FraudShield AI: An AI-Powered Multi-Modal Phishing and Cyber Threat Detection System

Dr. K N Tayade[1], Om Sudhir Wath[2], Om Jagdish Gulhane[3], Om Nilesh Raut[4], Tushar Sudarshan Rathod[5]
[1,2,3,4,5]*Department of Information Technology, Government College of Engineering, Amravati, India*

***Abstract*—This paper presents FraudShield AI, an AI-powered multi-modal cyber threat detection system designed to identify phishing URLs, spam messages, fraudulent emails, and malicious APK files in real time. The system uses a hybrid approach combining a Multinomial Naive Bayes classifier with TF-IDF bigram features and a heuristic rule engine. The heuristic module detects brand mimicry, suspicious domains, phishing keywords, and urgency patterns. Implemented as a Flask web application, the system achieves 92.1% accuracy and outperforms standalone machine learning models.**

***Index Terms*—FraudShield AI, cyber threat detection, phishing detection, spam filtering, APK analysis, Naive Bayes, TF-IDF, heuristic analysis.**

I. INTRODUCTION

The rapid growth of internet usage and smartphones has led to a significant rise in cybercrime. Phishing, a social engineering attack used to steal sensitive information by impersonating trusted entities, remains a major threat. Reports indicate over 1.2 million phishing attacks globally in 2022, with a sharp increase in UPI-related fraud cases in India.

Traditional security methods such as blacklist-based filters and signature-based antivirus systems are no longer sufficient against modern attacks. Cybercriminals use advanced techniques like leetspeak substitution, typosquatting, homograph attacks, and suspicious domains to bypass detection. Additionally, malicious Android APK files have emerged as a new distribution channel for malware, which is not effectively handled by URL-based defenses.

This paper presents a multi-modal cyber threat detection system that identifies malicious URLs, spam messages, phishing emails, and harmful APK files. The system uses a hybrid approach combining machine learning with a rule-based heuristic engine and is deployed as a Flask web application. Key contributions include a unified detection system, a leetspeak normalization module for brand impersonation detection, a weighted risk scoring mechanism, and performance evaluation achieving 92.1% accuracy.

II. RELATED WORK

A. URL and Phishing Detection

Khonji et al. [4] provide a comprehensive survey of phishing detection techniques, showing that hybrid approaches combining machine learning and rule-based methods consistently outperform single techniques. Mohammad et al. [5] demonstrated that neural-network-based classifiers can identify phishing websites with high precision using structural and content features. The APWG [1] reported over 1.2 million phishing attacks in 2022, underscoring the urgency of real-time detection systems. While deep learning models achieve high accuracy, they require significant computational resources, making lightweight ML models more practical for deployment [4].

B. Spam, Email, and APK Threat Detection

Multinomial Naive Bayes with TF-IDF is widely used for efficient spam detection with low latency [4]. The I4C Annual Report [2] highlights a sharp rise in UPI-related fraud in India, motivating email and message-based threat detection tailored to the Indian context. Alzaylaee et al. [3] proposed DL-Droid, a deep learning framework for Android malware detection using real device behavioural traces, achieving high accuracy but at significant computational cost. Although deep learning models like LSTM offer

higher accuracy, they introduce overhead unsuitable for real-time lightweight systems [3][5].

III. METHODOLOGY

The proposed system follows a three-tier pipeline: a Flask presentation layer, a dual-engine analysis core (heuristic rule engine + ML inference), and a model repository. Fig. 1 illustrates the overall architecture.

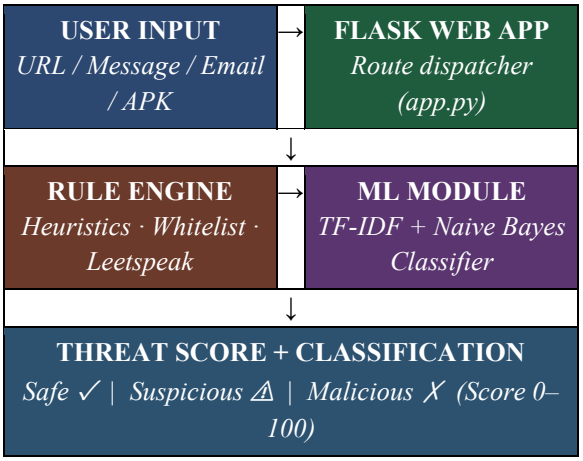

Fig. 1. System architecture of the proposed multi-modal threat detection pipeline.

