Intelligent Mobile Cyber Guard Platform using Machine Learning (KAYO)

Mr. D. Siva Theja[1], B. Pravallika[2], S. Alsaba[3], S. Thasmiya[4], P. Prasanna Lakshmi[5],B. ReddyParvathi[6]

[1]Assistant Professor, [2,3,4,5,6]UG Students, Aditya College of Engineering, Madanapalle – 517325, Andhra Pradesh, India
Corresponding author: sivatheja.devarinty@gmail.com

Abstract: With the widespread use of smartphones, mobile users are increasingly becoming targets of web-based attacks such as phishing, scam pages, and malware distribution sites. Existing desktop-focused tools often fail to capture behaviors that are unique to mobile web environments. This paper presents Kayo, a machine learning-based system specifically designed to detect malicious mobile webpages in real time. Kayo crawls target URLs using a mobile user agent, extracts 62 features across six categories (URL-based, HTML structural, JavaScript behavioral, content-based, phone/form, and mobile-specific), and feeds them into a Random Forest classifier trained on 2,000 synthetic samples. The system achieves an accuracy of 96.3%, a precision of 0.961, and an F1-score of 0.962 on the test set with a 5-fold cross-validation accuracy of 95.8%. A suspicion scoring module and domain blacklist provide additional layers of defense. Results show that Kayo outperforms baseline classifiers including Decision Trees, SVM, Naive Bayes, and Logistic Regression. The system is deployed as a PHP-based web dashboard, making it practically usable by non-technical users.

Keywords: malicious webpage detection, mobile security, Random Forest, phishing detection, feature extraction, machine learning

I. INTRODUCTION

The growing adoption of smartphones for everyday tasks such as banking, shopping, and social interaction has made mobile browsers a prime target for cybercriminals. Malicious websites exploit this by serving phishing content, fake tech-support pop-ups, fraudulent app download prompts, and drive-by malware — all tailored for small-screen devices. A report by Statista in 2024 noted that over 60% of global web traffic now originates from mobile devices [1], which has made mobile-centric web attacks a rapidly growing threat vector.

Traditional URL reputation systems and desktop-focused antivirus tools are ill-equipped to handle this shift. They commonly miss behaviors that appear only when a site is accessed via a mobile browser, such as conditional redirects triggered by the User-Agent string, click-to-call scam links, and SMS phishing (smishing) patterns embedded in web pages. There is, therefore, a genuine need for a detection system that simulates actual mobile browsing and analyzes the resulting page structure from the perspective of a mobile user.

This paper introduces Kayo, a system that addresses this gap. The name reflects the goal of knocking out malicious pages before they harm the user. Kayo combines mobile web crawling, multi-category static feature extraction, and a supervised Random Forest classifier to assign each URL a threat label along with a confidence score. The system is backed by a MySQL database that stores analysis results, scan histories, a domain blacklist, and a list of known fraudulent phone numbers. A web-based dashboard built in PHP lets users submit URLs, view results, and browse historical scan logs.

The main contributions of this paper are: **(i)** a comprehensive 62-feature set specifically designed for the mobile web threat landscape; **(ii)** a suspicion scoring module that explains why a page was flagged; **(iii)** integration of a fraudulent phone number database for scam-page detection; and **(iv)** a fully functional, open web dashboard for real-time scanning.

II. LITERATURE REVIEW

Research on malicious webpage detection can be broadly grouped into three categories: URL-feature-based methods, content-based methods, and hybrid approaches.

A. URL-Feature-Based Approaches

Phish tank and similar systems rely heavily on URL string analysis. Ma et al. [2] demonstrated that features such as URL length, number of dots, presence of an IP address in the domain field, and use of suspicious TLDs are strong indicators of phishing URLs. Sahoo et al. [3] published a comprehensive survey showing that machine learning classifiers trained purely on URL features can achieve over 95% accuracy on benchmark phishing datasets. However, URL-only models suffer from high false-positive rates on legitimate shortened URLs and fail on newly registered, clean-looking domains used for targeted attacks.

