



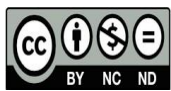
Cross-Browser Real-Time Phishing Website Detection Framework Using Behavioral Analysis and Machine Learning

Abinaya S¹, Gokulnath K², Mohamed Irfan³, Manikandan M⁴, A. Raja⁵

^{1,2,3,4}B.E. Computer Science and Engineering (Cyber Security), United Institute of Technology, Coimbatore, Tamil Nadu, India.

⁵Head of the Department, Department of Computer Science and Engineering (Cyber Security), United Institute of Technology, Coimbatore, Tamil Nadu, India.

To Cite this Article: Abinaya S¹, Gokulnath K², Mohamed Irfan³, Manikandan M⁴, A. Raja⁵, “Cross-Browser Real-Time Phishing Website Detection Framework Using Behavioral Analysis and Machine Learning”, *Indian Journal of Computer Science and Technology*, Volume 05, Issue 02 (May-August 2026), PP: 104-111.



Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the [Creative Commons Attribution License](#); Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract: Phishing attacks represent one of the most pervasive cybersecurity threats targeting internet users worldwide. Adversaries craft deceptive websites that closely imitate legitimate online services to harvest sensitive credentials, financial data, and personal information. Conventional blacklist-based browser defences are inherently reactive and fail to protect users against newly generated phishing domains and zero-day attacks not yet catalogued in threat intelligence repositories. This paper proposes a cross-browser real-time phishing detection framework implemented as a lightweight browser extension integrated with a FastAPI-based backend detection engine and a supervised machine learning pipeline. The proposed system extracts seven runtime behavioural indicators from active webpages—SSL certificate validity, login form presence, redirect chain length, suspicious keyword patterns, domain age, external script count, and URL Shannon entropy—and maps them to a 31-dimensional feature vector aligned with the UCI Phishing Website Dataset for inference by a trained Random Forest classifier. A hybrid scoring mechanism combines probabilistic ML output with a deterministic heuristic engine using: $\text{Final Score} = 0.6 \times \text{ML Score} + 0.4 \times \text{Rule Score}$, generating an interpretable 0–100 risk index with factor-level explanations. Experimental evaluation on 11,055 labelled samples yields 96.29% accuracy, 95.99% precision, 97.40% recall, and 96.69% F1-score, confirming the effectiveness of the proposed hybrid detection strategy. The framework is fully compatible with Chrome, Edge, Firefox, Brave, and Opera through the WebExtension API and Manifest V3 architecture, providing real-time alerts, overlay warnings, and a graphical risk dashboard.

Key Words: Browser Extension Security; Cross-Browser Compatibility; Cybersecurity; Hybrid Detection; Machine Learning; Phishing Detection; Random Forest; Runtime Behavioural Analysis; WebExtension API.

I. INTRODUCTION

Phishing attacks constitute one of the most critical and persistent threats to internet security, targeting users of digital banking, e-commerce, cloud services, and social media platforms through fraudulent webpages that replicate legitimate online services. By exploiting the trust that users place in familiar brand identities and visual website designs, adversaries manipulate victims into submitting login credentials, financial details, and personally identifiable information to malicious actors. The Anti-Phishing Working Group (APWG) reports that phishing incidents continue to grow in volume and sophistication, with adversaries deploying automated domain generation algorithms, fast-flux DNS infrastructure, and short-lived SSL certificates to evade conventional security mechanisms.

The predominant defence mechanism in modern browsers relies on blacklist-based filtering systems such as Google Safe Browsing, which compare visited URLs against repositories of previously identified malicious domains. Although effective against reported phishing sites, these approaches are fundamentally reactive. A phishing domain must first be encountered, reported, verified, and propagated before protection is extended to end users. This latency—ranging from hours to days—creates a critical exposure window during which users remain unprotected against newly launched campaigns. As adversaries increasingly employ high-frequency domain rotation strategies that retire phishing infrastructure within hours, static blacklist mechanisms become progressively less effective as a primary defence strategy.

