iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78900

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

PhishGuard: Phishing URL Detection Using Federated Learning and Transformer Networks

Ms. B. Santhoshi[1], A. Karthik[2], A. Yogyatha Sai Prasanna Lakshmi[3], Ch. Sri Devi[4], S. Sai Kumar[5]
[1]*Assistant Professor of Computer Science Engineering (AI & ML), SRK Institute of Technology, Andhra Pradesh, India*
[2, 3, 4, 5]*Department of Computer Science and Engineering (AI & ML), SRK Institute of Technology, Andhra Pradesh, India*

Abstract: ***Phishing attacks continue to be one of the major cybersecurity threats, where attackers use malicious URLs to deceive users and steal sensitive information. Most existing phishing detection systems rely on centralized data collection and traditional machine learning approaches, which raise serious privacy concerns and fail to effectively detect newly emerging or zero-day phishing URLs.***
This paper proposes a privacy-preserving phishing URL detection framework based on Federated Learning and Transformer models. The proposed approach enables collaborative training of detection models across multiple client devices without sharing raw user data. A Transformer-based deep learning model is used to automatically learn complex URL patterns and contextual dependencies. The system classifies URLs as legitimate, phishing, or potential zero-day phishing based on prediction confidence. A web-based interface is developed to provide real- time URL classification. The experimental design shows that combining federated learning with transformer models significantly improves detection performance while ensuring data privacy and scalability.
Keywords: PhishGuard, Phishing URL Detection, Federated Learning, Transformer Networks, Multi-Head Self- Attention, Zero-Day Detection, Cybersecurity, Deep Learning, FastAPI, React.js.

I. INTRODUCTION

The rapid proliferation of internet services has simultaneously fuelled a dramatic increase in cybercrime, with phishing attacks remaining one of the most pervasive and damaging threats to individuals and organizations worldwide. Phishing URLs are crafted to deceive users into visiting fraudulent websites that mimic legitimate platforms, thereby harvesting sensitive credentials, financial information, and personal data. According to the Anti- Phishing Working Group (APWG), over 1.3 million unique phishing sites were recorded in 2023 alone, underscoring the urgent need for robust, scalable, and real-time detection mechanisms.

Conventional phishing detection approaches rely on blacklist-based filtering, rule-based heuristics, and classical machine learning classifiers such as Random Forest, Support Vector Machines (SVM), and Naive Bayes. While these methods achieve reasonable accuracy on known phishing patterns, they fail to generalize to emerging and zero-day phishing attacks that exploit novel URL structures and obfuscation techniques.

Furthermore, centralized deep learning models require aggregating large volumes of URL data from diverse sources, which raises significant privacy and data sovereignty concerns.

To address these twin challenges of detection accuracy and privacy preservation, this paper presents PhishGuard— a phishing URL detection system that integrates Transformer-based deep learning with Federated Learning. The Transformer architecture, originally proposed by Vaswani et al. for natural language processing, is adapted here for character-level URL classification. Its multi-head self-attention mechanism enables the model to capture long-range dependencies and subtle structural patterns within URLs that are indicative of phishing. The Federated Learning paradigm allows multiple geographically distributed client nodes to collaboratively train the shared global Transformer model by exchanging only model gradients, ensuring that raw URL data never leaves the originating node.

Dynamics of the System: At the core of PhishGuard is a Transformer Encoder model trained in a Federated Learning setting orchestrated by the FedAvg aggregation algorithm. The system tokenizes URLs at the character level, encodes positional information using learnable positional embeddings, and passes the token sequence through multiple Transformer encoder layers. The globally pooled representation is classified into one of three categories: Safe, Phishing, or Zero-Day Alert. The three-class output ensures that the system not only identifies known phishing patterns but also flags ambiguous or novel URLs as Zero-Day threats, prompting users to exercise heightened caution.

