



Fenics: A Modular Framework for Security Evaluation in Decentralized Federated Learning

Downloaded from: <https://research.chalmers.se>, 2025-07-01 11:37 UTC

Citation for the original published paper (version of record):

Saha, S., Nova, S., Duvignau, R. et al (2025). Fenics: A Modular Framework for Security Evaluation in Decentralized Federated Learning. DEBS '25: Proceedings of the 19th ACM International Conference on Distributed and Event-based Systems: 146-151.
<http://dx.doi.org/10.1145/3701717.3730550>

N.B. When citing this work, cite the original published paper.



Fenics: A Modular Framework for Security Evaluation in Decentralized Federated Learning

Shubham Saha

Department of Computer Science and Engineering
Chalmers University of Technology
Göteborg, Västra Götaland, Sweden
shubhams@chalmers.se

Romaric Duvignau

Department of Computer Science and Engineering
Chalmers University of Technology
Göteborg, Västra Götaland, Sweden
duvignau@chalmers.se

Sifat Nawrin Nova

Department of Computer Science and Engineering
Chalmers University of Technology
Göteborg, Västra Götaland, Sweden
sifatn@chalmers.se

Carla Fabiana Chiasserini

Department of Computer Science and Engineering
Chalmers University of Technology
Göteborg, Västra Götaland, Sweden
Department of Electronics and Telecommunications
Politecnico di Torino
Turin, Metropolitan City of Turin, Italy
carla.chiasserini@polito.it

Abstract

This paper presents Fenics, a modular framework for evaluating the resilience of Decentralized Federated Learning (DFL) networks under adversarial conditions. As a nascent field, DFL raises security challenges in decentralized network settings under adversarial behaviors. To our knowledge, Fenics is the first fully open-source framework of its kind, enabling user-defined topologies, multiple communication protocols, and customizable attack models to study how malicious node placement affects network performance. It integrates core components of DFL, including data distribution, dynamic node participation, and aggregation to establish the DFL architecture. We demonstrate the framework's capabilities through different use cases under poisoning and delay attacks using the FashionMNIST dataset. The results validate its capability to reveal how node placement affects performance and expose network vulnerabilities. For example, poisoning attacks exhibit topology-dependent impacts, with accuracy dropping by over 55% in certain scenarios, leading to derailed convergence. Additionally, the extensive logging features of the framework enable post-simulation analysis and insightful interpretation. Its modular architecture, simple deployment, and customizable options make it a lightweight yet useful tool for in-depth research on DFL network security.

CCS Concepts

• **Networks** → **Cyber-physical networks; Peer-to-peer networks**; • **Computing methodologies** → **Distributed simulation**; • **Security and privacy** → **Distributed systems security**.

Keywords

Decentralized Federated Learning, Framework, Malicious nodes, Convergence



This work is licensed under a Creative Commons Attribution 4.0 International License.
DEBS '25, Gothenburg, Sweden
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1332-3/25/06
<https://doi.org/10.1145/3701717.3730550>

ACM Reference Format:

Shubham Saha, Sifat Nawrin Nova, Romaric Duvignau, and Carla Fabiana Chiasserini. 2025. Fenics: A Modular Framework for Security Evaluation in Decentralized Federated Learning. In *The 19th ACM International Conference on Distributed and Event-based Systems (DEBS '25)*, June 10–13, 2025, Gothenburg, Sweden. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3701717.3730550>

1 Introduction

Federated Learning (FL), introduced by Google in 2016, enables collaborative model training without sharing raw data, enhancing privacy and security [1]. Decentralized Federated Learning (DFL) extends FL in 2018 by eliminating the need for a central server [10]. In DFL networks, nodes communicate directly with each other while handling both training and aggregation locally. This decentralized structure protects local data privacy, reduces communication overhead, and eliminates single points of failure. However, this evolving technology introduces new security challenges as well. Without central coordination, the system can be more susceptible to attacks from malicious nodes, significantly impacting network performance. For example, in model poisoning attack, a compromised node can deliberately manipulate its local model parameters by injecting significant random noise after local training. Later, these poisoned models are shared during the aggregation phase to degrade overall network performance, cause misclassification, or lead to convergence on suboptimal solutions [15]. Similarly, in delay attacks, nodes intentionally postpone transmitting their updated model parameters during the model exchange phase, which disrupts the overall network performance [4]. As each node independently aggregates updates from its neighbors, the topology significantly influences how attacks propagate through the network as well. This makes the placement of malicious nodes a critical factor in DFL architectures, where a single compromised node can jeopardize the entire network. Despite being a nascent field, several contemporary solutions have contributed to the advancement of DFL. For instance, protection against poisoning attacks by leveraging Blockchain [12] or privacy preservation through partial homomorphic encryption [2]. However, some frameworks lack comprehensive modules for designing network topologies and modeling attacks based on custom

