IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

AEGISGUARD: A MULTI-VECTOR MALICIOUS URL AND SCAM PREVENTION SYSTEM

Mayuri Sharad Bhamare[1], Manoj V. Nikum[2]

Research Scholar MCA, Shri Jaykumar Rawal Institute of Technology, Dondaicha, India[1]

HOD MCA & Assistant Professor, Shri Jaykumar Rawal Institute of Technology, Dondaicha, India[2]

Abstract: Phishing and scam campaigns increasingly use deceptive URLs, short message service (SMS) lures, QR-code destinations, and multi-hop redirects to evade user awareness and conventional reputation-based controls. This paper presents AegisGuard, a lightweight web-based threat inspection system designed to analyze these vectors in a unified workspace without a backend SQL/NoSQL database or mandatory user authentication. The system combines lexical and structural URL indicators, a Random Forest classifier, rule-based smishing analysis, client-side QR decoding, and redirect-chain inspection. The source project report describes 21 URL feature extractors, model training on a stated dataset of 50,000 URLs using 100 decision trees with maximum depth 15, and risk categories of Safe, Suspicious, and Malicious. A Flask application exposes inspection endpoints, while browser localStorage retains a rolling history of up to 50 scan records. The report records 20 functional test scenarios passing and an average execution time of 142 ms for 100 concurrent scan requests. These figures are reported project results rather than independently reproduced benchmarks. The design prioritizes low-friction access and reduced server-side retention, while remaining limited by heuristic false positives, dependence on model and rule quality, and the absence of a live reputation feed.

Keywords: Malicious URL Detection, Phishing Prevention, Random Forest, Smishing, QR-Code Phishing, Redirect Analysis, Privacy-Preserving Security

I. INTRODUCTION

Digital communication channels are routinely used to distribute phishing links and social-engineering messages. Attackers can disguise destinations through brand impersonation, URL shorteners, unusual domain structures, QR codes, and urgency-based SMS content. A detection system that evaluates only a known-domain blacklist may not recognize a newly created or previously unindexed domain. Users also face friction when separate tools are required to inspect URLs, messages, QR images, and redirect chains.

AegisGuard is designed as a unified malicious URL and scam prevention workspace. Its central objective is to provide rapid, multi-vector inspection using local feature extraction and a machine-learning classifier, without requiring a backend database or user account. The project focuses on application-level payload inspection; it does not claim to provide network packet inspection or hardware firewall enforcement.

The contributions described in this paper are: (1) a 21-marker lexical URL analysis pipeline; (2) integration of Random Forest classification with rule-based risk scoring; (3) inspection support for smishing text, QR-code payloads, and HTTP redirects; and (4) local browser telemetry intended to reduce centralized retention of scan history.

II. RELATED WORK

Blacklist and domain-reputation services are useful for known threats, but their effectiveness for newly registered or short-lived malicious domains depends on discovery and update cycles. Feature-based URL classification instead examines structural signals such as length, character distribution, suspicious symbols, IP-address hosts, and domain similarity. Sahoo et al. survey machine-learning approaches to malicious URL detection [2], while Random Forest provides an ensemble method for combining nonlinear feature relationships [1].

Social-engineering attacks also exploit urgency, account-suspension claims, and financial lures. Jagatic et al. discuss social phishing and the role of contextual deception [5]. AegisGuard applies keyword and pattern matching to message text and combines that result with any embedded URL analysis. QR-code phishing, commonly called quishing, is addressed by decoding the QR payload before a user follows the destination. Redirect inspection complements these approaches by exposing intermediate HTTP 301/302 destinations.

Unlike cloud-dependent scanners, the proposed architecture emphasizes in-memory backend processing and browser-side storage for recent telemetry. This reduces dependence on a database service but does not, by itself, guarantee that all

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

operational data is private: deployment logs, hosting configuration, browser compromise, and redirect requests may still expose information.

III. METHODOLOGY

A. System Architecture

AegisGuard uses a three-tier design. Tier 1 is the browser dashboard implemented with HTML, CSS, and JavaScript. Tier 2 is a Python Flask application that validates requests and coordinates scanner modules. Tier 3 contains the pre-trained Random Forest model and its feature-processing pipeline. The browser sends JSON requests to the Flask API, receives a structured verdict and diagnostic details, and updates the dashboard without a full page reload.

B. URL Feature Extraction and Risk Scoring

The lexical engine extracts 21 structural markers, as described by the project report. Examples include URL length, character entropy, hyphen density, subdomain depth, the presence of an '@' symbol, raw IP-host patterns, suspicious top-level domains, and brand-like strings. Shannon entropy is calculated as $H(U) = -\Sigma\ P(x_i)\ \log_2 P(x_i)$, where $P(x_i)$ is the observed probability of character x_i in the analyzed string. Higher entropy can indicate a randomized or obfuscated string, but is not sufficient evidence of maliciousness on its own.