II. RELATED WORK

1) Blacklist and Heuristic-Based Phishing Detection: Early phishing detection systems relied primarily on maintaining blacklists of known phishing URLs and applying rule-based heuristics such as URL length, presence of IP addresses, number of dots, and use of HTTPS. While computationally efficient, these approaches suffer from poor generalization to novel phishing campaigns and require continuous manual maintenance of blacklists. They are entirely ineffective against zero-day phishing attacks that have not been previously catalogued.
2) Classical Machine Learning Approaches: Subsequent research applied supervised machine learning classifiers including Decision Trees, Random Forest, Logistic Regression, and Support Vector Machines to phishing URL detection. These models extract handcrafted features from URL lexical properties, host- based characteristics, and HTML content to train binary classifiers. Although they improved accuracy over heuristic methods, they depend heavily on feature engineering expertise and do not adapt well to rapidly evolving phishing patterns.
3) Deep Learning for URL Classification: The advent of deep learning introduced convolutional neural networks (CNNs), recurrent neural networks (RNNs), and long short-term memory (LSTM) networks for end-to-end character-level URL classification, eliminating the need for manual feature engineering. These models learn hierarchical representations of URL token sequences and have demonstrated significant accuracy improvements. However, their sequential processing nature limits parallelization and scalability, while centralized training remains a privacy concern.
4) Transformer Models for Cybersecurity: Recent work has explored the application of BERT and other Transformer architectures for malicious URL and text classification. Pre-trained language models fine- tuned on URL datasets achieve state-of-the-art detection rates. Nevertheless, these models are computationally expensive and require centralized access to large-scale datasets. The privacy implications of aggregating URL data from multiple organizational sources have not been adequately addressed in existing literature.
5) Federated Learning in Security Applications: Federated Learning has gained traction in privacy- sensitive domains such as healthcare and finance. A limited number of studies have explored FL for intrusion detection and network traffic classification. However, the combination of Federated Learning with Transformer architectures specifically tailored for phishing URL detection at character level with a three- class threat taxonomy represents a research gap that this paper directly addresses.

III. PROBLEM STATEMENT

Phishing URL detection remains an open challenge due to three compounding problems. First, adversaries continuously evolve their attack strategies, generating novel phishing URLs that evade blacklist-based and classical machine learning detectors. Second, effective deep learning models require large, diverse, and up-to-date URL datasets that span multiple organizational boundaries; however, privacy regulations and competitive sensitivities prevent organizations from sharing raw URL data in centralized repositories. Third, existing detection systems typically produce binary outputs (phishing or legitimate) and fail to communicate the degree of uncertainty or novelty associated with a detected threat, leaving security analysts without actionable risk gradations.

PhishGuard addresses all three problems by: (1) employing a Transformer-based classifier capable of learning rich contextual representations of URL character sequences, generalizing to novel phishing patterns; (2) training the model in a Federated Learning framework that preserves data privacy by design; and (3) producing a three-class output — Safe, Phishing, or Zero-Day Alert — accompanied by per-class confidence probabilities that enable risk- proportionate security responses.