Table 1: Comparison of Federated Learning Frameworks

Existing Frameworks	Architecture (DFL)	Supported Topologies	Modularity	Custom Attack Model	Lightweight	Open-source
TFF[14]	No	Star	Yes	No	Partial	Yes
Flower[3]	No	Star	Yes	No	Yes	Yes
PySyft[19]	No	N/S	Yes	No	No	Yes
FL-SEC[2]	Yes	P2P	No	No	Yes	No
P4L[2]	Yes	P2P	No	No	Yes	No
DEFEAT[5]	Yes	Random, P2P	No	No	Partial	No
Fedstellar[7]	Yes	Custom, P2P	Yes	Partial	No	Yes
EdgeFL[18]	Yes	P2P	No	No	Yes	Yes
Fenics (This paper)	Yes	Custom, P2P	Yes	Yes	Yes	Yes

N/S: Not specified by the authors.

configurations [12–14], while some require users to build the base DFL structure from scratch [3]. Additionally, these platforms tend to involve complex configurations, extensive deployment, and demand a high level of expertise. Considering these issues, this paper presents Fenics, an open-source¹ framework that provides support for comprehensive and in-depth security assessments of DFL networks across diverse adversarial settings. It provides flexible custom setups that also help to understand how different placements of adversarial nodes can affect the network and increase security risks. Real-world implications of this work include risk analysis and developing mitigation strategies. For that, the first crucial step is understanding the network’s underlying vulnerabilities, i.e., how trustworthy the network already is. Fenics enables such security evaluation by offering customizable attack strategies and simulation options. Fenics consists of modular components that enable options to customize network topologies, different attack strategies, communication protocols, machine learning models, aggregation algorithms, etc. Moreover, this framework is easy to deploy with minimal hardware configurations, which makes it lightweight.

To evaluate the effectiveness of the framework, the FashionMNIST dataset is used, and a series of scenario-based experiments are conducted by implementing two types of attacks: model poisoning and delay. The performance of the experiments is evaluated using accuracy, precision, recall, and F1-score at each round to monitor the convergence point. The findings indicate useful insights, e.g., poisoning attacks significantly degrade performance depending on attacker placement, while delay attacks cause resource consumption by extending execution time. These findings highlight the framework’s ability to analyze network resilience in DFL and validate its usability. By offering comprehensive support and features, Fenics stands as a valuable tool for students, academics, researchers, and security practitioners interested in exploring the security aspects of DFL networks. The rest of the paper is organized as follows: Section 2 reviews some existing works, Section 3 presents the framework architecture, Section 4 describes the experimental setup, Section 5 discusses the results, and lastly Section 6 concludes the paper.

2 Related Work

As FL continues to evolve, numerous frameworks have emerged to address different challenges in distributed and decentralized learning. Table 1 highlights some notable works that have contributed