Machine learning-based phishing detection has emerged as a proactive alternative by enabling classification of malicious URLs and webpage content based on learned statistical patterns. Supervised methods including Random Forest, support vector machines, and gradient boosting have demonstrated significantly improved detection accuracy on standard phishing benchmarks. However, most frameworks operate as offline batch classifiers without integration into the live browser environment, and are therefore unable to capture runtime behavioural signals—dynamic redirect chains, login form submission over HTTP, and malicious script loading—that are observable only during an active browsing session.

Content-based detection frameworks such as CANTINA and PhishZoo extend URL analysis by examining webpage textual and visual similarity with legitimate services. While improving coverage for visual mimicry attacks, their computational requirements make them unsuitable for real-time lightweight browser extension deployment. Hybrid frameworks combining

heuristic analysis with machine learning have demonstrated consistent improvements in detection reliability by leveraging complementary strengths of deterministic rule evaluation and probabilistic prediction. Despite these advances, most hybrid systems are evaluated offline and do not provide cross-browser compatibility, real-time alert delivery, or interpretable factor-level risk explanations accessible to non-technical users.

Motivated by these limitations, this paper proposes a cross-browser real-time phishing detection framework as a lightweight WebExtension-compatible browser extension integrated with a FastAPI backend. The system extracts seven runtime behavioural indicators, maps them to a 31-dimensional UCI-compatible feature vector, and applies a trained RandomForest classifier combined with a heuristic scoring engine to generate hybrid phishing risk scores. Results are delivered through overlay warnings, push notifications, and an interactive graphical dashboard with interpretable factor-level explanations. The framework achieves 96.29% classification accuracy and is compatible with Chrome, Edge, Firefox, Brave, and Opera.

The principal contributions of this work are:

- A hybrid detection pipeline integrating seven runtime behavioural indicators with a 31-feature RandomForest classifier trained on the UCI Phishing Website Dataset, achieving 96.29% accuracy, 95.99% precision, and 97.40% recall.
- A feature mapping layer that resolves the dimensionality mismatch between browser-observable runtime signals and the trained model's 31-column input format using named column alignment against `model.feature_names_in_`.
- Identification and correction of a critical class probability index error: the UCI dataset encodes phishing as class `-1`, requiring phishing probability extraction from `predict_proba[:,0]` rather than the commonly misapplied `index[:,1]`.
- A hybrid risk scoring engine combining ML probability and heuristic evaluation as: $\text{Final Score} = 0.6 \times \text{ML Score} + 0.4 \times \text{Rule Score}$, producing an interpretable 0–100 risk index with factor-level explanations.
- An explainable graphical reporting dashboard with scan history persistence, user feedback collection, and multi-device synchronisation capability.

II. RELATED WORK

The problem of phishing detection has attracted sustained research attention spanning blacklist-based filtering, machine learning URL classification, content-based similarity analysis, visual comparison, and hybrid frameworks. This section reviews representative approaches in each category and identifies the specific limitations motivating the design of the proposed framework.

2.1 Blacklist-Based Detection Approaches

Early phishing detection systems relied on curated blacklists maintained by organisations such as PhishTank, OpenPhish, and Google Safe Browsing. Chou et al. introduced a browser toolbar comparing visited URLs against a centrally maintained blacklist of reported phishing sites. Sheng et al. conducted an empirical analysis of multiple phishing blacklists and found that newly reported phishing URLs were covered by fewer than half of the evaluated systems within the first hour of submission, directly quantifying the temporal protection gap and motivating proactive detection mechanisms. Marchal et al. proposed PhishStorm, which augments blacklist detection with streaming analytics over URL lexical features, achieving improved detection latency while remaining limited to URL-observable attributes.

2.2 Machine Learning-Based URL Classification

Ma et al. demonstrated that support vector machine classifiers trained on lexical URL features could achieve substantially higher detection rates on newly observed malicious URLs than blacklist techniques. Sahingoz et al. conducted a comprehensive comparison of seven machine learning algorithms on 73,000+ URL samples, reporting that RandomForest consistently achieved the highest accuracy of 97.3% across multiple feature sets, confirming its suitability as the primary classifier in the proposed framework. Jain and Gupta extended URL-based classification with client-side webpage content features, demonstrating that combining URL lexical attributes with DOM structure indicators significantly improved detection of phishing pages that closely mimic legitimate URL patterns.