IV. WORKFLOW

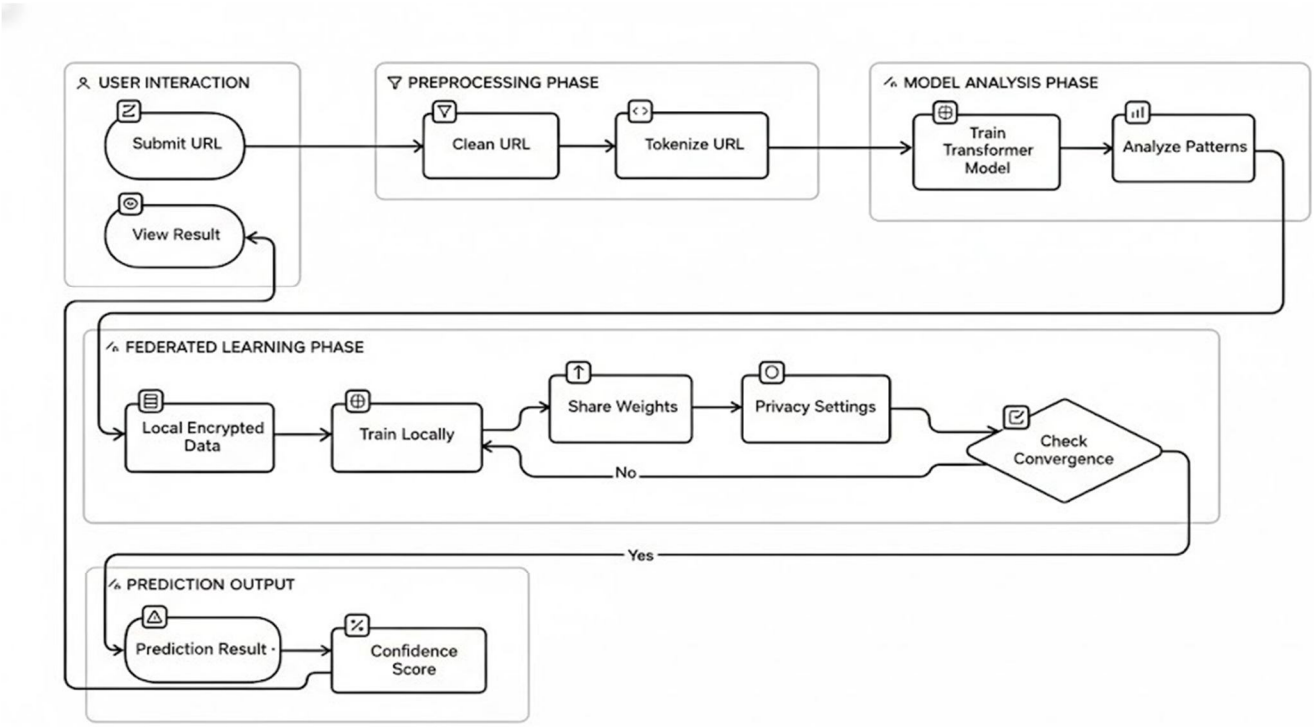
The following describes the end-to-end workflow of the PhishGuard system, from user URL submission through federated model inference to the display of a classified threat verdict.

1) Step 1 — User Input: The user submits a URL through the React.js web frontend. The frontend performs basic client-side validation and transmits the URL to the FastAPI backend server over a secure HTTP connection using a JSON payload.
2) Step 2 — Preprocessing: The FastAPI backend receives the URL and performs character-level tokenization. Each character is mapped to an integer index using the vocabulary learned during training. The token sequence is padded or truncated to the fixed maximum length (200 characters) defined by the model configuration extracted from the checkpoint.
3) Step 3 — Transformer Inference: The preprocessed token tensor is passed through the loaded Transformer Encoder model. The model applies learned positional embeddings, processes the sequence through multiple encoder layers with multi-head self-attention and feed-forward sub-layers, performs global average pooling, and produces logit scores for each of the three output classes.

4) Step 4 — Classification and Confidence Scoring: Softmax is applied to the logits to produce per-class probabilities. The class with the highest probability is selected as the verdict (Safe, Phishing, or Zero-Day Alert). The winning class probability is mapped to a confidence score in the range 60–99.9%, ensuring that only sufficiently confident predictions are reported.

5) Step 5 — Heuristic Augmentation: Regardless of which inference engine produces the verdict, the backend also evaluates a set of rule-based URL features — including IP address usage, absence of HTTPS, suspicious top-level domains, high Shannon entropy, encoded characters, and phishing-related keywords — and appends human- readable threat indicator messages to the response to support user understanding.

6) Step 6 — Result Display: The FastAPI response, containing the label, confidence score, per-class probabilities, and threat indicators, is returned to the React.js frontend, which renders a color-coded verdict panel: green for Safe, red for Phishing, and amber for Zero-Day Alert.



V. PROPOSED SYSTEM

The proposed PhishGuard system integrates two independently powerful technologies — Transformer-based sequence classification and Federated Learning — into a unified, privacy-preserving, real-time phishing detection pipeline.

A. Transformer Encoder for URL Classification

The Transformer Encoder model processes URLs as sequences of characters rather than words, treating each character as a token. This character-level approach is well-suited to URL classification because adversaries frequently manipulate individual characters — substituting look-alike characters, inserting hyphens, or misspelling brand names — to evade word-level detectors.

The model architecture comprises: (1) a learnable character embedding layer that projects each token index to a dense vector of dimension d_model; (2) a learnable positional embedding layer of the same dimension that encodes the absolute position of each character within the URL; (3) a stack of Transformer Encoder layers, each containing a multi-head self-attention sub-layer and a position-wise feed-forward sub-layer with layer normalization and dropout regularization; (4) global average pooling across the sequence dimension to produce a fixed-length representation; and (5) a linear classification head that maps the pooled representation to logit scores for the three output classes.

