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

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

Anonymous Author(s)

ABSTRACT

This paper presents Phoenix, a comprehensive open-source framework for evaluating Decentralized Federated Learning (DFL) networks in the presence of malicious users. The framework implements core DFL functionalities, including data distribution strategies, dynamic node participation, model aggregation, and attack model simulation, to assess DFL network resilience under various attack scenarios. Phoenix supports multiple communication protocols and custom network topologies, providing potential avenues to investigate the impact of malicious nodes based on their placement within the network. To the best of our knowledge, Phoenix is the first fully open-source and modular framework of its kind, allowing diverse topologies and attack scenarios to be easily incorporated. We demonstrate the framework's capabilities through experimental evaluation using the FashionMNIST dataset for poisoning and delay attacks. Experimental results reveal critical insights into DFL network vulnerabilities: poisoning attacks exhibit topology-dependent impacts, leading to derailed convergence, while delay attacks significantly degrade computational efficiency by forcing nodes to idle during malicious delays. The framework's modular architecture, comprehensive visualization capabilities, and scalability make it a lightweight and accessible tool for insightful analysis of DFL network security and further research.

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, Aggregation, Convergence, Customization

ACM Reference Format:

Anonymous Author(s). 2018. Phoenix: A Modular Framework for Security Evaluation in Decentralized Federated Learning. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Federated Learning (FL), introduced by Google in 2016, enables collaborative model training without sharing raw data, enhancing privacy and security [1]. While this technique has seen widespread adoption across various domains due to its ability to leverage decentralized data while maintaining confidentiality [7], recent research has identified several vulnerabilities that can compromise its privacy guarantees[3, 12]. Decentralized Federated Learning (DFL) extends FL by eliminating the need for a central server [11]. In DFL, nodes communicate directly in a peer-to-peer network, handling both training and aggregation locally. This decentralized structure enhances privacy, reduces communication overhead, and eliminates single-point failures. However, DFL remains vulnerable to malicious users, impacting network performance. As a nascent field, various solutions have been proposed to address these challenges by simulating and assessing the DFL networks. Contemporary platforms primarily focus on implementing specific security measures or attack prevention mechanisms. However, many existing solutions do not fully support a completely decentralized structure [16]. Some frameworks lack comprehensive modules for designing network topologies and modeling attacks based on customized configurations [14–16]. Additionally, they often require extensive deployment, complex configuration, and a high level of expertise. In this paper, Phoenix is presented as a lightweight, open-source [1] platform designed to address these limitations. Phoenix aims to provide a flexible and systematic analysis of attack impacts on DFL network. With a command-line interface (CLI) and a streamlined setup, Phoenix lowers the barrier for practitioners to explore security evaluation in DFL, experiment with various scenarios, and evaluate their impact. The framework supports a wide range of datasets, and network topologies while providing extensive attack simulation capabilities to assess network resilience. Its simple configuration system makes it highly accessible, offering robust capabilities for topology, attack model customization, performance monitoring, and visualization. Phoenix aims to analyze the impact of malicious users based on the attacker node placement on the network. This motive is central to understanding how different placements of adversarial nodes can affect the overall performance and security of FL systems. To thoroughly evaluate the effectiveness and robustness of

[1]*Link to the Github repository will be provided upon the acceptance of the paper.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

Table 1: Comparison of Federated Learning Solutions

Solution	FL Architecture	Topology	Scalability	Extensibility	Custom Attack Model	Lightweight	Open-source
TFF[16]	CFL	Star	≈	≈	✗	✓	✓
BrainTorrent[15]	DFL	Fully connected	✗	≈	✗	✗	✗
FL-SEC[14]	DFL	Star	✗	≈	✗	≈	✗
P4L[2]	DFL	Fully connected	✓	≈	≈	✓	✗
DEFEAT[6]	Hybrid	Custom	✗	✗	✗	✓	✗
DFedSN[4]	Hybrid	Custom	≈	✓	✗	✗	✗
Fedstellar[8]	DFL	All types	✓	✓	✗	✗	✓
Phoenix(This paper)	DFL	All types	✓	✓	✓	✓	✓

✓Implemented, ≈Partially Implemented, ✗Not Implemented.

