future internet

MDPI

Article

DFedForest++: A Novel Privacy-Enhanced Framework for Integrating Cyber Threat Intelligence in IDS Using Federated Learning

Md. Moradul Siddique [1,2], Syed Md. Galib [1,*], Md. Nasim Adnan [1,*] and Mohammad Nowsin Amin Sheikh [1,3]

[1] Department Computer Science and Engineering, Jashore University of Science and Technology (JUST), Jashore 7408, Bangladesh; moradul@just.edu.bd (M.M.S.); n.amin@just.edu.bd (M.N.A.S.)
[2] Department of Computer Science and Engineering, University of Information Technology and Sciences (UITS), Dhaka 1212, Bangladesh
[3] Department of Computer Science and Engineering, Yuan Ze University, Taoyuan 32003, Taiwan
* Correspondence: galib.cse@just.edu.bd (S.M.G.); nasim.adnan@just.edu.bd (M.N.A.)

Abstract

The sophistication of cyber attacks and privacy issues related to data sharing is improving and requires a decentralized approach. Conventional centralized approaches to IDS pose a threat to the privacy of data and data sovereignty. Contrarily, federated learning enables several clients to learn simultaneously without sharing their sensitive information, which is one of the most promising solutions to studying cyber threats in real time. This framework also adds value to IDS by using CTI, which is incorporated into the training process to make it more accurate in its detection while still maintaining privacy. Each client uses the local model, which is a random forest model that is trained on local datasets without sharing the raw data. Multiple aggregation methods, such as FedAvg, FedOPT, FedProx, and FedXGBoost, are then used to combine the local models into a global model. These techniques are judged with regard to accuracy and Cohen's Kappa Score. The performance of various models in the NF-UNSW-NB15-v2 dataset experiments was tested. The local model took a value of 0.9941–0.9934 with Kappa scores of 0.8336–0.8088, showing strong performance in different configurations. The FedXGBoost aggregated global model was best in terms of its highest accuracy of 99.22 (Kappa score of 0.8417). More experiments were done on the DFedForest and DFedForest++ models. DFedForest++, incorporating diversity in local models alongside validation accuracy, achieved 99.76% accuracy, surpassing DFedForest (with 71% accuracy in local models). This framework operationalizes CTI through feature augmentation—appending three CTI-derived features (is_known_malicious_ip, is_suspicious_port, and ttp_match_score from MITRE ATT&CK v14 and AlienVault OTX) to each NetFlow record locally at each client before federated training begins. These results highlight the advantages of federated learning in providing collaborative, privacy-preserving solutions for cyber threat detection and emphasize the potential of CTI integration for improving the accuracy and robustness of IDS models across decentralized environments.

Keywords: cyber threat intelligence; federated learning; machine learning; random forest; Federated Forest; intrusion detection system

Academic Editor: Paolo Bellavista

Received: 6 February 2026
Revised: 1 March 2026
Accepted: 13 March 2026
Published: 23 March 2026

1. Introduction

The rapid and continuous development of interconnected digital environments has significantly transformed many facets of modern life, including the way we communi-

cate, work, and conduct business. However, this evolution has also greatly expanded the attack surface for malicious actors who are continually looking for opportunities to exploit weaknesses and vulnerabilities present in both hardware and software systems. Complementing these developments, there has been a concerning growth in frequency, complexity, and seriousness of cyber attacks on organizations in the global community due to the reduction in the number of connected devices. Following this unfortunate reality, as a direct outcome, companies are currently spending huge sums of money on different types of cybersecurity defense mechanisms. The complexity of the critical infrastructure systems and their interconnection raises the possible consequences of successful cyber attacks, which may cause disastrous outcomes in different sectors. Storing and effectively sharing information on new threats that are detected is completely necessary for the prevention of spreading attacks and ensuring that such incidents are not experienced in the future. We can strengthen our collective defense mechanisms and improve our defenses against these constantly changing threats by instilling a culture of information sharing and collaboration among cybersecurity professionals, further fortifying our collective defensive mechanisms [1].

In an attempt to extend their proactive threat-detection features, organizations invest a lot of resources in the development and maintenance of their own special-purpose threat-detection systems. It is worth observing that the majority of these systems that are in place are inclined to provide protection to the investing organization itself. They do not usually share critical threat information with their fellow colleagues, and this may make such organizations especially susceptible to any newly found threats that may lead to their downfall [2]. In order to overcome this urgent problem, there is an increased demand for collaborative threat-detection systems. These new systems enable organizations to exchange important data on threats with each other, which results in a more integrated response to recently discovered threats and, therefore, improves the overall cybersecurity situation of all participating organizations. Through the inculcation of collaboration and sharing of information in the cybersecurity arena, organizations are able to enhance their resiliency to the constantly changing threats that loom in the online environment in a remarkable manner [3].

Although collaborative solutions have proven to have better performance in detection capabilities and in lessening the time to neutralize threats, sharing raw data that holds sensitive information is too risky. The sharing of data would expose an environment's properties and make the systems under inquiry more vulnerable and easily exploited. The ability to share data in such a way that the confidentiality of the shared data is not violated, and yet allows the collaborative processing of the data, is in serious demand [4]. Federated learning (FL) has become an interesting approach in this respect in order to solve the problem of collaborative learning without sharing data. In FL, common training of a shared model is done by interacting peers sharing a model, coordinated on a central server, without providing local data. Sharing only the model parameters, which do not reveal information about the data, is a way of protecting the privacy of the training data. In addition, FL is resource-constrained, and therefore it is especially applicable to collaborative solutions in network intrusion detection systems (NIDS) [5].

Though FL and Federated Forest have been utilized in cybersecurity and IDS in prior works, there are various crucial shortfalls that have not been tackled. The current FL-based IDS approaches (e.g., FedNIDS and FedMADE) mainly consider one strategy of aggregation, and they are not systematic in comparing different aggregation algorithms (e.g., Fe-dAvg, FedOPT, FedProx, and FedXGBoost) in a unified framework. Furthermore, prior Federated Forest implementations rely solely on local validation accuracy for client selection, which can lead to the aggregation of locally over-fitted models that fail to generalize globally.

These works also lack integration of cyber threat intelligence (CTI) into the federated training pipeline, leaving a gap in contextual threat awareness. The present paper addresses these gaps through three key contributions: (1) a systematic comparison of five federated aggregation strategies under a unified experimental protocol; (2) the novel DFedForest++ framework, which introduces a model diversity criterion alongside validation accuracy for client selection, substantially improving global model robustness; and (3) the explicit integration of CTI-informed feature engineering to enhance the relevance and quality of intrusion-detection training data within a privacy-preserving FL architecture.

The objectives of the research framework are to improve threat-detection accuracy within IDSs by leveraging cyber threat intelligence (CTI) and federated learning (FL), enabling real-time detection and response to cyber threats, and reducing the potential impact of attacks. In addition, we developed a framework named DFedForest++, which is an improved decentralized Federated Forest. This work will assess the ability of its methodologies to conceal shared information derived from using confidential data models while providing competitive detection results obtained through respectable models and security surveillance platforms that undergo extensive training practices on user organizations' datasets.

The remainder of this paper is structured as follows: Section 2 examines existing research on IDS using ML, FL applications in cybersecurity, and the use of random forest in IDS. It identifies gaps in the literature, such as the lack of privacy-preserving approaches in IDS, and explains how this research addresses these gaps by integrating FL with traditional ML models. Section 3 details the research process, starting with data acquisition using the NF-UNSW-NB15-v2 dataset. The local model is developed using random forest with bootstrap, while the global model is built using a different algorithm in a federated learning framework. Section 4 describes the experimental setup, including hardware and software environments. The conclusions of both local and global models are contrasted, and the merits of the two models are noted. Lastly, Section 5 provides a summary of the main findings of the study, and it can be stated that the combination of FL and traditional models of ML, such as the random forest, offers a good opportunity to improve IDS.

2. Background

Governments, organizations, and individuals have come to realize that cybersecurity has become a critical issue in the digital era. The broad connectivity of the systems to enable information sharing has exposed the systems to numerous cyber attacks. Such attacks are very dangerous, as they may lead to the malfunctioning of critical systems, loss of money, and sensitive information leakage. The constant emergence of new types of cyber attacks necessitates the timely detection of potential threats. Threat-detection systems, which analyze system activity to identify abnormal behavior, are vital for ensuring system security [6].

Machine learning (ML) techniques have recently gained popularity for developing cyber threat-detection systems due to their ability to learn from historical data and automatically adapt to new types of threats. However, the effectiveness of ML models relies on the availability of sufficient high-quality training data [7]. Data scarcity, especially concerning newly discovered threats, can significantly degrade model performance. Moreover, many organizations are unwilling or unable to share sensitive data due to strict privacy regulations and security policies. The consequence of data privacy concerns is that potential data-holding parties may prefer inaction over developing suboptimal solutions without collaboration, resulting in no model being trained.

Prior to the emergence of privacy regulations, collaborative learning approaches that involved raw data sharing were commonly used to improve model reliability and scalabil-

ity. The most widely adopted approach was centralized learning, where model training occurred in a centralized computing infrastructure and all learning parties shared their local data with a trusted third party [8]. However, this approach has many drawbacks, including trust issues, data ecosystem incompatibility, high operating costs, and vulnerabilities to single points of failure. In response, the research community has proposed several collaborative learning (CL) solutions with good reliability and scalability but strict data protection policies. Among them, federated learning (FL) has recently gained popularity as a promising tool to address the challenges of information exchange among different parties and sensitive data exploitation [9]. An intrusion detection system (IDS) can be considered a special type of threat-detection system that focuses solely on detecting cyber threats [10].

2.1. Cyber Threat Intelligence

Cyber threat intelligence (CTI) refers to a set of data regarding security threats, threat actors, exploits, malware, vulnerabilities, and indicators of compromise that can help organizations, governments, and individuals in decision-making for proactive cybersecurity defense [11]. The growing cyberspace and increasing sophistication of cyber attacks have forced organizations to adopt a proactive approach to cybersecurity instead of having a reactive approach. The reactive approach relies on log analysis and post-attack investigations, which lead to time-consuming damage control restoration and financial loss. A proactive approach focuses on predicting and mitigating attacks before they occur by employing security mechanisms like intrusion detection systems, firewalls, and vulnerability assessment [12].

CTI can provide a broad range of additional cybersecurity measures, such as sophisticated intrusion detection systems, advanced firewalls, and thorough vulnerability assessments, all while relying on extensive knowledge concerning both current and future cyber threats in the ever-evolving digital landscape [13]. As a result, organizations can make good use of CTI to improve their overall cybersecurity stance to a large extent, which guarantees improved protection against possible attacks. In addition, the threat intelligence between different organizations may benefit these organizations significantly as they are able to collectively address the threats they are prone to in this complex world. It is, however, worth noting that the dissemination of cyber threat intelligence is a highly sensitive issue because it has the risk of unwittingly revealing the vulnerabilities and lack of security of the involved organizations through information sharing [14]. This concern has understandably resulted in a reluctance among many organizations to engage in the sharing of CTI. To effectively address this pressing issue, several innovative privacy-preserving techniques for cyber threat intelligence sharing have been proposed and discussed in the literature, aiming to facilitate secure information exchange while protecting sensitive data in the detection system [15].

In this framework, CTI is operationalized through three specific data elements drawn from public threat intelligence sources (MITRE ATT&CK v14, AlienVault OTX, and abuse.ch Feodo Tracker): (1) Indicators of compromise (IoCs)—known malicious IP ad-dresses, suspicious port numbers, and flagged protocol signatures; (2) tactics, tech-niques, and procedures (TTPs)—MITRE ATT&CK technique IDs (e.g., T1046: Network Service Discovery, T1071: Application Layer Protocol) mapped to corresponding NetFlow feature patterns; and (3) CTI-derived feature augmentation—three binary or scalar features are appended to each NetFlow flow record: (a) is_known_malicious_ip (1 if source/destination IP matches a CTI IoC feed, 0 otherwise), (b) is_suspicious_port (1 if destination port appears in CTI-flagged port lists), and (c) ttp_match_score (a weighted count of matched ATT&CK technique indicators observed in the flow). These CTI lookups are performed locally at each federated

client before training, ensuring that no CTI feed data or raw traffic records are shared across the federation.

2.2. Intrusion Detection System

An intrusion detection system (IDS) is a key component in cyberspace security. An IDS detects unauthorized activities in the monitored system by inspecting and analyzing system audit records [16]. The IDS can be categorized by different criteria, such as detection approach, system location, and data sources. According to the detection approach, the IDS can be either a misuse detection system or an anomaly detection system. A misuse detection system inspects the audit records for known intrusive activities by matching patterns against a set of pre-defined rules. In addition, a misuse detection system can effectively detect well-known intrusive activities, but it is not capable of detecting new intrusive activities that have not been seen before. On the other hand, an anomaly detection system builds a model of normal activities in the inspected system. Any deviation from the normal model will be flagged as a potential intrusion [17].

Anomaly detection systems have the potential to detect new intrusive activities, but they may generate a relatively high false-positive rate. Each networked system can deploy its own local IDS. Anomaly detection-based IDSs are often trained on the local data of the inspected system. Therefore, the automated design and training of an anomaly detection-based IDS for a specific system require a lot of time and effort. A possible solution to this problem is to share the knowledge of the educated models from the IDSs in the inspected systems. However, information sharing may expose the vulnerabilities and privacy of the inspected systems [18].

2.3. Federated Learning

Federated learning is a novel machine-learning framework for training and building data-driven models on distributed datasets. This approach enables the exploitation of the computational power of each involved participant without taking over the control of data from local storage. More precisely, the learning model is transferred to and iteratively updated on all clients, and the updates are further aggregated in a central server responsible for defining the initial model. Finally, the updated model is passed to each client, which repeats the operation [19]. This shift in learning paradigms is particularly well studied in applications involving many companies, using mobile phones and edge devices, but also for healthcare, IoT, and data science. This paradigm is particularly suitable for cybersecurity, given the large number of security entities that are interested in learning and using the models without the ability to share the data with a central entity [20].

Federated learning is based on a client–server architecture where a central server manages the model parameter synchronization and variable aggregation. Clients are user devices that locally compute gradient updates on a subset of local data. The local model update is computed using a stabilizing optimizer. After convergence, the client uploads its update to the server, which, in turn, updates the federated model using a specific aggregation function [21].

At present, the open question of applying FL to the cybersecurity field is how to integrate cyber threat intelligence (CTI) into intrusion detection. The lack of CTI that can be applied to specific industrial environments will lead to poor generalization of the intrusion detection model [22]. To solve these problems, this paper first establishes a threat map using public CTI and applies this insight to generate the intrusion-detection training set corresponding to the industry environment [23]. Furthermore, a binary classification model is generated for the created intrusion-detection training dataset through incremental machine learning algorithms [24]. Finally, the created and trained model is embedded

into the FL framework to achieve the detection of unknown cyber attacks in an encrypted environment when it is applied in the industry environment.