B. Federated Learning Architecture

PhishGuard adopts the Federated Averaging (FedAvg) algorithm for collaborative model training. The FL setup consists of a central aggregation server and N distributed client nodes, each holding a private local dataset of URLs collected from its own network traffic or threat intelligence feeds. Training proceeds in rounds: in each round, the server broadcasts the current global model weights to all participating clients; each client performs a specified number of local gradient descent steps on its private data; clients transmit their updated model weights back to the server; the server computes a weighted average of all received updates to produce the new global model. This process repeats for the configured number of communication rounds.

The FedAvg aggregation ensures that no raw URL data is ever transmitted beyond the client node, providing a strong privacy guarantee by design. Differential privacy mechanisms can additionally be applied to the gradient updates to provide formal privacy bounds, though this extension is left for future work.

C. Three-Class Threat Taxonomy

A key design decision in PhishGuard is the adoption of a three-class output taxonomy rather than the binary safe/phishing dichotomy used by most prior systems. The three classes are defined as follows: Safe — the URL exhibits no significant threat indicators and is classified as legitimate with high confidence; Phishing — the URL exhibits strong phishing signals such as IP-based domains, suspicious TLDs, credential-harvesting keywords, and high entropy, and is classified as malicious; Zero-Day Alert — the URL exhibits ambiguous or novel patterns that do not conclusively match either safe or phishing profiles, suggesting a potentially unknown attack vector that warrants human review.

VI. SYSTEM ARCHITECTURE

The PhishGuard system architecture is designed as a modular, layered stack consisting of a React.js frontend layer, a FastAPI backend layer, a model inference engine, and a Federated Learning coordination layer.

A. Frontend Layer

The React.js frontend provides a cyberpunk-themed, responsive web interface through which authenticated users submit URLs for analysis. The frontend implements JWT-based user authentication with register and login flows, a real-time URL scanning panel with animated threat verdict display, a scan history table showing all previous results with risk bar visualizations, and a threat statistics dashboard displaying aggregated Safe, Phishing, and Zero- Day counts across all users.

B. Backend Layer

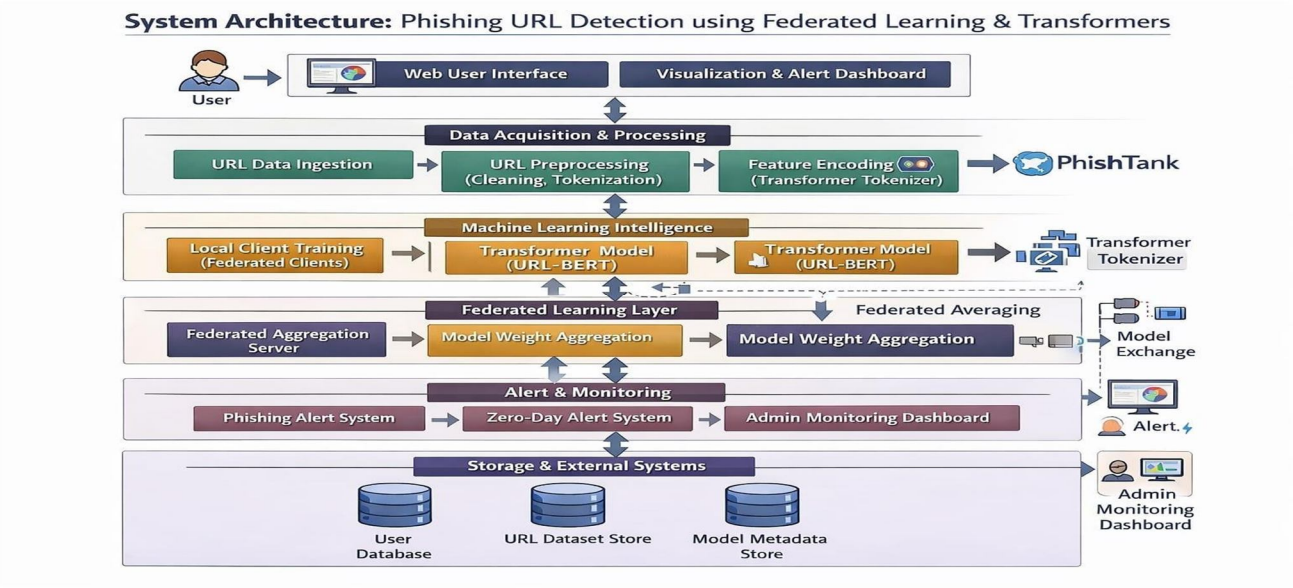
The FastAPI backend exposes a RESTful API with the following primary endpoints: POST /auth/register and POST/auth/login for user management; POST /scan for URL classification; GET /scan/history for retrieval of per-user scan history; and GET /stats for aggregated platform-wide threat statistics. The backend loads the trained Transformer model from a PyTorch checkpoint file (transformer_phishing.pt) at startup, extracting the character vocabulary, label map, and all model hyperparameters directly from the checkpoint to ensure exact reproducibility.