to the advancements in this field. To construct the table, we systematically reviewed existing popular FL frameworks. Our selection criteria included framework architecture, security features, open-source availability, and modularity. Key criteria for comparison included support for custom topologies, attack model integration, and lightweight deployment, which are crucial for evaluating security resilience in decentralized settings. To begin with, TensorFlow Federated (TFF), Flower, and PySyft are modular and open-source frameworks that support centralized federated learning (CFL) with privacy-preserving techniques such as differential privacy and secure aggregation [3, 14, 19]. Flower emphasizes lightweight deployment for resource-constrained devices, and TFF supports flexible simulations. However, PySyft can be harder to set up, and its strong focus on privacy might make it less convenient for developers who want something easy to use and scale [18]. Some frameworks specifically focus on enhancing decentralization, privacy, and security in federated learning through specialized techniques, mostly employing peer-to-peer (P2P) topologies. For instance, FL-SEC [12] employs blockchain technology for decentralization and protection against poisoning attacks. Similarly, P4L [2] implements partial homomorphic encryption for privacy preservation and supports fully connected topologies without centralized federation. While these approaches offer robust security features, their implementation complexity can affect scalability in resource-constrained environments. DEFEAT [5] presents a hybrid approach combining decentralized and semi-decentralized architectures with secure aggregation and custom algorithms. DEFEAT’s implementation balances communication costs and model accuracy through various message exchange schemes. Although this framework demonstrates effective solutions for specific use cases, its specialized architecture may present challenges for broader applications. Fedstellar [7], currently called “Nebula”, presents another notable advancement in DFL platforms, addressing various aspects like secure aggregation and communication encryption, with the capability of deploying all types of topologies. It is an open-source platform and supports multiple features, extensible to DFL research. However, with the availability of many features, the framework requires a heavy computational setup. Lastly, EdgeFL [18] is a lightweight and open-source DFL framework focused on ease of deployment and integration in edge environments. While it performs well under standard conditions, it lacks adversarial evaluation and advanced security features, emphasizing simplicity over customization.

¹<https://github.com/Shubham-Saha/Fenics—A-Modular-Framework-for-Security-Evaluation-in-Decentralized-Federated-Learning>

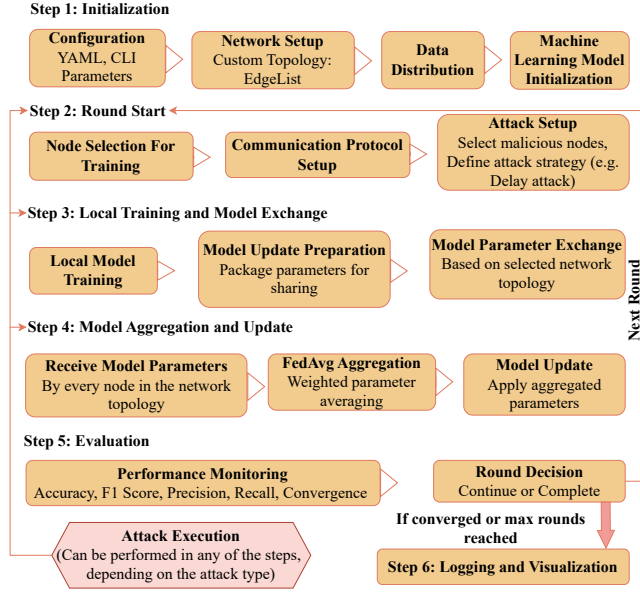


Figure 1: Architecture of Fenics.

Despite notable contributions, existing solutions often lack the flexibility and thorough security evaluation needed for decentralized FL networks. Limited customization, resource-heavy features, and the absence of open-source access further hinder usability. Hence, a necessity remains to address these limitations by proposing an accessible solution that enables comprehensive security evaluation in DFL through simple deployment and custom configuration options.

3 Architecture of Fenics

Fenics integrates core components of the DFL pipeline, including node selection, communication protocols, aggregation algorithms, and convergence techniques. This section provides an overview of its modular architecture with separated components. The key DFL components that are integrated in Fenics include:

Dynamic Node Participation: Node participation varies in DFL networks over the consecutive rounds, with the participation rate determining the proportion of active nodes during training and aggregation. To ensure universal participation of nodes while prioritizing larger datasets, MD client sampling approach is used in this framework, proposed in [11] with probabilities assigned based on local dataset sizes:

$$P_i = \frac{n_i}{\sum_{j=1}^N n_j}$$

where P_i is the probability of selecting node i , n_i is the size of node i 's dataset, and N is the total number of nodes.

Communication Mechanisms: This framework uses two different decentralized communication methods: P2P [6] and Gossip (GP) [9] protocol. P2P establishes deterministic communication patterns based on network topology, allowing nodes to exchange parameters with their immediate neighbors. Conversely, GP follows a stochastic approach where nodes randomly select peers for parameter exchange. Fenics dynamically accommodates these protocols based on network conditions and performance requirements.

Topologies: Beyond standard topologies like fully connected, ring, and star, Fenics is designed to support user-defined topologies via edgelist files, enabling custom configuration and scalable setups. This feature enhances the understanding of diverse network topologies in DFL, offering intuitive insights into network configurations.