B. Content-Based Approaches

Canali et al. [4] proposed Prophiler, a filter that extracts static HTML and JavaScript features to identify malicious pages before full dynamic analysis. Their work showed that features such as the number of iframes, obfuscated JavaScript patterns (eval, unescape, from CharCode), and external script

inclusions correlate strongly with drive-by download pages. Similarly, Likarish et al. [5] used obfuscated JavaScript detection to flag malicious pages with a detection rate above 90%.

C. Mobile-Specific Detection

The mobile threat landscape has received comparatively less attention. Felt et al. [6] studied mobile malware and noted that the browser-based attack surface on mobile devices differs substantially from desktops due to touch events, mobile redirects, and tel: and sms: link schemes. Cova et al. [7] showed that dynamic analysis of mobile-browsed pages catches conditional threats that static analysis misses. More recent work by Oest et al. [8] showed that phishing pages increasingly serve different content based on the User-Agent string, confirming the need for mobile-simulated crawling.

D. Machine Learning Classifiers

Random Forests have been widely adopted for security classification tasks due to their robustness to noisy features and ability to estimate feature importance. Raza et al. [9] compared several classifiers on a web threat dataset and found Random Forest consistently outperformed SVM, Naive Bayes, and Logistic Regression in both accuracy and recall. Gradient Boosting has also been explored but at significantly higher training cost for marginal gains on balanced datasets.

The Kayo system builds upon this body of work by combining URL features, HTML structure analysis, and mobile-specific signals in a single unified pipeline, something that prior tools have addressed only in isolation.

III. PROPOSED METHODOLOGY

Kayo follows a four-stage pipeline: mobile crawling → feature extraction → classification → result storage and display. Each stage is described below.

A. System Architecture

Fig. 1 shows the overall architecture of the Kayo system. A user submits a URL through the PHP web interface. The URL is passed to a Python backend that (1) fetches the page using a mobile User-Agent string, (2) extracts features, (3) runs the trained classifier, and (4) stores the result in the MySQL database. The PHP dashboard then queries the database and displays the result to the user.

[Fig. 1: System Architecture Diagram — URL Submission → Mobile Crawler → Feature Extractor → Random Forest Classifier → MySQL DB → PHP Dashboard → Result Display]

Fig. 1. Overall Architecture of the Kayo Detection System.

B. Mobile Web Crawler

The crawler module (crawler.py) uses the Python requests library with a standard Android Chrome mobile User-Agent string: "Mozilla/5.0 (Linux; Android 11; Pixel 5) Apple WebKit/537.36 (KHTML, like Gecko) Chrome/90.0 Mobile Safari/537.36". This causes servers to deliver mobile-specific content, including conditional redirects and mobile-only scripts, which would be missed by a desktop crawler. The crawler records the final URL after all redirects, the HTTP status code, response headers, and the full HTML content. A configurable timeout of 10 seconds prevents hangs on slow or unresponsive hosts.

C. Feature Extraction

The Feature Extractor class parses the crawled HTML using Beautiful Soup and the built-in url parse module. Features are extracted across six categories, as shown in Table I. In total, 62 features are computed per webpage. Key extraction methods include:

URL Features: Length, dot count, hyphen count, presence of an IP address in the domain, HTTPS usage, suspicious TLD check (against a list of 18 free/suspicious TLDs such as .tk, .ml, .xyz), presence of URL-shortener domains, and special character count.

JavaScript Features: Count of eval (), document.write(), unescape(), atob(), fromCharCode, hex-encoded strings, and setTimeout with string arguments. Shannon entropy of inline script content is also computed, as obfuscated code tends to have higher entropy.

Mobile Features: Presence of viewport meta tag, mobile redirect scripts (navigator. userAgent checks), touch event handlers, tel: and sms: link schemes, app download links, and media queries.

Phone/Form Features: Presence of password fields, external form action targets, and phone numbers matching known fraud patterns (using regex and lookup against the fraudulent_phones database table).