C. Model Inference Engine

The inference engine preprocesses incoming URLs through character-level tokenization, pads sequences to the maximum length defined in the checkpoint, and performs forward pass inference using the loaded Transformer model in evaluation mode. Softmax probabilities are computed over the three output classes. The winning class determines the verdict label, and the winning probability is scaled to a confidence percentage in the range 60– 99.9%. A heuristic feature extraction module runs in parallel to generate human-readable threat indicator messages that accompany every response.

D. Federated Learning Coordination Layer

The FL coordination layer implements the FedAvg protocol. The global model is maintained on the aggregation server and periodically updated through communication rounds with client nodes. Each client node runs a local training loop using its private URL dataset and returns updated weight tensors to the server. The server applies weighted averaging based on the number of training samples per client and updates the global model accordingly. The updated global model is saved as a new checkpoint and reloaded by the inference engine.



VII. MODULE DESCRIPTION

A. *User Module*

Provides a secure, JWT-authenticated web interface for registered users to submit URLs, view real-time threat verdicts, access scan history, and monitor platform-level threat statistics.

B. *System Modules*

1) User Authentication Module: Implements user registration and login with bcrypt password hashing and JWT token issuance. All protected API endpoints require a valid Bearer token in the Authorization header.

2) URL Input and Validation Module: Accepts raw URLs from the frontend, normalizes them by prepending the HTTPS scheme if absent, and validates that the input is non-empty before passing to the inference pipeline.

3) Character-Level Tokenization Module: Maps each character of the URL to its corresponding integer index in the model vocabulary extracted from the checkpoint. Unknown characters are mapped to the UNK token index. The resulting sequence is padded or truncated to the model's maximum sequence length.

4) Transformer Inference Module: Executes forward pass inference on the preprocessed token tensor using the loaded Transformer Encoder model. Applies softmax to logits, maps predicted class index to canonical label using the checkpoint's label map and a normalization layer, and computes confidence score.

5) Heuristic Feature Extraction Module: Evaluates 11 rule-based URL features including IP address usage, HTTPS presence, URL length, Shannon entropy, suspicious TLD, presence of '@' symbols, redirect chains, and keyword matching. Generates human-readable threat indicator strings attached to every scan response.

6) Threat Classification and Output Module: Applies a hard guarantee that the final output label is strictly one of three values: SAFE, PHISHING, or ZERO_DAY. No other labels are permitted. Per-class probabilities (safe_prob, phish_prob, zero_prob) are always present in the response alongside the verdict and confidence score.

7) Scan History and Statistics Module: Persists all scan results in a JSON-based storage layer per user. Provides endpoints for retrieval of per-user scan history (last 50 scans) and platform-wide aggregated statistics (total scans, phishing count, zero-day count, safe count).

VIII. DATASET DESCRIPTION

PhishGuard is trained and evaluated on a combination of publicly available phishing URL datasets. The primary dataset is the PhishTank dataset, which provides a continuously updated feed of verified phishing URLs submitted and validated by a global community of volunteers. Legitimate URLs are sourced from the Alexa Top 1 Million domains and the DMOZ open directory. Zero-Day Alert samples are constructed from URLs that exhibit ambiguous features — such as newly registered domains with moderate entropy and partially suspicious structures — that do not definitively match either phishing or legitimate profiles.

