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 10, October 2026

DOI: 10.17148/IARJSET.2026.131034

SentinelURL: AI Framework for Predictive Phishing Intelligence

Mr. Hemanth Kumar N P[1], Shruti Sharma[2], Pushpa S. Rathod[3], Roshini S.[4], Sahana S.[5]

Dept. of CSE, Don Bosco Institute of Technology, Bengaluru, India[1-5]

Abstract: Phishing websites copy trusted services to trick people into handing over passwords, banking details and other private data. Blacklists catch links that have already been reported, but they say nothing about a URL that was created or changed an hour ago. This paper presents SentinelURL, a machine-learning framework that judges a URL from its own lexical and structural characteristics before the user opens it. The workflow takes a URL, validates and preprocesses it, turns it into a numerical feature vector, and passes that vector to a trained classifier. The features include URL length, the number of dots and special characters, suspicious keywords, IP-address use, subdomain patterns, entropy and signs of obfuscation or typosquatting. The project report names Random Forest as the core classifier, and the project presentation lists Logistic Regression and XGBoost as models to compare it against. A Python backend and a simple interface return a clear category and warning, so that users can decide more safely. The project materials do not contain a full set of measured results, so this paper reports no accuracy, precision, recall, F1 or latency figures. It sets out the design, the implementation, the evaluation protocol, the present limitations and the steps needed for a rigorous test.

Keywords: phishing URL detection, machine learning, Random Forest, lexical features, cybersecurity, URL classification, predictive threat analysis

I. INTRODUCTION

People use links every day to bank, shop, study and talk to each other. That convenience gives attackers an easy opening: they send a user to a fake site that looks like a service they already trust. If the user falls for it, credentials and personal details can leak, and the result may be financial loss or someone else taking over an account.

Most traditional defences check a URL against a list of sites that were reported earlier, or apply fixed rules. Both still have a place, but a brand-new or slightly altered link may not be on any list yet. Attackers also make links hard to read on purpose, using long paths, odd symbols, misleading subdomains, raw IP addresses, or names that look almost like a real domain. For this reason SentinelURL looks at the structure of the URL itself instead of relying only on whether the link is already known.

The project follows a cautious, zero-trust-inspired idea: a URL should be assessed before the user depends on it. The framework measures a set of URL characteristics and gives them to a supervised model, which classifies the link. The aim is a warning that is practical and easy to understand, built in a modular way so that models can be compared and features improved over time.

A. Problem Statement

New and short-lived phishing URLs can slip past static blacklists, and convincing links can mislead people. A lightweight tool is needed that inspects URL features and gives a timely risk classification before the destination is opened. Because both missed attacks and false alarms reduce trust in such a tool, its classifications have to be evaluated with care.

B. Objectives

- Build a dataset of legitimate and malicious URLs, remove invalid and duplicate entries, and label them consistently.
- Extract lexical and structural features that can be expressed as numbers for supervised learning.
- Train and compare suitable classifiers, with Random Forest as the main model in the project report.
- Provide a simple interface that accepts a URL and shows the predicted category with an understandable warning.
- Evaluate the model with accuracy, precision, recall, F1-score, a confusion matrix and prediction latency.

II. RELATED WORK

Earlier research tackles phishing through several different signals. Email-focused systems study the wording of a message, the sender and the relationships between correspondents. Joseph and Srinivasan, whose paper served as the base paper for this project, analyse email content in real time using machine learning, natural-language processing,

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 10, October 2026

DOI: 10.17148/IARJSET.2026.131034

anomaly detection and adaptive feedback [1]. Their target is the email text and the social-engineering intent behind it, whereas SentinelURL concentrates on the URL. The two views complement each other without replacing each other: a URL-only model cannot read the body of an email, and an email-content model may not look closely at the structure of the link inside it.

The project presentation also lists studies on AI-driven email classification [2], DistilBERT-based separation of human-written from AI-generated phishing emails [3], and the role of generative AI in phishing detection [4]. Together they show that text semantics, sender relationships, explainability and changing attacker behaviour are all active research directions. Table I summarises them. Any results reported in those studies belong to those studies and are not performance achieved by SentinelURL.

