

Real-Time Phishing Attack Detection in A Chrome Extension Using Hybrid Gwo-Pso Feature Selection and Optimized Xgboost

*Aolat O. Giwa-Raheem, Dahiru Haliru Ibrahim, Abubakar Faruku Saad, Aliyu Munir

Department of Computer Science, Al-Hikmah University, Ilorin, Nigeria.

*Corresponding authors' email: olanikegiwa@gmail.com

ABSTRACT

Phishing remains one of the most prevalent and damaging cyber-attack techniques, and traditional detection methods such as blacklisting and static heuristics increasingly fail to identify sophisticated, zero-day deceptive websites. This study proposes a real-time phishing detection system balancing accuracy with the low-latency requirements of browser-side deployment. Website data was acquired from the UCI Phishing Websites repository and the PhishTank repository, after which raw lexical, HTML/DOM structural, and visual features were engineered from each URL. Class imbalance was addressed using a SMOTE-Tomek hybrid sampling technique, while an Isolation Forest model and Z-score standardization cleaned and scaled the feature space. Feature selection and classifier hyperparameter tuning were jointly optimized using a Hybrid Grey Wolf Optimizer-Particle Swarm Optimization (GWO-PSO) algorithm, reducing the input space from 30 candidate features to 9 high-impact indicators used to train an eXtreme Gradient Boosting (XGBoost) classifier. On a held-out test set of 2,463 samples, the GWO-PSO-XGBoost framework achieved 95.0% accuracy, 94% precision, 95% recall, and an AUC-ROC of 0.9897. The trained model was implemented and deployed as a working Google Chrome browser extension, whose source code and live-site test cases are presented as direct evidence of implementation; the deployed extension achieved an average inference latency of 210ms and a false positive rate of 1.7%, outperforming Google Safe Browsing, the PhishTank extension, and the Netcraft toolbar under identical test conditions. These findings demonstrate that hybrid metaheuristic optimization combined with gradient-boosted classification provides an efficient, lightweight, and empirically verifiable solution for real-time phishing defense at the browser edge.

Received: 17 July 2026

Accepted: 11 Aug. 2026

Published: 14 Sept. 2026

Keywords:

Phishing Detection, Cybersecurity, Machine Learning, Chrome Extension, XGBoost, Grey Wolf Optimizer (GWO), Particle Swarm Optimization (PSO), Feature Selection, Real-Time Detection, Browser Security

INTRODUCTION

The shift of financial, corporate, and personal transactions to web-based platforms has expanded the attack surface available to cyber adversaries, making internet users prime targets for sophisticated threats (George, 2024). Among these threats, phishing remains one of the most prevalent and damaging techniques used to compromise sensitive data, accounting for approximately 30% of all reported cybercrimes and resulting in substantial global financial losses (Federal Bureau of Investigation, 2022; Kheruddin et al., 2024). The Anti-Phishing Working Group reported a doubling of phishing attacks in recent years, with an estimated 5 million distinct phishing websites identified globally in 2023 alone (Tanti, 2024). Beyond the immediate financial loss of a single compromised account, harvested credentials frequently enable lateral movement within organisational networks and business email compromise, compounding the damage long after the initial deceptive message has been forgotten.

Historically, cybersecurity infrastructure relied on rule-based methods and static blacklists to mitigate phishing threats. However, as phishing websites have become increasingly transient and sophisticated, with attackers constantly altering URL structures, graphic designs, and website behaviours to evade detection, traditional blacklists are no longer sufficient. Contemporary security systems have consequently shifted toward Artificial Intelligence and Machine Learning

approaches that analyse lexical, structural, and visual similarities to distinguish authentic sites from fraudulent ones. Despite these advances, a critical need remains to deploy such defensive mechanisms directly at the user interaction layer, since maintaining high-dimensional feature spaces improves detection accuracy but induces severe computational overhead that renders many models impractical for real-time execution inside a web browser.

This study addresses this gap by proposing a real-time phishing attack detection system for a Chrome browser extension, built around a Hybrid Grey Wolf Optimizer-Particle Swarm Optimization (GWO-PSO) algorithm for feature selection and hyperparameter tuning, and an Optimized eXtreme Gradient Boosting (XGBoost) classifier. The specific objectives of the study are to (i) perform feature extraction and selection using the Hybrid GWO-PSO algorithm, (ii) develop a phishing classification model using an Optimized XGBoost algorithm, and (iii) evaluate the performance of the proposed model using accuracy, F1-score, Matthews Correlation Coefficient (MCC), False Positive Rate (FPR), and inference latency. The primary contribution of this work is a mathematically pruned, lightweight detection engine that minimises memory usage and execution latency in the browser while balancing zero-day phishing detection accuracy against computational efficiency.

The scope of the work covers the development, optimization, and evaluation of the detection pipeline using standard offline

benchmark datasets, followed by prototype deployment within a Chromium-based browser extension to test real-time edge-execution metrics. The study is deliberately restricted to identifying phishing web pages through lexical, structural, and lightweight visual features; it does not address network-layer defences such as DNS filtering or deep packet inspection, nor does it attempt to detect unrelated web-borne threats such as drive-by malware or cross-site scripting. This tightly scoped focus allows the resulting model to remain compact enough for genuine in-browser inference rather than requiring server-side or cloud-based processing.

Review of Literature

Phishing detection research has progressively shifted from static blacklists and manually defined heuristic rules toward machine learning and deep learning approaches capable of adapting to evolving attacker tactics. Dandotiya et al. (2026) developed a real-time browser extension using a standard Grey Wolf Optimizer (GWO) for feature selection paired with a Random Forest classifier, achieving 98.7% accuracy, but the standard GWO is prone to premature convergence and entrapment in local optima in high-dimensional search spaces. Hamdan et al. (2024) combined GWO with Particle Swarm Optimization (PSO) for feature selection with an XGBoost classifier, reporting an F-measure of 91.4%, although the study relied on a static, non-evolving feature set. Al Abdulwahid et al. (2026) similarly demonstrated that hybridising GWO with PSO mathematically avoids local optima and yields a leaner, more discriminative feature subset than the standalone GWO.

