section 4. Our initial intuition suggests that adversarial training alone should suffice to maintain model performance under attack scenarios. The integration of defense mechanisms such as differential privacy and adversarial training should not significantly compromise the computational performance of the edge devices. Interestingly, during the mission, it was observed that the *battery life of the drones remained unchanged despite the computational demands, suggesting that the current testing environment may not fully replicate the real-world impact of computational consumption on battery performance, which could be a critical factor in actual deployment scenarios.*

3.3 During flight computational workload

In this study, we explored two realistic operational scenarios to rigorously assess the performance and resilience of our FL model under different conditions. The experimental design was structured to simulate flight planning and real-time object detection tasks, focusing on computationally intensive operations to evaluate the system's efficiency and robustness. These scenarios were specifically chosen to stress-test the FL model by imposing high computational demands, thereby providing a comprehensive evaluation of the system's capability to maintain operational security and efficiency under resource-intensive conditions.

For the flight route planning component, we utilized the FlyPaw [8] software suite, which is integrated with the AERPAW testbed infrastructure. FlyPaw employs a time-optimized planning (TOP) algorithm to manage task offloading and route planning, taking into account variables such as network connectivity uncertainties and data collection requirements. This setup allowed for controlled experiments, including simulated poisoning attacks, to evaluate the resilience of the FL system. The analysis of post-experiment data, including iPerf synthetic traffic information and flight metrics, provided insights into system performance and helped refine our approach. This experimental framework enabled a thorough evaluation of the system's ability to handle high computational loads while maintaining robust security measures, demonstrating the practical applicability of our solutions in real-world drone operations.

In parallel, we developed an object detection model using the YOLOv8 [23] architecture, specifically trained on the VisDrone 2021 [28] dataset, which is tailored for aerial object detection tasks. The dataset comprises of more than 288 video clips with 261,908 frames and 10,209 static images, annotated with over 2.6 million bounding boxes representing various objects such as pedestrians, vehicles, and bicycles. Collected across 14 cities in diverse urban and rural settings, and under varying weather and lighting conditions, this dataset provides a robust foundation for object detection in real-world scenarios. An example of the images and annotations from this dataset is depicted in Figure 4.

The YOLOv8 model was also integrated into our FL framework using the Flower framework [6], presenting a realistic scenario for object detection on edge devices such as drones. During flight, drones must navigate obstacles and capture visual data in environments specific to their operational paths, making federated training critical for adapting to specific conditions. Object detection is essential for navigation, surveillance, and environmental monitoring, requiring drones to accurately identify and track objects in real-time. This capability is vital for the autonomous operation of drones, allowing them to dynamically adjust to changing environments and update their operational parameters. By training the model online, the system remains adaptive to new conditions and emerging threats, enhancing its responsiveness.

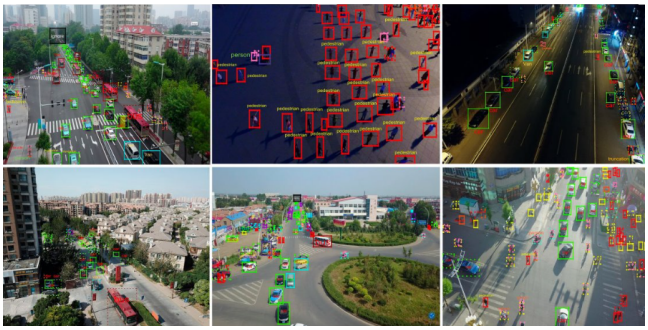

Figure 4: **Annotation exemplars in the VisDrone-DET2020 challenge. Reprinted from [28].**