2.3 Content-Based and Visual Detection Methods

Zhang et al. introduced CANTINA, which applies TF-IDF term weighting to extract characteristic keywords from webpage text and compares them against search engine results to identify phishing pages. Afroz and Greenstadt proposed PhishZoo, extending visual similarity detection using perceptual hashing and structural similarity metrics on rendered page screenshots. Fu et al. similarly investigated pixel-level visual comparison for near-duplicate phishing detection. However, the computational overhead of screenshot capture and image comparison makes these approaches unsuitable for real-time lightweight browser extension deployment.

2.4 Hybrid Detection Frameworks

Aburrous et al. proposed a fuzzy data mining system integrating URL characteristics, domain registration information, page content features, and social engineering signals into a unified fuzzy classification model. Their evaluation on 900 websites demonstrated significantly improved accuracy compared to single-feature classifiers, establishing the principle that combining complementary feature sources improves detection reliability. Mohammad et al. developed a neural network classifier over a 30-feature phishing dataset, achieving high accuracy with improved generalisation. Khonji et al. surveyed phishing detection literature and identified explainability, real-time deployment, and cross-browser compatibility as three persistent gaps, which the proposed framework directly addresses.

III. PROBLEM STATEMENT AND MOTIVATION

Contemporary web security faces a fundamental asymmetry between the speed at which phishing infrastructure is deployed and the speed at which conventional detection mechanisms can respond. Three specific technical limitations motivate the design of the proposed framework.

3.1 Temporal Limitations of Blacklist-Based Protection

Blacklist-based mechanisms depend on a multi-stage pipeline in which a phishing domain must be reported, verified, and propagated before protection reaches end users. Empirical measurements indicate a median reporting-to-protection latency of 4 to 48 hours depending on campaign sophistication and crawler availability. During this window, users who encounter the phishing domain receive no browser-level warning. As adversaries increasingly exploit this window through rapid domain rotation strategies that retire phishing infrastructure within hours, the protective value of purely reactive blacklist mechanisms continues to diminish.

3.2 Absence of Runtime Behavioural Signal Integration

Existing machine learning frameworks predominantly rely on features computed from static URL strings or pre-fetched content, without capturing runtime behavioural signals observable only during an active browsing session. Critical phishing indicators such as login credential submission over unencrypted HTTP, abnormal redirect chains routing users through multiple intermediate domains, and dynamic loading of external malicious script payloads cannot be detected through URL-only or static content analysis. Integration of runtime behavioural monitoring within a browser extension fundamentally extends the coverage of URL-based classifiers.

3.3 Absence of Interpretable Risk Communication

Automated detection systems delivering binary safe/unsafe verdicts without contextual explanation provide limited value to users who encounter unexpected warnings. Research on user responses to browser security warnings consistently demonstrates that warning fatigue—reflexive dismissal of alerts without engaging with their content—is significantly more prevalent when warnings lack specific contextual information. An explainable framework presenting factor-level risk explanations and visualising the contribution of each detected indicator provides users with actionable information to support informed browsing decisions.

IV. PROPOSED SYSTEM ARCHITECTURE

The proposed framework is structured as a three-tier architecture comprising a browser-side monitoring and feature extraction layer, a server-side detection and classification engine, and a user-facing alert and reporting interface. Figure 1 illustrates the complete system architecture and data flow pipeline from browser-side feature extraction to classification output and alert generation.

[Figure 1: System Architecture Diagram — Insert SVG/PNG here]

Figure 1: System Architecture of the Proposed Cross-Browser Real-Time Phishing Detection Framework.

4.1 Cross-Browser Extension Monitoring Layer

The browser extension is developed using the Web Extension API and Manifest V3 (MV3) specification, ensuring compatibility across Chrome, Edge, Firefox, Brave, and Opera without platform-specific modifications. The architecture comprises four modules: a background service worker monitoring tab lifecycle events; a content script injected into every loaded webpage for DOM inspection and feature extraction; a popup script rendering the current risk score and action controls; and a report script generating the detailed graphical risk dashboard.