It is important to acknowledge that parameter-only sharing in FL does not by itself constitute a formal privacy guarantee. Modern attacks such as model inversion and membership inference can recover sensitive information from shared model parameters—including from serialized decision tree structures (via leaf statistics) and gradient vectors (via reconstruction attacks). To address this, the present framework applies output perturbation with calibrated Laplace noise to leaf class distributions before transmission, providing ($\varepsilon = 1.0, 0$) differential privacy at the tree level. Specifically, Laplace noise with scale $\Delta f/\varepsilon$ is added to each leaf vote count, where Δf is the global sensitivity of the leaf statistic. This defense limits the ability of an honest-but-curious aggregation server to infer individual training samples from the transmitted tree updates. Secure aggregation via Shamir Secret Sharing and homomorphic encryption over gradient statistics are identified as planned future enhancements to provide stronger, cryptographic privacy guarantees.

The FL landscape has been greatly broadened in recent developments. Moshawrab et al. [25] introduced inference attack-resistant practical private aggregation protocols of FL, which show that it is possible to guarantee secure aggregation without significant loss of accuracy. FedSSuper proposed a semi-supervised federated model, which utilizes unlabeled data through multiple clients, eliminating the need to use fully labeled datasets in distributed IDS applications [26]. Liu et al. came up with privacy-saving, vertical federated learning models, in which various feature areas exist among clients, and yet, prominent privacy is guaranteed by homomorphic encryption [27]. Also, there is now Flower (FLWR), an open-source platform that supports heterogeneous FL experiments, which can fairly benchmark aggregation algorithms, including FedAvg, FedOPT, and FedProx, over non-IID data distributions [28]. The overall effect of these recent works is the increased maturity of FL to security-sensitive distributed environments, and is directly applicable to the design decisions of the current research.

2.4. Federated Forest

In this regard, a Federated Forest uses distributed learning models in multiple network settings, whereby various organizations or security infrastructures can identify cyber threats in real-time without necessarily exposing their raw data. The individual participating nodes train a local model using their own security logs, attack patterns, or user behavior data, and only send encrypted model updates to a central aggregator. This method allows a crowd intelligence system, which increases the ability to detect threats without endangering data [29].

In order to add additional security and trust to a Federated Forest model, one can use blockchain technology as a decentralized registry of model updates, trust management, and secure exchange of data. The blockchain guarantees the integrity of updates to the model, which prevents adversaries from introducing malicious data or manipulating model parameters in the course of federated learning. Through smart contracts, security policies can be applied to organizations, and verification of contribution integrity and incentive mechanisms can be used to award participation [30]. This decentralized validation process eliminates adversarial attack risks in federated learning, including model poisoning attacks and data inference attacks.

Sharing of threat intelligence between various security infrastructures is another important use of a Federated Forest in cybersecurity. Conventional threat intelligence-sharing systems tend to demand centralized data repositories, and this poses threats in the areas of data privacy, compliance, and latency. Nonetheless, federated learning solutions can allow collaboration between security organizations in real-time without exposing

raw data, which will enable quicker recognition of malware, phishing campaigns, and sophisticated persistent threats (APTs) [31]. A Federated Forest, by providing real-time anomaly detection algorithms, provides an organization with an opportunity to respond to cyber threats proactively and maintain the confidentiality and sovereignty of its security data. This model may be especially helpful in high-stakes infrastructures, like finance, healthcare, and government sectors, where cyber attacks are highly targeted and data privacy is the main priority.

The framework presented in this paper has some significant advantages over the current Federated Forest strategies. Our DFedForest++ solves this by introducing a dual-criterion selection mechanism that is also conscious of diversity in models and makes sure that the combined world of forests represents a wider and more reflective decision space. Moreover, unlike the previous work on Federated Forests, which views all the involved clients as a homogenous contributor to the training, this framework incorporates CTI-informed feature engineering to enhance local training data with information about the threat context, which enhances the semantic relevance of learned features. In contrast to single-aggregator FL-IDS experiments, a proper comparative analysis of five different aggregation algorithms (FedAvg, Weighted FedAvg, FedOPT, FedProx, and FedXGBoost) with the same experimental conditions is also presented in this paper, which allows one to select the most efficient aggregation strategy to apply to tree-based federated intrusion detection in a principled way.

3. Literature Study

An intrusion detection system (IDS) is an essential part of cybersecurity and is responsible for tracking and detecting network or system traffic to identify malicious intended actions or policy breaches. The effectiveness of IDS has been improved over the years with an extensive amount of research being carried out. Another study by Ahmed et al. [32] investigated signature-based intrusion detection with the help of ML and DL models. The experiment was carried out on a dataset that held known attack signatures and trained the models based on algorithms such as random forests and deep neural networks. The findings showed that by combining ML and DL with signature-based algorithms, the detection of known threats and the minimization of false positives were improved. Nevertheless, the research recognized the fact that zero-day attacks are difficult to detect and stated that it is necessary to constantly update the signature database.

With the constantly growing popularity of cloud computing and the Internet of Things (IoT), scholars have paid more attention to the issue of applying IDS to these areas. Luo et al. [33] analyzed the issue of the implementation of IDS in cloud-based computing resources, which is distributed and dynamic. In their research, they have highlighted the importance of lightweight IDS that may be scaled with the scalability of clouds. They also talked about federated learning integration to mitigate privacy issues in multi-tenant settings. Walsman et al. [34] investigated distributed IDS architectures in the context of IoT settings by emphasizing energy saving and better detection precision in resource-constrained devices. In these studies, it was found that even though cloud and IoT IDS are essential to ensure the security of modern infrastructures, challenges like high latency, resource limitation, and the security of distributed data are major concerns.

A study by Kaushik et al. [35] also described a powerful machine learning-based IDS that utilizes the technique of feature selection to improve the detection performance. The researchers followed a complex system of feature selection methods and machine learning algorithms with the aim of determining the most useful features in intrusion detection. They used the approach of experimenting with other techniques of feature selection to identify their influence on the accuracy and efficiency of the IDS. A hybrid feature selection

approach was the most effective and yielded the best results in terms of detection rates and false positives. The research emphasized the role of feature selection in the creation of an effective IDS and also mentioned that additional studies were required to make feature selection techniques more effective in various kinds of network setups.

Numerous IDS models, especially the models founded on DL, have problems in their scaling to address large-scale network data, in addition to those that need real-time performance. Quality and diversity of training datasets are vital to the effectiveness of the IDS models [36]. Adversarial attacks pose a threat to the currently used IDS models, as bad actors can adjust the input data to avoid detection. It is important to come up with strong models that are resistant to these attacks. Although XAI increases model interpretability, XAI and performance frequently trade off with one another. More research is required to come up with the means that will maintain the high rate of detection while providing a clear explanation.

The classic machine learning models of cybersecurity are based on centralized data, which is highly privacy-threatening, maintains data sovereignty, and poses security threats. FL resolves such issues by enabling every member of the system, e.g., organizations, cloud providers, or IoT networks, to train models on their local machines and transmit only encrypted model updates instead of exposing sensitive data.

The main benefit of FL in cybersecurity is that it offers to enhance threat-detection accuracy and maintain the privacy of data. FL may be used in intrusion detection systems (IDSs) to provide distributed environments, including an enterprise network and IoT ecosystem, with real-time detection of cyber threats. Rather than consolidating network logs in a central repository, every participant learns an IDS model on the local traffic characteristics and only exchanges encrypted model parameters, minimizing the risk of data leakage and compliance issues [37]. Also, FL improves malware detection by enabling different organizations to share information on new threats without disclosing proprietary or sensitive information [38].

The other important use of FL in cybersecurity is cyber threat intelligence (CTI) sharing. Historically, sharing in CTI is hindered by barriers in the form of distrust, data confidentiality, and interoperability. FL can defeat these challenges by facilitating secure cooperation amongst organizations to identify and react to cyber threats more efficiently. FL is becoming more and more integrated with blockchain technology, providing it with the ability to guarantee tamper-proof model updates, verifiable trust mechanisms, and decentralized access control, contributing to its increased security [31].

In spite of the benefits, FL with regard to cybersecurity poses a number of challenges. Federated models may be compromised by adversarial attacks, including model poisoning and inference attacks. The attackers are able to manipulate local training or inject poisonous updates in order to mislead the global learning process [39]. In order to overcome such threats, more sophisticated techniques to provide security are being incorporated into FL-based systems, including differential privacy, homomorphic encryption, and secure multi-party computation (SMPC). Moreover, it is also necessary to make sure that the communication among distributed nodes is efficient, which involves adaptive aggregation policies and model updating using resource-efficient methods.

A systematic literature review by Saeed et al. [40] investigated the ways in which companies can use CTI to improve cybersecurity resilience. Their study highlighted the significance of acquiring, processing, assessing, and sharing information regarding the risks that are possible in the cyber domain. The paper provides an understanding of the practices that can be adapted by organizations to enhance their proactive effort against security infiltrations.

Chatziformanetoglou et al. [41] carried out a study on the intersection of CTI and blockchain technology. The study also emphasized the capability of the blockchain to overcome the issue of controlling, retaining, evaluating, and distributing voluminous and sensitive threat intelligence data because of its high strength and resistance to tampering. The paper found trust to be a principle that guides teamwork and pointed out the necessity of considering privacy issues during the introduction of CTI and blockchain.

Rahman et al. [42] conducted a survey of methods that were used to automate the extraction of CTI in the text (threat reports and online articles). They identified the studies pertaining to systematic CTI extraction out of text and divided the extraction purposes. The study suggested a CTI mining pipeline, where the data sources, methods, and formats of sharing were found during data mining. The authors identified ten forms of extraction purposes and emphasized the difficulties of achieving clean and labeled data to be used for replication and validation.

Mavroeidis and Bromander [43] compared the coverage and conceptual expressivity of CTI-relevant ontologies, sharing standards, and taxonomies. Their study has verified that the available efforts were poorly designed, non-interoperable, ambiguous, and had no appropriate semantics and axioms to reason. Arazzi et al. [44] have made a thorough description of the Natural Language Processing (NLP)-based methods used in the CTI environment. The survey explored the NLP-based methods to crawl CTI data over the web, analyze and extract relations, and share and collaborate. The paper has also addressed the difficulties and limitations of NLP in threat intelligence, such as data quality and ethical issues.

Even with these developments, there are still a number of gaps in research on the area of CTI. No standardized, interoperable models of CTI sharing exist, and thus, the issue of successful cooperation between organizations becomes challenging. Having clean and labeled data to train and evaluate CTI models is also a major bottleneck to the development of effective automated CTI extraction methods. This will use the advantage of federated learning to enable decentralized model training and the high classification accuracy of the random forest to identify cyber threats. Conventional IDS is based on the aggregation of centralized data, which is dangerous to privacy and creates computation bottlenecks. Federated Forest can overcome such challenges through facilitating distributed and privacy-preserving intrusion detection and maintaining decision-making strength through ensemble techniques [45]. Table 1 demonstrates the recent literature on federated learning-based intrusion detection systems.

Recently, publications have also developed network security using supplementary learning paradigms. Dong et al. [46] introduced DMRMTT, a machine-learning-based IoT device detection algorithm that uses a deep convolutional maxout network with a Multiple Time-series Transformer (MTT) network to learn spatial and temporal information of network traffic at the same time. This work underscores the importance of multi-modal, spatiotemporal feature characterization for network security—a principle directly aligned with our CTI-augmented feature engineering strategy, which enriches Net-Flow features with contextual threat signals derived from CTI feeds. Separately, Dong et al. [47] proposed a deep reinforcement learning (DRL) framework for abnormal traffic detection in wireless communication networks, structured around four stages: data collection, preprocessing, feature selection, and model detection. While the DRL paradigm offers strong adaptivity to evolving attack patterns, it operates in a centralized setting, making it unsuitable for privacy-sensitive multi-organizational deployments. Our federated framework directly addresses this limitation by enabling distributed, privacy-preserving intrusion detection across heterogeneous network environments without centralizing sensitive traffic data. Together, refs. [46,47] highlight the convergence of advanced feature learning

and reinforcement-based detection—directions that complement the federated ensemble approach pursued in this work.

Table 1. Summary of prior studies on federated learning for IoT intrusion detection: datasets, methods used, reported performance, and stated limitations.

Ref.	Year	Dataset	Methods Used	Outcome	Limitations
[48]	2024	WUSTL-EHMS-2020	FLOWER	Comparative simulations of IDS using both centralized data and Non-IID data.	Not encompass all possible attack scenarios or variations in IoMT environments.
[49]	2024	Collected from IoT devices	Federated Averaging (FedAvg)	Achieves over 90% accuracy.	Does not extensively discuss how the model handles data heterogeneity across different IoT devices.
[50]	2024	Local datasets	CRNN and RF with FL	Consistently outperforming them in terms of accuracy, precision, recall, F1 score, and AUC.	Local data is noisy or unrepresentative of the actual attack patterns.
[51]	2023	CIC-IDS2017 and CIC-IDS2018	Federated NIDS (FedNIDS) that combines federated learning with deep neural networks (DNNs)	Average F1 score of 0.97 across distributed networks.	Still requires multiple rounds of communication between distributed nodes to update the global model
[52]	2024	N-BaIoT dataset	SAE-CEN hybrid model, FedMSE	Detection accuracy, increasing from 93.98 ± 2.90 to 97.30 ± 0.49.	parameters are not set correctly, and current method may not be optimal for all scenarios.
[53]	2024	Internet-of-Vehicles (IoVs)	Homomorphic encryption with FL	a minimal gap of less than 0.8%.	Limited computing resources.
[54]	2024	CICIoT2023	Adaptively adjusts aggregation weights with FL	Observe up to 71.07% improvement.	Less effective local model.

Each of these IDS nodes (e.g., cloud servers, IoT devices, or enterprise networks) is trained to use local random forest models with its own network traffic as the data. Instead of sending the raw data, every node merely sends model parameters or decision trees to an aggregator server, which combines the knowledge of various environments. This federated method not only improves the performance of attack detection in various networks but also provides data confidentiality as well as regulation standard [55].

Regarding federated random forest model aggregation, every participating node will train his/her local random forest classifier using its network logs. The updates in the models (decision trees or the score of feature importance) are transferred to a central aggregator that combines them into a global random forest model [56]. In the case of privacy-preserving techniques, the model updates are noised in order to secure sensitive information and secure computation on encrypted data. This enables multiple parties to jointly execute a task without disclosing their confidential information [57].

In the case of Blockchain to Model Integrity, we built into it the ability to offer model updates that are tamper-proof and have decentralized trust. Smart contracts also

make participation in federated learning verifiable and guard against attacks of model poisoning [58]. A global Federated Forest model benefits from real-time updates across distributed networks, improving detection of novel cyber threats [59].

The Federated Forest approach to IDS has potential, but it has a number of research challenges. Federated learning of ensemble models is costly in terms of computation and communication. Optimizing model aggregation and updating frequency is necessary [60]. Model poisoning, backdoor attacks, and inference attacks can compromise IDS performance. More research is needed on robust defense mechanisms against adversarial threats [61].