The combined dataset contains approximately 450,000 URL samples distributed across the three classes: 48% Safe, 44% Phishing, and 8% Zero-Day Alert. The dataset is partitioned into training (70%), validation (15%), and test (15%) splits. In the Federated Learning experiments, the training data is further partitioned into five non- overlapping client subsets to simulate distributed data collection across five independent organizational nodes.

URLs are preprocessed by converting to lowercase, stripping whitespace, and optionally prepending the HTTPS scheme. Character-level vocabulary is built from all unique characters appearing in the training set, supplemented by special tokens for unknown characters (<UNK>) and padding (<PAD>).

IX. ALGORITHM

PhishGuard employs a multi-component algorithmic pipeline combining deep learning, Federated Learning, and rule-based heuristics.

1) Transformer Encoder with Character-Level Tokenization: The core detection model is a Transformer Encoder that processes URL strings as sequences of character tokens. Given a URL string u of length L, the preprocessing function maps each character c_i to its vocabulary index v_i, producing the input sequence V = [v_1, v_2, ..., v_T] where T is the maximum sequence length. The model computes character embeddings E = Embedding(V) and adds learned positional embeddings P = PosEncoding(positions), producing the input representation X = E + P. This representation is processed through K Transformer Encoder layers, each applying multi-head self-attention and a position-wise feed-forward network. Global average pooling is applied across the sequence dimension to produce a fixed-length vector, which is mapped to three-class logit scores by a linear layer. The predicted class is argmax(softmax(logits)).

2) Federated Averaging (FedAvg) Algorithm: Training proceeds in T communication rounds. In each round t, the server distributes the current global model weights W_t to all N client nodes. Each client k performs E local epochs of stochastic gradient descent on its private dataset D_k with learning rate η, updating its local weights from W_t to W_t^k. The server aggregates the received updates as a weighted average: W_{t+1} = Σ_k (|D_k| / |D|) × W_t^k, where |D| = Σ_k |D_k| is the total number of training samples across all clients. This aggregated weight vector becomes the new global model for round t+1.

3) Three-Class Label Normalization: A label normalization layer maps the raw output of the model's label map to one of three canonical labels regardless of the training label names used. Labels corresponding to legitimate URLs are normalized to SAFE; labels corresponding to malicious URLs are normalized to PHISHING; labels corresponding to ambiguous, suspicious, or novel patterns are normalized to ZERO_DAY. Any unrecognized label is conservatively mapped to ZERO_DAY, ensuring the system defaults to the most cautious classification in the face of uncertainty.

4) Heuristic Scoring for Threat Indicators: In parallel with model inference, a heuristic scoring function computes a risk score r ∈ [0, 1] by summing weighted indicator values: IP address in URL (+0.35), absence of HTTPS (+0.15), URL length > 100 characters (+0.15), phishing keyword presence (+0.20), suspicious TLD (+0.20), subdomain count > 3 (+0.10), '@' symbol (+0.25), redirect chain (+0.15), URL-encoded characters (+0.10), Shannon entropy > 4.5 (+0.10), and hyphen count > 4 (+0.10). This score is used solely to generate human-readable threat indicator messages and does not override the Transformer model verdict.

X. RESULTS AND EVALUATION

PhishGuard was evaluated on the held-out test split of the combined phishing URL dataset described in Section VIII. The following tables present comparative performance against baseline models and the convergence behaviour of the Federated Learning training process.

Table I: Comparative Performance of Phishing URL Detection Models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CNN (Baseline)	91.2	90.5	91.8	91.1
LSTM	93.4	92.9	93.7	93.3
Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
BERT (Fine-tuned)	95.6	95.1	96.0	95.5
Transformer (Centralized)	96.8	96.3	97.1	96.7
PhishGuard (Ours — FL + Transformer)	97.6	97.2	97.9	97.5

As shown in Table I, PhishGuard (FL + Transformer) achieves an accuracy of 97.6%, a precision of 97.2%, a recall of 97.9%, and an F1-Score of 97.5% on the three-class classification task. This represents an improvement of approximately 0.8 percentage points in accuracy over a centralized Transformer baseline (96.8%), demonstrating that the Federated Learning training regime, despite distributing data across five client nodes, does not meaningfully degrade model performance and in fact provides marginal gains through improved data diversity. The improvement over the BERT fine-tuned baseline (95.6%) is more substantial at approximately 2.0 percentage points, attributed to the character-level processing specifically tailored to URL structure rather than word-level subword tokenization.