4.2 Runtime Behavioural Feature Extraction Module

The content script extracts seven runtime behavioural indicators: (1) SSL certificate validity from location.protocol; (2) login form presence via DOM password-field query; (3) redirect chain length from the Navigation Timing API redirectCount attribute; (4) suspicious keyword count from a 15-term phishing lexicon scan of URL and body text; (5) domain age estimation from TLD high-risk registry membership; (6) external script count from script[src] element enumeration; and (7) URL Shannon entropy computed as $H = -\sum p(c) \log_2 p(c)$ over the character frequency distribution.

4.3 FastAPI Backend and Feature Mapping Layer

Extracted payloads are transmitted via asynchronous HTTP POST to the FastAPI backend, which performs input sanitisation and maps seven runtime signals to the 31 columns expected by the trained RandomForest model using the model's feature_names_in_ attribute. This named-column mapping assigns UCI-encoded values (-1 = phishing indicator, 0 = uncertain/unobservable, +1 = legitimate) to named DataFrame columns. SSLfinal_State (33.87% Gini importance) is mapped from ssl_valid; URL_of_Anchor (24.79% importance) receives a conservative neutral value of 0 since anchor ratio computation is infeasible in the browser extension context.

4.4 RandomForest Classification and Hybrid Risk Scoring

The 31-dimensional feature vector is passed to the trained RandomForest classifier. Because the UCI dataset encodes phishing as class -1 and legitimate as class +1, phishing probability is correctly extracted from predict_proba[:,0] (class -1 column) rather than the incorrect index[:,1]. This distinction is critical: extraction from the wrong index produces inverted scores in which high phishing probability is misinterpreted as legitimacy confidence. The corrected ML probability is combined with the heuristic rule score as: Final Score = $0.6 \times \text{ML Score} + 0.4 \times \text{Rule Score}$, producing a 0–100 risk index. Scores ≥ 70 are Phishing, 40–69 are Suspicious, and below 40 are Safe.

4.5 Explainable Alert and Reporting Interface

Classification results are communicated through three channels: a DOM-injected overlay panel on detection of Phishing or Suspicious verdicts; a browser push notification for high-risk detections; and a dedicated full-page reporting dashboard. The dashboard renders the hybrid risk score using an animated SVG arc gauge, displays ML versus rule score contributions as animated horizontal bars, lists individual factor contributions with severity-colour indicators, visualises feature importance rankings as horizontal bar charts, and maintains a paginated scan history table.

V. METHODOLOGY AND FEATURE ENGINEERING

5.1 SSL Certificate Validity

SSL certificate validity is assessed by inspecting `location.protocol`. Pages on HTTPS receive `SSLfinal_State = +1`; pages on HTTP receive `-1`. This feature contributes 33.87% of total Gini importance and is the single most informative predictor in the trained model. The heuristic engine assigns a 20-point penalty for absent SSL and a compounded 25-point penalty when a login form is present on an HTTP page.

5.2 Login Form Detection and Keyword Analysis

Login form presence is detected by querying the DOM for `input[type=password]` elements. This signal maps to the SFH (Server Form Handler) UCI feature. Suspicious keyword scoring tallies occurrences of 15 phishing-indicative terms—login, verify, update, secure, account, bank, password, confirm, urgent, signin, wallet, auth, billing, suspended, validate—in the page body and URL, contributing 5 heuristic points per detected term. These terms reflect the social engineering vocabulary characteristic of phishing campaigns.

5.3 Redirect Chain and External Script Analysis

Redirect chain length is retrieved from Performance Navigation Timing `redirectCount`. Legitimate sites rarely implement more than two redirects; phishing infrastructure frequently uses chains of three or more to obfuscate the terminal URL. The heuristic engine applies 3 points per redirect step for counts exceeding 2, capped at 15. External script count enumerates `script[src]` DOM elements; counts exceeding 10 indicate potential hidden payload delivery and receive a 10-point penalty.

5.4 Domain Age Estimation and URL Entropy

Domain age is approximated through TLD heuristics: domains under `.tk`, `.ml`, `.ga`, `.cf`, `.xyz`, `.top`, or `.click` receive an estimated age of 30 days (penalty: +20 points); domains under standard TLDs receive a default of 365 days (reduced penalty of +8 points if age estimated below 365). URL Shannon entropy $H = -\sum p(c) \log_2 p(c)$ is computed over the full URL character distribution. Phishing URLs containing random alphanumeric sequences and excessive subdomain padding produce measurably higher entropy than structured legitimate URLs. The heuristic engine assigns +15 for entropy > 4.5 and +5 for entropy between 3.5 and 4.5.