Ref.	Study focus	Main signal	Relation to SentinelURL
[1]	Anti-phishing adaptive AI for social-engineering attacks	Email content; ML, NLP, anomaly detection, feedback	Base paper; works on email text, SentinelURL works on the URL
[2]	AI-driven email classification and phishing detection	Email content	Related message-level detection; different input
[3]	Human-written vs. AI-generated phishing emails	Email text; DistilBERT, explainable AI	Shows value of explainability; text-based
[4]	Generative AI in phishing detection	Survey of opportunities	Context on evolving attacks and research gaps

TABLE I. Related studies listed in the project materials and how they differ from SentinelURL.

SentinelURL deliberately takes a lightweight, feature-based route. URL features can be computed without downloading or running the destination page, which makes the approach suitable as a quick first check. The trade-off is that lexical signals are indirect evidence: a genuine URL can be long or unusual, and a malicious one can be made to look ordinary. Adding reputation, DNS, certificate, web-page or email-context signals is a sensible extension, but it should only be described as part of the system once it has actually been built and tested.

III. PROPOSED SYSTEM

A. System Overview

SentinelURL has three logical layers, shown in Fig. 1. The presentation layer receives a URL and displays the result. The application layer validates the input, calls the feature extractor and hands the resulting feature vector to the model. The intelligence layer applies the trained classifier and returns a category. The project report identifies a Scikit-learn Random Forest as the core classifier, and the presentation proposes comparing it with XGBoost and Logistic Regression.

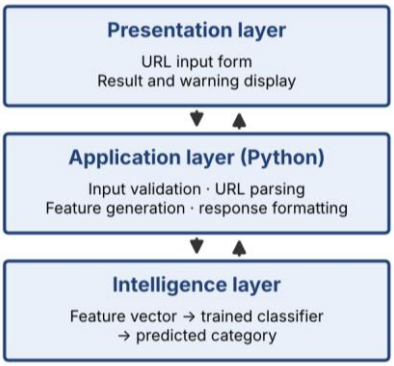


Fig. 1. Three-layer architecture of SentinelURL.

B. Data Collection and Labeling

According to the project report, legitimate URLs are collected from a trusted-domain source and phishing URLs from public cybersecurity repositories such as PhishTank [5]. The records are cleaned, formatted, stripped of duplicates and combined into a CSV file, with label 0 for safe and 1 for phishing. The presentation shows a wider set of output classes (safe, phishing, malware and defacement), but nothing in the materials shows that one dataset and model have already

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 10, October 2026

DOI: 10.17148/IARJSET.2026.131034

been validated for all four. The final experiment should therefore state the exact dataset sources, access dates, record count per class and label definitions.

Data preparation should be reproducible: malformed URLs are checked, duplicates removed, encoding standardised where appropriate and labels kept consistent. The train/test split must be made before any feature selection or tuning, and near-duplicate URLs or links from the same campaign must not be spread across both partitions, since that leakage makes results look better than they are.

C. URL Feature Extraction

This stage turns a URL into numbers. The project materials mention URL length, number of dots, special characters, suspicious keywords, IP-address use, subdomain patterns, entropy, the digit-to-letter ratio and signs of obfuscation or typosquatting. The presentation refers to 25 numerical features; the final version should list every feature by name together with its definition so that the work can be reproduced.

D. Classification and Output

The feature vector is passed to the trained classifier. Random Forest, the main model in the report, combines the votes of many decision trees and can capture non-linear relationships among features. Logistic Regression and XGBoost are candidates for comparison. The choice should rest on validation data and a stated metric, not on assumption.

The interface should show the predicted category with a clear safety message. A probability or risk score should be displayed only if it is calibrated, or if it is clearly described as a model score and not as a guaranteed chance of harm. The system should never open the submitted URL during analysis.

E. Threat Classes and Scope

The slide architecture lists four labels: safe/benign, phishing, malware and defacement. The detailed data-collection section of the report, however, describes only safe and phishing URLs. Multi-class detection is therefore an intended extension unless labeled malware and defacement records and a matching trained classifier exist. In the same way, the web application is a target deliverable, and the Chrome extension is a stated objective that should be reported as implemented only after it has been built and tested.