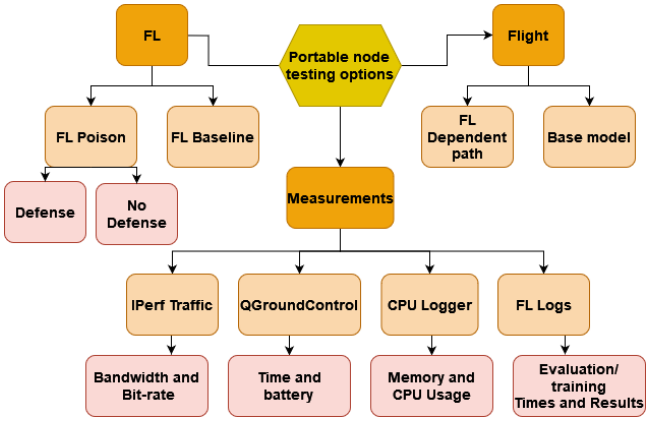

Figure 5: **Tree of the experiment design space.**

The inclusion of these advanced computer vision tasks significantly increases the computational demands on the system. Real-time image processing and model training, particularly for complex models such as YOLOv8, impose substantial computational requirements on the limited resources typically available onboard drones, including CPU, GPU, and memory. These high computational loads can lead to increased latency, potentially affecting the drone's responsiveness and compromising mission-critical tasks that require real-time decision-making. The integration of such resource-intensive tasks thus presents a rigorous test of the system's ability to manage high processing demands while maintaining operational integrity and security.

4 Results

4.1 Evaluation Framework

The workflow outlined in the Figure 5 begins with the selection of testing options for portable nodes, categorized into two primary pathways: FL and Flight. Within the FL pathway, two models are considered: FL Baseline, which represents standard conditions without any poisoning or defenses, and FL poisoned model. The poisoned model is further subdivided into scenarios with or without defenses. Thus, it represents diverse cases for benign, suspicious, malicious and safe conditions. Measurements play a central role across all testing options, encompassing various aspects such as iPerf synthetic traffic for evaluating bandwidth and bit-rate, QGroundControl for tracking time and battery usage, CPU Logger for monitoring memory and CPU usage, and FL Logs for assessing evaluation and training times and results.

On the other side, Flight pathway branches into the base model and the FL Dependent path, indicating the integration of FL models with flight operations. Both pathways feed into the broader measurement framework, ensuring comprehensive data collection and analysis across the different tested scenarios. Such a structured approach allows for a thorough evaluation of the system's performance under various experimental conditions.

To facilitate comprehensive testing, our workflow leverages various tools and scripts. The versatile network measurement tool, iPerf3[3], is integrated as a traffic generation software preloaded onto drone systems within AERPAW for the testing of bandwidth, delay, and packet loss on IP-enabled links. In the AERPAW testbed, a client-server pair, consisting of a large drone with an active flight plan and a fixed node, establishes a measurable network connection. During development, this connection traverses testbed radios, ensuring an accurate experimentation of real-world scenarios.

For the collection of system metrics such as memory and CPU usage, test times, and battery usage, we extend the measurement workflow with custom Python scripts tailored to AERPAW's environment. QGroundControl is employed to record time and battery usage during tests. Since AERPAW does not natively support metric logging, a Python script utilizing the `psutil` library was developed to monitor these parameters. The CPU Logger plays a critical role in tracking memory and CPU usage, ensuring the effective monitoring of computational resources. FL Logs are meticulously maintained to document evaluation and training times and results.

4.2 Hardware Specification

In Section 2.1, we introduced the preliminary capabilities of the AERPAW platform. Herein, we delve into the hardware capabilities of the system that we use in our experimentation. The Lake Wheeler site, the central area for drone experimentation within the AERPAW testbed, is equipped with five fixed nodes (LW1, LW2, LW3, LW4, LW5) [2]. These nodes are outfitted with NI Software Defined Radios, which offer versatile signal processing and communication functionalities. Additionally, the fixed nodes are integrated with Keysight RF Sensors for precise radio frequency measurement and analysis, and Fortem Drone Detection Radars for monitoring drone activities. For Internet of Things (IoT) applications, LoRa IoT equipment facilitates long-range, low-power communication. Moreover, an Ericsson 4G/5G Base Station supports advanced cellular communication experiments, providing the necessary bandwidth and low-latency connectivity essential for testing contemporary communication standards.