Phoenix, we conduct a series of experiments with various attack scenarios by implementing the most widely studied attacks like model poisoning attacks and delay attacks due to their relevance/impact on the training process [3]. To evaluate the framework, metrics like F1 score, accuracy, precision, and recall are used, to assess overall performance at every round and identify the convergence point. The findings indicate useful insights for example model poisoning attacks significantly degrade performance based on the attacker node placement on the network. In contrast, delay attacks primarily consume resources by prolonging the execution process compared to scenarios without any attacks. By focusing on attack analysis within a streamlined platform, Phoenix fills a crucial gap in current DFL network security issues. It also provides a primary assessment facility with scalability, minimal technical complexity, and extensibility by integrating custom machine learning models, datasets, aggregation protocols, topologies, and attack modules.

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 key works that have contributed to the advancements in this field. To begin with, TensorFlow Federated (TFF) has the foundation of Centralized Federated Learning (CFL) architecture and uses star topology with moderate scope in scalability [16]. Roy et al. have developed BrainTorrent DFL for healthcare applications using a fully connected topology [15] for medical image processing, which gives better performance compared to traditional server-based FL and pooled data training. However, BrainTorrent DFL lacks security features and broader extensibility beyond its specific domain. Several recent frameworks have focused on specific aspects of DFL security. FL-SEC [14] 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 [6] and DFedSN [4] represent hybrid approaches 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, while DFedSN addresses data heterogeneity in social networks through affine transformation. Although these frameworks demonstrate effective solutions for specific use cases, their specialized architectures may present challenges for broader applications. Lastly, Fedstellar [8] 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, this platform promises comprehensive support for the implementation of the DFL network with many features available that may require a heavy computational setup.

Despite contributing significantly to this field of research, many frameworks lack the flexibility and comprehensive security evaluation needed for decentralized structures in FL networks. Some frameworks with limited customization reduce flexibility and are resource-intensive due to their heavy features. Additionally, not being open-source creates accessibility challenges. Hence, a necessity remains to address these limitations by proposing a solution that enables comprehensive security evaluation in DFL through custom configurations and accessible deployment options.

3 ARCHITECTURE OF PHOENIX

The framework is designed with a modular architecture, that allows customization in node participation methods, communication protocols, attack models, aggregation algorithms, and convergence techniques. This section provides an overview of Phoenix's architecture, including its core components and workflow.

3.1 Core Components

The core components that are implemented in Phoenix include:
Dynamic Node Participation: In DFL, node participation varies over the training rounds with the participation rate that determines the proportion of active nodes during the training and the aggregation. This balance ensures universal participation while prioritizing nodes with larger datasets. Phoenix employs weighted random sampling for node selection [13], 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 Protocols: To enhance communication efficiency, the platform integrates the P2P and Gossip (GP) protocols. The P2P approach establishes deterministic communication patterns based on network topology, enabling nodes to exchange parameters with their immediate neighbors [7]. In contrast, GP follows a stochastic approach, where nodes randomly select peers for parameter exchange [10]. The framework dynamically adapts these protocols based on network conditions and performance requirements.

Topologies: Phoenix supports user-defined network topologies via edge list files, enabling experimental configurations ranging from small-scale setups to large-scale federations. In addition to standard topologies like fully connected, ring, and star, the platform also supports the construction of custom hybrid topologies like mesh networks. This feature enhances the understanding of DFL network topology exploration and dynamic visualization offers intuitive insights into network configurations.

Attack Simulation: Phoenix incorporates two fundamental attacks are implemented: Model poisoning [17] and Delay attack [5], as described below.

- *Model poisoning attack:* In model poisoning, nodes deliberately manipulate their local model parameters by injecting significant random noise into their state dictionary after local training. When these poisoned parameters are shared during the aggregation phase, they degrade performance, cause misclassification, or lead to convergence on suboptimal solutions.
- *Delay attack:* A delay attack occurs when nodes intentionally postpone transmitting their updated model parameters during training rounds. This disruption evaluates the system's performance under conditions where some nodes take longer than expected to share their updates.

Model Aggregation: FedAvg is a widely adopted aggregation method in FL that averages model parameters from all participating nodes to update their local models collaboratively [9]. In a DFL setup, the aggregation is performed through peer-to-peer interactions without relying on a central coordinating server. Mathematically, let K be 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 model 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 with which node k communicates for aggregation, and $\sum_{j \in N_k} n_j$ is the total number of data samples across node k and its neighbors.

Convergence Detection Mechanism: Detecting the point of convergence is crucial to determine when the global model has stabilized. This allows the system to optimize the duration and conserve computational resources. Convergence is identified based on the improvement in average test accuracy over consecutive rounds. This criterion is formalized through a moving window analysis of

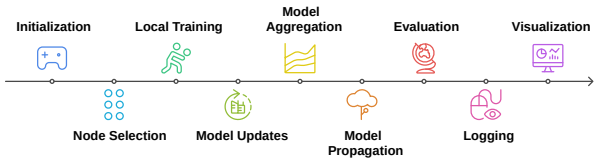

Figure 1: Workflow of Phoenix

performance metrics. For any given round r, convergence is detected if the improvements in the average test accuracy in the past rounds p satisfy:

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

where $\Delta_r = accuracy_r - accuracy_{r-1}$ 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.

3.2 Workflow

The complete workflow of the platform progresses through several steps and is designed to emulate a realistic decentralized environment. Figure 1 depicts the workflow divided into several steps.

Initialization: Phoenix is designed to offer flexibility in configuring and tuning various parameters through a YAML configuration file. The execution starts with data partitioning among nodes using Dirichlet distribution. Each node receives a portion of data to simulate real-world data heterogeneity.

Round-based training: Nodes are then selected for participation in training rounds using weighted random sampling. After local training, nodes share their model updates with other nodes according to the chosen communication protocol. The updates shared by nodes may be benign or adversarial, depending on the role assigned to each node. This allows for simulating decentralized systems with varying trust levels. After receiving updates, each node locally aggregates them into a global model, which serves as the starting point for the next training round. This cycle repeats until the specified rounds are completed. After each training round, evaluations on both training and testing datasets assess model performance to track learning progression over multiple rounds. This process highlights the impact of adversarial behaviors, data heterogeneity, and aggregation protocols. Throughout the simulation, detailed logs are captured, including training durations, aggregation times, performance metrics, and adversarial effects such as poisoned updates or delayed communications. These logs are systematically structured to ensure reproducibility and traceability of results.

Visualization: The final step is visualization, where logged metrics are translated into graphical representations such as *(i)* convergence trends of the global model, *(ii)* class distribution across nodes, *(iii)* performance metrics of individual nodes and the global model, and *(iv)* network topology diagrams. These visualizations serve as essential tools for interpreting and analyzing simulation outcomes. This modular architecture seamlessly integrates with various topologies and custom simulations and extends its functionalities by supporting user-defined attack models, datasets and aggregation protocols.

Table 2: Model Architecture and 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 v **Loss Function**	10 units Cross-Entropy Loss
Optimizer	Stochastic Gradient Descent (SGD)
Learning Rate	0.01

Table 3: Default parameters used in Phoenix

Parameter	Default Value
Dataset	FashionMNIST
Data Distribution	Dirichlet (α = 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

4 EXPERIMENTAL SETUP

This section assesses the framework's performance in the DFL setting particularly under adversarial conditions. The experimental analyses include the impact of the malicious nodes based on the placements and three distinct attack scenarios are compared with a baseline configuration.

System Configuration: The default model architecture used is a Convolutional Neural Network (CNN) designed for image classification tasks. The architectural setup is shown in Table 2. For evaluation, the FashionMNIST dataset is used, which consists of 70,000 grayscale images across 10 fashion categories [18]. Dirichlet distribution is applied to simulate non-i.i.d. data partitioning among federated nodes. The concentration parameter α in Dirichlet distribution controls the degree of heterogeneity where lower values $\alpha < 1$ produce more skewed distributions [19]. The framework is designed to optimize operational efficiency by leveraging Python's widespread adoption and extensive ecosystem of libraries and tools. It integrates PyTorch for local update computation and parameter aggregation to facilitate deep learning operations. Additionally, NumPy and Pandas handle data processing tasks, while NetworkX is employed to simulate complex network topologies that closely resemble real-world structures. Python's native logging module is used for monitoring and capturing runtime information, while Matplotlib and Seaborn enable dynamic visualization of performance metrics. Finally, to simulate decentralized processing, Python's multiprocessing and threading executors are utilized that allows separate processes for each node's training tasks and efficient handling of concurrent update exchanges. A custom topology comprising 10 nodes, as illustrated in Figure 2 is designed to evaluate the impact of attacks based on the node placements. To find the optimal configuration after several test runs with varying hyperparameter settings, the default parameters are summarized in Table 3 which serves as a baseline for custom configurations. Three distinct attack scenarios are simulated and executed over multiple rounds to closely resemble anonymous attack patterns observed in real-world settings. The framework utilizes a YAML-based configuration file that facilitates designing attack scenarios by adjusting existing parameters or introducing new ones. 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.

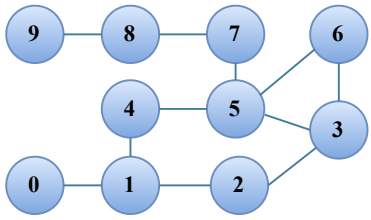

Figure 2: Custom topology of 10 nodes used in our evaluation.

Evaluation Metrics: The framework evaluates model effectiveness using accuracy, F1 score, precision, and recall across training rounds. Performance is monitored through computational metrics such as per-round training duration and aggregation time. Additionally, we evaluate convergence to assess the overall learning stability.

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

Phoenix provides precise control over attack execution through various tunable parameters and specific assessment preferences. To evaluate the performance of this framework we design three distinct attack scenarios based on the custom topology shown in Figure 2 while retaining the baseline configuration parameters:

Scenario 1: Node 5 is designated as a malicious node performing 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: 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: Node 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.

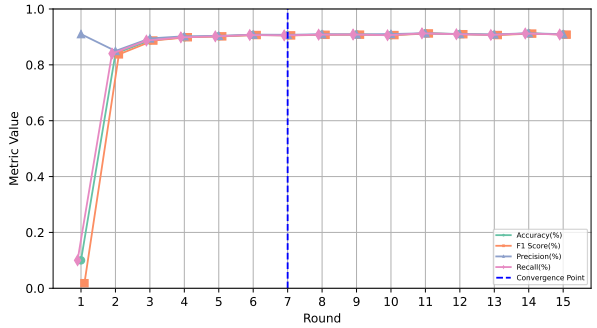

Figure 3: Performance metrics of Baseline

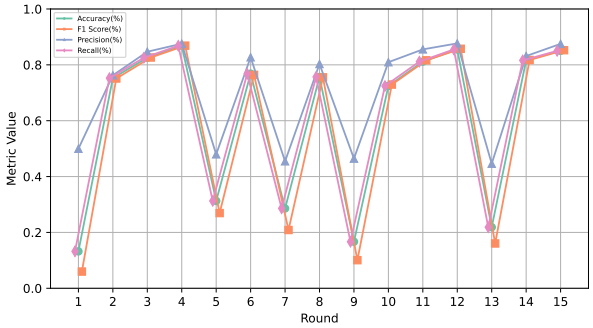

Figure 4: Performance metrics of Scenario 1

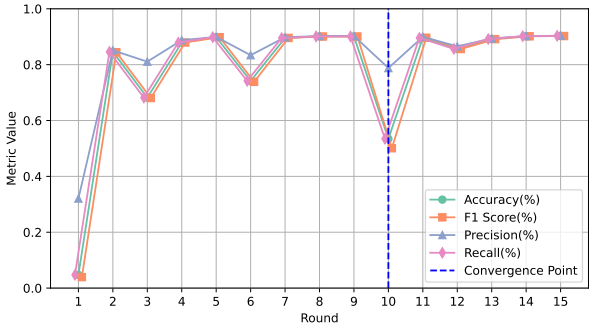

Figure 5: Performance metrics of Scenario 2

5 RESULTS AND DISCUSSION

The experimental evaluation of Phoenix revealed 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. The system achieves convergence at round 7 with an accuracy of 90.57% and maintained consistent performance thereafter indicating 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 delay and model poisoning 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 86.82% in round 4 to 31.28% 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. Here the attack is performed in rounds 1, 3, 6, 1,0, and 12, as depicted in Figure,5. Notable performance fluctuations are observed during these specific rounds when the attack is launched. Since this malicious node is connected to only one neighbor, it shares its corrupted model only with that node, while the remaining healthy nodes continue updating and exchanging models normally. Over successive rounds, the influence of the corrupted model decreases as most nodes receive updates from healthy peers. However, the overall performance drops by a delayed convergence compared to the baseline scenario shown in Figure 3. The model manages to achieve convergence by round 10, maintaining an accuracy level above 88%. This analysis emphasizes that an attack from a single node with minimal connections has a limited impact. It also highlights the variability introduced by different attack strategies, communication protocols, and topologies can substantially influence 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 1500 seconds with minor fluctuations. Conversely, the delay attack leads to erratic behavior, with sharp spikes in round times, particularly in rounds 3, 6, and 10-11, where round durations exceed 1700 seconds. This highlights the attack's effectiveness in disrupting normal operations by introducing unpredictable delays and prolonging round completion times.