Attack Simulation: Two fundamental attack simulation strategies: Poisoning [15] and Delay [4] are currently implemented in Fenics. The modular design also allows users to implement additional attack strategies based on specific requirements.

Model Aggregation: FedAvg is a widely adopted aggregation method in DFL that averages model parameters from all participating nodes to update their local models individually [8]. Let K denote the total number of nodes ($K=N$), n_k be the number of data samples at node k , and w_k^t be the model parameters from node k at round t . The updated parameters for node k at round $t+1$ are computed as:

$$w_k^{t+1} = \frac{1}{\sum_{j \in N_k} n_j} \sum_{j \in N_k} n_j w_j^t$$

where N_k is the set of neighboring nodes for node k , and $\sum_{j \in N_k} n_j$ is the total number of samples across k and its neighbors.

Convergence Detection Mechanism: Detecting convergence is crucial to determine model stabilization, optimizing execution time, and conserving computational resources. It is determined by analyzing improvements in average test accuracy over consecutive rounds using a moving window analysis of performance metrics. For any given round r , convergence is detected if the improvements in the average test accuracy in the previous rounds p satisfy:

$$\Delta_{r-j+1} < \theta \quad \forall j \in \{1, 2, \dots, p\}$$

where $\Delta_r = \text{accuracy}_r - \text{accuracy}_{r-1}$ represents the improvement in average test accuracy from round $r-1$ to round r , θ represents the minimum significant improvement threshold, and p is the patience.

Workflow: Figure 1 depicts the architecture of Fenics and illustrates how the components described above interact during each training round. The execution proceeds through a sequence of configurable steps to emulate realistic DFL behavior. It begins with initialization, where configuration parameters are loaded via YAML files and CLI inputs, and network topologies are established using edge list files. A Dirichlet distribution [17] is applied for non-i.i.d. data partitioning across nodes. After that, the machine learning model is initialized to establish the baseline for training. The above setup forms the foundation for the subsequent iterative training process, where nodes are selected for participation to perform local training using MD client sampling based on their respective datasets. Upon completion of the training, model parameters are exchanged according to the specified communication mechanism. After receiving model parameters from their neighbors, each node independently performs FedAvg aggregation using dataset-weighted averaging to update its local model. This creates a new baseline for the next round. Depending on the attack types, the execution of attacks can occur at different steps as illustrated in Figure 1. For instance, model poisoning occurs after local training but before parameter exchange, while delay attacks disrupt the model exchange phase.

After aggregation, the performance of the use case scenarios is evaluated using metrics such as accuracy, F1 score, precision,

Table 2: Model Training Parameters.

Layer Type	Details
Conv Layer 1	32 filters, 3×3 kernel
Conv Layer 2	64 filters, 3×3 kernel
Activation	ReLU, Softmax (Output layer)
Max-Pooling	2×2
Fully Connected Layer	128 hidden units, ReLU activation.
Output Layer	10 units
Loss Function	Cross-Entropy Loss
Optimizer	Stochastic Gradient Descent (SGD)
Learning Rate	0.01

Table 3: Default parameters used in Fenics.

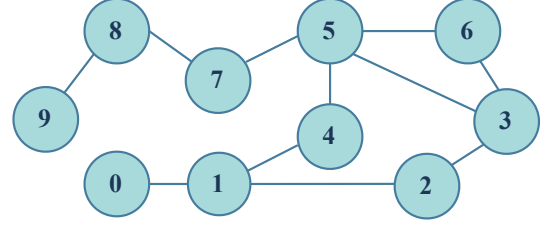
Parameter	Default Value
Dataset	FashionMNIST
Data Distribution	Dirichlet ($\alpha = 0.5$)
Number of Nodes	10
Training Rounds	15
Local Epochs	5
Participation Rate	0.7(70%)
Aggregation method	FedAvg
Convergence Threshold (θ)	0.01
Model	CNN
Topology	Custom
Patience (p)	3
Communication Protocol	P2P

and recall. This step also monitors convergence and captures the influence of adversarial behaviors across rounds. Based on the evaluation, the system decides whether to proceed to the next round or terminate training. Throughout the process, Fenics logs key metrics including training time, aggregation duration, and model performance. It also generates logging of convergence trends, class distributions, and network topologies that help in post-simulation analysis. The capabilities of Fenics are demonstrated and validated through scenario-based evaluations. The entire execution, from configurable setup to performance assessment and visualization, highlights its usability and potential to analyze diverse DFL scenarios under adversarial conditions.