Overall, we found that the AERPAW testbed's hardware infrastructure provides a versatile and powerful platform capable of supporting a comprehensive range of wireless communication and drone experimentation. The integration of high-capacity network infrastructure, powerful CPUs, ample memory, and Intel UHD Graphics enables the testbed to meet the demands of our work that features both fundamental and applied research in advanced wireless communication and drone technologies.

Through our experimentation, we concluded that the combination of these advanced hardware components, alongside the flexible "Sandbox" environment equipped with the Keysight Propsim Channel Emulator and the use of industry-standard tools such as USRPs, OpenAirInterface, srsLTE, GNU Radio, and Matlab, significantly enhanced our capability to simulate, emulate, and test diverse wireless and drone-related scenarios.

4.3 Dataset exploration

This section examines the CICIOT[1] and IOTBOTNET[2] datasets, comparing their attack variety, class imbalance, feature selection methodologies, and the design of Deep Neural Network (DNN) models used in FL, tailored to each dataset. The analysis of the CICIOT and IOTBOTNET datasets reveals distinct differences in the variety and distribution of attacks. Table 1 presents a comparison of the attacks detected in both datasets. CICIOT includes a broader range of detected attacks compared to IOTBOTNET, providing a more diverse dataset for analysis. However, both datasets exhibit class imbalance, albeit in different ways. In IOTBOTNET, the Normal Traffic class is significantly underrepresented, which poses challenges for effective model training, as the scarcity of normal data can bias the learning process. Conversely, in the CICIOT dataset, categories such as Web-Based attacks and Spoofing are notably underrepresented. While this imbalance is less problematic in binary classification tasks, it remains a factor to consider in model training. Additionally, both datasets show a heavy over-representation of DDoS attacks, which can reduce the impact of other attack classes during training. To address these imbalances, Random Undersampling was employed to create a more balanced training set, though this method also led to a reduced sample size, particularly affecting the IOTBOTNET dataset.

CICIOT-2023		IOTBOTNET-2020	
Binary Category	**General Category**	**Binary Category**	**General Category**
Attack	DDoS	Anomaly	DDoS
	DoS		DoS
	Recon		Scan
	Web-Based		Theft
	Brute Force		-
	Spoofing		-
	Mirai		-
Benign	Benign	Normal	Normal

Table 1: **IOTBOTNET 2020 and CICIOT 2023 Attack Categories and Labels**

For the CICIOT dataset, feature selection was conducted using mutual information, decision trees, and recursive feature elimination algorithms, which are standard techniques within the sklearn library. The IOTBOTNET dataset, on the other hand, has been extensively studied, and the selected features are supported by prior research, including studies from [22] and [24]. The features were chosen based on their stable performance in machine learning models, as determined by running the aforementioned algorithms on a baseline model prior to the training step.

DNN models employed for FL were specifically designed for each dataset, as outlined in Table 2. Both models were trained using identical hyperparameters, following commonly accepted defaults. For

[1]https://www.unb.ca/cic/datasets/iotdataset-2023.html
[2]https://sites.google.com/view/iotbotnetdatset/home

Parameters	Values
Number of epochs for local model	5
Batch size	64
Optimizer for local model	Adam
Learning rate for local model	0.001
Loss function	Binary cross entropy loss
Federated number of rounds	1, 3
Betas	0.9, 0.999