The results shown in Table 4 sum up a comprehensive comparison of model performance across scenarios. The baseline achieves strong performance across all metrics, while Scenario 1 experiences

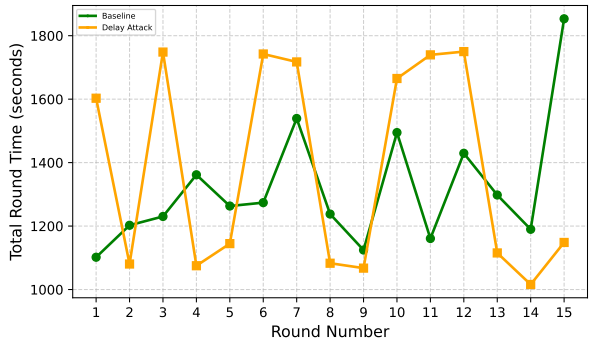

Figure 6: Comparison of Total Round Times: Baseline vs Delay Attack

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

the most significant degradation due to poisoning. On the other hand, Scenario 2 remains comparatively resilient as the malicious node resides in a less dense area. Notably, the average round time progressively increases from the baseline to Scenario 3, indicating the computational overhead introduced by the delay attack. Overall, these findings exhibit the effectiveness of Phoenix in the security evaluation of DFL networks. It also facilitates understanding the network performance under adversarial conditions.

6 CONCLUSION

This paper presents Phoenix, an open-source and comprehensive framework for security assessment in DFL with modular architecture and minimal dependencies. This framework is designed to be lightweight and easy to configure, enabling the implementation of diverse DFL scenarios with scalability while minimizing resource consumption. This accessibility, combined with comprehensive performance metrics and visualization capabilities, makes Phoenix a useful and cost-effective tool. The experimental results demonstrate its effectiveness 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 impact while also uncovering key trade-offs between model performance and computational efficiency. Phoenix acts as a foundational infrastructure for comprehensive security evaluation, whether to develop mitigation strategies, address defensive measures, analyze attack impacts, or improve aggregation protocols. By providing a flexible platform especially tailored for DFL settings, it paves the way for more resilient and secure strategies in distributed learning environments.

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] Chunlu Chen, Ji Liu, Haowen Tan, Xingjian Li, Kevin I-Kai Wang, Peng Li, Kouichi Sakurai, and Dejing Dou. 2025. Trustworthy federated learning: privacy, security, and beyond. *Knowledge and Information Systems* 67, 3 (March 2025), 2321–2356. https://doi.org/10.1007/s10115-024-02285-2

[4] Y. Chen, L. Liang, and W. Gao. 2023. DFedSN: Decentralized Federated Learning Based on Heterogeneous Data in Social Networks. *World Wide Web* (2023). https://doi.org/10.1007/s11280-023-01152-4

[5] 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. https://doi.org/10.1109/SmartGridComm.2019.8909732

[6] 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. https://doi.org/10.1016/j.hcc.2023.100128

[7] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme 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. https://doi.org/10.1109/COMST.2023.3315746

[8] Enrique Tomás Martínez Beltrán, Ángel Luis Perales Gómez, Chao Feng, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme 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. https://doi.org/10.1016/j.eswa.2023.122861

[9] 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

[10] 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.

[11] 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.

[12] Ehsan Nowroozi, Imran Haider, Rahim Taheri, and Mauro Conti. 2025. Federated Learning Under Attack: Exposing Vulnerabilities Through Data Poisoning Attacks in Computer Networks. *IEEE Transactions on Network and Service Management* (2025), 1–1. https://doi.org/10.1109/TNSM.2025.3525554

[13] 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). https://doi.org/10.1007/s10791-024-09478-x

[14] 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. https://doi.org/10.1109/IOTM.001.2100173

[15] 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

[16] TensorFlow Federated. 2023. TensorFlow Federated. https://www.tensorflow.org/federated Accessed: 2025-01-31.

[17] Geming Xia, Jian Chen, Chaodong Yu, and Jun Ma. 2023. Poisoning Attacks in Federated Learning: A Survey. *IEEE Access* 11 (2023), 10708–10722. https://doi.org/10.1109/ACCESS.2023.3238823

[18] 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

[19] 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