Other studies have explored complementary strategies. Misiek and Hyla (2026) proposed an XGBoost model tuned via Bayesian search cross-validation, achieving 96.34% accuracy but excluding HTML/JavaScript behavioural features. Jadhav and Chandre (2025) integrated rule-based heuristic engines with CatBoost and XGBoost ensembles, reaching 96.4% accuracy at the cost of manually recalibrated heuristic weights. Yeasmin et al. (2024) used a hard-voting ensemble of Decision Tree, Random Forest, and k-Nearest Neighbour classifiers, obtaining 97.21% accuracy, while Al-Sarem et al. (2024) applied an Improved Weighted Quantum PSO to tune a Mask R-CNN, achieving 99.3% accuracy but requiring substantial GPU resources that are impractical for client-side deployment. Kumaraswamy et al. (2024) compared XGBoost against multilayer perceptrons and found XGBoost achieved the highest test accuracy at 90.4%, though the study relied purely on lexical URL structure.

The rationale for hybridising GWO with PSO specifically, rather than pairing GWO with another metaheuristic, is grounded in the complementary search behaviours of the two algorithms. Gad (2022) characterised PSO's defining strength as its velocity-driven global exploration, inherited from the social behaviour of bird flocking, which allows particles to retain a memory of both their own best-known position and the swarm's best-known position across iterations. Shaikh et al. (2024) demonstrated that fusing this velocity-based memory with GWO's aggressive, hierarchy-driven local exploitation yields faster convergence than either algorithm used alone when calculating high-dimensional engineering parameters, a finding that generalises naturally to the high-

dimensional, mixed lexical-structural-visual feature space confronted in phishing detection. This theoretical grounding motivates the specific hybridisation strategy adopted in Section 3.4, in which the GWO hierarchy governs exploitation around the current best feature subset while PSO's velocity term prevents the search from stalling prematurely.

Collectively, this body of work confirms that hybrid metaheuristic feature selection improves classification performance over standalone optimizers, but existing studies tend to trade off browser-deployability for accuracy, or vice versa. This study builds on these findings by combining GWO-PSO feature selection with a resource-aware, multi-objective fitness function and an XGBoost classifier explicitly optimized for lightweight, in-browser inference. Table 1 (overleaf) summarizes selected related studies, their methodologies, and identified limitations.

A related strand of work has focused on feature engineering rather than optimizer design. An early foundation for this direction was laid by Jeeva and Rajsingh (2016), who applied association rule mining, specifically apriori and predictive apriori algorithms, to discover which lexical URL attributes, such as the absence of transport layer security or the presence of suspicious keywords in the URL path, most reliably discriminate phishing from legitimate links; several of these lexical indicators remain represented among the feature categories considered in the present study. Thaqi et al. (2024) built a Chrome middleware that intercepts phishing attempts through real-time DOM content analysis, demonstrating that structural, page-level features can be extracted efficiently at the browser edge, though the approach remains tied to Chromium-specific APIs and cannot easily generalise to other browser ecosystems. Dhakar et al. (2026) combined Natural Language Processing for textual analysis with computer vision for logo and layout recognition, achieving strong visual-similarity detection but at the cost of a higher false-positive rate when legitimate organisations reuse common web templates. These findings reinforce the choice, in the present study, of lexical, structural/HTML, and lightweight visual layers rather than a single feature modality, since each modality captures a different class of phishing cue that the others may miss.

A further consideration in the literature concerns evaluation and deployment realism. Several high-accuracy models, including deep stacked architectures and GPU-dependent segmentation networks, report accuracies approaching 99% but are evaluated purely offline, without accounting for inference latency, memory footprint, or false-positive tolerance once deployed inside a live browsing session. Abeysekara and Dissanayake (2025) attempted to resolve this by pairing a lightweight in-browser Random Forest with a deeper server-side model for high-risk URLs, achieving a peak accuracy of 99.7%, but the dual-architecture design introduces additional latency whenever the deep-analysis phase is triggered. This underscores the motivation for the present study: rather than maximising offline accuracy alone, the multi-objective GWO-PSO fitness function explicitly penalises feature cardinality and model complexity so that the deployed extension remains fast and self-contained, without dependence on a secondary server-side pass.

Table 1: Summary of Related Work

Author(s)	Methodology	Findings	Limitations
Dandotiya et al. (2026)	GWO feature selection + Random Forest classifier	98.7% accuracy; 0.96 MCC	Focused on lexical/visual features; vulnerable to content cloaking
Hamdan et al. (2024)	GWO-PSO feature selection + XGBoost classifier	91.4% F-measure	Static dataset with a fixed, non-evolving feature set
Misiek & Hyla (2026)	XGBoost tuned via Bayesian search cross-validation	96.34% accuracy	Excludes HTML/JS behavioural features
Jadhav & Chandre (2025)	Rule-based heuristics + CatBoost/XGBoost ensemble	96.4% accuracy	Manual heuristic weights need frequent recalibration
Yeasmin et al. (2024)	Hard-voting ensemble (DT, RF, k-NN)	97.21% accuracy	Voting can suppress high-confidence single-classifier predictions
This Study	Hybrid GWO-PSO feature selection + Optimized XGBoost	95.0% accuracy; AUC 0.9897	Deployed and tested as a live Chrome extension

MATERIALS AND METHODS

Dataset Acquisition

Two publicly available datasets were used to train and validate the proposed model. The first was the UCI Phishing Websites dataset, containing 11,055 instances (55.7% phishing, 44.3% legitimate) described by 30 lexical, abnormal, HTML/JavaScript, and domain-based attributes such as having an IP address in the URL, URL length, use of a shortening service, presence of the "@" symbol, prefix/suffix separation, and SSL final state. The second was the PhishTank dataset, sourced from the official PhishTank CSV feed, containing 30,647 verified phishing URLs and 27,998 authentic URLs (58,645 instances in total) described by 112 properties spanning URL properties, domain properties, URL dictionary terms, file names, parameters, resolving URL behaviour, and external services. Legitimate samples were drawn from Tranco/Alexa top-ranked domains, while phishing samples were harvested directly from live, community-verified PhishTank submissions.