Table 2: **Default hyperparameter settings for the baseline model**

Dataset	Defense Strategies	Poisoning Volume	Accuracy	Precision	Recall	Loss	Log-Cosh	Time Elapsed(s)
CICIOT	Baseline	0%	93.2%	95.4%	90.9%	21.9%	2.5%	194
	Differential Privacy	33%	76.5%	99.8%	74.4%	54.1%	6.4%	435
		66%	72.5%	99.8%	69.9%	67.7%	8.7%	514
	Adversarial Training	33%	85.8%	99.7%	84.6%	39.4%	4.9%	668
		66%	81.9%	99.7%	90.3%	52.0%	6.6%	645
	All Defenses	33%	91.1%	95.7%	94.5%	76.5%	11.2%	881
		66%	71.6%	99.8%	68.9%	71.7%	10.1%	890
IoTBotNet	Baseline	0%	91.2%	99.8%	82.6%	39.1%	5.1%	465
	Differential Privacy	33%	81.1%	100%	76.4%	55.6%	8.0%	552
		66%	26.4%	100%	8.1%	84.5%	10.3%	547
	Adversarial Training	33%	83.7%	91.8%	87.5%	56.6%	5.7%	500
		66%	80.3%	99.8%	75.5%	64.2%	7.0%	496
	All Defenses	33%	79.5%	79.9%	99.4%	57.9%	8.4%	489
		66%	67.7%	78.9%	81.4%	63.3%	9.8%	594

Table 3: **Feature noise attack metrics for CICIOT-FL-DNN and IoTBotNet-FL-DNN models under data poisoning scenarios**

the CICIOT dataset, the DNN model was structured sequentially with layers comprising 64, 32, 16, 8, 4, and 1 neuron(s), each incorporating batch normalization and a dropout layer with rate of 40%, except for the final layer. When the differential privacy defense strategy was applied, the 64-neuron layer was omitted, and the dropout rate was reduced by 10% to enhance training stability [24]. DNN model for the IOTBOTNET dataset was constructed with a sequential layer structure of 16, 8, 4, and 1 neuron(s), accompanied by batch normalization and 30% dropout rate. This model structure remained consistent due to its proven efficiency for this data.

4.4 Metrics & Model Performance Discussion

4.4.1 FL results. We employed accuracy, precision, and recall as key performance metrics to evaluate the FL model. Precision and recall are particularly crucial for assessing the model's capacity to manage the diverse data distributions encountered across different clients. These metrics often exhibit an inverse relationship, where improvements in one may lead to a decline in the other. By presenting both precision and recall, we provide a more nuanced and comprehensive analysis of the model's performance. While accuracy offers a general overview, it can be misleading in FL contexts characterized by significant class imbalances. Therefore, it is essential to supplement accuracy with precision and recall to ensure the model robustness across all clients, particularly when evaluating the effects of various data poisoning strategies and defense mechanisms on model performance.

The performance of various defense strategies under different attack scenarios was systematically evaluated over three rounds at the global model level. While individual nodes might be impacted by poisoned training data, the global model is expected to demonstrate robust behavior in the face of such attacks. We start by presenting the machine learning metrics in Table 3 for *feature noise attack* across both the datasets. The second column details the defense strategies employed, while the third column provides the volume

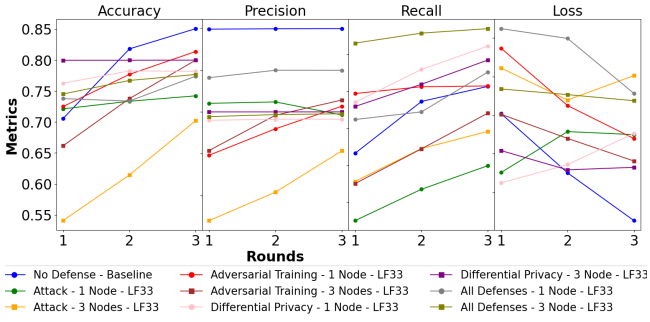

Figure 6: **Time-series for Global Model Evaluation Metrics over 3 Rounds with varying defenses against Label Flipping 33 percent volume attack**