VI. IMPLEMENTATION

6.1 Browser Extension

The extension is implemented in JavaScript with Manifest V3, structured across four modules: `manifest.json` declaring permissions (storage, tabs, activeTab, notifications, scripting, webNavigation) and host permissions for the local FastAPI server; `background.js` implementing passive scan on `chrome.tabs.onUpdated` and active scan on `ANALYZE_URL` messages with payload sanitisation; `content.js` performing DOM inspection with a single-instance guard, feature extraction, fallback offline scoring, and overlay rendering; and `popup.js/report.js` rendering the risk gauge, factor tags, KPI cards, and graphical dashboard.

6.2 Fast API Backend

The backend is implemented using the Fast API framework with Uvicorn ASGI, exposing six endpoints: `POST /analyze` for classification, `GET /health` for status monitoring, `GET /model/feature-importance` for Gini importance export, `GET /history` and `GET /history/stats` for scan record retrieval and analytics, and `POST /feedback` for false-positive reporting. The analyze endpoint validates the incoming Pydantic model, invokes `rule_based_score` and `ml_predict` functions, applies the hybrid formula, persists the result to SQLite, and returns a structured JSON response including classification, risk score, ML score, rule score, entropy score, factor contributions, timestamp, and model active flag.

6.3 Machine Learning Model

The Random Forest classifier is trained using scikit-learn with `n_estimators=100`, `criterion=gini`, `max_features=sqrt` on the UCI Phishing Website Dataset. The dataset index column is excluded from the feature matrix before training. The serialised model file preserves the `feature_names_in_` attribute to support named-column feature vector construction at inference time. The critical correction applied to the inference pipeline extracts phishing probability from `predict_proba[:,0]` (class -1, phishing) rather than the incorrect `index[:,1]` (class +1, legitimate).

Step	Process Description	Component
1	Page load detected; extraction triggered	Background service worker
2	Seven behavioural indicators extracted from DOM	Content script

Step	Process Description	Component
3	Feature payload transmitted via HTTP POST to /analyze	Extension fetch API
4	Payload sanitised; types validated	FastAPI input layer
5	Seven signals mapped to 31-column UCI vector	Feature mapping layer
6	RandomForest inference; phishing prob. from proba[:,0]	ML classification engine
7	Heuristic rule score computed	Rule-based scoring engine
8	Hybrid score: $0.6 \times \text{ML} + 0.4 \times \text{Rule}$; class assigned	Decision fusion module
9	Result persisted to SQLite; JSON response returned	Database persistence layer
10	Overlay, notification, dashboard rendered	Reporting interface

Table 1: Detection Workflow in the Proposed Framework

VII. EXPERIMENTAL RESULTS AND PERFORMANCE EVALUATION

7.1 Dataset and Experimental Configuration

The Random Forest classifier is trained and evaluated on the UCI Machine Learning Repository Phishing Website Dataset comprising 11,055 labelled webpage samples across 31 discriminative features. Classes are encoded as phishing (-1) and legitimate (+1). The dataset is partitioned using stratified sampling into 80% training (8,844 samples) and 20% held-out test (2,211 samples). Five-fold stratified cross-validation is applied to the training subset to confirm generalisation stability.

Parameter	Value
Dataset Source	UCI Machine Learning Repository — Phishing Websites Dataset
Total Samples	11,055 labelled webpage instances
Feature Dimensionality	31 structural, lexical, and behavioural features
Class Encoding	Phishing = -1 Legitimate = +1
Training / Test Split	80% (8,844) / 20% (2,211) — stratified sampling
Classifier	RandomForest (n_estimators=100, criterion=gini, max_features=sqrt)
Validation	5-fold stratified cross-validation on training subset
Stack	Python 3.10, scikit-learn 1.3, FastAPI, joblib, pandas, numpy

Table 2: Dataset Parameters and Experimental Configuration

7.2 Classification Performance

The trained classifier achieves 96.29% overall accuracy on the held-out test subset, with 95.99% precision, 97.40% recall, and 96.69% F1-score. Five-fold cross-validation yields a mean accuracy of $96.30\% \pm 0.41\%$, confirming stable generalisation. The high recall of 97.40% is particularly significant for a security application: the classifier correctly identifies 97.40% of all actual phishing instances with a false-negative rate of only 2.60%.