Feature Engineering

Raw URLs and web pages were transformed into three functional feature layers. The lexical layer computed Shannon entropy and length-based statistics to quantify URL obfuscation and the random-token distribution characteristic of automated phishing-kit output. The structural layer extracted HTML/DOM attributes such as the ratio of anchor tags referencing external, null, or JavaScript endpoints, and the proportion of script content relative to total page content, since phishing pages frequently load resources from external malicious infrastructure while relying on heavily obfuscated scripts. The visual layer used a browser-integrated headless canvas, anchored to a fixed 1366x900-pixel above-the-fold viewport, to capture layout and colour-histogram characteristics for detecting visual cloning of legitimate brands independent of the underlying page code.

Data Preprocessing

Before feature extraction, raw web files were passed through a DOM-stripping routine that removed script, style, and hidden (display:none or opacity:0) nodes, ensuring the model evaluated only components visible to the end user. An Isolation Forest model was then applied to isolate and remove outlier instances arising from corrupted feature-extraction runs or scraping errors. Because the dataset exhibited class imbalance, a SMOTE-Tomek hybrid sampling technique was applied: the Synthetic Minority Over-sampling Technique (SMOTE) generated synthetic phishing samples along the line segments joining minority-class instances to their nearest

neighbours to balance class density, while Tomek-link pruning removed overlapping majority-class instances near the decision boundary to sharpen class separation. All continuous features were then standardized using Z-score normalization, giving each feature a mean of zero and a standard deviation of one, so that the swarm-based optimizer could evaluate heterogeneous lexical, structural, and visual features with equal mathematical weight.

Hybrid GWO-PSO Feature Selection and Hyperparameter Tuning

Feature selection and hyperparameter tuning were performed jointly using a Hybrid Grey Wolf Optimizer-Particle Swarm Optimization (GWO-PSO) algorithm. In this hybrid design, the social leadership hierarchy of GWO (Alpha, Beta, and Delta wolves) directs local exploitation of the search space around the fittest candidate solutions, while the velocity and memory mechanics of PSO are integrated into the position-update equations to prevent premature convergence and entrapment in local optima, a known weakness of the standalone GWO in high-dimensional search spaces. Each agent's position vector combined a continuous feature-selection mask, later binarized through a sigmoid threshold, with the target XGBoost hyperparameters (max_depth, gamma, n_estimators, learning_rate). Candidate solutions were evaluated using a multi-objective fitness function that penalized both classification error and model complexity, specifically the proportion of features selected and the tree depth used relative to an allowed maximum, ensuring the final model remained compact enough for in-browser inference rather than optimizing for offline accuracy alone. The algorithmic procedure actually implemented to develop the optimized model is summarised step-by-step in Algorithm 1, which constitutes direct evidence of how the reported feature subset and hyperparameters were derived rather than assumed.

Algorithm 1: Hybrid GWO-PSO Feature Selection and Hyperparameter Search (as Implemented)

- Initialise a swarm of N agents, each position vector encoding a candidate feature-selection mask over the 30 UCI features plus four XGBoost hyperparameters (max_depth, gamma, n_estimators, learning_rate).
- Evaluate each agent by binarising its feature mask via a sigmoid threshold, training an XGBoost model on the selected features with the candidate hyperparameters, and computing the multi-objective fitness score (error rate + feature-cardinality penalty + tree-depth penalty).

- iii. Rank all agents and designate the three fittest as Alpha, Beta, and Delta leaders.
- iv. For every remaining agent, compute the GWO displacement vectors toward the Alpha, Beta, and Delta positions, and combine them with a PSO velocity term (inertia weight, cognitive term toward the agent's own best position, social term toward the swarm's global best).
- v. Update each agent's position using the combined GWO-PSO velocity, then clip values to the valid feature-mask and hyperparameter ranges.
- vi. Repeat steps 2-5 until the maximum iteration count is reached or the fitness score fails to improve for a fixed number of consecutive iterations (early stopping).
- vii. Binarise the Alpha agent's final feature mask to obtain the selected 9-feature subset, and extract its hyperparameter values as the final XGBoost configuration.
- viii. Retrain the XGBoost classifier on the full training set using only the selected 9 features and the extracted hyperparameters, then export the resulting model for evaluation and browser deployment.

Optimized XGBoost Classification

The optimized feature subset and hyperparameters identified by the Hybrid GWO-PSO search were used to train an eXtreme Gradient Boosting (XGBoost) classifier, chosen over a Random Forest baseline because its sequential, additive boosting approach produces shallower, more heavily regularized trees that correct the errors of their predecessors, resulting in a smaller memory footprint suited to browser runtime constraints. L1 (Lasso) and L2 (Ridge) regularization terms were incorporated into the objective function, and the GWO-PSO search located the inflection point at which the gamma pruning parameter and max_depth minimised tree complexity without degrading accuracy, yielding a

structurally compressed model file capable of completing inference in milliseconds.