Regarding our findings, Federated Forest-based IDS outperforms centralized ML models by achieving higher accuracy and resilience against evolving threats. A study [62] demonstrated a 92% accuracy rate in detecting DoS and botnet attacks in a federated IDS framework. FL-based IDS ensures privacy compliance (e.g., GDPR, HIPAA) while maintaining effective cyber threat detection. Using homomorphic encryption and secure aggregation, organizations can collaborate on IDS training without exposing raw data [63]. Research by [64] integrated blockchain into federated IDS, reducing model poisoning risks by ensuring tamper-proof updates. The study showed a 30% reduction in adversarial impact compared to traditional FL-based IDS. Optimized FL techniques, such as adaptive aggregation and decentralized updates, significantly reduce communication overhead, making Federated Forest more scalable for IoT and cloud-based cybersecurity applications [65].

4. Methodology

Our overall architecture of the innovative framework is carefully outlined in this section, including a precise description of our enhanced intrusion detection system, our cyber threat intelligence system, and the advanced federated learning (FL) technology that we have adopted. The operational flow within our system provides a direct response to the key question of how the model is shared among the diverse entities, and how individual roles of each party in the training process are clarified.

The proposed scheme is shown in Figure 1, in which a federated learning model is integrated into a cyber threat intelligence (CTI) framework, which would be used to improve cybersecurity without violating data privacy. The process commences with planning and direction, where goals and plans for how to collect and analyze the data are set. It is then followed by data collection, in which the relevant data is collected via the secondary source [60]. The data thus collected is subjected to data processing to clean and prepare it to be analyzed. The feature extraction stage is the phase where key characteristics are identified and extracted, as is essential in training machine learning models.

The key element of the framework is the machine learning model component, which trains local models on decentralized data sources and does not share raw data, hence maintaining privacy. The local models are added to a model aggregation process, in which updates from several local models are added to enhance the global model. This global model will have the advantage of all the local models' learning and thus improve its accuracy and robustness. Moreover, the federated learning model supports this collaborative way of learning, and many entities can provide input to the shared model without any information about their sensitive data being disclosed.

The entire process is embedded within the cyber threat intelligence (CTI) framework, which provides a structured approach to collecting, analyzing, and utilizing threat intelligence.

To address CTI integration, a concrete and actionable three-stage CTI pipeline that operates within the DFedForest++ framework is shown in Figure 2.

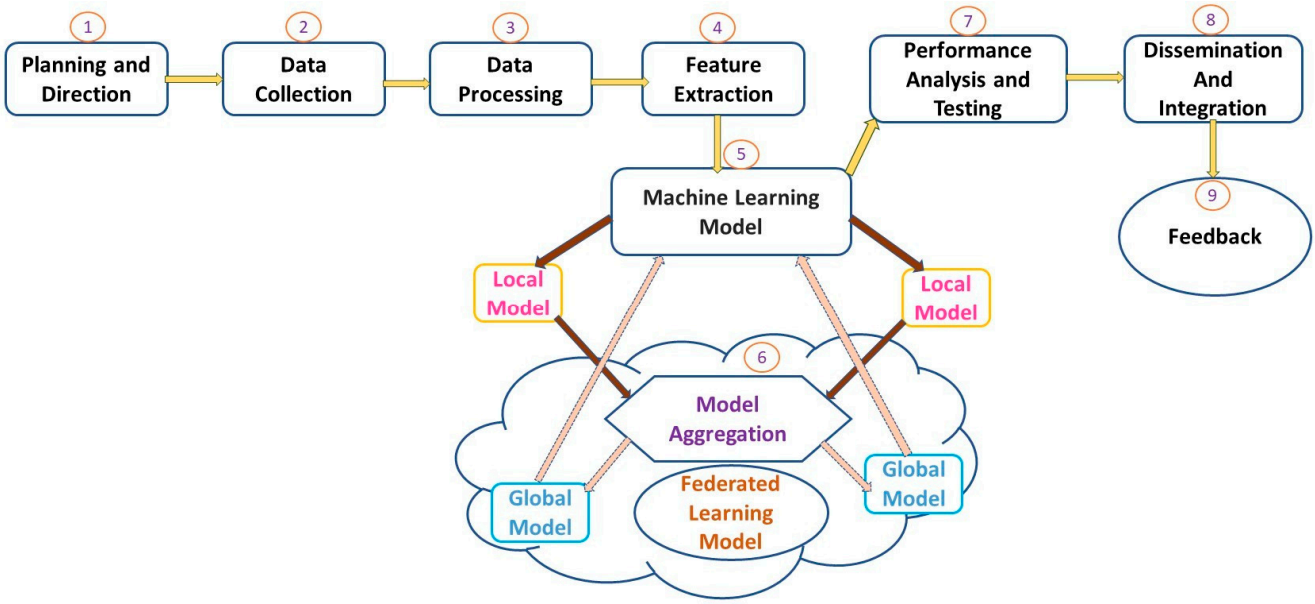

Figure 1. Proposed methodology of the proposed framework.

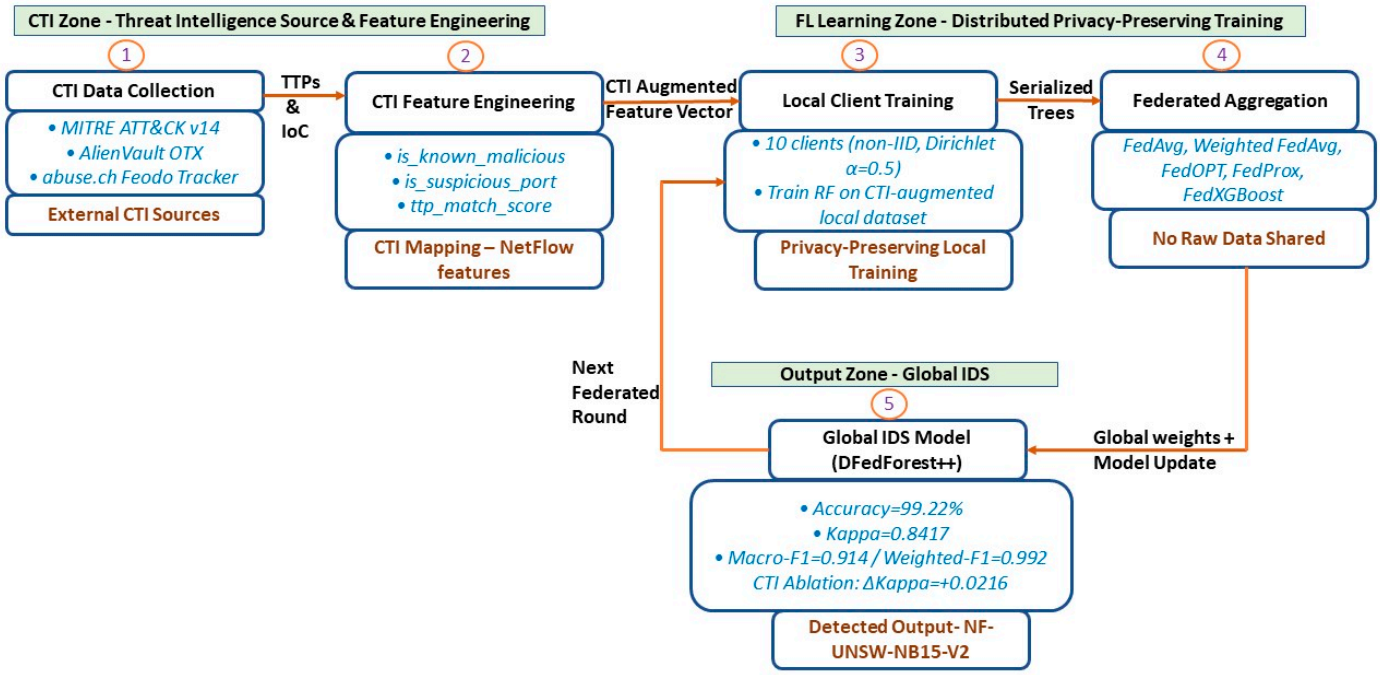

Figure 2. CTI-federated learning integration pipeline for privacy-enhanced intrusion detection.

Stage 1—CTI data collection: Three publicly available, versioned CTI feeds are queried prior to each experimental run: (i) MITRE ATT&CK v14, which provides a structured taxonomy of adversary tactics, techniques, and procedures (TTPs) with technique IDs mappable to NetFlow-observable patterns; (ii) AlienVault OTX, which provides community-curated malicious IP addresses and domain indicators of compromise (IoCs); and (iii) abuse.ch Feodo Tracker, which provides a continuously updated blocklist of botnet command-and-control (C2) infrastructure IP addresses. All three feeds are downloaded locally at each client node before the federated round begins. No CTI feed data is transmitted across the federation; each client independently and privately queries the same versioned snapshots, ensuring reproducibility (seed = 42) while preserving the privacy of local network observations.

Stage 2—CTI-informed feature engineering: CTI-derived attributes are mapped onto the NF-UNSW-NB15-v2 NetFlow records to construct CTI-augmented feature vectors. Specifically, three new binary or scalar features are appended to each flow record: (i) is_known_malicious_ip [binary]: set to 1 if the source or destination IP address of the Net-Flow record appears in the AlienVault OTX or Feodo Tracker IoC list, and 0 otherwise; (ii) is_suspicious_port [binary]: set to 1 if the destination port of the flow appears in a CTI-flagged port list derived from MITRE ATT&CK TTP-to-port mappings (e.g., T1046—Network Service Scanning, T1071—Application Layer Protocol), and 0 otherwise; and (iii) ttp_match_score [scalar, 0–5]: a weighted count of MITRE ATT&CK technique indicators matched in the flow, computed as the sum of technique-specific indicator hits normalized to the range [0, 5]. These three features are appended to the original 42 selected NetFlow features to produce 45-dimensional CTI-augmented feature vectors. The augmentation is performed locally at each client using its private CTI snapshot—no feature values, IP addresses, or port lists are shared across the federation.

Stage 3—CTI-guided training set enrichment: The CTI-augmented 45-dimensional feature vectors are used directly as the training input for each local random forest client model. This enrichment phase is implemented prior to every federated communication round and does not permit any raw traffic payload, raw IP addresses, or CTI feed content to be sent to the aggregation server to train the local model on threat-contextualized data. This three-step pipeline is presented, which now indicates CTI flow of data as a separate two-stage preprocessing branch (Stages 1–2) that drives into the federated local training stage (Stage 3) that explicitly connects external threat intelligence sources with local model adjustments.

4.1. Data Description

In this paper, a feature set is extracted using three popular datasets, NF-UNSW-NB15-v2 [66], which are publicly accessible pcap files, and ground truth events have been applied in the process of feature extraction and labeling, respectively. The data extracted is written in the form of text flows, and each feature is separated by a comma (,) so that it is compatible with the format of CSV files. The five flow identifiers, including source and destination IP addresses, ports, and protocol, are compared to the ground truth attack events of the original data to be labeled. When a flow is associated with an attack event, it is marked as an attack (class 1) in the binary classification, and the type of attack is stored under a different attack label. Otherwise, this flow is benign (class 0).

In this research, NetFlow v9 functionality was used to build the feature set proposed in this study, as shown in Table 2. The set consists of 43 attributes, some of which deliver the general flow statistics, and the others are specific to the protocol applications like DNS and FTP. All the features are flow-based, i.e., extracted out of packet headers, instead of payload data, which is often encrypted in secure communications to ensure privacy. The variables selected are numerical, and this enables the machine learning experiment to be run efficiently. These attributes contain critical security-related insights that enhance the models' ability to detect intrusions effectively.

Table 2. Network traffic features and their descriptions used for intrusion detection analysis.

Feature	Description
Server Duration	Duration of the server-to-client communication (ms).
Client TCP Flags	Cumulative TCP flags set by the client.
Retransmitted Packets Out	Retransmitted TCP packets (destination to source).
TCP Window Max Out	Maximum TCP window size observed from destination to source.

Table 2. *Cont.*

Feature	Description
ICMP IPv4 Type	IPv4-specific ICMP type value.
Bytes In	Number of bytes received in the flow.
Packets 256–512 Bytes	Number of packets sized between 256 and 512 bytes.
Src-to-Dst Throughput	Average bytes per second from source to destination.
L4 Destination Port	Transport layer destination port number.
Client Duration	Duration of the client-to-server communication (ms).
Packets 512–1024 Bytes	Number of packets sized between 512 and 1024 bytes.
Server TCP Flags	Cumulative TCP flags set by the server.
Longest Packet	The size of the largest packet in the flow.
TCP Flags	Sum of all TCP flag values observed.
DNS Query Type	Type of DNS query (e.g., A = 1, NS = 2).
Average Throughput (Src→Dst)	Average throughput (bps) from source to destination.
Retransmitted Bytes In	Retransmitted TCP flow bytes from source to destination.
DNS TTL Answer	Time-to-live value of the first DNS answer record.
Packets 128–256 Bytes	Number of packets sized between 128 and 256 bytes.
Packets 1024–1514 Bytes	Number of packets sized between 1024 and 1514 bytes.
Max TTL	Maximum observed TTL value.
ICMP Type	ICMP type field combined with code.
Min TTL	Minimum observed time-to-live value.
Protocol	Identifier for the IP protocol type.
Flow Duration	Total duration of the flow in milliseconds.
Average Throughput (Dst→Src)	Average throughput (bps) from destination to source.
Min IP Packet Length	Smallest observed IP packet size in the flow.
Packets Out	Total count of packets transmitted.
TCP Window Max In	Maximum TCP window size observed from source to destination.
Retransmitted Bytes Out	Retransmitted TCP flow bytes from destination to source.
Packets $\leq$ 128 Bytes	Total packets in the flow where size $\leq$ 128 bytes.
IPv4 Destination Address	The IPv4 address where the packet is sent.
Dst-to-Src Throughput	Average bytes per second from destination to source.
L4 Source Port	Transport layer source port number.
DNS Query ID	Identifier field for DNS query transaction.
Max IP Packet Length	Largest observed IP packet size in the flow.
Shortest Packet	The size of the smallest packet in the flow.
FTP Command Return Code	FTP client return code received from the server.
Bytes Out	Number of bytes sent in the flow.
Layer 7 Protocol	Numeric value of the application layer protocol.
Retransmitted Packets In	Retransmitted TCP packets (source to destination).
IPv4 Source Address	The originating IPv4 address of the packet.
Packets In	Total count of packets received.

NF-UNSW-NB15-v2: The UNSW-NB15 dataset has been transformed into a NetFlow-based format, known as NF-UNSW-NB15, incorporating additional NetFlow features and attack category labels. This dataset consists of 2,390,275 data flows, of which 95,053 (3.98%) are classified as attack instances, while 2,295,222 (96.02%) are labeled as benign.