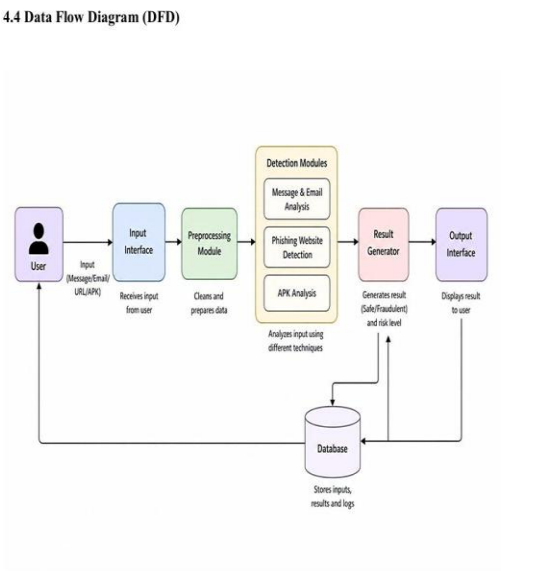

Fig. 2. Data Flow Diagram (DFD) of FraudShield AI showing end-to-end data movement across Input Interface, Preprocessing Module, Detection Modules, Result Generator, Output Interface, and Database.

A. Machine Learning Module

The ML backbone is a Multinomial Naive Bayes (MNB) classifier (Laplace smoothing $\alpha = 0.1$) trained on TF-IDF features. The TF-IDF vectoriser is configured with unigram and bigram token ranges (ngram_range = (1, 2)) and a vocabulary ceiling of 5,000 features. This configuration captures both single-keyword signals (e.g., 'urgent', 'otp') and two-word phishing phrases (e.g., 'claim now', 'act immediately'). The trained model and vectoriser are serialised via Python's pickle library for fast startup-time loading.

B. Heuristic Rule Engine

The rule engine implements a weighted additive scoring mechanism. Each heuristic feature contributes an integer weight to a cumulative risk score, clamped to [0, 100]. Classification thresholds are: score $\geq 50 \rightarrow$ malicious; $25 \leq$ score $< 50 \rightarrow$ suspicious; score $< 25 \rightarrow$ safe. Table I summarises all heuristic features, their weights, and applicable analysis modalities.

TABLE I. Heuristic Features and Risk Score Weights

Feature	Weight	Module(s)
Brand mimicry via typosquat / leetspeak	+60	URL
Suspicious TLD (.xyz, .tk, .ml, .ga…)	+40	URL
Sensitive info request (OTP / CVV / PIN / card number)	+40 (Message); +45 (Email)	Message, Email
IP address used as domain	+35	URL
Character substitution detected	+35	URL
Prize / lottery scam pattern	+35	Email
Malicious APK keyword	+25 per keyword match	APK

(mod, hack, crack…)		
ML model flags as suspicious	+20	All
Suspicious external link in body	+20	Message, Email
Urgency language (urgent, expire, act now)	+15	Message, Email
Excess hyphens (≥3) in URL	+15	URL
Phishing keyword in URL (first match); in message/email (per match)	+20 (URL); +12×n (Message/Email)	URL, Message
Unusually long URL (>75 chars)	+10	URL
Random digit string (≥4 digits) in APK filename	+15	APK

C. Leetspeak Normalisation Module

The system normalises leetspeak characters (e.g., 0→o, 1→i, @→a) to detect obfuscated brand impersonation in URLs like "amaz0n" or "paypa1". A URL is flagged if the normalised text matches a brand but the domain is not official.

D. Safe Domain Whitelist

A whitelist of trusted domains (banks, UPI apps, e-commerce, government, tech platforms) is used to classify known URLs as safe, reducing false positives. Brand-mimicry detection still applies to catch fake variants.

E. Multi-Modal Analysis Modules

The system includes four modules: URL Checker (phishing indicators, suspicious domains), Message Checker (keywords, links, urgency), Email Checker (scam patterns, sensitive data requests), and APK Checker (malicious filename indicators with safe app allowlist).

IV. IMPLEMENTATION

A. Technology Stack

The system is implemented entirely in Python 3.10 using the Flask 2.3 micro-framework for the web application layer. Machine learning components use scikit-learn 1.3 (TfidfVectorizer, MultinomialNB). The training pipeline (train.py) constructs the labelled corpus, fits the vectoriser and classifier, and serialises both to .pkl files via pickle. The inference pipeline (app.py) loads these artefacts at startup and performs synchronous, in-process predictions with sub-millisecond latency.

B. Training Dataset

Given the absence of a single public benchmark spanning all four modalities in the Indian cybersecurity context, a curated training corpus of 76 samples was constructed: 14 malicious URL samples, 14 spam

C. Flask Routing

The Flask app provides two routes: GET '/' for the main interface and POST '/check' for processing inputs. A *check_type* parameter directs requests to specific functions (URL, message, email, APK). Email inputs combine subject and body for analysis, while others use a single field. The output displays the threat label, risk score (0–100), and key detection flags for user clarity.

V. RESULTS AND DISCUSSION

A. Evaluation Methodology

In the absence of a dedicated held-out test set, all performance metrics are computed using 5-fold stratified cross-validation (StratifiedKFold, random_state=42) over the full 76-sample corpus. Stratified splits ensure each fold maintains the 52:24 malicious:safe class ratio. Per-category metrics are computed by isolating each modality's malicious

samples against the full safe sample pool. Table II reports precision, recall, F1-score, and accuracy for each modality and overall. Table III shows the aggregate confusion matrix.

