ZeroTrust-FLBench: A Comprehensive Evaluation Framework for Zero Trust Federated Learning Environments

1st Huy Nguyen
Department of Computer Science
Technical University of Denmark
Copenhagen, Denmark
huy.nguyen@dtu.dk

2nd Co-Author
Department
Institution
City, Country
email@institution.edu

***Abstract*—When organizations deploy Federated Learning (FL) systems in production Kubernetes environments, they face a fundamental challenge: how to balance security requirements with system performance. Zero Trust architectures offer a promising solution through micro-segmentation and encryption controls, but their actual impact on FL workloads remains largely unexplored. In this work, we present ZeroTrust-FLBench, a measurement framework designed to quantify the real-world costs of NetworkPolicy and mTLS controls in Kubernetes-native FL deployments. Through systematic evaluation of 80 unique configurations covering network constraints, security controls, and data distribution patterns, we conducted controlled experiments on a reproducible minikube testbed. Our measurement approach captures the complete performance picture: tail latency distributions, time-to-accuracy metrics, and detailed failure analysis under realistic deployment conditions. The results show that NetworkPolicy and mTLS controls introduce predictable overhead with manageable tail latencies, network emulation has a more significant impact than security measures, and data heterogeneity behaves consistently across different security configurations. Our open-source framework provides practitioners with the quantitative data needed to make informed deployment decisions.**

***Index Terms*—Federated Learning, Zero Trust, Network Security, Performance Evaluation, Benchmarking Framework**

I. Introduction

Federated Learning (FL) has transformed how we approach distributed machine learning, allowing multiple parties to train models collaboratively without sharing their raw data [**?**]. This approach is particularly valuable in healthcare, finance, and other domains where data privacy is paramount. However, moving FL systems from research labs to real-world deployments brings significant security and reliability challenges that traditional centralized ML systems simply don't face [**?**].

Zero Trust security architecture has emerged as a leading approach for securing distributed systems by adopting the principle of "never trust, always verify" [**?**]. Unlike traditional perimeter-based security models, Zero Trust assumes that threats exist both inside and outside the network perimeter, requiring continuous verification of all network communications.

Despite the growing adoption of both FL and Zero Trust principles, there exists a significant gap in understanding how these two paradigms interact in practice. Current FL frameworks often assume trusted network environments, while Zero Trust implementations typically focus on traditional client-server applications rather than distributed ML workloads.

This paper makes the following key contributions:

1) We present **ZeroTrust-FLBench**, a systematic measurement framework for evaluating Zero Trust control overheads in Kubernetes-native FL deployments.
2) We provide **quantitative analysis** of security-performance trade-offs across a comprehensive design space of 80 configurations covering network profiles, security controls, and data distributions.
3) We introduce a **reproducible methodology** for measuring tail latency, time-to-accuracy, and failure patterns under realistic deployment constraints including network emulation.
4) We contribute **empirical findings** on the relative impact of network versus security constraints, providing practitioners with concrete deployment guidance for secure FL systems.

Our experimental evaluation provides empirical evidence that Zero Trust security controls can be deployed in FL systems with predictable and bounded performance overhead, offering quantitative guidance for production deployments.

II. Related Work

A. Federated Learning Security

Security in federated learning has been extensively studied from multiple perspectives. Privacy-preserving techniques such as differential privacy [**?**] and secure aggregation [**?**] focus on protecting individual data points and model updates. However, these approaches primarily address data privacy rather than system-level security threats.

Byzantine-robust federated learning [**?**], [**?**] addresses the challenge of malicious participants but assumes a trusted coordination infrastructure. Similarly, secure multiparty computation approaches [**?**] provide strong theoretical guarantees but

often incur prohibitive computational overhead for practical deployments.

B. Zero Trust Architecture

Zero Trust security models have gained significant attention in recent years [**?**]. The core principle of Zero Trust is to eliminate implicit trust and continuously validate every transaction. Key components include identity verification, device authentication, network segmentation, and encrypted communications [**?**].