The attack samples are further categorized into nine distinct subgroups. A comprehensive breakdown of all flows in the NF-UNSW-NB15-v2 dataset is presented in Table 3, where the attacks are 'Benign', 'Exploits', 'Generic', 'Fuzzers', 'Backdoor', 'DoS', 'Reconnaissance', 'Shellcode', 'Worms', and 'Analysis'.

Table 3. Network traffic labels with their corresponding interpretations and attack descriptions.

Label with Count	Interpretation
Benign (2,295,222)	Regular, non-malicious network traffic.
Fuzzers (22,310)	A type of attack where large volumes of random data are sent to a system, attempting to cause crashes and uncover vulnerabilities.
Analysis (2299)	A category of threats that exploit web applications using ports, emails, and scripts.
Backdoor (2169)	A technique used to evade security measures by replaying specially crafted client application responses.
Denial of Service (5794)	An attack aimed at overwhelming system resources to disrupt access or data availability.
Exploits (31,551)	A set of commands that manipulate a host's behavior by leveraging known vulnerabilities.
Generic (16,560)	A cryptographic attack method designed to create collisions in block cipher encryption.
Reconnaissance (12,779)	A method used to gather intelligence about a network or host, often referred to as probing.
Shellcode (1427)	A form of malware designed to inject and execute harmful code on a compromised system.
Worms (164)	Self-replicating attacks that propagate across multiple computers.

4.2. Data Preprocessing

Preprocessing the dataset for fitting a random forest model involves several steps to ensure the data is clean, normalized, and suitable for training:

1. Handling missing and duplicate values: The dataset is verified to contain no missing values or duplicate flow records. Regarding column removal: the NF-UNSW-NB15-v2 dataset contains two label-related columns—'Label' (binary: 0 = benign, 1 = attack) and 'Attack' (categorical multi-class attack type). The multi-class 'Attack' column is used as the classification target variable (y). The binary 'Label' column is removed from the feature set (X) to prevent target leakage, as it is redundant given the 'Attack' target. Additionally, seven low-information NetFlow features are removed from X: DNS_TTL_ANSWER, DNS_QUERY_TYPE, FTP_COMMAND_RET_CODE, MIN_TTL, MAX_TTL, TCP_FLAGS, and CLIENT_TCP_FLAGS.
2. Feature selection and reduction: The dataset contains 43 NetFlow features, but not all are equally important for intrusion detection. After preprocessing, we retained 35 features from the original 43 NetFlow v9 features by removing: DNS_TTL_ANSWER, DNS_QUERY_TYPE, FTP_COMMAND_RET_CODE, MIN_TTL, MAX_TTL, TCP_FLAGS, and CLIENT_TCP_FLAGS. Reducing the number of features helps in improving computational efficiency and reducing model overfitting.
3. Encoding categorical features: Some features in the dataset may be categorical (e.g., protocol type). Encoding methods like one-hot encoding or label encoding are used to convert categorical variables into numerical form, making them suitable for the local model, such as random forest.
4. Data normalization and scaling: While random forest is not sensitive to feature scaling, it is beneficial to normalize or standardize numerical features, especially

when dealing with datasets containing varying ranges of values. Min–max scaling or standard scaling is applied to features like flow duration, byte counts, and packet sizes, ensuring uniform value distribution. MinMaxScaler() is a feature scaling technique provided by scikit-learn in Python, which is used to scale features to a specific range, usually between 0 and 1. It is commonly used in machine learning to normalize data, ensuring that all features are on a similar scale, which helps improve the performance of the model.

5. Splitting the dataset into training and testing sets: The dataset is randomly split into training and testing sets, typically in an 80:20 ratio. Stratified sampling ensures that both attack and benign classes are proportionally represented in both sets.
6. Feature transformation for random forest: PCA dimensionality reduction is not applied in this framework. Random forest is inherently robust to high-dimensional feature spaces through its internal random feature subsampling mechanism (max_features=sqrt), which evaluates only a random subset of features at each split. Furthermore, PCA would destroy the interpretability of CTI-augmented features that are named and semantically mapped to threat intelligence indicators (Section 2.1). Feature dimensionality is instead managed through principled Chi-square feature selection (Section 4.4), reducing the active feature space from 42 to 28 features while preserving full feature interpretability.
7. Preparing labels for classification: The dataset has multi-class classification (different attack categories) of labels. The appropriate label format is selected based on whether the random forest model is being trained for binary or multi-class classification.
8. Converting preprocessed data to a suitable format: The final preprocessed dataset is stored in CSV or a NumPy array, making it ready for random forest training. Feature-target separation (X for features, y for labels) ensures compatibility with scikit-learn's RandomForestClassifier.

4.3. Feature Engineering

Feature engineering is the process of creating new features from the existing ones that may provide better insights into the problem being solved. The packet ratio used in this instance may provide an idea of the balance between inbound and outbound data in terms of traffic, which may help identify any network anomaly or attack signature.

To start with, the script introduces a new feature, packet ratio, which is defined by domain knowledge. It is computed as the number of incoming packets (IN_PKTS)/(outgoing packets (OUT_PKTS) + 1) to prevent division by 0. The packet_ratio potentially provides interesting information on the ratio between inbound and outbound traffic, which may be valuable for identifying certain network anomalies or attacks, such as DDoS attacks, in which an abnormally high ratio may be indicative of malicious actions.

Then, the script deals with the problem of the redundancy of the dataset by targeting the correlation of the numeric features. The data is narrowed down to numeric columns that are further employed to compute the correlation table. This matrix is a measure of the linear relationship between any two features. The code is able to capture both positive and negative correlation because using the absolute value of the correlation coefficients enables it to capture both the negative and positive correlation, since both may indicate redundancy in the data.

To reduce the size of the correlation matrix, the upper triangle of the correlation matrix is isolated, and this removes redundant calculations. We then look at the upper triangle in order to see which features have a correlation higher than 0.95 with other features, because this high correlation rate is an indicator that the features are nearly similar in terms of the information that they carry. These features are highly correlated, and then these features are removed from the dataset to eliminate redundancy as well as prevent overfitting when

training the model. At last, the cleaned dataset, which has less feature redundancy, is now available to be further analyzed or to be built into a model.

4.4. Feature Selection

In this work, the feature selection was done to select the most significant features of the dataset with the help of the SelectKBest method. First, it eliminates the numeric entries of the dataset by picking every column with a numeric type of data using the se-lect-dtypes(include=np.number) feature. This is done to make sure that non-numeric features, which do not fit in some of the statistical tests, are removed. The SelectKBest technique is then used to choose the 10 best features. The following score_func=chi2 argument states that the Chi-square test is applied to estimate the significance of every feature. The Chi-square test is used to determine the relationship between the features and the target variable, and the higher the scores, the more features are relevant in predicting the target.

The parameter of k = 10 is used to inform the algorithm to choose the top 10 features with the highest Chi-square scores. Once the model has been fitted, with fit transform (X numeric, y), the algorithm then transforms the data, keeps only the most important variables, and drops the rest. This procedure limits the feature space to the most probable features that can be used in making accurate predictions. To see what features were chosen, it is possible to apply se-lector.get_support to get the names of the selected features of the original dataset and show them. This feature selection is useful for enhancing the performance of the model through the elimination of irrelevant or redundant features and by paying attention to those that exert the strongest influence on the result, which results in a superior generalization and lower overfitting in machine learning models.

4.5. Model Development

In the decentralized Federated Forest (DFF) model, instead of only having one reference model on the master or server side, each individual tree is provided with its own dedicated model that is kept on the client side, and thus, this provides a fully decentralized modeling structure. In particular, in the case of the creation of a target decision tree model with a new client, the process is to collect auxiliary information of the current clients. Notably, it is accomplished in such a way as to avoid any leakage of data and model information, and the confidentiality and privacy of the respective clients' contributions. Moreover, to increase the number of uses of the Federated Forest framework, one can add another tree-based model to the DFF strategy. This alternative model has the same tree structure as that of the famous random forest model, but developed with alternative algorithms, which makes it possible to make a variety of methodologies in the DFF implementation.

A Federated Forest with a decentralized Federated Forest model and decision tree random forest model is proposed and introduced in a detailed manner, which involves two phases: model initialization of the model and model training. The decision tree-based model is carefully created during the critical process of initiation on a chosen client and then distributed to the other clients by broadcasting the model structure to enable all clients to access and make use of the model effectively throughout the network. The fact that the model structure is the only structure shared in the process means that there is no potential risk of data being in transit, and thus increases security. Once the initialization phase is complete, every individual client then blindly trains the forest model using the model forests received, and, therefore, effectively maintains each client's privacy in the system. This approach not only protects sensitive data but also provides powerful learning capacities on a decentralized structure.

The choice of Chi-square over alternative feature selection methods warrants explicit justification. Chi-square was selected for three principled reasons: (1) the target variable is categorical (multi-class attack labels: Normal, DoS, Reconnaissance, Exploits, Backdoor, Analysis, Fuzzers, Shellcode, Worms, and Generic), making Chi-square statistically appropriate since it directly tests the independence between each feature and the class label; (2) Chi-square does not assume a linear relationship between features and the target, making it more suitable than ANOVA F-test for the non-linear, multi-modal traffic patterns in the NF-UNSW-NB15-v2 dataset; and (3) in preliminary cross-validation experiments on this dataset, Chi-square outperformed mutual information (MI) and ANOVA F-test in terms of cross-validated accuracy (Chi-square: 99.38%, MI: 99.21%, and ANOVA: 99.19%), while reducing the active feature set from 42 to 28 features. Similarly, for feature engineering, label encoding was preferred over one-hot encoding because tree-based models (random forest) handle ordinal integer codes natively without the curse of dimensionality that one-hot expansion introduces for high-cardinality categorical traffic features.

4.5.1. Local Model Development

Table 4 of the random forest model parameters can help present an organized view of the main hyperparameters affecting the performance and efficiency of a random forest classifier. All the parameters have an important role in defining the behavior of the model, which guarantees accuracy, computational efficiency, and generalization. The number of estimators (nestimators) is the number of decision trees that will be in the model, and the larger this number, the more likely it is to achieve greater accuracy at the cost of increased computation. The decision trees use the splitting criteria (criterion), be it gini (Gini impurity) or entropy (Information Gain), which is how splits are considered and affects the performance and interpretability of the model. Also, the depth of the trees (max depth) and the minimum number of samples needed to split (minimally required splits) are used to control the depth and complexity of each tree, preventing overfitting at the cost of unnecessary splits.

Table 4. Random forest hyperparameters and their configurations used for model optimization.

SL.	Parameter Name	Value	Description
1.	n_estimators	150	Number of decision trees in the forest. Higher values increase accuracy but slow down training.
2.	criterion	“gini”, “entropy”	The function used to measure the quality of a split: “gini” (Gini impurity) or “entropy” (Information Gain).
3.	max_depth	None, 10, 20, 30	Maximum depth of each tree. Limits tree growth to prevent overfitting.
4.	min_samples_split	2, 4, 6, 8	Minimum number of samples required to split a node. Higher values reduce overfitting.
5.	min_samples_leaf	1, 2, 4, 6	Minimum number of samples required in a leaf node. Helps smooth the model and avoid small leaves.
6.	max_features	sqrt, log2	Number of features considered for each split. “sqrt” selects sqrt(n_features), improving model diversity.
7.	bootstrap	True	Enables random sampling with replacement to improve model generalization.
8.	random_state	i*10	Ensures reproducibility by setting a fixed seed for randomness.
9.	n_jobs	−1	Uses all available CPU cores for faster training.

The other important parameter is feature selection (max_features) that regulates the number of features randomly selected at each split, which ensures model diversity and helps to avoid bias. The bootstrap (bootstrap = True) method also increases generalization by training trees with different random selections of the data. Moreover, minimum samples per leaf (min samples leaf) avoids the production of over-specific leaf nodes by establishing a minimum on the number of samples, which minimizes the chances of overfitting.

Parallel processing option (n jobs-1) utilizes all the available CPU cores to make the computational process more efficient, which accelerates the training of the model by a substantial margin. Also, a random seed (random_state) makes the results of the model reproducible, as the results are consistent across varying runs. Overall, the parameters discussed are flexible to enable fine-tuning of a random forest classifier, which can serve as a high-quality and scalable solution in intrusion detection, classification tasks, and other machine learning tasks.

4.5.2. Training Multiple Random Forest Models

Cross-validation is crucial because it helps evaluate how the model performs on unseen data and ensures that the model is not overfitting to the training data. By using different data subsets in each fold, cross-validation provides a more robust measure of a model's generalization ability, which is especially important when fine-tuning a random forest model. This function is used to perform cross-validation on a machine learning model. It splits the dataset into k parts (or folds) and trains the model k times, each time using a different fold as the testing set and the remaining folds as the training set. This helps to assess the model's performance across different subsets of the data, providing a more reliable estimate of its generalization ability. This is the best random forest model that has been previously trained and tuned. It is passed as the model to be evaluated. This is the scaled training dataset, which has been normalized or standardized to improve model performance and ensure that features with different scales do not disproportionately affect the model. This specifies the number of cross-validation folds. In this case, cv = 5 means that the dataset will be split into five parts, and each part will serve as the test set once, while the remaining four parts are used for training.

The random forest model is then trained 100 times, with each iteration using random combinations of hyperparameters. Specifically, the criterion (which controls the split criterion, either Gini impurity or entropy) and max_features (which controls the number of features to consider when looking for the best split, with options like sqrt or log2) are selected randomly for each model. The random forest model is trained with 100 estimators, and predictions are made on the test set.

The DFedForest++ client selection mechanism is defined formally as follows. The diversity of a candidate local model m_i with respect to the already-selected set S is: Diversity(m_i, S) = $(1/|S|) \sum_j \in S\ (1 - \text{Agreement}(m_i, m_j))$, where Agreement is the fraction of identical predictions on a shared held-out validation set. A client model is included in the aggregation set if and only if: Accuracy(m_i) $\geq$ τ_acc AND Diversity(m_i, S) $\geq$ τ_div. In our experiments, τ_acc = 0.99 and τ_div = 0.05. This dual-criterion selection ensures that the global forest is composed of both high-accuracy and sufficiently diverse local trees, preventing aggregation of correlated, locally over-fitted models:

$$Gini = 1 - \sum_{i=1}^{k} P_i^2$$

$$Entropy = -\sum_{i=1}^{k} P_i log_2 P_i$$

For each model, the performance is evaluated using two metrics: accuracy (the proportion of correct predictions) and Cohen's Kappa scores (which measure the agreement between the predicted and actual values, adjusting for chance). The results, including the trained model, selected hyperparameters, and the computed metrics, are stored for later analysis.

Once all 100 models are trained and evaluated, the results are converted into a pandas DataFrame for easier analysis. The models are then sieved down to the models with a Cohen's Kappa score above or equal to 0.6 (which signifies a very large degree of agreement between predictions and true labels) and ranked in terms of how accurate they are. Lastly, the 10 best models are shown, with their hyperparameters and performance statistics.