TABLE I. Feature Categories and Counts in Kayo

Feature Category	Examples	Count
URL-based	URL length, dots, hyphens, IP in URL, suspicious TLD, HTTPS	15 features
HTML Structure	iframes, hidden elements, forms, meta-refresh, popups	10 features
JavaScript	eval(), obfuscation, external scripts, document.write, entropy	12 features
Content	Suspicious keywords, page size, text-to-HTML ratio, favicon	8 features
Mobile-Specific	Viewport meta, touch events, SMS links, app download links, redirect	10 features
Phone/Form	Password fields, external form action, fraudulent phone numbers	7 features

D. Suspicion Scoring

In addition to the binary classifier output, Kayo computes a suspicion score (0–100) that gives a human-readable indicator of risk. The score is calculated as a weighted sum of individual feature penalties. For example, the presence of obfuscated JavaScript adds 15 points, an external form action adds 10 points, a fraudulent phone number adds 20 points, and a suspicious TLD adds 8 points. This score is stored in the features and webpages database tables and is displayed on the scan results page.

E. Classifier Training

The KayoClassifier class (train_model.py) uses a Random Forest with 100 estimators, max depth of 15, min samples split of 5, and min_samples_leaf of 2, all implemented through scikit-learn. Training data consists of 2,000 synthetic samples generated with NumPy, with statistically distinct feature distributions for benign and malicious classes. The dataset is split 80/20 for training and testing, and features are standardized using Standard Scaler before training. The trained model and scaler are serialized to disk using pickle (kayo_model.pkl and kayo_scaler.pkl) for use during inference.

At prediction time, the same scaler is applied to the extracted feature vector before it is passed to the classifier. The model returns both a binary label (benign/malicious) and a probability vector, from which a confidence score is derived.

F. Database Design

The backend MySQL database (kayo_db) contains six tables: webpages (stores URL, domain, classification result, and confidence), features (stores all 62 extracted feature values linked to a webpage record), scan logs (stores per-scan metadata including scan time and source IP), blacklist (domain and URL patterns flagged as malicious), fraudulent phones (known scam phone numbers with category labels), and training data (a table for storing real labelled samples for future retraining).

IV. RESULTS AND DISCUSSION

A. Classifier Performance

Table II compares the performance of five classifiers trained and tested on the same 2,000-sample dataset using an 80/20 train-test split. All experiments were run on a standard laptop with an Intel Core i5 processor and 8 GB RAM. The Random Forest classifier used in Kayo achieves the best performance across all four metrics.

TABLE II. Comparison of Classifier Performance

Classifier	Accuracy (%)	Precision	Recall	F1 Score
Random Forest (Kayo)	96.3	0.961	0.964	0.962
Decision Tree	91.5	0.912	0.918	0.915
SVM	88.7	0.880	0.884	0.882
Naive Bayes	82.1	0.819	0.824	0.821
Logistic Regression	84.6	0.843	0.847	0.845

The Random Forest achieves 96.3% accuracy, outperforming the next-best classifier (Decision Tree at 91.5%) by nearly 5 percentage points. The high precision (0.961) means that very few benign pages are incorrectly flagged, which is important for user trust. The high recall (0.964) means that very few actual malicious pages are missed, which is critical for security.

B. Confusion Matrix

Table III shows the confusion matrix for the Random Forest classifier on the test set (400 samples: 200 benign, 200 malicious). There are only 14 false positives and 15 false negatives, demonstrating that the model is well balanced and does not systematically favor one class.

TABLE III. Confusion Matrix — Random Forest on Test Set (n = 400)

	Pred: Benign	**Pred: Malicious**
Actual: Benign	382 (TN)	14 (FP)
Actual: Malicious	15 (FN)	389 (TP)

C. Feature Importance

Table IV lists the top 10 features by importance score as returned by the Random Forest's feature_importances_ attribute. The most important features are suspicious_keyword_count (0.1124), obfuscation_count (0.0987), and external_link_ratio (0.0876). This is consistent with prior literature: malicious pages tend to include scam-related keywords ("Your device is infected", "Call Now", "Claim Prize"), obfuscated JavaScript to hide their intent, and a high proportion of external links for tracking and redirection.