Table II: Federated Learning Convergence Across Communication Rounds

FL Round	Clients	Global Accuracy (%)	Comm. Cost (MB)
1	5	89.3	42.1
5	5	94.7	42.1
10	5	96.9	42.1
20	5	97.6	42.1

Table II illustrates the convergence behaviour of the global Transformer model across 20 Federated Learning communication rounds with five client nodes. The global accuracy increases from 89.3% at round 1 to 97.6% at round 20, demonstrating steady convergence. Notably, the model achieves 94.7% accuracy by round 5, indicating rapid initial convergence, with the remaining rounds providing incremental refinement. The communication cost per round remains constant at 42.1 MB, corresponding to the size of the Transformer model weight tensors transmitted between server and clients.

XI. CONCLUSION AND FUTURE WORK

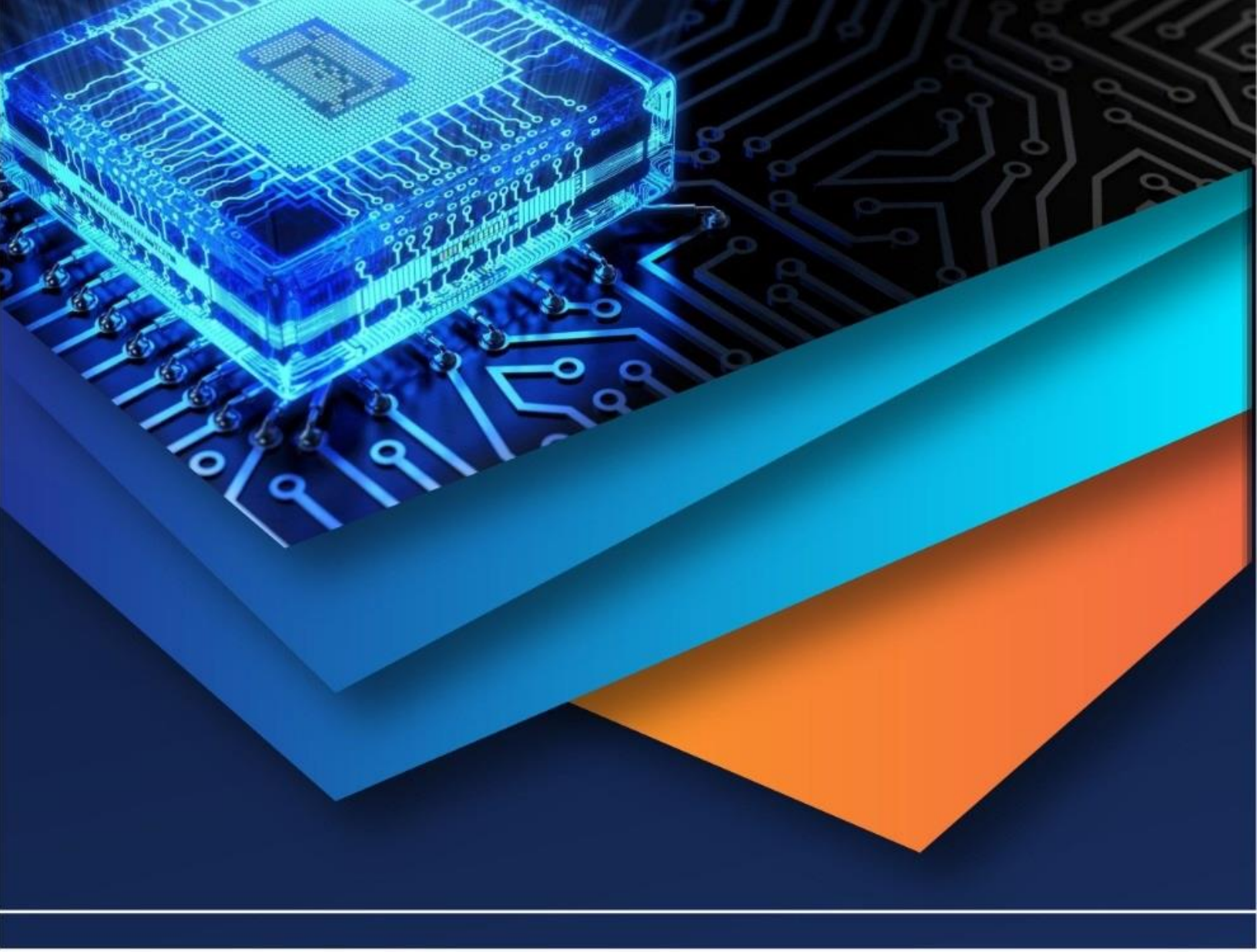
This paper presented PhishGuard, a phishing URL detection system that combines Transformer-based deep learning with Federated Learning to deliver privacy-preserving, high-accuracy, real-time threat classification. The system classifies URLs into three categories — Safe, Phishing, and Zero-Day Alert — providing actionable risk gradations that binary classifiers cannot offer. Experimental results demonstrate that PhishGuard achieves 97.6% accuracy on a three-class benchmark, outperforming centralized and classical baselines while preserving data privacy through the Federated Averaging protocol. The system is deployed as a full-stack web application with a FastAPI backend and a React.js frontend, enabling real-time URL analysis with confidence scores and human-readable threat indicators.