4 Experimental Setup

This section demonstrates the framework’s effectiveness through experimental analyses, including the impact of malicious nodes based on the placements, with three distinct attack scenarios compared with a baseline.

System Configuration: For the evaluation of the experimental scenarios, FashionMNIST dataset, which consists of 70,000 grayscale images across 10 fashion categories [16], is used with a Convolutional Neural Network (CNN) model. The model’s training parameters are shown in Table 2. The framework is implemented in Python to support efficient and scalable development. It leverages PyTorch for local training and aggregation, while NumPy and Pandas handle data processing. NetworkX is used to simulate network topologies, and built-in logging modules track runtime behavior. Matplotlib

**Figure 2: Custom topology of 10 nodes used in our evaluation.**

and Seaborn are used to visualize performance metrics. Python’s multiprocessing and threading libraries are used to simulate decentralized processing that allows parallel execution for each node’s training and concurrent update exchanges. The default parameters used in Fenics are summarized in Table 3, which serves as a baseline for custom configurations. The concentration parameter α in Dirichlet distribution controls the degree of heterogeneity, where lower values $\alpha < 1$ produce more skewed distributions [17]. A custom topology comprising 10 nodes, as illustrated in Figure 2 is designed to evaluate the impact of attacks based on the node placements. Three distinct attack scenarios are simulated and executed over multiple rounds to closely resemble anonymous attack patterns observed in real-world settings. All experiments are conducted on a system equipped with an AMD Ryzen 7 processor, an NVIDIA GeForce RTX 3060 Ti GPU, and 32GB of DDR4 RAM.

Baseline Configuration: In the baseline setting, the system operates without attacks to establish reference performance metrics. This serves as a benchmark for evaluating the impact of the subsequent attack implementations.

Attack Scenarios: Fenics provides precise control over attack execution through various tunable parameters and specific assessment preferences. We design three distinct attack scenarios based on the custom topology shown in Figure 2 while retaining the baseline configuration parameters. Through these targeted attack scenarios assessing DFL network performance, the usability and capabilities of the framework are validated.

Scenario 1 (Poison attack from the Center): Node 5 is designated as a malicious node to perform poison attack. The node selection is based on its central position in the network topology to observe the attack impact on neighboring nodes in a dense area.

Scenario 2 (Poison attack from the Edge): Following the previous scenario, the malicious node is repositioned from 5 to 0 to investigate how an attacker’s position with minimum neighbor-connected affects the system.

Scenario 3 (Delay attack from both positions): Nodes 0 and 5 are performing the delay attack to evaluate the system’s behavior under temporal disruptions rather than data corruption, while the attackers are positioned at both central and edge locations.

5 Results and Discussion

The experimental evaluation using Fenics reveals several analytical insights into system behavior under various attack scenarios. This section presents a detailed analysis of the findings across different attack configurations and a comparison with the baseline.

Baseline Performance: The baseline configuration demonstrates robust performance and stable convergence, as shown in Figure 3.

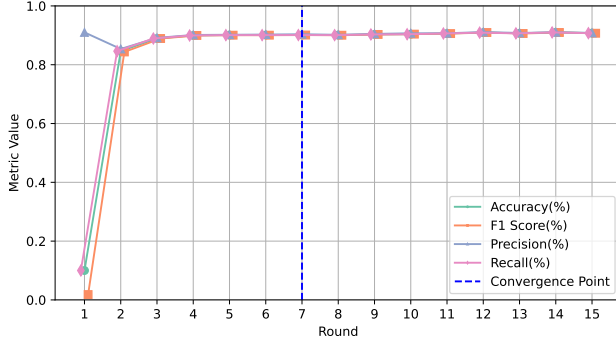


Figure 3: Performance metrics of Baseline.

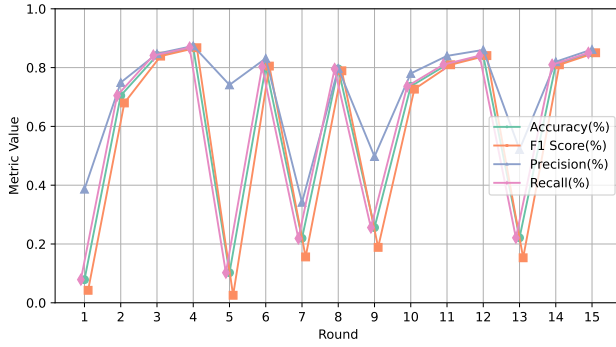


Figure 4: Performance metrics of Scenario 1.

The system achieves convergence at round 7 with an accuracy of 90.57% and maintains consistent performance. This indicates robust distributed learning behavior in the absence of attacks.

5.1 Impact of Attack Scenarios

A comparative analysis of the DFL network under three attack scenarios is presented to evaluate the impact of poisoning and delay attacks based on the placement of malicious nodes.

Performance Under Poison Attack: This analysis reveals that the placement of malicious nodes within the network significantly influences the system performance. In Scenario 1, node 5 performed poison attacks in rounds 1, 5, 7, 9, and 13 during the execution. As illustrated in Figure 4, this attack leads to substantial degradation in model performance. Using P2P communication protocol, node 5 shares its corrupted model updates with its immediate neighbors. This contamination spreads through the network, causing significant disruptions in the subsequent rounds. For example, accuracy drops from a peak of 86.84% in round 4 to 10.23% in round 5, consequently preventing the system from achieving convergence. These results underscore the critical role of centrally positioned malicious nodes in undermining the learning process. This provides a significant insight into how the placements of the nodes affect model performance. Thus, a strategic placement of an attacker can severely disrupt the global model's performance by injecting poisoned updates into key areas of the network.

In Scenario 2, the attacker is repositioned to node 0 and launches attacks in rounds 1, 3, 6, 10, and 12, as shown in Figure 5. Compared

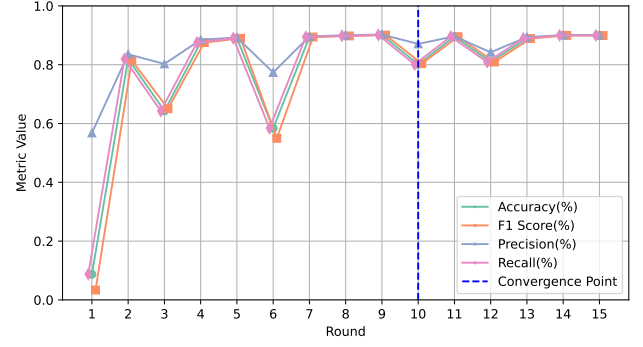


Figure 5: Performance metrics of Scenario 2.

to Scenario 1, this setup shows improved performance but still falls short of the baseline. This is because the attacker is placed at the edge of the topology, connected to only one neighbor, limiting the spread of corrupted updates. The overall performance drops due to delayed convergence compared to the baseline scenario in Figure 3. However, the model achieves convergence by round 10, maintaining an accuracy level above 80%. This analysis emphasizes that an attack from a single node with minimal connections has a limited impact. It also highlights how attack strategies, communication protocols, and topologies can significantly impact the resilience and performance of a DFL network.

Performance Under Delay Attack: In Scenario 3, both nodes 0 and 5 execute delay attacks. Although accuracy levels remain nearly identical to the baseline (Figure 3), the key impact lies in the time metrics. The deliberate postponement of model updates by these nodes introduces significant delays in the training process. As shown in Figure 6, the total round time (in seconds) is compared across multiple rounds for both the baseline and the delay attack scenario. The baseline shows relatively stable round times, ranging between 1100 and 1900 seconds with minor fluctuations. Conversely, the delay attack leads to erratic behavior, with sharp spikes in round times, particularly in rounds 3, 6, 7, and 10-12, where round durations exceed 1600 seconds. This behavior highlights the attack's effectiveness in disrupting normal operations by introducing delays and prolonging round completion times.