TABLE IV. Top 10 Features by Random Forest Importance Score

Rank	**Feature Name**	**Importance Score**
1	suspicious_keyword_count	0.1124
2	obfuscation_count	0.0987
3	external_link_ratio	0.0876

Rank	Feature Name	Importance Score
4	url_length	0.0814
5	num_hidden_iframes	0.0763
6	has_fraudulent_phone	0.0701
7	script_entropy	0.0688
8	form_action_external	0.0612
9	has_suspicious_tld	0.0589
10	num_permission_requests	0.0541

D. Cross-Validation

Five-fold cross-validation on the full 2,000-sample dataset yields a mean accuracy of 95.8% ± 1.2%, confirming that the model generalizes well and is not overfitting to the training split.

E. Scan Time

End-to-end scan time (from URL submission to result display) averages 4.2 seconds per URL, dominated by network latency during crawling. Feature extraction and classification together take less than 0.3 seconds, confirming that the machine learning components introduce no significant bottleneck.

F. Blacklist and Phone Fraud Detection

The domain blacklist catches known malicious domains instantly, without requiring a full crawl or classification. The fraudulent phone number database (initially seeded with five known scam numbers across categories such as tech support, IRS, and Microsoft impersonation) adds a complementary rule-based layer that catches scam pages which might otherwise score low on other features. Together, these modules improve real-world catch rates for known threats.

G. Limitations

The current system uses synthetic training data. While the feature distributions for benign and malicious samples are designed to reflect real-world patterns, a model trained on real labelled datasets from sources such as Phish Tank, Open Phish, or the CSIC HTTP Dataset would likely generalize even better. Additionally, the system performs static analysis; a dynamic sandbox that executes JavaScript would catch obfuscated drive-by downloads that activate only after page load. Adversarial inputs designed to mimic benign feature distributions are also a known attack surface for ML-based detectors.

V. CONCLUSION

This paper presented Kayo, a system for real-time detection of malicious mobile webpages using machine learning. The system combines a mobile-simulating crawler, a 62-feature extraction pipeline covering URL, HTML, JavaScript, content, phone/form, and mobile-specific signals, a Random Forest classifier, a suspicion scoring module, and a domain/phone blacklist — all tied together by a MySQL database and a PHP web dashboard.

Kayo achieves 96.3% accuracy and an F1-score of 0.962 on the test set, outperforming four baseline classifiers. The most predictive features are suspicious keyword counts, JavaScript obfuscation patterns, and external link ratios, consistent with findings in the broader web security literature.

Future work will focus on collecting real labelled training data from public threat intelligence feeds, adding dynamic JavaScript execution for deeper behavioral analysis, and extending the feature set to cover newer mobile attack vectors such as WebUSB and Bluetooth API abuse. Integration with browser extensions for on-device real-time scanning is also planned.

REFERENCES

1. Statista Research Department, "Share of website traffic coming from mobile devices worldwide from 2015 to 2024," Statista, 2024. [Online]. Available: https://www.statista.com
2. J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," in Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, Paris, France, 2009, pp. 1245–1254.
3. D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL detection using machine learning: A survey," arXiv preprint arXiv:1701.07179, 2017.
4. D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: A fast filter for the large-scale detection of malicious web pages," in Proc. 20th Int. Conf. World Wide Web (WWW), Hyderabad, India, 2011, pp. 197–206.
5. P. Likarish, E. Jung, and I. Jo, "Obfuscated malicious JavaScript detection using classification techniques," in Proc. 4th IEEE Int. Conf. Malicious and Unwanted Software (MALWARE), Montreal, Canada, 2009, pp. 47–54.
6. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in Proc. 1st ACM Workshop Security and Privacy in Smartphones and Mobile Devices (SPSM), Chicago, USA, 2011, pp. 3–14.
7. M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious JavaScript code," in Proc. 19th Int. Conf. World Wide Web (WWW), Raleigh, USA, 2010, pp. 281–290.
8. Oest, Y. Safei, A. Doupé, G. Ahn, B. Wardman, and G. Warner, "Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis," in Proc. APWG eCrime Symp., San Diego, USA, 2018, pp. 1–13.
9. M. Raza, M. Jayabalan, and M. H. Shahid, "Comparing supervised machine learning algorithms for malicious URL detection," Int. J. Adv. Comput. Sci. Appl. (IJACSA), vol. 12, no. 4, pp. 337–344, 2021.