IV. SYSTEM WORKFLOW

The workflow, illustrated in Fig. 2, is: User Submits URL → Validation and Preprocessing → Feature Extraction → Classification → Category and Warning. The user pastes a link; the backend checks that it is well formed and rejects invalid input; the extractor computes the feature vector; the classifier predicts a category; and the interface presents the result. Keeping the stages separate means the model can be replaced or retrained without redesigning the interface.

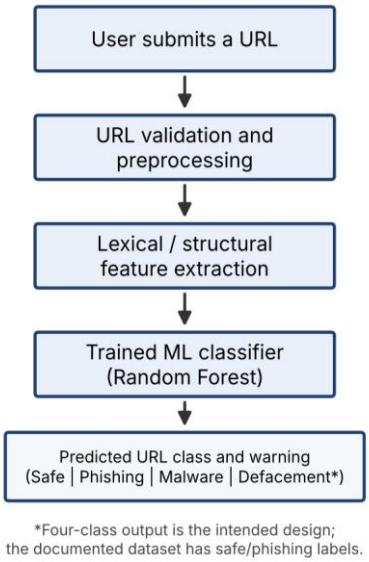


Fig. 2. SentinelURL URL-analysis workflow, redrawn from the project presentation.

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 10, October 2026

DOI: 10.17148/IARJSET.2026.131034

V. METHODOLOGY

A. Dataset preparation. Collect URLs from documented sources, record retrieval dates and usage conditions, validate URL syntax, remove duplicates, assign labels and report the number of examples per class. Keep a held-out test set that is never used for feature selection or tuning.

B. Preprocessing. Normalise the representation only where doing so does not erase meaningful signals. Parse each URL into scheme, hostname, path, query and port, handle missing or malformed values consistently, and apply the same transformation in training and in prediction.

C. Feature extraction. Compute the exact lexical and structural attributes used by the implementation, and store their names and order together with the model so that training and prediction use the same representation.

D. Model training. Train the Random Forest described in the report. If Logistic Regression and XGBoost are available, train them on the same split with a comparable tuning procedure. Use cross-validation on the training data where feasible and pick the model with a declared validation metric.

E. Application integration. Connect the trained model to a Python backend and a front-end interface. The backend validates inputs, preprocesses, extracts features, calls the model and returns a structured result. Errors should produce safe messages and never expose stack traces.

F. Evaluation. Report accuracy, precision, recall, F1-score, a confusion matrix and latency. For imbalanced data add per-class precision and recall and macro-F1. State the hardware, software versions, split strategy, random seed and dataset date so another team can repeat the experiment.

VI. IMPLEMENTATION DESIGN

A. Presentation layer. The user pastes a URL and submits it. The page shows a clear category, a short explanation and a reminder that automated classification is not a guarantee. The report describes a simple interface and the presentation names a React front end as part of the planned integration; the final manuscript should name the framework actually used in the working code.

B. Application layer. The Python backend links the interface to the model through input validation, URL parsing, feature generation, model invocation and response formatting. Separating these duties allows the model to be updated without touching the user interface.

C. Intelligence layer. The Scikit-learn Random Forest is the primary classifier. Training and prediction must share the same feature order and preprocessing, and the model file should be versioned with its feature definitions and evaluation record. A feedback mechanism could later collect reviewed false positives and false negatives, but model updates should be controlled and tested and not applied automatically from unverified reports.

D. Security and privacy. By default the system should not visit the destination, because a live request can expose the user to a malicious page or reveal information to it. Logs should keep sensitive query parameters to a minimum and should not hold full URLs longer than needed. The interface must also withstand malformed input and must make clear that a model prediction is not a verified security verdict.