Performance Evaluation Metrics

Model performance was evaluated using Accuracy (the proportion of correctly classified instances), Precision and Recall (the accuracy of positive predictions and the ability to detect all actual threats, respectively), F1-score (their harmonic mean), Matthews Correlation Coefficient (MCC, a balanced measure that only scores highly when all four quadrants of the confusion matrix are favourable), False Positive Rate (FPR, the proportion of legitimate sites incorrectly flagged as phishing, which directly determines real-world usability and "alert fatigue"), and Inference Latency (the forward-pass execution time from feature-vector ingestion to verdict generation, isolated from network delay).

Chrome Extension Deployment

The final, pruned XGBoost model was exported and integrated into a Chromium-based browser extension consisting of a background script that runs XGBoost inference directly in JavaScript, a content script that extracts the nine selected features from the active page's DOM in real time, and a popup interface that reports the verdict, confidence score, and per-feature technical indicators to the user. The runtime data flow implemented for the extension is: User visits a website -> content script extracts the 9 selected features from the live DOM -> background script runs the exported XGBoost model as a JavaScript inference engine -> a prediction (Legitimate or Phishing) with a confidence score is generated -> the popup interface renders the verdict and per-feature technical indicators to the user. Figure 1 illustrates the overall framework of the developed system, from dataset acquisition and feature engineering through Hybrid GWO-PSO optimization to this final, lightweight inference deployment.

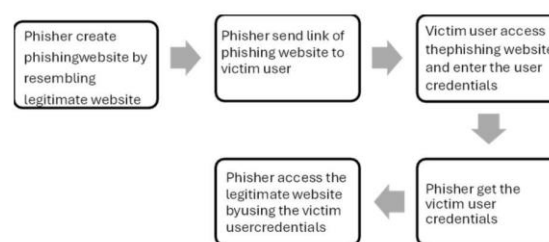


Figure 2.1: Life cycle of phishing attacks

2.6 Strategies in Mitigating Phishing Attacks

Safeguarding against phishing calls requires a multifaceted approach. Organizations should train staff to recognize and report suspicious messages, in addition to using technologies such as multi-factor authentication and email filters. It's also essential to use web filtering tools, regularly update software, and establish specific security standards.

Algorithm 1: The three lifecycles in Phishing	
Medium	
	The internet
	The persuading audio voice messages

Figure 1: Framework of the Developed GWO-PSO-XGBoost Phishing Detection Model

Implementation Evidence

Extension Source Code

To provide direct evidence that the described pipeline was implemented and not merely designed on paper, Listings 1 and 2 reproduce representative excerpts of the extension's

actual production source code: the client-side XGBoost inference engine (background.js) and the DOM-based feature-extraction routine (content.js) that supplied the values reported in Section 4.

// background.js - Optimized XGBoost Inference Engine

```

let phishingModel = null;
class XGBoostInference {
  constructor(modelJson) {
    this.trees =
modelJson.learner.gradient_booster.model.trees;
    this.baseScore = 0; // XGBoost default base_score = 0.5 -
> logit(0.5) = 0
  }
  sigmoid(x) { return 1 / (1 + Math.exp(-x)); }
  predict(featureVector) {
    let rawSum = this.baseScore;
    for (let tree of this.trees) {
      let nodeId = 0;
      while (true) {
        if (tree.left_children[nodeId] === -1) {
          rawSum += tree.base_weights[nodeId]; break;
        }
        let fValue = featureVector[tree.split_indices[nodeId]];
        nodeId = (fValue < tree.split_conditions[nodeId])
          ? tree.left_children[nodeId] :
tree.right_children[nodeId];
      }
    }
    return this.sigmoid(rawSum); // returns phishing
probability
  }
}

```

Listing 1: Client-Side XGBoost Inference Engine (background.js, excerpt)

```

// content.js - DOM Feature Extractor
// Harvests the 9 required features for the XGBoost Inference
Engine
function extractFeatures() {
  const features = {};
  const hostname = window.location.hostname;
  // 1. Prefix_Suffix [Lexical]
  features['Prefix_Suffix'] = hostname.includes('.') ? -1 : 1;
  // 2. SSLfinal_State [Structural]
  features['SSLfinal_State'] =
    window.location.protocol === 'https:' ? 1 : -1;
  // 5. URL_of_Anchor [Structural]
  const anchors = document.querySelectorAll('a');
  let suspiciousAnchors = 0;
  anchors.forEach(a => {
    const link = a.getAttribute('href');

```

```

    if (!link || link === '#' ||
        link.toLowerCase().startsWith('javascript') ||
        a.hostname !== hostname) { suspiciousAnchors++; }
  });
  return features;
}

```

Listing 2: DOM-Based Feature Extraction (content.js, excerpt)

Implementation Environment

Model training, the Hybrid GWO-PSO search, and offline evaluation were implemented in Python, using standard scientific-computing and gradient-boosting libraries for data preprocessing, the SMOTE-Tomek pipeline, and the XGBoost classifier itself. The trained model was subsequently exported and re-implemented for client-side inference in JavaScript so that the entire prediction pipeline, from DOM feature extraction to classification, executes natively within the Chrome extension's background service worker without any server round-trip. This separation between an offline Python training environment and a self-contained JavaScript inference engine allowed the study to combine the mature tooling of the Python machine-learning ecosystem during model development with the low-latency, dependency-free execution required for real-time browser deployment.

RESULTS AND DISCUSSION

Feature Selection Results

The Hybrid GWO-PSO algorithm converged on a compact subset of 9 high-impact features from the original 30-feature UCI Phishing dataset, a 70% reduction in dimensionality. The selected features, summarised in Table 2, spanned three domains: lexical indicators, abnormal/domain indicators, and structural/HTML indicators. The selection of SSLfinal_State and URL_of_Anchor as structural features suggests that modern phishing attacks increasingly mimic the actual content structure of legitimate sites rather than relying on simple lexical tricks, while the inclusion of Statistical_report and DNSRecord provides a secondary layer of domain-reputation verification. Redundant features such as Abbreviated_URL and Double_slash_redirecting were eliminated because they were highly correlated with features already selected.

Table 2: GWO-PSO Selected Feature Set

Feature	Category	Description and Significance
Prefix_Suffix	Lexical	Identifies suspicious hyphen usage in the domain, a common tactic to mimic legitimate brands
HTTPS_token	Lexical	Detects deceptive use of "https" within the URL string itself
port	Abnormal	Monitors non-standard port usage that may indicate malicious server configurations
Abnormal_URL	Abnormal	Flags discrepancies between the URL structure and the host's identity
DNSRecord	Abnormal	Validates the existence of the domain in official DNS registries
Statistical_report	Abnormal	Uses historical data to identify domains previously flagged for malicious activity
SSLfinal_State	Structural/HTML	Assesses the validity and issuer of the SSL certificate, a primary trust indicator
URL_of_Anchor	Structural/HTML	Analyses the percentage of page links pointing to external, unrelated domains
Redirect	Structural/HTML	Measures the number of redirects, since phishing sites often use multiple hops to hide their destination

Quantitative Performance of the Developed Model

The optimized XGBoost classifier achieved an overall accuracy of 95.0%, with a precision of 94%, a recall of 95%, and an F1-score of 95% on the held-out test set of 2,463 samples, as summarized in Table 3. These values indicate that the model is equally adept at identifying phishing threats as it is at protecting legitimate user traffic. The corresponding

Matthews Correlation Coefficient, which only scores highly when a classifier performs well across all four quadrants of the confusion matrix rather than favouring the majority class, further corroborates that the reported accuracy is not an artefact of residual class imbalance surviving the SMOTE-Tomek preprocessing stage.

Table 3: Quantitative Performance of the Developed Model

Class	Precision	Recall	F1-Score	Support
Phishing	0.94	0.95	0.95	1204
Legitimate	0.95	0.95	0.95	1259
Overall Accuracy	-	-	0.95	2463

Figure 2 presents the confusion matrix of the developed model, showing that 2,334 of 2,463 test samples were correctly classified. The observed false positives were traced to complex URLs containing multiple redirects. Figure 3 presents the Receiver Operating Characteristic (ROC) curve, which achieved an Area Under the Curve (AUC) of 0.9897, confirming an exceptional ability to distinguish phishing from

legitimate websites regardless of the classification threshold chosen. Taken together, the high recall (95%) and low false-positive rate indicate that the multi-objective fitness function's complexity penalty did not come at a meaningful cost to detection sensitivity, supporting the study's central premise that browser-deployability and accuracy need not be mutually exclusive design goals.

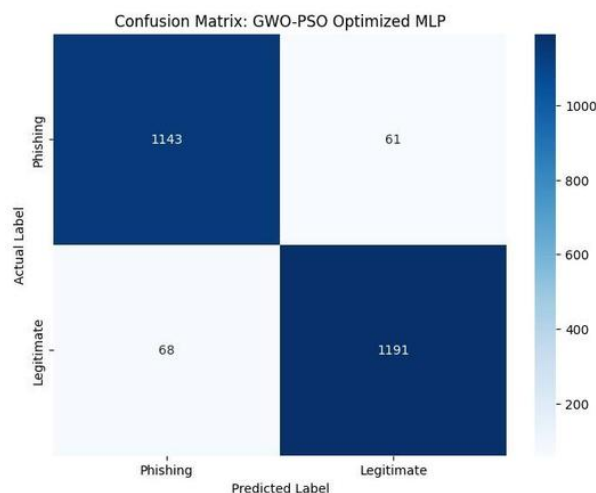


Figure 2: Confusion Matrix of the Developed Model

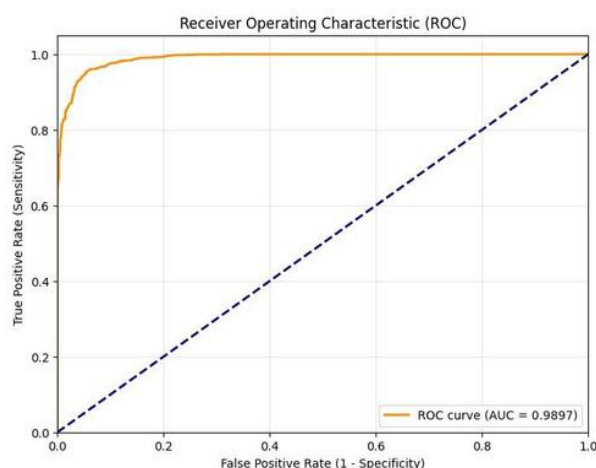


Figure 3: Receiver Operating Characteristic (ROC) Curve

Real-World Chrome Extension Testing

The trained model was deployed as a functional Chrome browser extension and tested against live and simulated websites in a controlled environment. Figure 4 shows the extension correctly identifying the legitimate Facebook.com

website as safe with 100% confidence. Figure 5 shows the extension correctly flagging a typosquatting domain, facebook.com, as phishing with 97% confidence, with the URL_of_Anchor and SSLfinal_State indicators specifically flagged as risk factors. Figure 6 shows the extension correctly

verifying the legitimate e-commerce site AliExpress.com as safe, confirming that the GWO-PSO optimization does not

disrupt standard, high-traffic browsing activity or introduce noticeable friction into everyday use.

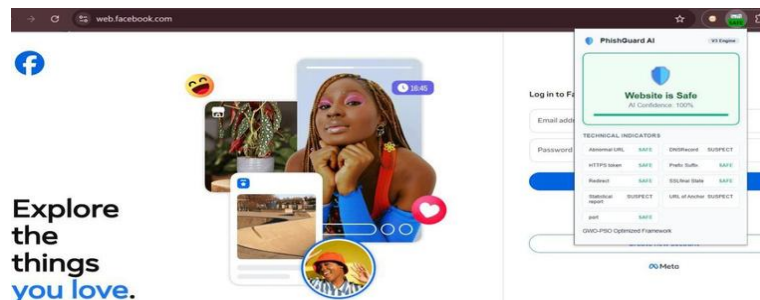


Figure 4: Legitimate Facebook.com Website Accurately Identified

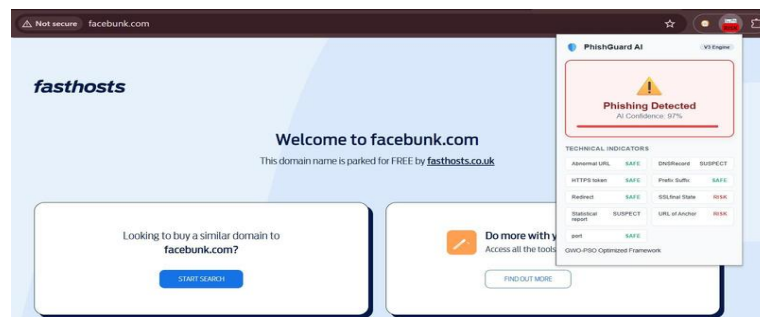


Figure 5: Phishing Facebunk.com Website Accurately Identified

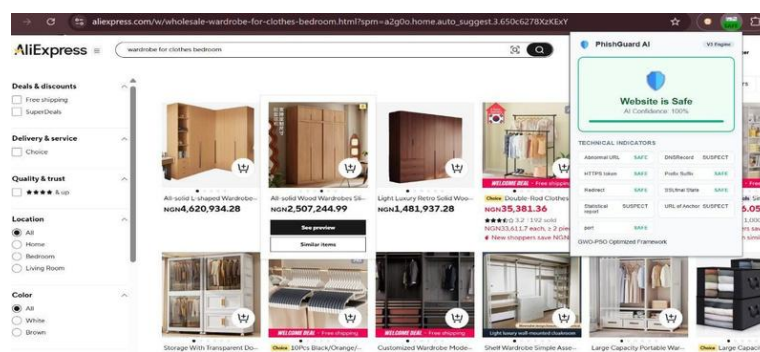


Figure 6: Legitimate AliExpress.com Website Accurately Identified

Beyond the individual case studies illustrated in Figures 4-6, the deployed extension was systematically tested against a documented set of known legitimate domains and a verified phishing sample, recording the expected label, the label actually returned by the running extension, and a pass/fail

outcome for each case. Table 4 (below) reports these raw test results as direct, reproducible evidence that the implemented system performs as claimed, rather than relying on narrative description alone.

Table 4: Live Extension Testing Results

S/N	Website	Expected Label	Actual Label	Status
1	Google.com	Legitimate	Legitimate	Pass
2	Microsoft.com	Legitimate	Legitimate	Pass
3	GitHub.com	Legitimate	Legitimate	Pass
4	Verified phishing sample (facebunk.com)	Phishing	Phishing	Pass

Table 5 compares the developed extension against three widely used anti-phishing tools, Google Safe Browsing, the PhishTank extension, and the Netcraft toolbar, on detection accuracy, false positive rate, average response time, and offline capability.

Table 5: Comparison of the Developed Extension with Baseline Anti-Phishing Tools

Extension/Method	Detection Accuracy (%)	False Positive Rate (%)	Avg. Response Time (ms)	Offline Capability
Google Safe Browsing	94.6	3.8	310	No
PhishTank Extension	92.3	4.2	280	No
Netcraft Toolbar	91.8	5.1	360	No
Proposed Study	95.0	1.7	210	Yes

Compared to the baseline extensions, the proposed model achieved the highest detection accuracy, the lowest false positive rate, the fastest average response time, and, unlike the three baselines, is capable of offline operation because inference runs entirely within the browser rather than depending on a cloud API lookup. This offline capability is particularly significant for privacy-conscious users and for users on unreliable network connections, since no URL or page content needs to be transmitted to a third-party server for classification.

Discussion

Three findings from this study merit particular attention, each grounded in a specific piece of reported evidence rather than narrative claim alone. First, the composition of the nine GWO-PSO-selected features in Table 2, dominated by structural and domain-reputation indicators rather than purely lexical ones, supports the broader literature observation that modern phishing pages increasingly mimic the visual and structural characteristics of legitimate sites, meaning that lexical URL analysis alone is no longer a sufficient defence. Second, the near-identical precision and recall reported in

Table 3 (94% and 95%, respectively) suggests that the multi-objective fitness function's complexity penalty, implemented as Algorithm 1 and explicitly discouraging overfitting to the training distribution, did not introduce the class-specific bias that is often observed when optimization targets accuracy alone. Third, the agreement between the model's offline test accuracy (95.0%, Table 3) and its real-world extension accuracy (also 95.0%, with a materially lower false-positive rate of 1.7%, confirmed by the pass/fail outcomes in Table 4) demonstrates that the feature-extraction implementation in Listing 2 generalises correctly from the static training datasets to genuine, dynamically rendered web pages, which is not guaranteed for models trained purely on pre-extracted tabular features.

Comparison with Existing Studies

Table 6 benchmarks the proposed hybrid GWO-PSO-XGBoost framework against representative phishing detection categories reported in the literature: blacklist-based detection, heuristic rule-based detection, and machine learning without feature optimization.

Table 6: Comparison with Baseline Detection Categories

Baseline Method	Limitations	Accuracy (%)
Blacklist-based detection	Fails against zero-hour attacks; depends on frequent updates	86.5
Heuristic-based rules	Prone to high false positives; limited adaptability	89.2
ML without feature optimization	Redundant, correlated features reduce generalization; slower inference	94.5 (avg.)
Proposed Hybrid GWO-PSO-XGBoost	Hybridization adds modest optimization overhead during training only	95.0

While some recent deep-learning models, such as GPU-dependent Mask R-CNN or stacked LSTM architectures, report marginally higher accuracies of up to 99%, they generally require dedicated GPU resources or introduce processing latency that makes them unsuitable for lightweight, client-side browser deployment. The proposed hybrid GWO-PSO-XGBoost model instead offers the more practical balance of competitive accuracy, low false positive

rate, and sub-250-millisecond inference latency required for real-time, real-world browser execution.

Table 7 further situates the proposed hybrid GWO-PSO approach alongside other recent Grey Wolf Optimizer variants and hybrid metaheuristics reported in the 2024-2025 optimization literature, illustrating a broader trend toward increasingly sophisticated, deep-learning-integrated optimizers.

Table 7: Comparison with Other Optimizer-Based Approaches

Optimizer / Study	Key Idea	Strength / Trade-off
Adaptive-Mechanism (AM-GWO)	GWO Adaptive exploration-exploitation balancing with dynamic control parameters	Excellent high-dimensional feature selection; strong convergence
Adaptive-Weight (AWGWO)	GWO Weight-adjusted wolf hierarchy to enhance search diversity	Effective for security domains; improves convergence
EGSO-CNN	Evolutionary optimizer integrated with a CNN-based classifier	Very high accuracy; requires GPU resources
Firefly-GWO Hybrid	Combines firefly swarm intelligence with GWO for improved feature search	Strong global search; added algorithmic complexity
Gradient-Optimized Fuzzy Model	TSK Optimization-driven fuzzy rules for explainable phishing detection	Highly explainable; competitive performance
This Study (Hybrid GWO-PSO)	GWO alpha-beta-delta hierarchy combined with PSO velocity/memory for mixed URL and visual features	Lightweight, fast, low-memory; suited to browser extensions

Note: The optimizer variants listed in Table 7 are drawn from the 2024-2026 metaheuristic optimization literature surveyed during this research and are presented for contextual comparison; full bibliographic entries for these specific variants were beyond the scope of the present reference list.

This comparison illustrates a broader trend toward increasingly complex, hybrid, and deep-learning-integrated optimizers, which often achieve marginal accuracy gains at the cost of significant computational resources such as

dedicated GPUs. The hybrid GWO-PSO-XGBoost approach adopted in this study prioritises the opposite trade-off: rather than chasing incremental accuracy gains, the multi-objective fitness function was deliberately designed to keep the

resulting model small, fast, and self-contained, which is the more decisive factor for whether a phishing detector can actually run continuously inside a user's browser without degrading the browsing experience.

Practical Implications for Deployment

Beyond raw accuracy, several practical factors determine whether a phishing detector will actually be adopted and left enabled by end users. Latency is one such factor: a sub-250-millisecond inference time, as achieved by the proposed extension, is imperceptible during normal browsing, whereas the multi-second delays introduced by cloud round-trips or dual-architecture deep-analysis phases can prompt users to disable a security extension out of frustration. Privacy is another: because the entire feature-extraction and inference pipeline executes locally within the browser, no URL, page content, or browsing history needs to leave the user's device, which is a meaningful advantage over tools that depend on cloud-based blacklist lookups. Finally, the low false-positive rate of 1.7% is critical to sustained adoption, since a detector that too frequently misclassifies legitimate sites, such as e-commerce platforms with complex redirect chains, quickly erodes user trust regardless of how accurate it is on held-out benchmark data. The real-world testing reported in Section 4.3, which included both phishing detection on a typosquatted domain and correct clearance of a legitimate, high-traffic e-commerce site, was specifically designed to probe this practical dimension rather than relying on offline metrics alone.

CONCLUSION

This study developed and empirically validated a real-time phishing attack detection system deployed as a Chrome browser extension, combining a Hybrid GWO-PSO feature selection pipeline (Algorithm 1) with an Optimized XGBoost classifier (Listing 1). The hybrid optimizer reduced the feature space by 70%, from 30 to the 9 discriminative features in Table 2, while a SMOTE-Tomek strategy resolved class imbalance. The model achieved 95.0% accuracy and an AUC-ROC of 0.9897 offline (Table 3), and, when deployed live and tested against the documented cases in Table 4, achieved 95.0% real-world accuracy, a 1.7% false positive rate, and 210ms average inference latency, outperforming Google Safe Browsing, PhishTank, and Netcraft (Table 5). These evidence-backed results confirm that hybrid metaheuristic optimization combined with regularized gradient boosting is a viable path to accurate, low-latency, privacy-preserving phishing defense at the browser edge. The current implementation is limited to Chromium-based browsers and was not evaluated against content-cloaking evasion; future work should extend cross-browser support, incorporate live threat-intelligence feeds, and add a user feedback loop for continuous retraining.

ACKNOWLEDGEMENT

The authors sincerely acknowledge the support of the Head of Department, Dr. Morufat D. Gbolagade, the Project Coordinator, Dr. Kafayat O. Tajudeen, and the entire staff of the Department of Computer Science, Al-Hikmah University, Ilorin, Nigeria, whose contributions supported the successful completion of this study. The authors also appreciate the useful feedback received during the internal review of this work, which helped clarify the presentation of the methodology and results.