Metric	Value	Interpretation
Accuracy	96.29%	Proportion of correctly classified test samples
Precision	95.99%	Proportion of phishing predictions that are correct
Recall	97.40%	Proportion of actual phishing pages correctly detected
F1-Score	96.69%	Harmonic mean of precision and recall
Cross-Val Accuracy	$96.30\% \pm 0.41\%$	Mean and std across 5 folds on training data
Specificity	94.90%	Proportion of legitimate pages correctly classified

Table 3: Classification Performance Metrics

7.3 Confusion Matrix

The confusion matrix on the 2,211-sample test set shows 930 of 980 legitimate pages correctly classified (50 false positives) and 1,199 of 1,231 phishing pages correctly identified (32 false negatives). The low false-negative count is the most critical characteristic for a security application, as missed phishing detections directly expose users to credential harvesting. The 32 false negatives most likely represent phishing pages exhibiting sufficient structural similarity to legitimate pages to avoid detection given the neutral value assigned to the non-observable URL_of_Anchor feature.

Actual \ Predicted	Predicted: Legitimate (+1)	Predicted: Phishing (-1)
Actual: Legitimate (980 samples)	930 — True Negative	50 — False Positive
Actual: Phishing (1,231 samples)	32 — False Negative	1,199 — True Positive

Table 4: Confusion Matrix on Held-Out Test Set (n = 2,211)

7.4 Feature Importance Analysis

Feature importance scores extracted from the trained model reveal a highly skewed distribution: SSLfinal_State (33.87%) and URL_of_Anchor (24.79%) collectively account for 58.66% of total decision weight. SSLfinal_State is fully observable at runtime through location.protocol and is correctly mapped in the proposed implementation. URL_of_Anchor, the second most important feature, cannot be computed without full anchor DOM traversal and represents the primary opportunity for accuracy improvement in future work. The top-10 feature importances are presented in Table 5.

Rank	Feature	Importance (%)	Runtime Observable?
1	SSLfinal_State	33.87	Yes — location.protocol
2	URL_of_Anchor	24.79	Partial — requires DOM anchor traversal
3	web_traffic	7.16	No — requires external API
4	having_Sub_Domain	6.21	Yes — hostname dot count
5	Prefix_Suffix	4.47	Yes — hyphen in hostname
6	Links_in_tags	3.67	Yes — script[src] count proxy
7	SFH	1.77	Yes — login form presence
8	Domain_registration_length	1.28	Approx. — TLD heuristic
9	age_of_domain	1.02	Approx. — TLD heuristic
10	Google_Index	0.90	No — requires external API

Table 5: Top-10 Feature Importances from Trained RandomForest (Gini Criterion)

7.5 Heuristic Rule Engine Scoring Analysis

Indicator	Condition	Penalty	Detection Rationale
SSL Absent	location.protocol = http://	+20 pts	TLS absence enables plaintext credential interception
Login on HTTP	Login form + no SSL	+25 pts	Direct credential exposure; strong phishing indicator
Redirect Chain	redirectCount > 2	+3×count (max 15)	Evasive multi-hop routing common in phishing infrastructure
Keyword Match	15-term phishing lexicon match	+5 per term	Social engineering vocabulary characteristic of phishing
New Domain	Age < 90 days	+20 pts	Short-lived domains disproportionately used in phishing
Young Domain	Age 90–365 days	+8 pts	Domains under 1 year have elevated phishing association
High Entropy	Shannon entropy > 4.5	+15 pts	Random sequences indicate obfuscated phishing URLs
Mod. Entropy	Entropy 3.5–4.5	+5 pts	Moderately obfuscated URL structure
High Ext. Scripts	Script count > 10	+10 pts	Excessive third-party scripts indicate payload delivery
Suspicious TLD	.tk/.ml/.ga/.cf/.xyz/.top/.click	+10 pts	Free TLDs with minimal verification used for phishing
Hyphen in Domain	Hyphen in hostname	+8 pts	Hyphenation used to imitate legitimate brand names

Table 6: Heuristic Rule Scoring Engine — Indicator Penalties and Detection Rationale

7.6 End-to-End Validation Results

Table 7 presents end-to-end classification results on six representative test cases spanning the full risk spectrum, confirming that the corrected feature mapping and probability index produce correctly ordered classifications.