The results in Table 4 present a comparative view of model performance across scenarios. While the baseline maintains high accuracy and stability, Scenario 1 shows major degradation due to a centrally placed poisoning attack. In contrast, Scenario 2 remains comparatively resilient, with the attacker at a less connected edge node. Scenario 3 highlights increased round times from delay attacks, despite minimal accuracy loss. These insights are obtained through Fenics' modular attack simulation and detailed logging, which allow tracking convergence behavior, per-round metrics, and the topological influence of malicious nodes. This exhibits the effectiveness of Fenics in the security evaluation of DFL by facilitating network performance observation under adversarial settings.

6 Conclusion

This paper presents Fenics, an open-source and comprehensive framework for security assessment in DFL networks with a modular architecture. This framework is designed to be lightweight and

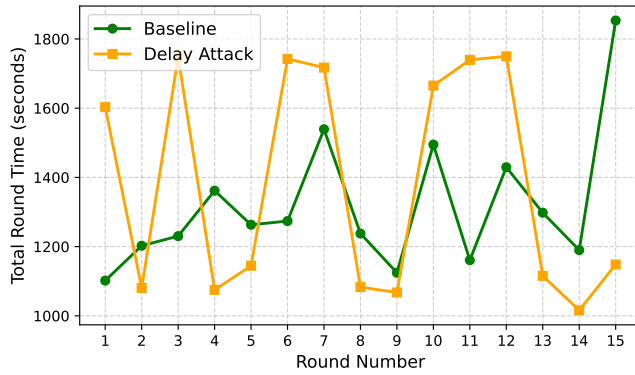


Figure 6: Comparison of Total Round Times: Baseline vs Delay Attack.

Table 4: Performance Comparison Across Different Scenarios.

Metric	Baseline	Scenario 1	Scenario 2	Scenario 3
Accuracy	0.909 ± 0.001	0.844 ± 0.010	0.885 ± 0.035	0.893 ± 0.023
F1 Score	0.908 ± 0.002	0.843 ± 0.012	0.884 ± 0.034	0.891 ± 0.023
Precision	0.909 ± 0.001	0.865 ± 0.005	0.887 ± 0.030	0.893 ± 0.023
Recall	0.908 ± 0.001	0.844 ± 0.010	0.885 ± 0.033	0.895 ± 0.022
Avg Round Time(s)	1168.81 ± 127.65	1226.41 ± 51.40	1289.10 ± 83.92	1303.26 ± 111.03

easy to configure, enabling the in-depth security research of diverse DFL scenarios. Integration of key DFL components combined with thorough analysis and visualization capabilities makes Fenics a useful and cost-effective tool. The experimental results demonstrate its effectiveness and validate its potential in analyzing DFL behavior under various scenarios, particularly on the network behaviors depending on the placement of the malicious nodes. The findings emphasize the critical relationship between network topology and attack impacts while also uncovering key trade-offs between model performance and computational efficiency. Fenics serves as a foundational tool for these extensive security evaluations, whether to analyze attack impacts, improve aggregation protocols, develop mitigation strategies, or address defensive measures. We plan to extend Fenics with support for additional attack types, adaptive defense mechanisms, and real-world deployment scenarios to strengthen its utility in secure decentralized learning. By providing a flexible platform especially tailored for DFL security evaluation, it paves the way for more resilient strategies towards secured distributed learning environments.

Acknowledgments

We appreciate the reviewers' useful feedback and suggested improvements. This framework originated from a course project focusing on malicious users in DFL during the course on data-driven support for cyber-physical systems. We want to thank Jonathan Lindqvist and Anton Lundh for being part of the original project team and for their support, which inspired this work.