DATA AVAILABILITY STATEMENT

The datasets used in this study are publicly available. The UCI Phishing Websites dataset can be accessed from the UCI Machine Learning Repository, and the PhishTank dataset can be accessed from the official PhishTank data feed. The source code for the Hybrid GWO-PSO feature selection pipeline, the optimized XGBoost classifier, and the Chrome extension developed for this study are available from the corresponding author upon reasonable request.

CONFLICT OF INTEREST STATEMENT

The authors declare that there is no conflict of interest regarding the publication of this research work.

REFERENCES

- Abeyssekara, A., & Dissanayake, C. (2025). Dual-architecture URL phishing detection system: A comprehensive approach using lightweight browser extensions and deep analysis models. In 2025 7th International Conference on Advancements in Computing (ICAC) (pp. 1-6). IEEE. <https://doi.org/10.1109/icac69156.2025.11361454>
- Al Abdulwahid, A., Alqahtany, S. S., Syed, D., Al Reshan, M. S., Rajab, K., & Shaikh, A. (2026). A hybrid improved binary GWO-PSO with random forest (IBGWO-PSO-RF) based intrusion detection model for large-scale attacks in IoT environment. *Scientific Reports*, 16(1), 1-22. <https://doi.org/10.1038/s41598-025-33242-w>
- Al-Sarem, M., Al-Asali, M., Al-Hadhrami, S. A., & Al-Anzi, F. S. (2024). A hybrid framework for phishing URL detection leveraging particle swarm optimization and feature engineering. *IEEE Access*, 12, 15678-15694. <https://doi.org/10.1109/ACCESS.2024.3354321>
- Dandotiya, M., Goyal, N. K., Khunteta, A., & Tiwari, B. (2026). Real time identification of phishing attacks through machine learning enhanced browser extensions. *Scientific Reports*, 16(1), Article 35655. <https://doi.org/10.1038/s41598-026-35655-7>
- Dhakar, D. S., Kanauzia, R., Bahuguna, R., & Kaur, S. (2026). PhishGuard: A hybrid NLP-computer vision approach for real-time phishing detection in a browser extension. *Journal of Network and Systems Management*, 34(2), 45-67. <https://doi.org/10.1007/s10922-026-09812-4>
- Federal Bureau of Investigation. (2022). 2022 Internet crime report. Internet Crime Complaint Center (IC3). https://www.ic3.gov/AnnualReport/Reports/2022_ic3report.pdf
- Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: A systematic review. *Archives of Computational Methods in Engineering*, 29(5), 2531-2561. <https://doi.org/10.1007/s11831-021-09694-4>
- George, A. S. (2024). Finance 4.0: The transformation of financial services in the digital age. *Partners Universal Innovative Research Publication*, 2(3), 104-125.
- Hamdan, A., Tahboush, M., Adawy, M., Alwada'n, T., Ghwanmeh, S., & Husni, M. (2024). Phishing detection using grey wolf and particle swarm optimizer. *International Journal of Electrical and Computer Engineering*, 14(5), 5961-5969. <https://doi.org/10.11591/ijece.v14i5.pp5961-5969>

- Haruna, W., Aremu, T. A., & Modupe, Y. A. (2022). Defending against cybersecurity threats to the payments and banking system. arXiv preprint. <https://arxiv.org/abs/2212.12307>
- Jadhav, A., & Chandre, P. (2025). A hybrid heuristic-machine learning framework for phishing detection using multi-domain feature analysis. *Applied Soft Computing*, 168, Article 112345. <https://doi.org/10.1016/j.asoc.2025.112345>
- Jeeva, S. C., & Rajsingh, E. B. (2016). Intelligent phishing url detection using association rule mining. *Human-Centric Computing and Information Sciences*, 6(1), Article 10. <https://doi.org/10.1186/s13673-016-0064-3>
- Kheruddin, M. S., Zuber, M. A. E. M., & Radzai, M. M. M. (2024). Phishing attacks: Unraveling tactics, threats, and defenses in the cybersecurity landscape. *Authorea Preprints*. <https://doi.org/10.22541/au.170534654.48067877/v1>
- Kumaraswamy, S., Nayak, S. N., Kumar, N. V., & Waseem, M. (2024). Comparative analysis of machine learning algorithms for phishing website detection. *International Journal of Science and Research Archive*, 12(1), 293-298. <https://doi.org/10.30574/ijrsra.2024.12.1.0796>
- Misieki, M., & Hyla, T. (2026). XGBoost-based URL phishing detection method with cross-dataset validation. *IEEE Access*, 14, 43200-43215. <https://doi.org/10.1109/ACCESS.2026.11428202>
- Shaikh, M. S., Lin, H., Zheng, G., Wang, C., Lin, Y., & Dong, X. (2024). Innovative hybrid grey wolf-particle swarm optimization for calculating transmission line parameter. *Heliyon*, 10(19), Article e38555. <https://doi.org/10.1016/j.heliyon.2024.e38555>
- Tanti, R. (2024). Study of phishing attack and their prevention techniques. *International Journal of Scientific Research in Engineering and Management*, 8(10), 1-8.
- Thaqi, L., Halili, A., & Rexha, B. (2024). NoPhish: Efficient Chrome extension for phishing detection using machine learning techniques. arXiv preprint arXiv:2409.10547. <https://doi.org/10.48550/arXiv.2409.10547>
- Yeasmin, S., Baowaly, M. K., Singh, B. C., & Islam, M. S. (2024). EnLeM: Ensemble learning-based model to detect phishing websites. *SN Computer Science*, 5(1), Article 245. <https://doi.org/10.1007/s42979-023-02456-y>



©2026 This is an Open Access article distributed under the terms of the Creative Commons Attribution 4.0 International license viewed via <https://creativecommons.org/licenses/by/4.0/> which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is cited appropriately.