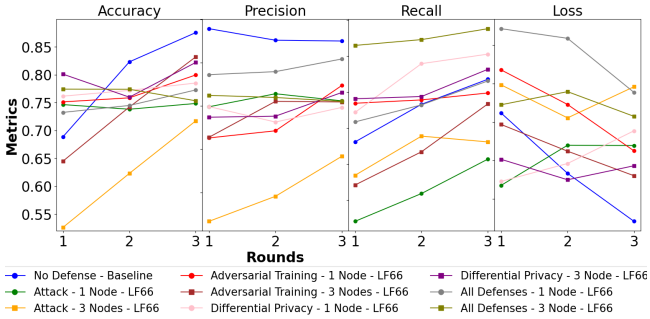

Figure 7: **Time-series for Global Model Evaluation Metrics over 3 Rounds with varying defenses against Label Flipping 66 percent volume attack**

of poisoning applied. The final column displays the wall clock time, which reflects the computational cost associated with each scenario.

We begin by showing the baseline values (no attack, no defense) for anomaly detection for the benign use case. As the effect of volume of poisoning data doubles from 33% to 66% for each defense strategy, the effectiveness of defenses in recovering accuracy slightly decreases with the drop of <5% in most cases. The "all defenses" strategy where both differential privacy and adversarial training are applied is not always the best due to poor accuracy (4th column) at higher computational cost (last column) and hence is unnecessary. *In case of feature noise attack, adversarial training (for both datasets) seems to be the most effective defense irrespective of the poisoning levels.* Although the defense results are not all comprehensive or perfect w.r.t. baseline values, they demonstrate effectiveness as shown by the other FL metrics (particularly precision), supporting their overall value in mitigating attacks.

We present the time-series results for *label-flipping attack* for IoTBotNet dataset with two different volumes of poisoning: 33% and 66% in Figures 6 and 7, respectively. Results are shown for both 1 and 3 client nodes. Although we focus on one dataset for brevity, the findings are broadly applicable. Adversarial training proves to be the most effective defense across both poisoning levels in terms of accuracy and stable progression. However, these results are less optimistic than those for feature noise due to the targeted nature of the label-flipping attack. This attack directly manipulates critical labels, making it harder for the model to maintain accuracy,

FL-DNN Model Scenario	Max CPU Usage	AVG CPU Usage	Max Memory Usage	Energy Usage	Time Elapsed (s)
Stationary Drones (No FLYPAW)					
Baseline	127%	98%	0.3%	0%	172
Baseline + LF33	134%	101%	0.4%	0%	221
DP + LF33	147%	119%	0.6%	0%	471
AT + LF33	121%	97%	0.7%	0%	536
AD + LF33	152%	116%	1.0%	0%	839
Baseline + FL-YOLO	217%	183%	1.0%	0%	10041
Active Drones (FLYPAW)					
Baseline	207%	150%	0.6%	7%	204
Baseline + LF33	216%	157%	0.4%	7%	232
DP + LF33	225%	181%	0.7%	7%	476
AT + LF33	204%	154%	0.6%	7%	531
AD + LF33	233%	179%	1.0%	7%	857
Baseline + FL-YOLO	307%	253%	1.0%	7%	10124

Table 4: **System metrics for different FL-DNN model scenarios for Stationary Drones and Active Drones.**

especially when the training data has not been exposed to such manipulations before.

4.4.2 Usage results. Table 4 details the system metrics for different FL-DNN model scenarios for stationary drones and active drones. The results indicated that implementing defense mechanisms, especially in active drone scenarios, incurs significant computational costs. For stationary drones not equipped with the FlyPaw system, deploying all defenses (FL-DNN-AD + LF33) led to a peak CPU usage of 152% and an average CPU usage of 116%, with mission durations extending up to 839 seconds. In contrast, when active drones operated with the FlyPaw system, the computational load escalated sharply, with maximum CPU usage reaching 233% and average CPU usage rising to 179%, extending mission times to as long as 857 seconds. This marked increase in computational demand underscores the resource-intensive nature of these defense strategies, particularly when applied in real-time operational contexts.