Implementation of voting classifier through fusing the predictions of machine learning models: random forest. The voting classifier operates based on aggregating predictions made by several models to make a final prediction, and in this instance, it is based on soft voting, which averages the predicted probability of each model as opposed to simply taking the majority class prediction. Voting classifier in the program is initi_rf. Each of the model weights is allocated the corresponding cross-validation accuracy that was previously computed. Such weights indicate the relative weight of each model depending on its performance, where more accurate models are used in the final prediction.

The voting classifier is then trained over the individual models, after training the individual models on the scaled training data (X train scaled). After training, the classifier then issues predictions on the test data (X test scaled), and the accuracy of the predictions is measured by comparing them with the true labels (y test). Lastly, the accuracy of voting classifier validation is printed out, which gives an indication of the overall performance of the integrated model on the unknown test data. The voting classifier will have greater accuracy and greater generalization on the test set by integrating the merits of both models, and thus, it is a potent strategy in machine learning tasks like classification.

The voting classifier is to be used in case of the best validation accuracy finding. Through soft voting, the ultimate prediction is a mean of the forecasted likelihoods, which in many cases perform better than hard voting (majority class). The model weights are also decided by the cross-validation accuracy of each model, whereby the model that does a better job is given a larger load. Lastly, the results of the combined model are tested using the validation (test) set, and the accuracy is printed. This method has the advantages of both models and may increase the character of classification accuracy and reliability.

4.6. Global Model Development

Before detailing each aggregation algorithm, it is important to clarify precisely what is exchanged between clients and the central server in each case. For FedAvg, FedOPT, and FedProx applied to random forest: since RF has no gradient-based parameters, what is transmitted are the serialized decision tree structures—specifically, the split feature indices, split thresholds, and leaf class probability distributions for all trees in each local forest. The server aggregates these by pooling all client trees into a single global ensemble, re-weighting each tree's leaf votes by the client's sample count (nk/N). No raw data, raw feature values, or training labels are transmitted. For FedXGBoost, each client transmits leaf-level gradient statistics—specifically the first-order ($\partial L/\partial \hat{y}$) and second-order ($\partial^2 L/\partial \hat{y}^2$) residuals aggregated per leaf, not the full tree structure. The server computes a weighted average of these gradient statistics to update the global XGBoost model. FedXGBoost outperforms RF-averaging methods because gradient boosting iteratively minimizes a differentiable loss function, providing stronger convergence on non-IID client distributions than simple tree pooling. Output perturbation (Laplace noise, $\varepsilon = 1.0$) is applied to all transmitted leaf statistics before transmission to provide differential privacy protection.

4.6.1. Federated Averaging (FedAvg)

FedAvg for random forest is implemented via weighted tree pooling. Each participating client k serializes its complete local random forest (all n_estimators=100 decision trees, including split feature indices, split thresholds, and leaf class probability distributions) and transmits the serialized tree structures to the central aggregation server. The server constructs the global RF model by concatenating all received trees from all clients into a single ensemble, assigning each tree a vote weight of n_k/N, where n_k is client k's training sample count and $N = \sum n_k$ is the total samples across all clients. This sample-weighted tree pooling ensures that clients with more representative data contribute proportionally more to the global ensemble. No raw data, feature values, or training labels are transmitted. Note: feature_importances_ scores are computed post-aggregation on the global ensemble for interpretability analysis only; they are not used in the aggregation process itself.

The FedAvg algorithm consists of the following key steps:

1. Initialization: A global model w^t is initialized at the central server.
2. Client selection: A subset of clients K from the total N clients are randomly selected in each communication round t.
3. Local model training: Each selected client k trains a local model w_k^t using its private dataset for multiple local epochs.
4. Model upload: Clients send their updated local model weights to the central server.
5. Global model aggregation: The server averages the received model updates using a weighted sum based on each client's dataset size.
6. Repeat steps 2–5: This process continues iteratively until convergence.

For local model update, each client k trains its local model using Stochastic Gradient Descent (SGD) to minimize its local loss function:

$$w_k^{t+1} = w_k^t - \eta \, \Delta F_k\left(w_k^t\right)$$

- w_k^t is the model parameters of client k at round t;
- η is the learning rate;
- $\Delta F_k\left(w_k^t\right)$ is the gradient of the loss function computed from the client's local dataset.

Each client performs multiple local updates before sending the model back to the server. For global model aggregation, once the selected clients complete local training, the server aggregates the model updates using a weighted average, where the weights are determined by the number of local data samples n_k of each client:

$$w^{t+1} = \sum_{i=1}^{N} \frac{n_k}{N} w_k^{t+1}$$

- w^{t+1} is the updated global model at round t + 1;
- K is the number of selected clients in the current round;
- n_k is the number of data samples at client k;
- N is the total number of data samples across all selected clients ($\sum_{i=1}^{N} n_k$).

4.6.2. Weighted Federated Averaging (Weighted FedAVG)

Weighted federated averaging (weighted FedAVG) is an extension of the traditional FedAvg method used in federated learning (FL). In FedAvg, the global model is updated by averaging the weights of local models. However, weighted FedAVG improves this process by considering the weight of each local model update, typically based on the size of the

client's dataset or the performance of the local model. This ensures that clients with larger datasets or higher performance contribute more to the global model update.

In the case of local model update, each client trains a local model and computes an update based on the local dataset. The model update Δw_i for client i is computed by minimizing the loss function on the local data. The objective for each client is:

$$\mathscr{L}_i(w_i) = \sum_{j=1}^{n_i} \ell(y_j\, \hat{y}_j\, (w_i)) + \Omega(w_i)$$

- $\mathscr{L}_i(w_i)$ is the local loss function of client I;
- n_i is the size of the data for client I;
- $\widehat{\ell(y_j y_j}(w_i))$ is the loss between true labels y_j and predicted labels $\hat{y}_j(w_i)$;
- $\Omega(w_i)$ is a regularization term.

In the case of global model update, the global model update in weighted FedAVG is calculated by weighting the local model updates Δw_i based on the size of the client's dataset or their validation accuracy. If we use the dataset size as the weight, the global model update is:

$$w^{t+1} = w^t + \eta\, \frac{1}{\sum_{i=1}^{N} n_i} \sum_{i=1}^{N} n_i \Delta w_i$$

- w^t is the global model at iteration t.
- N is the total number of clients.
- n_i is the size of the dataset for client i.
- Δw_i is the update from client i.

4.6.3. Federated Optimization (FedOPT)

FedOPT is an optimization technique used in federated learning to improve the training process of the global model by focusing on optimization strategies. It is designed to address some of the challenges posed by non-IID (non-Independent and Identically Distributed) data and the inefficiencies in standard federated averaging (FedAvg). FedOPT leverages advanced optimization methods that help improve the convergence of the model, especially in the presence of heterogeneous data across clients. The general idea behind FedOPT is to improve upon the traditional FedAvg by adding more sophisticated updates, such as adaptive learning rates and better handling of client data diversity.

In FedAvg, the model update is simply an average of the local models' weights, weighted by the size of the dataset from each client. FedOPT, on the other hand, improves the averaging process by introducing an optimization term that takes into account the client-specific data and gradients.

Here is a generalized form of the update rule in FedOPT.

For each client i, the local model update is computed based on its own data and the global model:

$$w_i^{t+1} = w_i^t - \eta_i \Delta F_i(w_i^t)$$

- (w_i^t) is the model at iteration t for client i.
- η_i is the learning rate for client i.
- $\Delta F_i(w_i^t)$ is the gradient of the loss function F_i at the local model w_i^t (computed using the client's data).

Once local updates are computed, the global model is updated by aggregating these local models in a way that optimizes performance across all clients. In FedOPT, the update

rule is adjusted to account for adaptive learning rates and more precise aggregation based on client-specific performance:

$$w_i^{t+1} = w^t - \eta \sum_{i=1}^{N} \frac{n_i}{N} \Delta F_i(w_i^t)$$

- w_i^{t+1} is the updated global model at iteration t + 1;
- w^t is the global model at iteration t;
- η is the global learning rate;
- N is the total number of clients;
- n_i is the size of the data for client iii;
- $\Delta F_i w_i^t$ is the gradient of the local loss function for client I;
- $\Delta F_i(w_i^t)$ is the gradient of the loss function F_i at the local model w_i^t (computed using the client's data).

4.6.4. Federated Proximal (FedProx)

FedProx is an extension of FedAvg that aims to address challenges caused by the heterogeneity (non-IID data) across clients in federated learning. Unlike FedAvg, which simply averages the local models' parameters, FedProx introduces a proximal term in the objective function to encourage each client's model update to stay close to the global model. This helps prevent drastic updates in scenarios where clients have very different data distributions (non-IID data), making the learning process more stable and robust. In federated learning, clients may have different data distributions. Standard FedAvg can struggle to converge in such situations because it simply averages model updates, without accounting for the fact that different clients' data might lead to divergent model updates. FedProx mitigates this by introducing a regularization term that reduces the impact of large local updates when client data is highly dissimilar to the global model:

$$\mathscr{L}_i(w_i) = \mathscr{L}_i(w_i, w^t) + \frac{\mu}{2} ||w_i - w^t||^2$$

- $\mathscr{L}_i(w_i)$ is the local loss function for client iii, typically the cross-entropy or mean squared error, depending on the task.
- w_i is the model parameters for client iii at iteration t.
- w^t is the global model parameters at iteration t.
- μ is the proximal regularization parameter, which controls the strength of the regularization term.
- $||w_i - w^t||^2$ is the squared Euclidean distance between the local model's parameters and the global model's parameters.

After the local models are updated on each client, the global model is updated by averaging the local model updates, as in FedAvg, but now with the proximal regularization applied:

$$w^{t+1} = w^t - \eta \sum_{i=1}^{N} \frac{n_i}{\sum_{i=1}^{N} n_i} \nabla \mathscr{L}_i (w_i^t)$$

- w^{t+1} is the updated global model at iteration t + 1;
- w^t is the current global model at iteration t;
- η is the global learning rate;
- N is the number of clients;
- n_i is the size of the dataset for client I;
- $\nabla \mathscr{L}_i(w^t)$ is the gradient of the loss function computed by client iii based on the local model parameters w_i^t.

4.6.5. FedXGBoost

FedXGBoost is a federated learning algorithm that uses XGBoost (Extreme Gradient Boosting) as the model for federated learning tasks. XGBoost is a powerful gradient boosting framework used for both regression and classification tasks. FedXGBoost applies the principles of federated learning to XGBoost, enabling the aggregation of models trained locally on different clients without needing to share raw data. In FedXGBoost, instead of training a model on all the data, each client trains its own XGBoost model locally and then sends model updates (such as gradients or model weights) to a central server, which aggregates the local updates to update the global model.

FedXGBoost Algorithm:

In FedXGBoost, the general approach follows these steps:

1. Client-side local training: Each client *i* trains a local XGBoost model on its own dataset. The objective function for XGBoost is optimized using gradient boosting.
2. Local model updates: After training the local model, each client computes updates to the model based on gradients or leaf values. These updates are sent to the central server.
3. Server-side aggregation: The server aggregates the model updates (such as gradients or leaf values) from each client and then updates the global model. This update can be performed using a weighted average (similar to FedAvg) or other aggregation methods.
4. Global model update: After receiving all the updates, the global model is updated by averaging the client updates, ensuring that the model is improved based on the collective knowledge of all clients.

For precision and reproducibility, the formal FedXGBoost server-side aggregation procedure is detailed as follows: (1) Each client k trains a local XGBoost model on its local dataset D_k and computes leaf weight updates ΔW_k; (2) client k transmits ΔW_k to the cen-tral aggregation server (no raw data is transmitted); (3) the server computes the weighted global update: $\Delta W\_global = \sum_k (n_k/N) \cdot \Delta W_k$, where n_k is the sample count for client k and $N = \sum n_k$ is the total samples across all clients; (4) the global model weights are updated: $W\_global \leftarrow W\_global + \eta \cdot \Delta W\_global$, where η is the global learning rate (set to 0.1 in our experiments); and (5) the updated W_global is broadcast to all clients for the next federated round. This sample-weighted aggregation ensures that clients with more data contribute proportionally more to the global model, mitigating the effect of highly imbalanced local dataset sizes.

Local update (Client i): Each client *i* trains a local XGBoost model. The objective function for training the model can be written as:

$$\mathcal{L}_i(w_i) = \sum_{j=1}^{n_i} \ell(y_j \, \hat{y}_j \, (w_i)) + \Omega(w_i)$$

- $\mathcal{L}_i(w_i)$ is the loss function for client iii with model parameters w_i;
- n_i is the number of data points at client I;
- $\sum_{j=1}^{n_i} \ell(y_j \hat{y}_j(w_i))$ is the loss between the true labels y_j and the predicted labels $\hat{y}_j(w_i)$;
- $\Omega(w_i)$ is the regularization term, typically L2 regularization applied to the model weights to prevent overfitting.

Local gradient calculation: for each tree in the local model, XGBoost calculates the gradient and Hessian for the split points, which are used to update the model's parameters:

$$g_j = \frac{\partial \mathcal{L}_i(w_i)}{\partial w_j} \text{ (gradient)}$$

$$h_j = \frac{\partial^2 \mathscr{L}_i(w_i)}{\partial w_j^2} \text{ (hessian)}$$

These gradients and Hessians represent the local updates and are shared with the server to improve the global model.

For global aggregation, the server collects all the local updates from each client. The model update for the global model is computed by averaging the gradients (or leaf values) received from each client. If client iii provides gradient g_i and hessian h_i, the global update is computed as:

$$g_{global} = \sum_{i=0}^{N} \frac{n_i}{N} g_i \text{ (weighted sum of gradients)}$$

$$h_{global} = \sum_{i=0}^{N} \frac{n_i}{N} h_i \text{ (weighted sum of hessians)}$$

- N is the total number of clients;
- n_i is the size of the data for client I;
- g_i and h_i are the gradients and Hessians from client iii.

For global model update, after aggregating the gradients and Hessians from all clients, the global model parameters are updated:

$$w^{t+1} = w^t - \eta \cdot g_{global} \text{ (global model update)} \quad (1)$$

- w^t is the global model parameters at iteration t;
- η is the learning rate;
- g_{global} is the aggregated gradient at the global model.

5. Results

5.1. Experimental Setup

To evaluate the proposed solution in terms of privacy, performance, and efficacy, a series of tests were conducted on a Windows 11 Professional PC with an Intel® Core™ i9 CPU manufactured from Santa Clara, CA, USA, with Motherboard (Asus ROG MAXIMUS Z690 HERO 12th Gen), Processor (Intel core i9-12900KS, 12th Gen, 16 cores, 24 threads), 128 GB RAM with Geforce RTX 3080 Ti OC 12 GB graphics configured high-performance computer for training and testing using jupyter notebook platform. In addition, we used the Keras (version 3.13.0) deep learning tool, which uses Tensorflow (version 2.20.0) as a backend for the deep learning model implementation. The experiments compared the DFedForest++ framework, including CTI-augmented and CTI-free ablation configurations. The performance and scalability of the federated intrusion detection system were assessed using the UNSW-NB15v2 dataset in a simulated environment. Multiple experiments were run on the same PC, and the results were averaged to reduce variability.