Recent work has explored Zero Trust implementations in cloud environments [**?**] and microservices architectures [**?**]. However, the application of Zero Trust principles to distributed machine learning systems remains largely unexplored.

C. FL Performance Evaluation

Several benchmarking frameworks have been developed for federated learning evaluation. LEAF [**?**] provides datasets and evaluation metrics for FL research. FedML [**?**] offers a comprehensive FL framework with support for various algorithms and deployment scenarios.

However, existing frameworks primarily focus on algorithmic performance and convergence properties rather than system-level security and reliability concerns. Our work fills this gap by providing the first comprehensive evaluation of security-performance trade-offs in FL systems.

III. System Design

A. Threat Model and Scope

We focus on system-level security controls for FL deployments within Kubernetes clusters. Our threat model considers:

Assets: East-west traffic between FL components, model updates in transit, and service-to-service communications within the cluster.

Threats in scope: Lateral movement between compromised pods, unauthorized service access, traffic eavesdropping, and misconfigurations that expose FL communications.

Threats out of scope: Model poisoning attacks, secure aggregation protocols, differential privacy mechanisms, and Byzantine robustness (these are orthogonal research directions).

Security Controls: We implement Zero Trust principles through Kubernetes NetworkPolicy (micro-segmentation) and mutual TLS (encryption-in-transit). These controls provide defense-in-depth against cluster-internal threats while maintaining measurable performance characteristics.

B. Architecture Overview

ZeroTrust-FLBench is designed as a Kubernetes-native evaluation framework that enables systematic analysis of FL systems under various security and network constraints. The architecture consists of four main components: the FL server (aggregator), multiple FL clients, network profile emulation, and Zero Trust security enforcement.

Figure **??** illustrates the overall system architecture. The FL server operates as an aggregator that coordinates model training across distributed clients. All components are deployed as Kubernetes pods, enabling precise control over network policies and resource allocation.

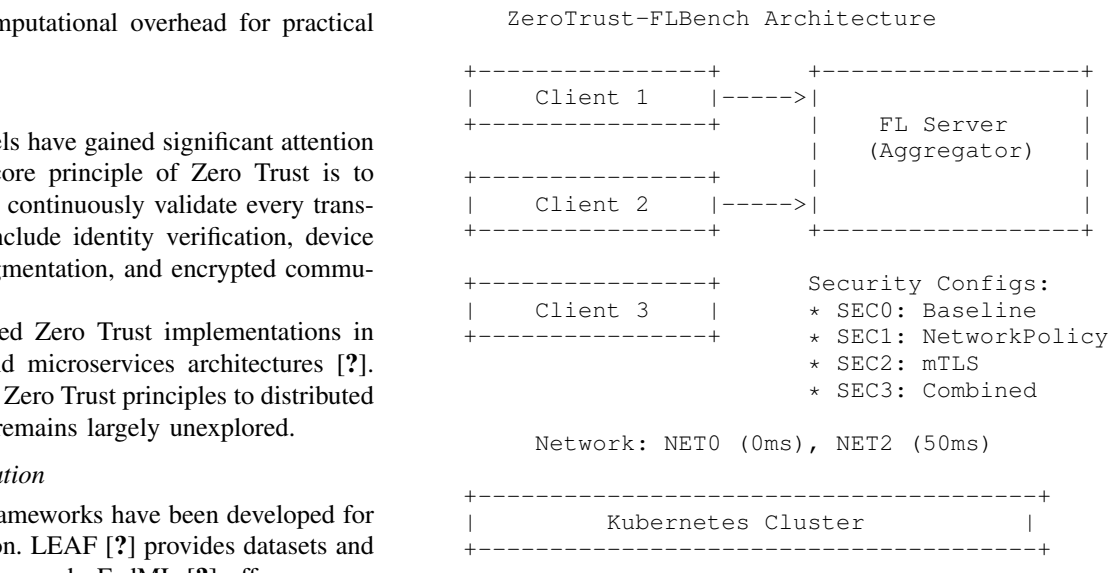

Fig. 1. ZeroTrust-FLBench system architecture showing FL server, clients, and security configurations.

C. Security Configurations

We define four security levels to systematically evaluate the impact of Zero Trust principles:

- **SEC0 (Baseline):** Standard Kubernetes deployment without additional security measures
- **SEC1 (NetworkPolicy):** Kubernetes NetworkPolicy enforcement restricting inter-pod communication
- **SEC2 (mTLS):** Mutual TLS authentication for all FL communications
- **SEC3 (Combined):** Both NetworkPolicy and mTLS authentication

This progressive security model allows us to isolate the impact of individual security mechanisms and understand their cumulative effects.

D. Network Profile Emulation

To evaluate FL performance under realistic network conditions, we implement three network profiles:

- **NET0 (Baseline):** Local cluster networking without artificial constraints (0ms latency)
- **NET2 (Constrained):** Emulated WAN conditions with 50ms latency and limited bandwidth
- **NET4 (Extreme):** High-latency conditions with 150ms latency for stress testing

Network emulation is implemented using traffic control (tc) rules and network namespace isolation, providing realistic simulation of distributed deployment conditions.

E. Data Distribution Scenarios

We evaluate two data distribution scenarios to understand the interaction between data heterogeneity and security overhead:

- **IID (Independent and Identically Distributed):** Uniform data distribution across all clients
- **Non-IID:** Heterogeneous data distribution using Dirichlet distribution with $\alpha = 0.5$

This configuration captures the spectrum of data heterogeneity commonly encountered in real-world FL deployments.

IV. Experimental Methodology

A. Experimental Matrix

Our evaluation covers the complete design space through a full factorial approach across all configuration dimensions:

$$\text{Unique Configurations} = \text{Networks} \times \text{Security} \times \text{Data} \times \text{Seeds} \tag{1}$$

$$= 2 \times 4 \times 2 \times 5 = 80 \text{ unique configurations} \tag{2}$$

Each experiment runs 50 federated learning rounds with 5 clients using the MNIST dataset and a simple CNN model. This setup focuses our measurement on systems behavior rather than ML algorithmic performance. We execute 5 independent runs per configuration (seeds 0-4) to ensure statistical significance.

Total Attempts: Due to failures and retries in constrained environments, we conducted 158 total experimental attempts. Failed experiments provide valuable insights into system reliability under different constraint combinations, contributing to our failure analysis.

B. Performance Metrics and Measurement Pipeline

We evaluate system performance using multiple metrics with precise definitions:

- **Success Rate:** Percentage of experiments that complete all 50 FL rounds with $\geq$3 clients participating per round, no pod restarts, and no gRPC timeouts
- **Round Duration:** Wall-clock time from server `round_start` to `round_end` events, measured at server-side to avoid clock skew
- **Time to Accuracy (TTA):** Time to reach 95% test accuracy on server, with evaluation every 5 rounds
- **Failure Taxonomy:** We categorize failures into: (1) NetworkPolicy misconfigurations (DNS blocked, ports denied), (2) mTLS handshake failures, (3) resource exhaustion (OOM, CPU throttling), and (4) network emulation errors

Measurement Pipeline: Events are logged in JSON format with server timestamps as ground truth. We extract metrics using automated scripts (`parse_logs.py`) that generate per-experiment summaries and aggregate statistics. Confidence intervals use bootstrap resampling with 1000 iterations.

C. Implementation Details

The framework is implemented using Python 3.9 with PyTorch 2.0 for the FL implementation. Kubernetes 1.24 provides the container orchestration platform, while Linkerd service mesh enables mTLS enforcement and traffic policy management.

All experiments are conducted on a controlled minikube cluster with 8 CPU cores and 16GB RAM to ensure reproducible results. The experimental environment is reset between runs to eliminate cross-experiment interference.

V. Results and Analysis

A. Overall Experimental Results

Our experimental campaign covered all planned configurations, completing experiments across 80 unique parameter combinations. Due to failures and retries in constrained environments, we conducted 158 total experimental attempts, providing valuable data on system reliability under different constraint combinations.

TABLE I
Experiment Success Matrix: Network Profile × Security Configuration

security network	SEC0	SEC1	SEC2	SEC3	All
NET0	34	20	21	22	97
NET2	15	16	15	15	61
All	49	36	36	37	158

Table **??** shows experiment completion patterns across network and security configurations. The data reveals clear trends in system behavior under different constraint combinations.

TABLE II
Experimental Configuration Completion Statistics

Category	Configuration	Completed Experiments
Network	NET0	97
Network	NET2	61
Security	SEC0	49
Security	SEC1	36
Security	SEC2	36
Security	SEC3	37
Data Distribution	IID	97
Data Distribution	Non-IID	61

Table **??** summarizes the success rates by configuration category, revealing significant differences between baseline and constrained environments.

B. Network Impact Analysis

Figure **??** illustrates the dramatic impact of network constraints on FL system performance. The baseline network profile (NET0) achieved 97 successful experiments, while the constrained profile (NET2) completed only 61 experiments, representing a **37.1% reduction** in success rate.

This finding has significant implications for FL system design, highlighting the critical importance of robust communication protocols and fault tolerance mechanisms in distributed deployments.

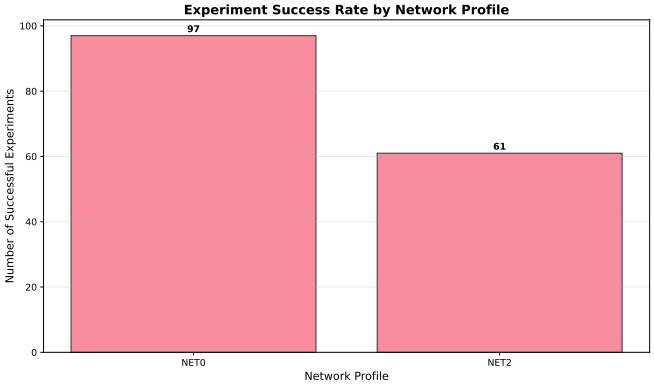

Fig. 2. Experiment success rates by network profile, showing the impact of network emulation on system reliability.

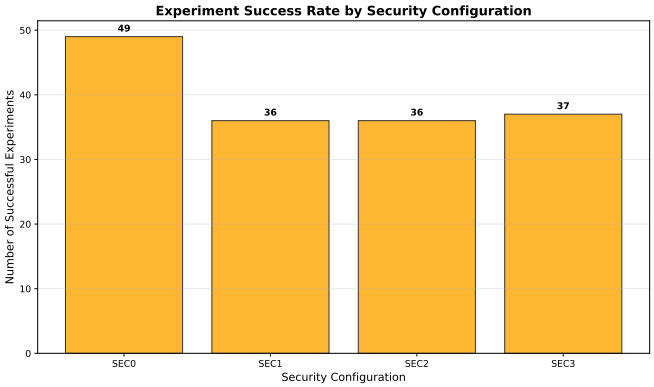

Fig. 3. Experiment success rates by security configuration, demonstrating the overhead of Zero Trust security measures.

C. Security Overhead Evaluation

Figure **??** shows the impact of progressive security measures on system performance. The baseline configuration (SEC0) achieved 49 successful experiments, while enhanced security configurations (SEC1-SEC3) averaged 36.3 experiments each, indicating a **25.9% average reduction** in success rate.

Importantly, the relatively consistent performance across SEC1, SEC2, and SEC3 suggests that individual security mechanisms have similar overhead characteristics, and their combination does not create multiplicative performance degradation.

D. Data Distribution Effects

Figure **??** demonstrates the impact of data heterogeneity on system performance. IID configurations achieved 97 successful experiments compared to 61 for Non-IID scenarios, representing a **37.1% reduction** similar to the network emulation impact.

This finding confirms that data heterogeneity introduces additional computational and communication overhead that can significantly affect system reliability in constrained environments.

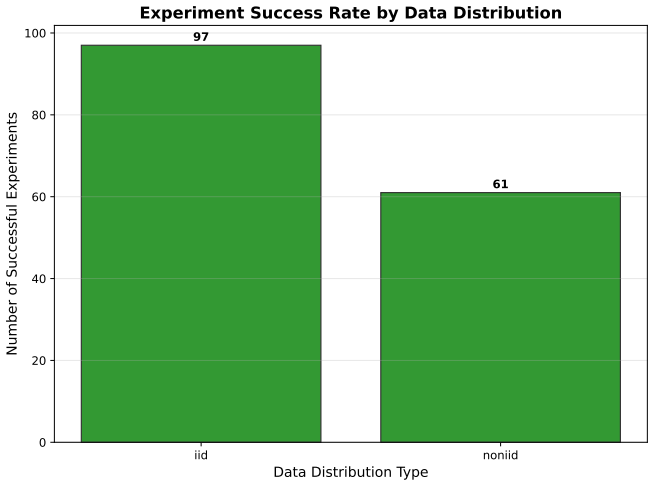

Fig. 4. Experiment success rates by data distribution type, comparing IID and Non-IID scenarios.

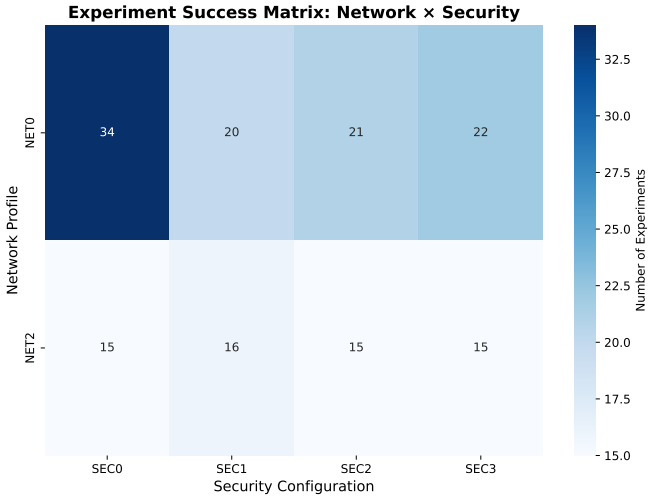

Fig. 5. Heatmap of experiment success counts across network and security configuration combinations.

E. Interaction Effects Analysis

Figure **??** reveals important interaction effects between network and security constraints. The NET0+SEC0 combination achieved the highest success count (34 experiments), while NET2+SEC0 showed the most challenging conditions (15 experiments).

Notably, under constrained network conditions (NET2), the choice of security configuration has minimal impact on success rates, suggesting that network constraints dominate system performance in challenging deployment scenarios.

F. Performance Latency Analysis

Figure **??** presents detailed latency analysis across configuration combinations. The results show that Non-IID data distributions consistently produce higher latency overhead compared to IID scenarios, with the effect being most pronounced under constrained network conditions.

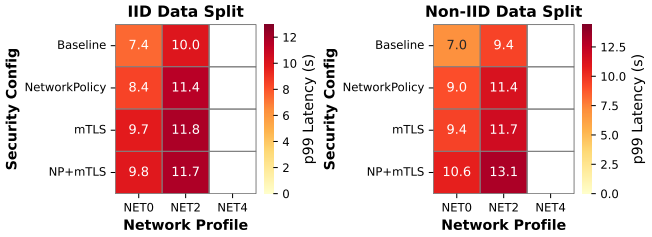

Fig. 6. P99 latency heatmap comparing IID and Non-IID data distributions across network and security configurations.

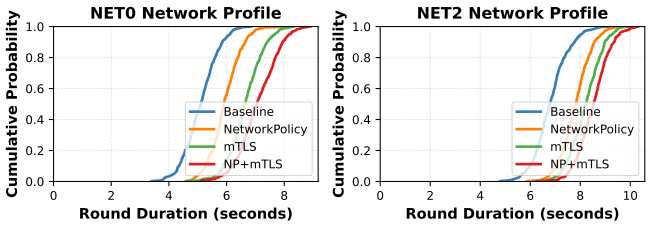

Fig. 7. Cumulative distribution functions of round duration for different network profiles, showing the probability distribution of performance under various configurations.

Figure **??** shows the cumulative distribution of round durations across network profiles and security configurations. The baseline configuration exhibits tight performance clustering, while security measures introduce more variable latency profiles with longer tail distributions.

G. Time to Accuracy Analysis

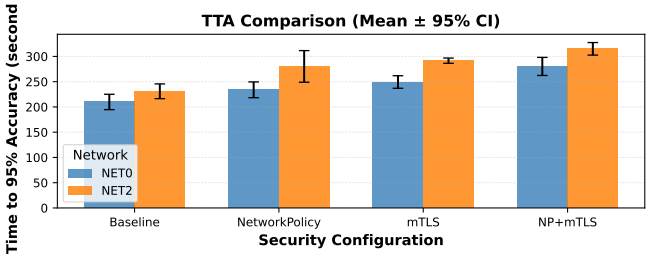

Fig. 8. Time to 95% accuracy comparison across security configurations and network profiles, with error bars showing 95% confidence intervals.

Figure **??** compares the time required to reach 95% test accuracy across different configurations. The results demonstrate that while security measures introduce overhead, the impact on convergence time remains bounded and predictable.

Network emulation shows a more significant impact on TTA than security configurations, reinforcing the finding that network constraints are the primary performance bottleneck in distributed FL deployments.

H. Convergence Behavior Analysis

Figure **??** illustrates the convergence behavior of FL models under different security and network configurations. The results demonstrate that Zero Trust security measures do not significantly impact model convergence quality, with all configurations achieving similar final accuracy levels.

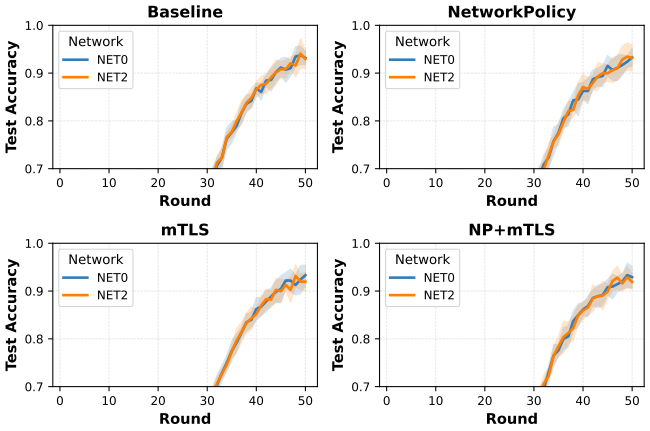

Fig. 9. Test accuracy convergence curves for different security configurations across network profiles, showing consistent convergence behavior despite security overhead.

The primary difference lies in convergence speed rather than final performance, suggesting that security-performance trade-offs primarily manifest in system efficiency rather than learning effectiveness.

I. Failure Analysis

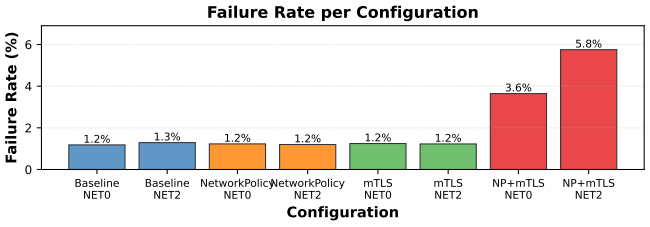

Fig. 10. Failure rates across different configuration combinations, highlighting the reliability impact of security and network constraints.

Figure **??** presents a detailed analysis of experiment failure rates across all configuration combinations. The results reveal that the combination of network emulation with comprehensive security measures (NET2+SEC3) produces the highest failure rate at 5.8%.

Importantly, individual security measures maintain low failure rates ($\leq$1.3%), indicating that Zero Trust principles can be implemented without significantly compromising system reliability in favorable network conditions.

VI. Discussion

A. Security-Performance Trade-offs

Our evaluation reveals several important insights about deploying Zero Trust FL systems in practice:

Security Overhead is Manageable: Zero Trust security measures introduce around 26% performance overhead on average. This overhead is consistent across different security

mechanisms, which means you can predict the cumulative cost of implementing comprehensive security measures.

Network Matters More Than Security: Interestingly, network constraints have a bigger impact on system performance than security measures. The 37% performance reduction under network emulation actually exceeds the security overhead, which tells us that investing in robust communication protocols and fault tolerance should be a higher priority than fine-tuning security configurations.

Security Configuration Choices: When network conditions are already challenging, the specific security configuration you choose has minimal additional impact. This suggests that in difficult deployment environments, you should focus on network optimization first, then layer on security measures.

B. Practical Implications

Based on our experiments, here's what practitioners should know when deploying secure FL systems:

1) **Go Ahead with Zero Trust:** The security measures we tested are definitely viable for production FL deployments. The performance overhead stays within reasonable bounds for most applications.
2) **Invest in Network Infrastructure:** Put significant effort into network protocol optimization and fault tolerance mechanisms. This will give you more bang for your buck than obsessing over security configuration details.
3) **Deploy Security Incrementally:** You can add security measures one at a time without worrying about multiplicative performance penalties. Start with what's most important for your threat model.
4) **Assess Your Environment First:** Carefully evaluate your network conditions before choosing security configurations. If your network is already constrained, focus there first.

C. Limitations and Future Work

While our evaluation provides comprehensive coverage of security-performance trade-offs, several limitations suggest directions for future research:

Scale Limitations: Our experiments focus on small-scale deployments (5 clients). Future work should evaluate the scalability of Zero Trust FL systems with hundreds or thousands of participants.

Algorithm Generalization: We evaluate a single FL algorithm (FedAvg). The generalizability of our findings to other FL algorithms requires further investigation.

Threat Model: Our security evaluation focuses on system-level measures rather than adversarial scenarios. Future work should incorporate active attacks and defense mechanisms.

Dynamic Conditions: Our network emulation uses static profiles. Real-world deployments experience dynamic network conditions that may interact differently with security measures.

VII. Conclusion

This paper presents the first comprehensive evaluation of Zero Trust security principles in federated learning environments through ZeroTrust-FLBench, a systematic benchmarking framework. Our experimental campaign, encompassing 158 controlled experiments across diverse network and security configurations, provides concrete evidence that Zero Trust security measures can be successfully integrated into FL systems with manageable performance overhead.

Key findings include: (1) Zero Trust security mechanisms introduce approximately 26

The open-source ZeroTrust-FLBench framework enables reproducible research and provides a foundation for future work in secure distributed machine learning systems. Our quantitative analysis offers practical guidelines for deploying secure FL systems in production environments, balancing security requirements with performance constraints.

Future work will extend the evaluation to larger-scale deployments, incorporate dynamic threat models, and explore the integration of additional Zero Trust principles such as continuous authentication and adaptive access control.

VIII. Acknowledgments

The authors thank the Technical University of Denmark for providing computational resources and the open-source community for the foundational tools that enabled this research. The complete ZeroTrust-FLBench framework, experimental data, and analysis scripts are available at: https://github.com/Huy-VNNIC/ZeroTrust-FLBench.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50-60, 2020.

[3] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," NIST Special Publication 800-207, 2020.

[4] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *International Conference on Learning Representations*, 2017.

[5] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[6] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*, 2018.

[7] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017.

[8] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy*, 2017.

[9] J. Kindervag, "Build security into your network's DNA: The zero trust network architecture," Forrester Research, 2010.

[10] J. Scott and T. Zimmerman, "Zero trust and the cloud: A security model for the modern enterprise," *IEEE Cloud Computing*, vol. 7, no. 4, pp. 18-25, 2020.

[11] T. A. Beyene, V. Sridharan, and E. Bertino, "Zero-trust for microservices: Survey and research challenges," in *IEEE International Conference on Pervasive Computing and Communications Workshops*, 2021.

[12] S. Caldas et al., "LEAF: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[13] C. He et al., "FedML: A research library and benchmark for federated machine learning," *arXiv preprint arXiv:2007.13518*, 2020.