References

- [1] Google AI. 2017. Federated Learning: Collaborative Machine Learning without Centralized Training Data. <https://research.google/blog/federated-learning-collaborative-machine-learning-without-centralized-training-data/> Google AI Blog.
- [2] I. Arapakis, P. Papadopoulos, K. Katevas, and D. Perino. 2023. P4L: Privacy Preserving Peer-to-Peer Learning for Infrastructureless Setups. *arXiv preprint arXiv:2302.13438* (2023). <https://arxiv.org/abs/2302.13438>
- [3] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchu Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Hei Li Kwing, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. 2020. Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390* (2020).
- [4] Xin Lou, Cuong Tran, David K.Y. Yau, Rui Tan, Hongwei Ng, Tom Zhengjia Fu, and Marianne Winslett. 2019. Learning-Based Time Delay Attack Characterization for Cyber-Physical Systems. In *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. 1–6. doi:10.1109/SmartGridComm.2019.8909732
- [5] G. Lu, Z. Xiong, R. Li, N. Mohammad, Y. Li, and W. Li. 2023. DEFEAT: A Decentralized Federated Learning Against Gradient Attacks. *High-Confidence Computing* (2023), 100128. doi:10.1016/j.hcc.2023.100128
- [6] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Jérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. 2023. Decentralized Federated Learning: Fundamentals, State of the Art, Frameworks, Trends, and Challenges. *IEEE Communications Surveys & Tutorials* 25, 4 (2023), 2983–3013. doi:10.1109/COMST.2023.3315746
- [7] Enrique Tomás Martínez Beltrán, Ángel Luis Perales Gómez, Chao Feng, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Jérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. 2024. Fedstellar: A Platform for Decentralized Federated Learning. *Expert Systems with Applications* 242 (2024), 122861. doi:10.1016/j.eswa.2023.122861
- [8] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *International Conference on Artificial Intelligence and Statistics*. <https://api.semanticscholar.org/CorpusID:14955348>
- [9] Dishita Naik, Paul Grace, Nitin Naik, Paul Jenkins, Durgesh Mishra, and Shaligram Prajapat. 2023. An Introduction to Gossip Protocol Based Learning in Peer-to-Peer Federated Learning. In *2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG)*. 1–8.
- [10] Dishita Naik and Nitin Naik. 2024. An Introduction to Federated Learning: Working, Types, Benefits and Limitations. In *Advances in Computational Intelligence Systems*, Nitin Naik, Paul Jenkins, Paul Grace, Longzhi Yang, and Shaligram Prajapat (Eds.). Springer Nature Switzerland, Cham, 3–17.
- [11] C. Ouyang, Y. Li, J. Mao, et al. 2024. Enhancing Federated Learning with Dynamic Weight Adjustment Based on Particle Swarm Optimization. *Discover Computing* 27, 35 (2024). doi:10.1007/s10791-024-09478-x
- [12] Y. Qu, C. Xu, L. Gao, Y. Xiang, and S. Yu. 2022. FLSEC: Privacy-Preserving Decentralized Federated Learning Using SignSGD for the Internet of Artificially Intelligent Things. *IEEE Internet of Things Magazine* 5 (2022), 85–90. doi:10.1109/IOTM.001.2100173
- [13] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger. 2019. BrainTorrent: A Peer-to-Peer Environment for Decentralized Federated Learning. *arXiv preprint arXiv:1905.06731* (2019). <https://arxiv.org/abs/1905.06731>
- [14] TensorFlow Federated. 2023. TensorFlow Federated. <https://www.tensorflow.org/federated> Accessed: 2025-01-31.
- [15] Geming Xia, Jian Chen, Chaodong Yu, and Jun Ma. 2023. Poisoning Attacks in Federated Learning: A Survey. *IEEE Access* 11 (2023), 10708–10722. doi:10.1109/ACCESS.2023.3238823
- [16] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *ArXiv abs/1708.07747* (2017). <https://api.semanticscholar.org/CorpusID:702279>
- [17] Mikhail Yurochkin, Mayank Agarwal, Soumya Shubhra Ghosh, Kristjan H. Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. 2019. Bayesian Nonparametric Federated Learning of Neural Networks. *ArXiv abs/1905.12022* (2019). <https://api.semanticscholar.org/CorpusID:168170092>
- [18] Hongyi Zhang, Jan Bosch, and Helena Holmström Olsson. 2025. Enabling efficient and low-effort decentralized federated learning with the EdgeFL framework. *Information and Software Technology* 178 (2025), 107600. doi:10.1016/j.infsof.2024.107600
- [19] Alexander Ziller, Andrew Trask, Antonio Lopardo, Benjamin Szymkow, Bobby Wagner, Emma Bluemke, Jean-Mickael Nounahon, Jonathan Passerat-Palmbach, Kritika Prakash, Nick Rose, Théo Ryffel, Zarreen Naawal Reza, and Georgios Kaissis. 2021. PySyft: A Library for Easy Federated Learning. Springer International Publishing, Cham, 111–139. doi:10.1007/978-3-030-70604-3_5