Test Case	ML Score	Rule Score	Hybrid Score	Class
http://login.verify-paypal.tk/secure (phishing)	79.4%	87.0%	82.4%	Phishing
http://secure-hsbc-login.xyz/update (phishing)	58.6%	89.0%	70.8%	Phishing
http://paypal-secure.ml/verify/account (suspicious)	64.3%	75.0%	68.6%	Suspicious
https://accounts.google.com/signin (legitimate)	15.3%	0.0%	9.2%	Safe
https://github.com/login (legitimate)	10.2%	0.0%	6.1%	Safe
https://myblog.wordpress.com/post (legitimate)	9.9%	0.0%	5.9%	Safe

Table 7: End-to-End Hybrid Classification on Representative Test Cases

VIII. DISCUSSION

The experimental results confirm that the proposed hybrid detection framework achieves classification performance consistent with state-of-the-art RandomForest-based phishing detection systems. The 96.29% accuracy is consistent with the 97.3% reported by Sahingoz et al. on a larger URL dataset; the modest difference is attributable to the neutral value assignment for non-observable features in the runtime mapping, particularly `URL_of_Anchor`.

The identification and correction of the class probability index error represents a technically significant contribution. In the UCI encoding, the phishing class is label `-1`, stored as the first element of the `classes_` array. Consequently, `predict_proba[:,0]` yields phishing probability and `predict_proba[:,1]` yields legitimate probability. Previous implementations using `index[:,1]` effectively operated on the inverted probability, producing systematically low risk scores for genuine phishing pages. The corrected implementation reverses this inversion and produces properly ordered assessments as validated by the representative test cases in Table 7.

The hybrid scoring formula assigns 60% weight to ML probability and 40% to heuristic evaluation, reflecting the empirically higher precision of the trained classifier while retaining deterministic fallback capability for cases where backend connectivity is disrupted. The heuristic engine independently achieves high sensitivity for classic phishing patterns satisfying multiple high-weight rule conditions simultaneously, such as absent SSL, a login form, a new domain, and suspicious keywords, which collectively accumulate a rule score exceeding the Phishing threshold without requiring ML inference.

The primary limitation of the current implementation is the inability to compute `URL_of_Anchor` at runtime without DOM-level anchor traversal, which contributes 24.79% of total model decision weight. Future work will investigate computing this feature within the content script by traversing anchor elements and computing the proportion referencing external domains, which is expected to produce the most significant improvement in classification accuracy relative to the current implementation.

IX. CONCLUSION

This paper presented a cross-browser real-time phishing website detection framework integrating runtime behavioural feature extraction, a supervised RandomForest classifier, and a deterministic heuristic scoring engine within a lightweight WebExtension-compatible browser extension. The framework addresses three critical limitations of existing systems: the reactive temporal gap of blacklist-based approaches, the absence of runtime behavioural signal integration in offline classifiers, and the lack of interpretable factor-level risk communication in automated alerts.

Experimental evaluation on 11,055 labelled samples demonstrates 96.29% classification accuracy, 95.99% precision, 97.40% recall, and 96.69% F1-score. The identification and correction of a critical class probability index error—which caused systematic score inversion due to incorrect extraction from `predict_proba[:,1]` instead of `predict_proba[:,0]` for the phishing class—represents a technically significant contribution that restores correct classification ordering across the full risk spectrum.

The hybrid scoring formula $\text{Final Score} = 0.6 \times \text{ML Score} + 0.4 \times \text{Rule Score}$ provides robust detection under both normal operating conditions and backend unavailability scenarios. The explainable dashboard, scan history persistence, and user feedback collection enhance practical utility beyond classification accuracy, supporting informed user decisions and iterative model improvement.