4.4.3 Object Detection results. The integration of YOLOv8 for real-time object detection further increased these demands. When defenses were applied alongside YOLOv8 training (FL-DNN-Baseline + FL-YOLO Training), the computational requirements spiked, with stationary drones experiencing up to 217% maximum CPU usage and 10124 seconds of mission time, while active drones saw a staggering 253% maximum CPU usage and the same extended mission duration of 10124 seconds. These results illustrate the challenges of maintaining both security and operational efficiency under such heavy workloads, especially in scenarios where drones must make real-time decisions with limited computational resources.

The study measured energy usage during the experiments but found it remained constant due to the emulation setup on the AERPAW testbed, which did not factor in the impact of computational consumption on battery life. This creates an unrealistic representation of real-world scenarios where increased computational loads, such as those from implementing defenses, would likely lead to higher energy consumption, reducing battery life and limiting mission duration. This limitation underscores the need for more realistic energy modeling in future experiments to accurately evaluate the practicality of resource-intensive tasks in drone operations.

4.4.4 Network results. The analysis of network measurements shown in Table 5 allows us to observe a moderate increase in resource demands when advanced defenses are implemented. Such

FL-DNN Model Scenario	AVG Bitrate (Gbit/sec)	AVG RTT (s)	Bandwidth (GB/sec)	Time Elapsed (s)
Baseline	8.65	0.939	1.08	204
Baseline + FL-YOLO	6.02	0.939	0.75	232
Baseline + LF33	8.65	0.939	1.08	476
DP + LF33	8.04	0.939	1.01	531
AT + LF33	8.35	0.939	1.04	857
AD + LF33	9.33	0.939	1.17	10124

Table 5: **Network metrics for different FL-DNN model scenarios for Active Drones.**

an increase is expected given that these defenses are applied at the model level rather than at the network layer. For instance, the average bitrate increased from 8.65 Gbits/sec in the baseline scenario to 9.33 Gbits/sec when both anomaly detection and label flipping defenses were applied. Similarly, bandwidth usage saw a moderate rise from 1.08 GBytes/sec to 1.17 GBytes/sec under the same conditions. While these increases are notable, they are not drastic. However, the mission time experienced a significant extension, jumping from 204 seconds in the baseline scenario to 10,124 seconds with the advanced defenses in place. This suggests that while network demands moderately increased, the impact on operational duration was much more substantial, reflecting the higher computational and communication overhead introduced by the defenses. The moderate increase in network resource usage is aligned with expectations, as the defenses do not directly alter the network protocols but rather add complexity to the model processing, leading to extended mission times.

Overall, these results underscore the importance of selecting appropriate defense mechanisms based on the specific attack scenario, poisoning volume, and dataset used. While Differential Privacy provides a strong defense against label-flipping attacks. However, it was not able to achieve expected levels with no defense, showing that it is less effective against high-volume feature noise attacks, where Adversarial Training is better. The combined defense approach may not always work due to its higher computational costs and occasional performance trade-offs, especially as observed in the CICIOT dataset, where the time elapsed was significantly higher.

5 Conclusion

The growing use of collaborative drones demands real-time processing of large volumes of sensitive data for immediate decision-making. However, the decentralized nature of these systems makes them vulnerable to security threats, highlighting the need for advanced threat intelligence frameworks. Our work addresses this by developing a custom FL model using a Deep Neural Network for anomaly detection within a drone swarm, deployed on the AERPAW testbed. We demonstrated effective threat detection while preserving data privacy and validated the model's resilience against grey-box data poisoning attacks. To mitigate these threats, we optimized defense solutions with adversarial training. Additionally, we showcased the practical application of our FL model by running it alongside computer vision tasks during in-flight drone operations, ensuring its viability in real-time scenarios. A custom measurement workflow for AERPAW was also developed to balance the trade-offs in FL tasks.