VII. EVALUATION PLAN AND RESULTS

The project materials describe the intended metrics but give no complete numerical test report for SentinelURL. For that reason no accuracy, precision, recall, F1, latency or winning model is claimed here, and the email-detection results of the base paper, which concern a different task, are not reused. The metrics to be reported are defined below, with TP, TN, FP and FN denoting true positives, true negatives, false positives and false negatives.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (1)$$

$$Precision = TP / (TP + FP) \quad (2)$$

$$Recall = TP / (TP + FN) \quad (3)$$

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 10, October 2026

DOI: 10.17148/IARJSET.2026.131034

$$F1 = 2PR / (P + R) \quad (4)$$

Here P and R are precision and recall. Precision shows how many flagged URLs really are malicious, and recall shows how many malicious URLs are caught. Latency is the time from submitting a URL to receiving a prediction, measured over repeated runs and summarised with the median and a high percentile. For a multi-class experiment, per-class scores, macro and weighted averages and a confusion matrix should be given. If the final dataset is binary, it must be reported as a binary task, without implying that malware and defacement detection has been demonstrated.

Model	Acc.	Prec.	Recall	F1	Latency
Random Forest	Pending	Pending	Pending	Pending	Pending
Logistic Regression	Pending	Pending	Pending	Pending	Pending
XGBoost	Pending	Pending	Pending	Pending	Pending

TABLE II. Results template. Entries are to be filled from one common held-out test set once experiments are run.

A. Discussion

A feature-based classifier is attractive because it is light and can predict without analysing the whole page. Its main weakness is that URL structure is only indirect evidence of intent. Legitimate URLs can have long paths, unusual parameters or many subdomains, and a carefully made phishing URL may show few warning signs. Reliability therefore depends on how diverse and fresh the dataset is, how good the labels are, and whether the evaluation reflects real use.

A fair comparison must also count false positives and false negatives instead of looking at accuracy alone. A model that flags too many safe links becomes annoying, while one that misses phishing links gives a false sense of security. Thresholds, calibration and the wording shown to users should be chosen with this balance in mind.

VIII. LIMITATIONS

- The materials lack a fully reproducible dataset description: exact record counts, retrieval dates, class balance and the final split protocol.
- The report documents safe and phishing labels; the malware and defacement categories in the presentation need to be confirmed against the actual dataset and trained model.
- No complete set of measured SentinelURL results exists yet, so quantitative claims must wait for experiments.
- URL-only features cannot fully capture page content, email context, redirects, or changes made after a URL is first seen.
- The browser extension and continuous or adaptive learning remain objectives or future work until implemented and tested.

IX. CONCLUSION AND FUTURE WORK

This paper described SentinelURL, a machine-learning framework that predicts whether a URL may be malicious by examining its lexical and structural features. The design keeps the interface, Python backend, feature extraction and classifier separate, with Random Forest as the principal model in the project report. The goal is a timely, understandable warning that does not depend only on URLs someone has already reported.

The next step is a reproducible evaluation on a documented dataset with a held-out test set, reporting accuracy, precision, recall, F1-score, false-positive and false-negative counts and latency. Later work may add a verified multi-class dataset, explanations of feature contributions, domain-reputation and redirect analysis, a carefully designed browser extension, privacy-preserving logging and a controlled feedback process for periodic retraining. Each of these should be claimed as complete only after it has been built and evaluated.

REFERENCES

[1] A. S. K. Joseph and S. Srinivasan, “Anti-Phishing Adaptive AI Systems: Efficiently Countering Social Engineering Attacks by Real-Time Analysis of Email Content,” in Proc. Int. Conf. Computational Innovations and Engineering Sustainability (ICCIES), 2025, doi: 10.1109/ICCIES63851.2025.11032758.

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 10, October 2026

DOI: 10.17148/IARJSET.2026.131034

[2] L. Naik, B. K. S., S. S. S., and S. K., "AI driven email classification and phishing attack detection for secure communication," in Proc. 9th Int. Conf. Computational System and Information Technology for Sustainable Solutions (CSITSS), 2025.

[3] A. Anil Kumar, T. P. Imthias Ahamed, and N. A. Shukoor, "Detecting Human-Written and AI-Generated Phishing Emails with DistilBERT and Explainable AI," in Proc. IEEE 2nd Int. Conf. for Women in Computing (InCoWoCo), 2025.

[4] O. Perera and J. Grob, "Generative AI in Phishing Detection: Insights and Research Opportunities," in Proc. Cyber Awareness and Research Symposium (CARS), 2024.

[5] PhishTank, "PhishTank: Join the fight against phishing." [Online]. Available: https://phishtank.org/

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