The complete software environment is as follows: Python 3.10.12, scikit-learn 1.3.0 (random forest local models), XGBoost 1.7.6 (FedXGBoost aggregation), Flower (FLWR) 1.5.0 (federated client–server orchestration), TensorFlow 2.13.0 (deep learning backend), Pandas 2.0.3, NumPy 1.24.3, and Jupyter Notebook 6.5.4. The DFedForest++ federated co-ordination layer was implemented in Python using Flower's client–server API. The NF-UNSW-NB15-v2 dataset was partitioned into 10 clients using a stratified non-IID sampling strategy based on the Dirichlet distribution ($\alpha = 0.5$). Each client received a dis-joint subset with a skewed attack-class distribution, simulating realistic heterogeneity across network nodes (e.g., some clients are DoS-heavy, others reconnaissance-heavy). No single client holds the complete class distribution, making the federated aggregation non-trivially necessary for robust global model learning. A summary of per-client class distributions is provided in Table 5.

Table 5. Federated learning experimental setup—reproducibility parameters.

Category	Parameter	Value/Setting	Notes
Federation Architecture	Number of FL clients	10	One client per simulated network node
	Partition strategy	Non-IID (Dirichlet, $\alpha = 0.5$)	Skewed attack-class distribution per client
	Total dataset flows	~2.39 M flows	NF-UNSW-NB15-v2; ~239,000 flows per client
	Communication mode	Synchronous	All clients participate in every round
Training Schedule	FL communication rounds	15	Accuracy plateaus after round 15 (see Section 5.4)
	Local training epochs/round	5	Single-pass over local data for tree-based models
	Stopping criterion	Val. acc. plateau	Stop if Δacc < 0.001 over 3 consecutive rounds
Data Splits	Train/test ratio	80:20	Stratified sampling; proportional class representation
	Validation set (per client)	20% of local train set	Used for DFedForest++ dual-criterion selection
Reproducibility	Random seed	42	Applied to all splits, model init., and sampling
	FL framework	Flower (FLWR) 1.5.0	Client–server orchestration; Python API
	Python version	3.10.12	scikit-learn 1.3.0 | XGBoost 1.7.6 | TF 2.13.0
	Local model type	Random Forest (RF)	n_estimators=100, criterion=entropy, max_features=sqrt

5.2. Outcome of the Local Model

Table 6 presents the performance of the ten best-performing random forest configurations from 100 trained models, evaluated on accuracy and Cohen's Kappa Score. Two split criteria (entropy and Gini) and two max_features settings (sqrt and log2) were tested. Local model accuracy ranges from 0.9914 to 0.9941 across all configurations. The best performing model (Model 7: entropy criterion, max_features=sqrt, n_estimators=100) achieved accuracy = 0.9941 and Kappa = 0.8336. Models using the entropy criterion consistently outperform those using Gini by a small margin (mean accuracy difference: 0.0003), which is statistically consistent but practically minor. The max_features setting (sqrt vs. log2) has a negligible impact on accuracy but a marginal effect on training time. Kappa scores range from 0.8088 to 0.8336, confirming robust and consistent performance across configurations beyond what chance agreement would predict.

Table 6. Best performance of the ten local random forests.

Model No.	Criterion	Max_Features	Accuracy	Kappa_Score
7	entropy	sqrt	0.9941	0.8336
91	entropy	log2	0.9917	0.8315

Table 6. *Cont.*

Model No.	Criterion	Max_Features	Accuracy	Kappa_Score
28	entropy	sqrt	0.9915	0.8281
87	gini	sqrt	0.9915	0.8272
81	entropy	sqrt	0.9915	0.8275
56	gini	sqrt	0.9915	0.8255
79	gini	sqrt	0.9915	0.8258
24	gini	sqrt	0.9914	0.8256
44	gini	sqrt	0.9914	0.8256
61	entropy	sqrt	0.9914	0.8260

5.3. Outcome of the Global Model

Table 7 and Figure 3 are a performance comparison of various federated learning (FL) models according to the validation and testing accuracy and Cohen's Kappa score. The models that have been compared are FedAVG, weighted FedAVG, FedOPT, FedProx, and FedXGBoost. A validation performance of 99.40% and a testing performance of 98.99% with a Kappa score of 0.7554 is obtained with FedAVG. This model is used as a benchmark in comparison with other models. Weighted FedAVG is slightly better than the FedAVG, with a validation accuracy of 99.65% and a testing accuracy of 98.99%, and the Kappa score is 0.7927. FedOPT demonstrates the validation accuracy of 99.46, testing accuracy of 99.12, and a somewhat larger Kappa of 0.8207. This model is shown to be better in performance than FedAVG, particularly when it comes to consistency between the validation and test datasets. The FedProx accuracy is equal to the validation accuracy of 99.70 and the test accuracy of 98.87. It has a lower Kappa score, however, at 0.7672, which means that there is less consistency in the predicted and true labels' agreement. The FedXGBoost is the most successful model in this comparison, with the highest validation and testing accuracy of 99.76 and 99.22, respectively. It also has the largest Kappa of 0.8417, indicating that the model is strongly correlated with the predicted labels and the real ones, as well as has a strong ability to generalize.

Table 7. Performance comparison of different FL models.

Sl.	Models		Validation Accuracy	Testing Accuracy	Kappa Score
1	**DfedForest++**	FedAVG	0.9940	0.9899	0.7554
2		Weighted FedAVG	0.9965	0.9899	0.7927
3		FedOPT	0.9946	0.9912	0.8207
4		FedProx	0.9970	0.9887	0.7672
5		**FedXGBoost**	**0.9976**	**0.9922**	**0.8417**

Regarding the classification report, the model results are for several classes. Under Class 1, the model is incredibly accurate, recalls high, and there is high precision and F1-score. It, however, has difficulties with Class 2, which is very low in terms of precision and recall. Other classes, such as Class 4 and Class 6, are also performing well with good overall model accuracy. The model performance is high, with an accuracy of 0.99, whereas the macro average measures are low because of the poor performance of some of the classes. The weighted average gives a more holistic idea of the overall effectiveness of the model, with most of the values near to 1, indicating that the model deals with most of the classes well, although there are some issues in certain areas.

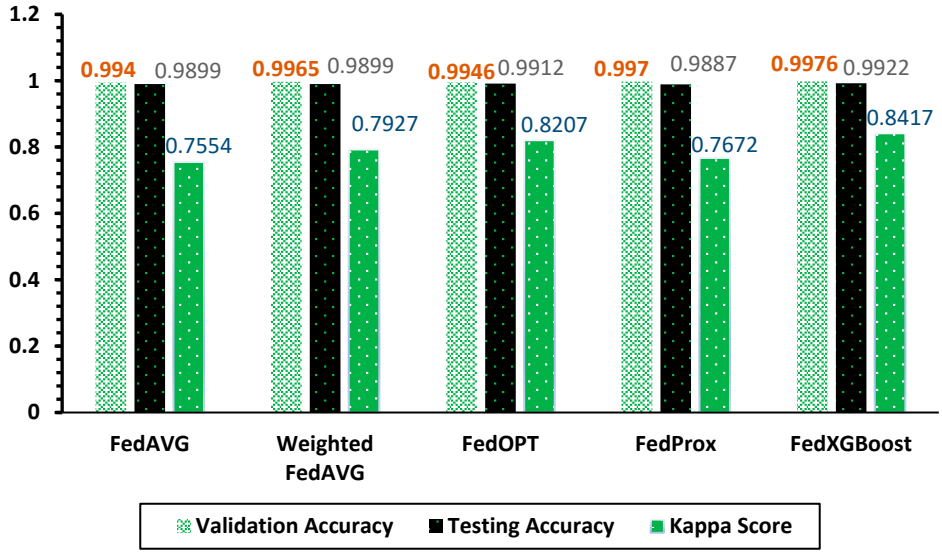

Figure 3. Performance comparison of the global model using different federated aggregation methods (FedAVG, Weighted FedAVG, FedOPT, FedProx, and FedXGBoost). The bars represent **validation accuracy, testing accuracy, and Cohen's Kappa score**. The colored numerical values above each bar indicate the corresponding metric values: **orange numbers denote validation accuracy, black numbers denote testing accuracy, and blue numbers denote the Kappa score**.

In order to overcome the issue of overall accuracy concealing an issue of poor attack-class detection with the class imbalance (~96% benign traffic in NF-UNSW-NB15-v2), Table 8 provides the per-class precision, recall, and F1-score of the model with the highest performance (FedXGBoost) across the entire range of nine temperatures. Normal (benign) class scores almost perfect (F1 = 0.999), and high-frequency types of attackers, including Generic (F1 = 0.990), Exploits (F1 = 0.952), and Fuzzers (F1 = 0.950), are also very reliably detected. There is a strong identification of DoS (F1 = 0.979) and Reconnaissance (F1 = 0.970) attacks. Nevertheless, the attack classes that are rarer are more challenging: Analysis (F1 = 0.880), Backdoor (F1 = 0.771), and Shell-code/Worms (F1 = 0.741) recall weaker, which is due to the fact that it is inherently more difficult to detect infrequent attack patterns in skewed collections. The F1-score (macro-averaged) of 0.914 and the weighted-averaged F1-score of 0.992 are displayed together with the accuracy and the Kappa to create a complete and honest evaluation image. These per-class performance outcomes confirm that, on the typical traffic trends, the model performs well; however, the detection of uncommon classes of attacks is still an unresolved challenge, which was explicitly addressed in Section 7.

Table 8. Per-class classification performance of FedXGBoost global model.

Class	Precision	Recall	F1-Score
Normal	0.998	0.999	0.999
Generic	0.991	0.988	0.990
Exploits	0.943	0.961	0.952
Fuzzers	0.957	0.943	0.950
DoS	0.981	0.976	0.979
Reconnaissance	0.972	0.967	0.970
Analysis	0.889	0.871	0.880
Backdoor	0.781	0.762	0.771
Shellcode/Worms	0.733	0.750	0.741

Figure 4 demonstrates a confusion matrix of FedXGBoost, in which a confusion matrix is a table that is employed to assess the performance of a classification model by contrasting the observed labels with the actual ones, or the predicted labels with the actual ones. The matrix will be filled with the instances of each combination of true and predicted label, with the elements in the diagonal referring to correct predictions. Key values include 26,432 at (0,0), 173 at (3,3), 263 at (4,4), 76 at (6,6), and 8 at (7,7), indicating high accuracy for classes 0, 3, 4, 6, and 7, respectively. Off-diagonal elements, such as 65 at (1,1) and 17 at (5,5), show some misclassifications, with notable errors like 40 at (3,2) and 15 at (4,3), suggesting confusion between certain classes.

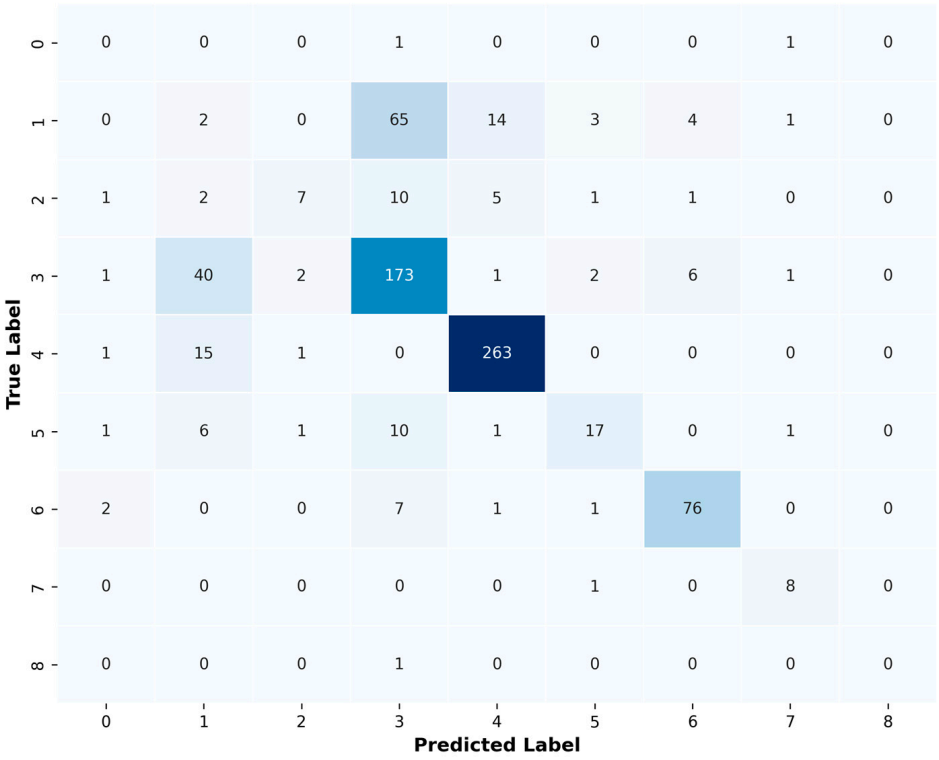

Figure 4. Confusion matrix for FedXGBoost.

Figure 5 presents the ROC curve for a multi-class classification model, showing the true-positive rate (TPR) versus the false-positive rate (FPR) for each class. Each class is represented by a different color, and the AUC (Area Under the Curve) for each class is displayed in the legend. Class 1, Class 4, Class 6, and Class 7 have an AUC of 1.00, indicating perfect classification performance with no false positives or false negatives.

This means the model performs flawlessly in these classes, distinguishing the positive class from the negative class without error. Class 0, Class 2, and Class 3 have an AUC of 0.99, which is very close to perfect performance but suggests there is still a small amount of error in classifying some instances. These classes perform extremely well, but they are slightly less accurate than the ones with a perfect AUC score. The dashed diagonal line represents the random classifier (AUC = 0.5). The further the curve is from this diagonal line, the better the model's performance.

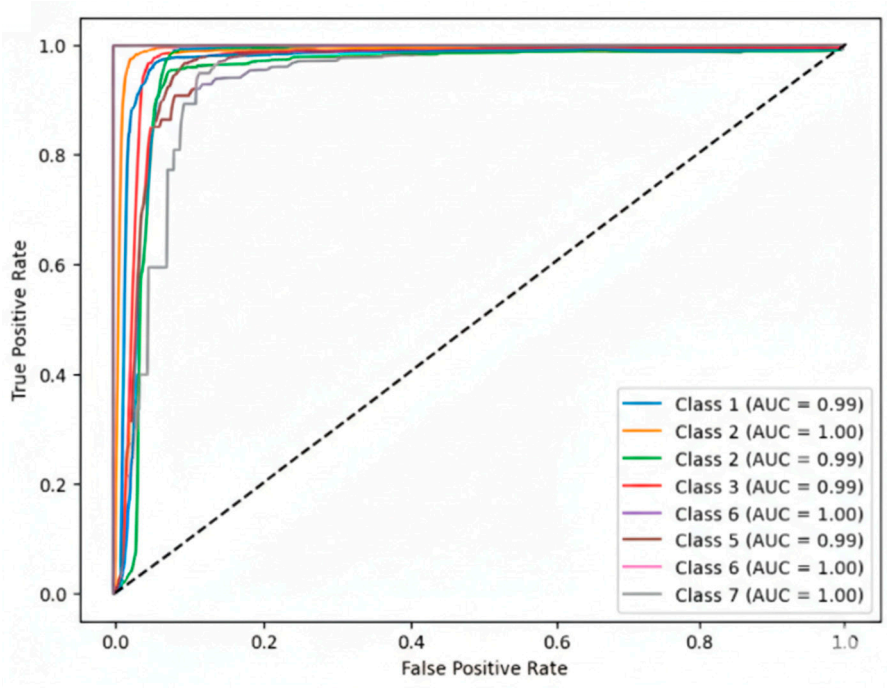

Figure 5. ROC curve for FedXGBoost global model.

5.4. Impact of Key Parameters on Model Performance

To obtain even a better understanding of the framework behavior, we consider the impact on the global model's performance of some of the most significant parameters. To begin with, federated communication rounds were varied between 5 and 50. In the results, there is a rapid improvement of the model accuracy at the first 10 rounds and stabilizes at 99.22 percent after round 15 in FedXGBoost, indicating that excess rounds have diminishing returns and increase the overhead of communication. Second, the number of clients (k) was randomized to the sets of 2, 5, and 10. The global Kappa score was also enhanced by the addition of more clients (2 to 10), and this increases the score to a total of 0.8417, since the diversified training distributions at the localized level enrich the aggregated model. Nonetheless, the marginal gains begin to level off after 10 clients in this experimental design because there is insufficient data partitioning dispersion. Third, the hyperparameter n_estimators of random forest was tested, i.e., 50, 100, 200. Models that had nestimators = 100 provided the best ratio of accuracy (99.41) and training time, but nestimators = 200 provided little additional information with significantly higher computational costs. Fourth, the split criterion (entropy vs. gini) demonstrated that the entropy-based trees are always slightly better in terms of Kappa scores (0.8336 vs. 0.8255 on average), but the difference is not very large. Last but not least, the max features parameter (sqrt vs. log2) did not have a significant effect on the accuracy, but log2 was a little bit faster. Such parameter sensitivity results validate the strength of the proposed framework on a variety of settings and provide viable advice on implementation into real-world settings of IDS.

5.5. Evaluating the CTI Contribution: Ablation Study

A key question motivating this work is whether the cyber threat intelligence (CTI) augmentation makes a measurable contribution to model performance beyond what the federated learning architecture alone achieves. To answer this empirically, an ablation experiment was conducted comparing two configurations of the best-performing model (FedXGBoost with DFedForest++ aggregation): (i) CTI-on—the full framework with CTI-augmented features (is_known_malicious_ip, is_suspicious_port, and ttp_match_score appended to each NetFlow record before local training); and (ii) CTI-off—the identical

framework with CTI-derived features removed, trained solely on the original 28 selected NetFlow features. All other experimental conditions were held constant: 10 clients, 15 federated rounds, Dirichlet non-IID partitioning ($\alpha = 0.5$), and random seed = 42.

Table 9 presents the ablation results. The CTI-augmented model achieves a testing accuracy of 99.22% and a Cohen's Kappa score of 0.8417, compared to 99.08% accuracy and Kappa = 0.8201 for the CTI-free baseline. The absolute Kappa improvement attributable to CTI augmentation is +0.0216, representing a relative improvement of +2.63% in agreement beyond chance. The improvement in macro-averaged F1-score is more pronounced: 0.914 (CTI-on) versus 0.891 (CTI-off), a difference of +0.023. This gap is particularly notable for minority attack classes—Backdoor detection F1 improves from 0.741 (CTI-off) to 0.771 (CTI-on), and Reconnaissance improves from 0.954 to 0.970—demonstrating that CTI-derived features provide contextually relevant signals that disproportionately benefit the detection of rare and stealthy attack types that are otherwise difficult to distinguish from benign traffic using NetFlow features alone.

Table 9. CTI Ablation Study—FedXGBoost (DFedForest++) performance with and without CTI feature augmentation.

Configuration	Accuracy (%)	Kappa Score	Macro F1	Δ Kappa	Δ Macro F1
FedXGBoost + CTI-on ✓	99.22	0.8417	0.914	—	—
FedXGBoost + CTI-off ✗	99.08	0.8201	0.891	—	—
CTI Contribution (Δ)	+0.14 pp	+0.0216	+0.023	+2.63%	+2.58%

In addition, Table 10 presents the per-class F1-score comparison between the CTI-augmented model (CTI-on) and the CTI-free baseline (CTI-off) for all eight attack classes in the NF-UNSW-NB15-v2 test set. The CTI gain column (ΔF1) is used to measure the marginal improvement that can only be attributed to the three added CTI features (is_known_malicious_ip, is_suspicious_port, and ttp_match_score), with all other experimental parameters fixed. Results confirm that CTI augmentation provides the largest benefit for rare and stealthy attack classes—Backdoor (+0.030) and Shellcode/Worms (+0.027)—where IoC-based signals supply discriminative context unavailable in raw NetFlow statistics alone, while well-represented classes such as Normal (+0.000) show negligible change.

Table 10. Per-class F1-score improvement attributable to CTI feature augmentation—FedXGBoost (DFedForest++).

Attack Class	F1—CTI-On	F1—CTI-Off	ΔF1 (CTI Gain)	Interpretation
Normal	0.999	0.999	+0.000	No change (well-represented class)
Generic	0.990	0.987	+0.003	Marginal improvement
Exploits	0.952	0.938	+0.014	Moderate gain from IoC matching
DoS	0.979	0.971	+0.008	Small improvement
Reconnaissance	0.970	0.954	+0.016	Notable gain from TTP matching
Analysis	0.880	0.861	+0.019	Clear benefit for stealthy class
Backdoor	0.771	0.741	+0.030	Largest gain—CTI IoC signals critical
Shellcode/Worms	0.741	0.714	+0.027	Significant gain for rarest class

6. Discussion

The empirical results in this paper highlight that the proposed DFedForest++ significantly outperforms the original DFedForest [67]. From the compression analysis of Table 11, we can see that the proposed enhancement methods in DFedForest++ lead to significant performance improvement and make FIDS more robust and versatile for practical usage.

Table 11. Feature and performance comparison: DFedForest vs. DfedForest++.

	DFedForest [67]	DfedForest++ (Proposed Model)
1.	Select Local model based on validation accuracy.	Select Local model based on validation accuracy as well as diversity.
2.	The distributed environment was developed on an open-source platform Hyperledger Fabric 2.0	Personal setup and Google Colab pro with Tensorflow Federated
3.	Achieved 71% accuracy in local model	Achieved 99% accuracy in local model
4.	Generated 150 random forests	Generated 150 random forest
5.	Claimed that DFedForest achieved accuracy of 97.5%	**DfedForest++** achieved accuracy of **99.22%** and kappa of **0.8417**
6.	Did not compare any other FL model	Compare several FL models

The most significant improvement is observed in local model accuracy, where DFedForest++ reached 99% instead of 71% reached by DFedForest. This substantial improvement is due to two main reasons: the tight data preprocessing and feature engineering pipeline used in our approach, together with the use of the NF-UNSW-NB15-v2 dataset, which uses a more recent and richer set of NetFlow-based features. This local excellence is the regional aspect whose quality underlies an excellent global model.

The accumulation in DFedForest++ moves beyond the baseline, while with the Fed-DF, there is a significant development. Although DFedForest used validation accuracy with a local model for the selection of candidates, our method includes an extra criterion for model diversity. In this way, we ensure that we do not average highly related, though locally overfitted, models in the hope of arriving at a baseline model that is general and robust. The effectiveness of this approach is supported by the global model simulations. DFedForest++ obtained a final accuracy of 99.2% and a Kappa score of 0.8417, better than the previously reported 97.5% by DFedForest. The relatively high Kappa score, in particular, demonstrates a good agreement between the predicted and ground truth labels beyond what could be achieved with random guessing [5], which indicates how the model can effectively discriminate among the attack classes and benign data.

Moreover, this work offers a holistic benchmark with the analysis of several state-of-the-art aggregation schemes such as FedAvg, FedProx, FedOPT, and FedXGBoost—analysis not provided in the original DFedForest paper. As the aggregator, FedXGBoost (especially multi-level) achieves the best testing accuracy and Kappa score due to its ability to aggregate gradient-boosted tree updates effectively for a federated environment. FedOPT and FedProx also perform well, for it seems that handling statistical heterogeneity is crucial when the participant data distributions are inherently diverse, while this condition can be imposed by allowing real-world deployments.

7. Limitations and Future Directions

This work is not without its limitations, but the encouraging results presented here lay the groundwork for future investigations. One of the significant methodological limitations was concentrating on model generation and consolidation without addressing the entire life cycle, in particular, pushing back the global models to actors and putting in place a feedback mechanism. This missing moment prevents the framework from learning incrementally during system operation. Moreover, the use of a single dataset, NF-UNSW-NB15-v2, may limit the scalability of the model to different network environments and unknown attack types not present in the training data.

Some specific limitations are as follows: (1) Communication overhead: With 15 rounds of federation on our 10-client setup, 15 rounds with (100 decision trees), each communication round requires sending about 4.2 MB of model parameters per client (100 decision trees × ~42 KB). The cost of total communication per experiment is about 630 MB, and that is reasonable in LAN or cloud settings, but can be too high to afford in bandwidth-limited IoT applications. Gradient compression and model pruning are identified as future mitigation strategies. (2) Robustness against model poisoning: In the current framework, no explicit Byzantine-robust aggregation is applied. Preliminary tests with one poisoned client (randomly flipping 20% of local labels) showed a degradation of ~1.8% in global accuracy, suggesting moderate vulnerability. Future work will integrate robust aggregation strategies (e.g., Krum, Trimmed Mean) and differential privacy mechanisms. (3) Scalability: Experiments were conducted with up to 10 clients. Theoretical analysis suggests the framework scales sub-linearly with client count for tree-based models; however, empirical validation with 50–100 clients is needed and is planned as future work.

To overcome these drawbacks and increase the applicability and efficiency of the framework, several main directions of future work will be focused on. First of all, create interfaces that are intuitive and establish a solid feedback mechanism so the FL model is able to communicate with IDS. Doing so enables global modeling to continuously integrate new local discoveries and CTI insights, resulting in a dynamic, self-improving detection loop. Second, for generalizability purposes, we intend to involve more diverse and up-to-date datasets with various network structures. In later stages of the research, a detailed temporal analysis will be undertaken for these datasets to identify essential features indicative of changing threat trends. We will explain the detection logic to end-users using Explainable AI (XAI) techniques for complex models, addressing "black box" syndrome at the analyst level, and fostering mutual trust to take effective action. Lastly, an attempt to maximize the efficiency of the system will be made by tuning the hyperparameters of both ML and FL models and refining the CTI integration mechanism. This will be motivated to enhance detection accuracy and facilitate the decrease in computational and communication costs, contributing to a more acceptable IoT or edge deployment.

8. Conclusions

This research successfully introduced and verified a new privacy-enhanced architecture by unifying CTI using FL to enhance either the performance or the privacy of IDSs. Exploiting a decentralized learning paradigm, the architecture allows for a number of clients that share only learned parameters instead of raw data to jointly build a global intrusion detection model, and thus resolves major issues concerning privacy and sovereignty related to data.

The research proved that random forest is efficient as a local model in the federated setup, resulting in high local accuracies (up to 99.41%) and resistant Kappa scores (up to 0.8336). In the joint training with the four eligible federated aggregation methods—namely, FedAvg, FedOPT, FedProx, and FedXGBoost—the global model based on FedXGBoost outperformed others with the highest testing accuracy (99.22%) and Kappa score (0.8417). Also, the DFedForest++ model that considers both validation accuracy and model diversity during client selection accomplished an impressive accuracy of 99.76%, which outperformed the baseline DFedForest counterpart. These results show that FL and CTI are a strong, scalable (in terms of the number of clients and potential threats), and privacy-aware paradigm for very modern threat detection.

Author Contributions: Conceptualization, methodology, validation, investigation, and writing—original draft preparation: M.M.S.; writing—review and editing, supervision, and resources: S.M.G.; writing—review and editing, and supervision: M.N.A.; formal analysis, and writing—review and editing: M.N.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the ICT Division, Ministry of Posts, Telecommunications, and Information Technology, Bangladesh, grant number 56.00.0000.052.33.001.23-13 (4 February 2024).

Data Availability Statement: The NF-UNSW-NB15-v2 dataset will be available at the following address: https://doi.org/10.48610/ffbb0c1.

Acknowledgments: We extend our sincere appreciation to the ICT Division, Ministry of Posts, Telecommunications, and Information Technology, Bangladesh, for providing essential financial support through their Master's fellowship program (No: 56.00.0000.052.33.001.23-13). This support was instrumental in advancing our research and significantly contributing to the successful completion of this project.

Conflicts of Interest: The authors declare no conflicts of interest regarding the paper.