In the future, our framework can be enhanced by exploring advanced defense techniques, such as generative adversarial networks, to further strengthen the model's resilience against sophisticated attacks. Additionally, we propose extending our FL approach to integrate IoT networks, specifically focusing on smart home environments. By designing an FL-integrated platform for IoT devices in smart homes. This platform would allow IoT devices, such as smart cameras and sensors, to collaboratively train a global model without sharing sensitive data. Future work could include optimizing communication efficiency between IoT devices and the cloud, and ensuring robust responses for critical security events.

Acknowledgments

This material is based upon work supported in part by the U.S. Army Corps of Engineers, Engineering Research and Development Center—Information Technology Laboratory (ERDC-ITL) under Contract W912HZ23C0041, and the National Science Foundation (NSF) under Award No. OAC-2018074. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the U.S. Government or agency thereof.

References

[1] Mohamed Abdelkader, Samet Güler, Hassan Jaleel, and Jeff S Shamma. 2021. Aerial swarms: Recent applications and challenges. *Current robotics reports* 2 (2021), 309–320.

[2] AERPAW. 2024. AERPAW User Manual: Equipment Information. https://sites.google.com/ncsu.edu/aerpaw-wiki/aerpaw-user-manual/1-aerpaw-overview/1-4-equipment-information. Accessed: 2024-08-05.

[3] AERPAW. 2024. iPerf - Traffic Generation Software. https://sites.google.com/ncsu.edu/aerpaw-wiki/aerpaw-user-manual/4-sample-experiments-repository/4-3-traffic-generation-software/4-3-2-iperf Accessed: 2024-07-30.

[4] Naveed Akhtar, Ajmal Mian, Navid Kardan, and Mubarak Shah. 2021. Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey. *IEEE Access* 9 (2021), 155161–155196. https://doi.org/10.1109/ACCESS.2021.3127960

[5] Syreen Banabilah, Moayad Aloqaily, Eitaa Alsayed, Nida Malik, and Yaser Jararweh. 2022. Federated learning review: Fundamentals, enabling technologies, and future applications. *Information processing & management* 59, 6 (2022), 103061.

[6] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. 2020. Flower: A Friendly Federated Learning Research Framework. *CoRR* abs/2007.14390 (2020). arXiv:2007.14390 https://arxiv.org/abs/2007.14390

[7] Nan Cheng, Shen Wu, Xiucheng Wang, Zhisheng Yin, Changle Li, Wen Chen, and Fangjiong Chen. 2023. AI for UAV-assisted IoT applications: A comprehensive review. *IEEE Internet of Things Journal* 10, 16 (2023), 14438–14461.

[8] A. Grote, E. Lyons, K. Thareja, G. Papadimitriou, E. Deelman, A. Mandal, P. Calyam, and M. Zink. 2023. FlyPaw: Optimized Route Planning for Scientific UAVMissions. In *2023 IEEE 19th International Conference on e-Science (e-Science)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–10. https://doi.org/10.1109/e-Science58273.2023.10254831

[9] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[10] K Naveen Kumar, C Krishna Mohan, and Linga Reddy Cenkeramaddi. 2024. The Impact of Adversarial Attacks on Federated Learning: A Survey. *IEEE transactions on pattern analysis and machine intelligence* 46 (01 2024), 1–20. https://doi.org/10.1109/tpami.2023.3322785

[11] K. Naveen Kumar, C. Vishnu, Reshmi Mitra, and C. Krishna Mohan. 2020. Black-box Adversarial Attacks in Autonomous Vehicle Technology. , 7 pages. https://doi.org/10.1109/AIPR50011.2020.9425267

[12] Dongcheng Li et al. 2021. Detection and Mitigation of Label-Flipping Attacks in Federated Learning Systems with KPCA and K-Means. In *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*. 551–559. https://doi.org/10.1109/DSA52907.2021.00081

[13] Vuk Marojevic et al. 2020. Advanced wireless for unmanned aerial systems: 5G standardization, research challenges, and AERPAw architecture. *IEEE Vehicular Technology Magazine* 15, 2 (2020), 22–30.