The system also uses Levenshtein edit distance to identify domain names that resemble monitored brands. The classifier described in the report is a Scikit-learn Random Forest trained on 50,000 benign and phishing URL records, with 100 trees and a maximum tree depth of 15; StandardScaler is reported for continuous features. The report gives the final labels as SAFE for scores below 30, SUSPICIOUS for scores from 30 to below 65, and MALICIOUS for scores of 65 or higher. A simplified heuristic in the supplied implementation excerpt adds risk points for long URLs, raw IP hosts, '@' symbols, and entropy above 4.2. The paper therefore treats the classifier and heuristic rules as complementary components rather than assuming that every risk score is a calibrated probability.

C. Smishing, QR, and Redirect Inspection

The smishing analyzer searches message text for urgency language, account-suspension claims, financial lures, brand impersonation, and embedded URLs. Extracted URLs can be passed to the lexical scanner so that text context and URL structure contribute to a combined assessment. The QR inspector decodes uploaded PNG, JPG, or WebP images in the browser using HTML5 Canvas and jsQR, allowing the encoded destination to be reviewed before navigation. The redirect unmasker follows HTTP 301/302 responses and

E. Evaluation Approach

records the destination chain, with a stated maximum traversal depth of 10 hops.

D. Privacy and Local Telemetry

AegisGuard does not use a backend SQL/NoSQL database for scan history. The frontend stores recent diagnostic records and counters using browser localStorage, with a rolling maximum of 50 scan entries. This reduces server-side persistence of user scan history, but localStorage is not encrypted storage and can be accessible to scripts running in the same origin. Production deployments should apply standard input validation, secure redirect-fetch controls, rate limiting, and appropriate logging minimization.

The project report describes a 20-case functional test matrix covering URL categories, smishing messages, QR input handling, redirect traversal, local telemetry, user-interface presets, and performance. It reports all 20 scenarios as passing and an average execution time of 142 ms for 100 concurrent scan requests. The supplied report does not provide a reproducible dataset split, confusion matrix, precision, recall, F1-score, hardware specification, or raw timing logs. Accordingly, this paper reports the figures as project-reported observations, not as independently verified generalization or production-performance claims.

IV. RESULTS AND DISCUSSION

The source report records a 100% pass rate across 20 functional scenarios and a mean execution time of 142 ms in a test described as 100 concurrent scan requests. This is consistent with the project's target of sub-200 ms inspection. The modular design also consolidates four operational tasks—URL scanning, message analysis, QR decoding, and redirect tracing—within one interface. Browser-side QR decoding avoids opening the decoded destination automatically, and local telemetry avoids a dedicated database for scan-history persistence.

These results should be interpreted within the limits of the available evidence. A functional pass rate measures the tested scenarios, not detection accuracy across real-world traffic. The report does not provide enough raw data to calculate false-positive rate, false-negative rate, precision, recall, or statistical confidence intervals. Moreover, lexical signals and keyword rules can be evaded or can flag legitimate URLs; a Random Forest model may degrade when threat patterns or

data distributions change. The claimed response time may also vary with deployment hardware, network behavior, and redirect targets.

Table I. Project-Reported Validation Summary

Metric	Reported result	Interpretation
Functional test cases	20 of 20 passed	Project report's test matrix
Concurrent scan test	100 requests	Test setup stated in report
Average execution time	142 ms	Reported average; raw logs unavailable
URL feature set	21 markers	Structural/lexical features described
Model configuration	100 trees; depth 15	Training details stated in report
Local telemetry history	Up to 50 entries	Browser localStorage rolling history

V. CONCLUSION

This paper presented AegisGuard, a lightweight, database-independent workspace for inspecting suspicious URLs, smishing text, QR-code destinations, and redirect chains. The design combines lexical analysis, a Random Forest classifier, rule-based message analysis, browser-side QR decoding, and local telemetry. According to the supplied project report, all 20 listed functional tests passed and average execution time was 142 ms for a 100-request test.

Future work should evaluate the detector on a documented, time-separated dataset and publish confusion matrices, precision, recall, F1-score, and per-module latency measurements. Additional improvements include calibrated probability outputs, continuous model refresh, carefully controlled reputation/WHOIS integrations, browser-extension packaging, adversarial testing, and stronger protections for redirect fetching. Until such evaluation is available, AegisGuard should be considered a decision-support tool and not a substitute for browser protections or security awareness.

ACKNOWLEDGMENT

The author(s) may acknowledge the project guide, department, and institution here. Replace this placeholder with the appropriate acknowledgment before submission.

REFERENCES

[1]. L. Breiman, “Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[2]. D. Sahoo, C. Liu, and S. C. H. Hoi, “Malicious URL detection using machine learning: A survey,” IEEE Access, vol. 7, pp. 17296–17316, 2019.
[3]. Google Safe Browsing Team, “Evaluating Domain Risk at Scale,” Google Security Whitepaper, 2023. Details should be verified against the original publication before submission.
[4]. M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed. Sebastopol, CA: O’Reilly Media, 2018.
[5]. T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, “Social phishing,” Communications of the ACM, vol. 50, no. 10, pp. 94–100, 2007.