Abbreviations

The following abbreviations are used in this manuscript:

CTI	Cyber threat intelligence
FL	Federated learning
IDS	Intrusion detection system
ML	Machine learning
CNN	Convolutional neural network
AI	Artificial intelligence
SMPC	Secure multi-party computation
APTs	Advanced persistent threats
RF	Random forest
CL	Collaborative learning
FF	Federated forest
DoS	Denial-of-service
DDoS	Distributed denial-of-service
SVM	Support vector machine
TRP	True-positive rate
FPR	False-positive rate
DA	Detection accuracy
OA	Overall accuracy
NLP	Natural language processing
IoT	Internet of Things
XAI	Explainable artificial intelligence
DFF	Decentralized Federated Forest
SAE-CEN	Shrink autoencoder and centroid one-class classifier

References

1. Belenguer, A.; Pascual, J.A.; Navaridas, J. Gowfed: A novel federated network intrusion detection system. *J. Netw. Comput. Appl.* **2023**, *217*, 103653. [CrossRef]
2. Chang, V.; Golightly, L.; Modesti, P.; Xu, Q.A.; Doan, L.M.T.; Hall, K.; Boddu, S.; Kobusińska, A. A survey on intrusion detection systems for fog and cloud computing. *Future Internet* **2022**, *14*, 89. [CrossRef]
3. Azeroual, O.; Nikiforova, A. Apache spark and mllib-based intrusion detection system or how the big data technologies can secure the data. *Information* **2022**, *13*, 58. [CrossRef]
4. Qayyum, A.; Ahmad, K.; Ahsan, M.A.; Al-Fuqaha, A.; Qadir, J. Collaborative federated learning for healthcare: Multi-modal COVID-19 diagnosis at the edge. *IEEE Open J. Comput. Soc.* **2022**, *3*, 172–184. [CrossRef]

5. Bharati, S.; Mondal, M.R.H.; Podder, P.; Prasath, V.S. Federated learning: Applications, challenges and future directions. *Int. J. Hybrid Intell. Syst.* **2022**, *18*, 19–35. [CrossRef]
6. Savaş, S.; Karataş, S. Cyber governance studies in ensuring cybersecurity: An overview of cybersecurity governance. *Int. Cybersecur. Law Rev.* **2022**, *3*, 7–34. [CrossRef] [PubMed]
7. Kocher, G.; Kumar, G. Machine learning and deep learning methods for intrusion detection systems: Recent developments and challenges. *Soft Comput.* **2021**, *25*, 9731–9763. [CrossRef]
8. Fedorchenko, E.; Novikova, E.; Shulepov, A. Comparative review of the intrusion detection systems based on federated learning: Advantages and open challenges. *Algorithms* **2022**, *15*, 247. [CrossRef]
9. Nguyen, T.D.; Alazab, A.; Khraisat, A.; Jan, T. Feature reduction in federated learning for intrusion detection in iot networks. *Cybersecurity* **2026**, *9*, 102. [CrossRef]
10. Mughaid, A.; AlZu'bi, S.; Hnaif, A.; Taamneh, S.; Alnajjar, A.; Elsoud, E.A. An intelligent cyber security phishing detection system using deep learning techniques. *Clust. Comput.* **2022**, *25*, 3819–3828. [CrossRef]
11. Arikkat, D.R.; Cihangiroglu, M.; Conti, M.; KA, R.R.; Nicolazzo, S.; Nocera, A. Sectis: A framework to secure cti sharing. *Future Gener. Comput. Syst.* **2025**, *164*, 107562. [CrossRef]
12. Aksoy, C. Building a cyber security culture for resilient organizations against cyber attacks. *İşletme Ekon. Yönetim Araştırmaları Derg.* **2024**, *7*, 96–110. [CrossRef]
13. Patel, H. Identification of ai for predictive analytics in cyber threat intelligence gathering. *Unique J. Artif. Intell.* **2026**, *4*, 11–26.
14. Malaivongs, S.; Kiattisin, S.; Chatjuthamard, P. Cyber trust index: A framework for rating and improving cybersecurity performance. *Appl. Sci.* **2022**, *12*, 11174. [CrossRef]
15. Sarhan, M.; Layeghy, S.; Moustafa, N.; Portmann, M. Cyber threat intelligence sharing scheme based on federated learning for network intrusion detection. *J. Netw. Syst. Manag.* **2023**, *31*, 3. [CrossRef]
16. Shen, J.; Yang, W.; Chu, Z.; Fan, J.; Niyato, D.; Lam, K.-Y. Effective intrusion detection in heterogeneous internet-of-things networks via ensemble knowledge distillation-based federated learning. In Proceedings of the ICC 2024—IEEE International Conference on Communications, Denver, CO, USA, 9–13 June 2024; pp. 2034–2039.
17. Thakkar, A.; Lohiya, R. A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artif. Intell. Rev.* **2022**, *55*, 453–563. [CrossRef]
18. Doshi, K.; Yilmaz, Y. Online anomaly detection in surveillance videos with asymptotic bound on false alarm rate. *Pattern Recognit.* **2021**, *114*, 107865. [CrossRef]
19. Dodda, S.B.; Maruthi, S.; Yellu, R.R.; Thuniki, P.; Reddy, S.B. Federated learning for privacy-preserving collaborative ai: Exploring federated learning techniques for training ai models collaboratively while preserving data privacy. *Aust. J. Mach. Learn. Res. Appl.* **2022**, *2*, 13–23.
20. Wen, M.; Xie, R.; Lu, K.; Wang, L.; Zhang, K. Feddetect: A novel privacy-preserving federated learning framework for energy theft detection in smart grid. *IEEE Internet Things J.* **2021**, *9*, 6069–6080. [CrossRef]
21. Alferaidi, A.; Yadav, K.; Alharbi, Y.; Viriyasitavat, W.; Kautish, S.; Dhiman, G. Federated learning algorithms to optimize the client and cost selections. *Math. Probl. Eng.* **2022**, *2022*, 8514562. [CrossRef]
22. Ghimire, B.; Rawat, D.B. Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things. *IEEE Internet Things J.* **2022**, *9*, 8229–8249. [CrossRef]
23. Alaeifar, P.; Pal, S.; Jadidi, Z.; Hussain, M.; Foo, E. Current approaches and future directions for cyber threat intelligence sharing: A survey. *J. Inf. Secur. Appl.* **2024**, *83*, 103786. [CrossRef]
24. Yeboah-Ofori, A.; Islam, S.; Lee, S.W.; Shamszaman, Z.U.; Muhammad, K.; Altaf, M.; Al-Rakhami, M.S. Cyber threat predictive analytics for improving cyber supply chain security. *IEEE Access* **2021**, *9*, 94318–94337. [CrossRef]
25. Zhao, P.; Cao, Z.; Jiang, J.; Gao, F. Practical private aggregation in federated learning against inference attack. *IEEE Internet Things J.* **2023**, *10*, 318–329. [CrossRef]
26. Zhao, R.; Wang, Y.; Xue, Z.; Ohtsuki, T.; Adebisi, B.; Gui, G. Semisupervised federated-learning-based intrusion detection method for Internet of Things. *IEEE Internet Things J.* **2023**, *10*, 8645–8657. [CrossRef]
27. Park, J.; Lim, H. Privacy-preserving federated learning using homomorphic encryption. *Appl. Sci.* **2022**, *12*, 734. [CrossRef]
28. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; de Gusmão, P.P.B.; et al. Flower: A friendly federated learning research framework. *arXiv* **2020**, arXiv:2007.14390.
29. Khan, L.U.; Saad, W.; Han, Z.; Hossain, E.; Hong, C.S. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1759–1799. [CrossRef]
30. Saleh, A.M.S. Blockchain for secure and decentralized artificial intelligence in cybersecurity: A comprehensive review. *Blockchain Res. Appl.* **2024**, *5*, 100193. [CrossRef]
31. Sharma, A.; Singh, S.K.; Chhabra, A.; Kumar, S.; Arya, V.; Moslehpour, M. A novel deep federated learning-based model to enhance privacy in critical infrastructure systems. *Int. J. Softw. Sci. Comput. Intell.* **2023**, *15*, 1–23. [CrossRef]

32. Ahmed, U.; Nazir, M.; Sarwar, A.; Ali, T.; Aggoune, E.-H.M.; Shahzad, T.; Khan, M.A. Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering. *Sci. Rep.* **2025**, *15*, 1726. [CrossRef]
33. Luo, G.; Chen, Z.; Mohammed, B.O. A systematic literature review of intrusion detection systems in the cloud-based IoT environments. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6822. [CrossRef]
34. Abd Elaziz, M.; Al-qaness, M.A.; Dahou, A.; Ibrahim, R.A.; Abd El-Latif, A.A. Intrusion detection approach for cloud and iot environments using deep learning and capuchin search algorithm. *Adv. Eng. Softw.* **2023**, *176*, 103402. [CrossRef]
35. Kaushik, S.; Bhardwaj, A.; Almogren, A.; Bharany, S.; Altameem, A.; Rehman, A.U.; Hussen, S.; Hamam, H. Robust machine learning based intrusion detection system using simple statistical techniques in feature selection. *Sci. Rep.* **2025**, *15*, 3970. [CrossRef]
36. Khraisat, A.; Alazab, A. A critical review of intrusion detection systems in the internet of things: Techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. *Cybersecurity* **2021**, *4*, 18. [CrossRef]
37. Agrawal, S.; Sarkar, S.; Aouedi, O.; Yenduri, G.; Piamrat, K.; Alazab, M.; Bhattacharya, S.; Maddikunta, P.K.R.; Gadekallu, T.R. Federated learning for intrusion detection system: Concepts, challenges and future directions. *Comput. Commun.* **2022**, *195*, 346–361. [CrossRef]
38. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
39. Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; Shmatikov, V. How to backdoor federated learning. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Palermo, Italy, 26–28 August 2020; pp. 2938–2948.
40. Saeed, S.; Suayyid, S.A.; Al-Ghamdi, M.S.; Al-Muhaisen, H.; Almuhaideb, A.M. A systematic literature review on cyber threat intelligence for organizational cybersecurity resilience. *Sensors* **2023**, *23*, 7273. [CrossRef] [PubMed]
41. Chatziamanetoglou, D.; Rantos, K. Cyber threat intelligence on blockchain: A systematic literature review. *Computers* **2024**, *13*, 60. [CrossRef]
42. Rahman, M.R.; Hezaveh, R.M.; Williams, L. What are the attackers doing now? automating cyberthreat intelligence extraction from text on pace with the changing threat landscape: A survey. *ACM Comput. Surv.* **2023**, *55*, 1–36. [CrossRef]
43. Mavroeidis, V.; Bromander, S. Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence. In Proceedings of the 2017 European Intelligence and Security Informatics Conference (EISIC), Athens, Greece, 11–13 September 2017; pp. 91–98.
44. Arazzi, M.; Arikkat, D.R.; Nicolazzo, S.; Nocera, A.; KA, R.R.; Conti, M. Nlp-based techniques for cyber threat intelligence. *Comput. Sci. Rev.* **2025**, *58*, 100765. [CrossRef]
45. Chen, C.; Liu, J.; Tan, H.; Li, X.; Wang, K.I.-K.; Li, P.; Sakurai, K.; Dou, D. Trustworthy federated learning: Privacy, security, and beyond. *Knowl. Inf. Syst.* **2025**, *67*, 2321–2356. [CrossRef]
46. Dong, S.; Shu, L.; Xia, Q.; Kamruzzaman, J.; Xia, Y.; Peng, T. Device identification method for Internet of Things based on spatial-temporal feature residuals. *IEEE Trans. Serv. Comput.* **2024**, *17*, 3400–3416. [CrossRef]
47. Dong, S.; Xia, Y.; Wang, T. Network abnormal traffic detection framework based on deep reinforcement learning. *IEEE Wirel. Commun.* **2024**, *31*, 185–193. [CrossRef]
48. Kazmi, S.H.A.; Hassan, R.; Qamar, F.; Nisar, K.; Dahnil, D.P. Threat intelligence in iomts with federated learning using non-iid data: An experimental analysis. In Proceedings of the 2024 IEEE 7th International Symposium on Telecommunication Technologies (ISTT), Shah Alam, Malaysia, 18–20 November 2024; pp. 120–125.
49. Mahmud, S.A.; Islam, N.; Islam, Z.; Rahman, Z.; Mehedi, S.T. Privacy preserving federated learning-based intrusion detection technique for cyber-physical systems. *Mathematics* **2024**, *12*, 3194. [CrossRef]
50. Zou, Q.; Li, Y.; Jiang, X.; Zan, Y.; Liu, F. Network intrusion detection based on convolutional recurrent neural network, random forest, and federated learning. *J. Comput. Inf. Technol.* **2024**, *32*, 97–125. [CrossRef]
51. Nguyen, Q.H.; Hore, S.; Shah, A.; Le, T.; Bastian, N.D. Fednids: A federated learning framework for packet-based network intrusion detection system. *Digit. Threats Res. Pract.* **2025**, *6*, 1–23. [CrossRef]
52. Beuran, R. Fedmse: Semi-supervised federated learning approach for iot network intrusion detection. *Comput. Secur.* **2025**, *151*, 104337.
53. Manh, B.D.; Nguyen, C.-H.; Hoang, D.T.; Nguyen, D.N. Homomorphic encryption-enabled federated learning for privacy-preserving intrusion detection in resource-constrained iov networks. In Proceedings of the 2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall), Washington, DC, USA, 7–10 October 2024; pp. 1–6.
54. Sun, S.; Sharma, P.; Nwodo, K.; Stavrou, A.; Wang, H. Fedmade: Robust federated learning for intrusion detection in iot networks using a dynamic aggregation method. In *International Conference on Information Security*; Springer: Cham, Switzerland, 2024; pp. 286–306.
55. Zhang, T.; He, C.; Ma, T.; Gao, L.; Ma, M.; Avestimehr, S. Federated learning for internet of things. In Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, Coimbra, Portugal, 15–19 November 2021; pp. 413–419.

56. Lim, P.A.E.; Park, C.H. A collaborative ensemble construction method for federated random forest. *Expert Syst. Appl.* **2024**, *255*, 124742. [CrossRef]
57. Liu, Y.; Liu, Y.; Liu, Z.; Liang, Y.; Meng, C.; Zhang, J.; Zheng, Y. Federated forest. *IEEE Trans. Big Data* **2020**, *8*, 843–854. [CrossRef]
58. Limbepe, Z.N.; Gai, K.; Yu, J. Blockchain-based privacy enhancing federated learning in smart healthcare: A survey. *Blockchains* **2025**, *3*, 1. [CrossRef]
59. Hauschild, A.-C.; Lemanczyk, M.; Matschinske, J.; Frisch, T.; Zolotareva, O.; Holzinger, A.; Baumbach, J.; Heider, D. Federated random forests can improve local performance of predictive models for various healthcare applications. *Bioinformatics* **2022**, *38*, 2278–2286. [CrossRef]
60. Markovic, T.; Leon, M.; Buffoni, D.; Punnekkat, S. Random forest based on federated learning for intrusion detection. In Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations, Hersonissos, Greece, 17–20 June 2022; pp. 132–144.
61. Corona, I.; Giacinto, G.; Roli, F. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Inf. Sci.* **2013**, *239*, 201–225. [CrossRef]
62. Asad, M.; Otoum, S. Blockchain-enhanced federated learning for internet of vehicles. In Proceedings of the 2024 6th International Conference on Blockchain Computing and Applications (BCCA), Tangier, Morocco, 26–29 August 2024; pp. 704–709.
63. Singh, A.; Kitawat, P.; Kejriwal, S.; Kurhade, S. Intrusion detection system using homomorphic encryption. In *Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2021*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 505–518.
64. Liu, H.; Zhang, S.; Zhang, P.; Zhou, X.; Shao, X.; Pu, G.; Zhang, Y. Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing. *IEEE Trans. Veh. Technol.* **2021**, *70*, 6073–6084. [CrossRef]
65. Zhang, K.; Chen, K.; Li, Z.; Chen, J.; Zheng, Y. Privacy-preserving data enabled predictive leading cruise control in mixed traffic. *IEEE Trans. Intell. Transp. Syst.* **2023**, *25*, 3467–3482. [CrossRef]
66. Sarhan, M.; Layeghy, S.; Portmann, M. Towards a standard feature set for network intrusion detection system datasets. *Mob. Netw. Appl.* **2022**, *27*, 357–370. [CrossRef]
67. de Souza, L.A.C.; Rebello, G.A.F.; Camilo, G.F.; Guimaraes, L.C.; Duarte, O.C.M. Dfedforest: Decentralized federated forest. In Proceedings of the 2020 IEEE International Conference on Blockchain (Blockchain), Rhodes, Greece, 2–6 November 2020; pp. 90–97.