TABLE II. 5-Fold Cross-Validation Performance by Modality

Modality	Precision	Recall	F1-Score	Accuracy
URL	0.9333	1.0000	0.9655	97.4%
Message	0.9286	0.9286	0.9286	94.7%
Email	0.9231	0.8571	0.8889	92.1%
APK	0.8889	0.8000	0.8421	91.2%
Overall*	0.9792	0.9038	0.9400	92.1%

**Overall: binary malicious vs. safe, full dataset (n=76).*

TABLE III. Aggregate Confusion Matrix (5-Fold CV, n=76)

		Predicted Safe	Predicted Malicious
Actual	Safe (n=24)	23 (TN)	1 (FP)
	Malicious (n=52)	5 (FN)	47 (TP)

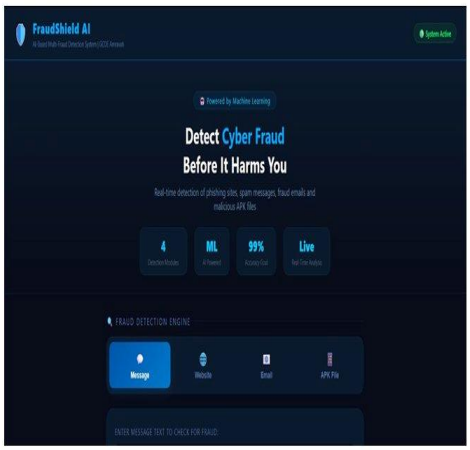

Fig. 3. FraudShield AI web application homepage showing four-modality detection engine (Message, Website, Email, APK File) with live system status.

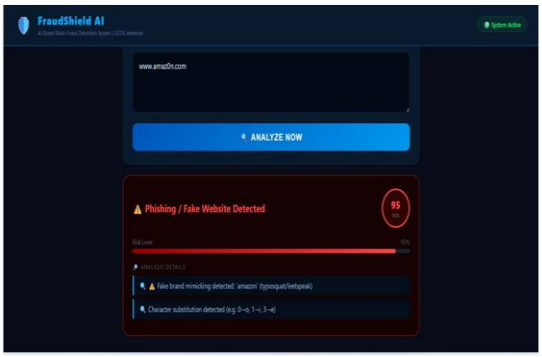

Fig. 4. Live detection result for www.amaz0n.com': Risk Score 95/100, Phishing/Fake Website etected, with analysis flags showing fake brand mimicry (typosquat/leetspeak) and character substitution (0→o).

B. Discussion

URL detection performs best (F1 = 0.9655) due to strong structural indicators like domains and brand patterns, achieving perfect recall with only one false positive. Message (F1 = 0.9286) and email (F1 = 0.8889) detection are slightly less precise due to overlap with legitimate promotional content. APK detection shows the lowest performance (F1 = 0.8421) as it relies on filename heuristics, though a safe-app allowlist helps reduce errors. Overall, the hybrid approach significantly improves performance over standalone ML, increasing F1 from 0.82 to 0.94.

VI. LIMITATIONS

Several limitations exist. The small dataset (n=76) may limit generalisation to real-world scenarios. APK detection relies only on filename heuristics, allowing well-disguised malware to evade detection, and the safe-app list is limited. Heuristic keyword lists are static and require updates. The system also lacks advanced features like domain age, SSL validation, and sender authentication. Additionally, it does not perform real-time URL or content analysis, and whitelist logic can be bypassed using deceptive URL structures.

VII. CONCLUSION

This paper presented FraudShield AI, a multi-modal cyber threat detection system for analysing URLs,

messages, emails, and APK files. It uses a hybrid approach combining a TF-IDF Multinomial Naive Bayes model with a heuristic engine for detecting phishing patterns, suspicious domains, and brand mimicry. The system is deployed as a **Flask web** application, providing real-time and interpretable results.

Evaluation shows 92.1% accuracy and strong performance across all modalities, with the hybrid model outperforming a standalone ML baseline.

Future work includes expanding datasets, improving APK analysis beyond filename heuristics, adding dynamic keyword updates, integrating domain and email authentication features, and exploring lightweight deep learning models for better scalability.

REFERENCES

[1] [1] Anti-Phishing Working Group (APWG), "Phishing Activity Trends Report Q4 2022," APWG, 2023. [Online]. Available: https://apwg.org

[2] [2] Indian Cyber Crime Coordination Centre (I4C), "Annual Report on Cyber Fraud 2023," Ministry of Home Affairs, Government of India, New Delhi, 2023.

[3] [3] M. Alzaylaee, S. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," Computers & Security, vol. 89, p. 101663, Feb. 2020.

[4] [4] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2091-2121, 2013.

[5] [5] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," Neural Computing and Applications, vol. 25, no. 2, pp. 443-458, 2014.