[14] Yassine Mekdad, Ahmet Aris, Leonardo Babun, Abdeslam El Fergougui, Mauro Conti, Riccardo Lazzeretti, and A. Selcuk Uluagac. 2023. A survey on security and privacy issues of UAVs. *Computer Networks* 224 (2023), 109626. https://doi.org/10.1016/j.comnet.2023.109626

[15] Alicia Esquivel Morel et al. 2020. Enhancing network-edge connectivity and computation security in drone video analytics. In *2020 IEEE Applied Imagery*

Pattern Recognition Workshop (AIPR).

[16] Mohammad Mozaffari et al. 2019. A tutorial on UAVs for wireless networks: Applications, challenges, and open problems. *IEEE communications surveys & tutorials* 21, 3 (2019), 2334–2360.

[17] Thuy Dung Nguyen et al. 2024. Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions. *Engineering Applications of Artificial Intelligence* 127 (2024), 107166.

[18] Ashwin Panicker, Ozgur Ozdemir, Mihail L Sichitiu, Ismail Guvenc, Rudra Dutta, Vuk Marojevic, and Brian Floyd. 2021. AERPAW Emulation Overview and Preliminary Performance Evaluation. *Computer Networks* 194 (2021), 108083.

[19] Arnau Rovira-Sugranes, Abolfazl Razi, Fatemeh Afghah, and Jacob Chakareski. 2022. A review of AI-enabled routing protocols for UAV networks: Trends, challenges, and future outlook. *Ad Hoc Networks* 130 (2022), 102790.

[20] Siva Raja Sindiramutty et al. 2024. *Data Security and Privacy Concerns in Drone Operations.* 236–290. https://doi.org/10.4018/979-8-3693-0774-8.ch010

[21] Khaled Telli, Okba Kraa, Yassine Himeur, Abdelmalik Ouamane, Mohamed Boumehraz, Shadi Atalla, and Wathiq Mansoor. 2023. A Comprehensive Review of Recent Research Trends on Unmanned Aerial Vehicles (UAVs). *Systems* 11, 8 (2023). https://doi.org/10.3390/systems11080400

[22] Imtiaz Ullah and Qusay H Mahmoud. 2020. A Technique for Generating a Botnet Dataset for Anomalous Activity Detection in IoT Networks. (10 2020). https://doi.org/10.1109/smc42975.2020.9283220

[23] Ultralytics. 2023. YOLOv8. https://github.com/ultralytics/ultralytics.

[24] Xiaofeng Wang et al. 2023. Federated deep learning for anomaly detection in the internet of things. *Computers & Electrical Engineering* 108 (05 2023), 108651–108651. https://doi.org/10.1016/j.compeleceng.2023.108651

[25] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2022. FedAttack: Effective and covert poisoning attack on federated recommendation via hard sampling. In *Proceedings of the 28th ACM SIGKDD.* 4164–4172.

[26] Jean-Paul Yaacoub, Hassan Noura, Ola Salman, and Ali Chehab. 2020. Security analysis of drones systems: Attacks, limitations, and recommendations. *Internet of Things* 11 (2020), 100218. https://doi.org/10.1016/j.iot.2020.100218

[27] Evşen Yanmaz, Saeed Yahyanejad, Bernhard Rinner, Hermann Hellwagner, and Christian Bettstetter. 2018. Drone networks: Communications, coordination, and sensing. *Ad Hoc Networks* 68 (2018), 1–15.

[28] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. 2021. Detection and Tracking Meet Drones Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 1–1. https://doi.org/10.1109/TPAMI.2021.3119563