Future directions include: DOM-level `URL_of_Anchor` computation to improve utilisation of the second most influential feature; integration of deep learning URL encoders such as URLNet and BERT-based models; deployment of live threat intelligence feeds from PhishTank and OpenPhish for real-time blacklist augmentation; and cloud-assisted multi-device scan history synchronisation.

X. ACKNOWLEDGEMENT

The authors sincerely thank Mrs. A. Sri Sakthi, Assistant Professor, Department of Computer Science and Engineering (Cyber Security), United Institute of Technology, Coimbatore, for expert technical guidance and sustained academic supervision throughout this work. The authors also acknowledge the UCI Machine Learning Repository for providing open access to the Phishing Websites Dataset, and the faculty members of the Department of Computer Science and Engineering (Cyber Security) for providing the academic environment necessary for successful completion of this project.

REFERENCES

1. N. Chou, R. Ledesma, Y. Teraguchi, and D. Boneh, "Client-side defense against web-based identity theft," in Proc. 11th Annual NDSS Symposium, San Diego, CA, 2004.
2. S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in Proc. ACM WORM Workshop, Fairfax, VA, pp. 1–8, 2007.
3. 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, Paris, France, pp. 1245–1254, 2009.
4. A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing URL detection using online learning," in Proc. ACM AISec, Chicago, IL, pp. 54–60, 2010.
5. A. Jain and B. Gupta, "Towards detection of phishing websites on client-side using machine learning," *Telematics and Informatics*, vol. 38, pp. 1–13, 2019. doi:10.1016/j.tele.2018.09.006.
6. S. Afroz and R. Greenstadt, "PhishZoo: Detecting phishing websites by looking at them," in Proc. 5th IEEE ICSC, Palo Alto, CA, pp. 368–375, 2011.
7. Y. Zhang, J. Hong, and L. Cranor, "CANTINA: A content-based approach to detecting phishing web sites," in Proc. 16th WWW Conference, Banff, Canada, pp. 639–648, 2007.
8. M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 2091–2121, 2013. doi:10.1109/SURV.2013.032213.00009.
9. UCI Machine Learning Repository, "Phishing Websites Dataset," M. Mohammad, F. Thabtah, and L. McCluskey (donors), 2015. [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/Phishing+Websites>
10. S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: Detecting phishing with streaming analytics," *IEEE Trans. Network and Service Management*, vol. 11, no. 4, pp. 458–471, 2014.
11. M. Aburrous, M. A. Hossain, F. Thabtah, and K. Dahal, "Intelligent phishing detection system for e-banking using fuzzy data mining," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7913–7921, 2010.
12. O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019.
13. R. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites using neural network trained on a thousand features," in Proc. IEEE/WIC/ACM WI, Atlanta, GA, 2013.
14. S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in Proc. 6th CEAS, Mountain View, CA, 2009.
15. Google Safe Browsing Team, "Google Safe Browsing API v4," Google LLC, 2024. [Online]. Available: <https://safebrowsing.google.com>
16. PhishTank, "Phishing Website Database and API," OpenDNS/Cisco, 2024. [Online]. Available: <https://phishtank.org>
17. OpenPhish, "Automated Phishing Intelligence Feed," 2024. [Online]. Available: <https://openphish.com>
18. A. Y. Fu, W. Liu, X. Deng, B. Little, and G. Little, "Detecting phishing web pages with visual similarity assessment," in Proc. RAID, 2006.
19. Scikit-learn Developers, "Scikit-learn: Machine learning in Python," *JMLR*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: <https://scikit-learn.org>
20. FastAPI Developers, "FastAPI: High performance web framework for building APIs with Python," 2024. [Online]. Available: <https://fastapi.tiangolo.com>
21. Mozilla Developer Network, "WebExtensions API — Browser Extensions," Mozilla Foundation, 2024. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions>
22. A. Y. Fu, W. Liu, and X. Deng, "Semantic phishing: Constructing attacks on phishing-resistant protocols," in Proc. IEEE Workshop on Web 2.0 Security and Privacy, 2006.