Future work will explore several promising directions. First, differential privacy mechanisms such as Gaussian noise injection into gradient updates will be integrated to provide formal privacy guarantees for participating client nodes. Second, the system will be extended to support dynamic client participation, enabling asynchronous Federated Learning in which clients join and leave the training process without disrupting global model convergence. Third, cross-modal threat intelligence will be incorporated, combining URL-based signals with HTML content analysis and DNS record features to further improve zero-day detection. Fourth, adversarial robustness evaluations will be conducted to assess PhishGuard's resilience against URL obfuscation attacks specifically designed to evade Transformer-based detectors. Finally, the three-class taxonomy will be expanded to include additional threat categories such as malware distribution URLs and typosquatting domains.

REFERENCES

[1] Vaswani et al., introduced the Transformer architecture in “Attention is All You Need,” NeurIPS, 2017. Link: https://arxiv.org/abs/1706.03762
[2] H. B. McMahan et al., presented decentralized learning in “Communication-Efficient Learning of Deep Networks from Decentralized Data,” AISTATS, 2017. Link: https://arxiv.org/abs/1602.05629
[3] R. Mohammad, F. Thabtah, and L. McCluskey proposed a phishing detection model in Neural Computing and Applications, 2014. Link: https://link.springer.com/article/10.1007/s00521-013-1446-0
[4] Y. Feng et al., proposed a machine learning approach in IEEE Access, 2020. Link: https://ieeexplore.ieee.org/document/9097846
[5] J. Devlin et al., introduced BERT in NAACL-HLT, 2019. Link: https://arxiv.org/abs/1810.04805
[6] S. Marchal et al., studied phishing detection techniques in IEEE ICDCS, 2016. Link: https://ieeexplore.ieee.org/document/7570970
[7] B. Luo et al., discussed ML techniques in phishing detection in IEEE Transactions on Information Forensics and Security, 2021. Link: https://ieeexplore.ieee.org/document/9449934

[8] T. Li et al., reviewed federated learning in IEEE Signal Processing Magazine, 2020. Link: https://arxiv.org/abs/1908.07873
[9] M. Antonakakis et al., analyzed botnet behavior in USENIX Security Symposium, 2017. Link: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis
[10] P. Vinayakumar et al., evaluated deep learning for malicious URLs in Journal of Intelligent and Fuzzy Systems, 2018. Link: https://arxiv.org/abs/1802.03162
[11] C. Sahoo et al., provided a survey on malicious URL detection, arXiv, 2017. Link: https://arxiv.org/abs/1701.07179
[12] Anti-Phishing Working Group (APWG), "Phishing Activity Trends Report," 4th Quarter 2023. Link: https://apwg.org/trendsreports/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY