Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X
Volume 4, Issue 5, 2026

NEXORASCAN: A MACHINE LEARNING–DRIVEN CHROME EXTENSION FOR REAL-TIME DETECTION OF MALICIOUS WEBSITES AND BROWSER PERMISSION ABUSE

[*1]Muhammad Ali, [2]Muhammad Ishaq, [3]Asma Mukhtiar, [4]Zain Ul Abidin
[*1]Department of Information & Communication Engineering, The Islamia University of Bahawalpur, Punjab, Pakistan
[2]Department of Information & Communication Engineering The Islamia University of Bahawalpur, Punjab, Pakistan
[3]Department of Information & Communication Engineering The Islamia University of Bahawalpur, Punjab, Pakistan
[4]Department of Information & Communication Engineering The Islamia University of Bahawalpur, Punjab, Pakistan
[*1]malikmul00786@gmail.com., [2]asmamukhtiar211@gmail.com, [3]malikishaq5657@gmail.com, [4]engr.zain@iub.edu.pk

DOI: https://doi.org/10.5281/zenodo.20313609

Keywords
Browser extension, Chrome Manifest V3, mali-cious website detection, machine learning, phishing, browser per-mission abuse, JavaScript analysis, domain reputation, random forest, privacy - preserving classification, NexoraScan

Article History
Received: 19 April 2026
Accepted: 13 May 2026
Published: 14 May 2026

Corresponding Author: * Muhammad Ali

Abstract
Malicious websites and abusive browser permissions continue to pose serious cybersecurity threats to internet users, while traditional blacklist-based protection mechanisms often fail to detect newly emerging and adaptive attacks in real time. This paper presents NexoraScan, a machine learning–driven security framework designed to identify malicious websites and browser permission abuse through a lightweight and privacy-preserving approach. The proposed system consists of three integrated com-ponents: (1) a Google Chrome extension developed using Mani-fest V3 for real-time website monitoring, (2) a web-based URL scanning platform, and (3) a companion Android application for accessible cross-platform protection. The framework extracts six behavioural and infrastructural security features directly from active browsing sessions and evaluates them using supervised ma-chine learning models, including Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Decision Tree (DT), Logistic Regression (LR), and Naive Bayes (NB). Experimental evaluation was conducted on a balanced dataset containing 3,000 labelled website instances. Among all classifiers, Random Forest achieved the best performance with 95.0% accuracy, a macro F1-score of 0.95, and a ROC-AUC score of 0.98 under 5-fold cross-validation. Furthermore, real-time inference latency remained below 300 ms, making the solution suitable for practical browser-based deployment. Feature importance analysis demonstrates that SSL certificate validity, domain age, redirect behaviour, and JavaScript obfuscation indicators provide the strongest discriminative capability for malicious website detec-tion. The proposed framework uniquely

combines infrastructure-level indicators with behaviour-level JavaScript analysis to detect both phishing-oriented and permission-abusing web activity. All prediction and analysis operations are executed locally within the browser environment, ensuring that no user browsing history or metadata is transmitted externally, thereby preserving user privacy and supporting GDPR-oriented data minimisation prin-ciples. Experimental results indicate that NexoraScan provides an effective, lightweight, and deployable solution for real-time malicious website detection on resource-constrained systems.

I. Introduction

The web browser is now the primary interface through which people bank, consult doctors, attend classes, and stay in touch. That centrality makes it a target. Phishing, drive-by downloads, cryptojacking, and permission-abusing scripts together account for a large share of credential compromises and financial losses each year [1]. The Anti-Phishing Working Group (APWG) reported that 2023 was the worst year for phishing on record, with over 4.9 million unique phishing sites documented across all four quarters [2]. Attack volumes stayed near those levels through mid-2024 [3].

Blacklists like Google Safe Browsing and PhishTank are reactive by design: a site has to be spotted, reported, and added before anything gets blocked. The problem is that phishing sites often go dark within 24–48 hours of launch, which is faster than most blacklists update [4]. Signature-based systems face the same timing problem and are further undermined by polymorphic JavaScript obfuscation [5]. Rule-based ap-proaches are brittle and need constant expert maintenance as threat patterns shift.

Machine learning offers a different angle: a model trained on observable web properties can flag previously unseen malicious sites that share structural patterns with known bad infrastructure, without needing to recognise specific domains or signatures [6], [7]. Browser extensions are a natural home for such a detector — they run inside the browser process and have privileged access to URL metadata, DOM content, network request logs, and the Permissions API [8], [9]. Chrome Manifest V3 (MV3) makes this deployment model more secure by banning remotely-hosted code and narrowing permission declarations [10].

Browser permission abuse deserves attention on its own. Malicious or deceptive sites that trick users into granting camera, microphone, geolocation, or notification access are a growing threat vector that gets less coverage than phishing. Unlike technical exploits, permission abuse works through the browser's own dialog box — no malware artefact, no software vulnerability needed [11]. Studies confirm the pattern is widespread: fake "robot check" prompts push users toward granting notification access, after which spam or malicious payloads follow.

This paper makes the following contributions:

1)We present NexoraScan, a deployable real-time detec-tion system comprising a Chrome MV3 extension, a web scanner portal, and an Android APK, all sharing a common ML inference backend.

2)We compare six ML classifiers under identical con-ditions on a 3,000-sample balanced dataset with six infrastructure- and behaviour-derived features, using 5-fold cross-validation with reported standard deviations. Random Forest achieves 95.0% ± 1.2% accuracy and ROC-AUC of 0.98.

3)We show that a minimal, browser-extractable feature set — domain age, SSL validity, SSL expiry, HTTP redirect count, eval() density, and document.write() density — is sufficient for competitive performance without DOM parsing or server-side WHOIS queries.

4)We introduce the fusion of infrastructure-layer and behaviour-layer features as a proxy for permission-abusing code obfuscation, targeting a threat vector largely unaddressed in prior browser-extension ML systems.

5)We present a leave-one-feature-out ablation study that quantifies each feature's contribution, identifying do-main age as the single most informative feature (7.3-point accuracy drop on removal).

6)We characterise the system's false positive and false negative failure modes and outline concrete extensions, including direct browser permission monitoring and con-tinual learning.

The rest of the paper is organised as follows. Section II cov-ers related work. Section III describes the system architecture. Section IV presents the feature engineering. Section V covers classifiers and experimental setup. Section VI reports results. Section VII discusses findings and limitations. Section VIII outlines future work. Section IX concludes.

II. Related Work

A. URL- and Domain-Based Phishing Detection

URL lexical features remain a heavily studied signal for phishing detection. Kustiawan and Ghauth [6] trained multiple classifiers over a large URL dataset spanning URL, HTML, and derived features, and found Random Forest consistently strong. URL analysis avoids rendering potentially malicious content, but it can be fooled by attackers who route through legitimate cloud hosts or URL shorteners. Ahammad et al. [7] showed that pairing URL lexical features with domain in-frastructure signals substantially improves recall on zero-day phishing sites. Almujahid et al. [12] compared ML algorithms for phishing detection across temporally diverse benchmarks. Karim et al. [13] demonstrated that hybrid pipelines combining URL and domain-level features cut classifier error while staying competitive on accuracy. Zamir et al. [14] combined URL features with visual similarity metrics to catch brand-impersonation attacks. Tang et al. [15] used a graph-based method to extract structural URL features that capture token relationships and generalise better to unseen domains. Sheng et al. [16] showed that bidirectional transformer encoding of URL character sequences achieves strong multi-task detection across phishing benchmarks. Basit et al. [17] surveyed AI-based phishing detection and noted that hybrid approaches consistently outperform single-modality systems.

B. Machine Learning for Malicious Website Detection

Systematic comparisons of RF, SVM, DT, and neural base-lines consistently find ensemble methods outperforming single classifiers on heterogeneous feature sets [12], [18]. The advan-tage is attributed to variance reduction, which is especially pro-nounced when infrastructure-level and behaviour-level features are combined. Ejaz et al. [19] proposed a continual learning framework that adapts to new phishing patterns without full retraining, directly targeting temporal model decay. Dandotiya et al. [9] found that carefully engineered features enable browser-integrated RF classifiers to match state-of-the-art F1 scores at sub-500 ms latency. Karim et al. [13] showed high precision on novel PhishTank and OpenPhish URLs using hybrid URL and network-layer features. Liang et al. [20] captured structural relationships among web entities for mali-cious URL detection. Researchers also applied CNNs to raw URL character sequences, reporting competitive accuracy with less feature engineering. Sahoo et al. [4] surveyed ML-based malicious URL detection and found ensemble methods and domain-age features consistently at the top. Thakur et al. [21] reviewed deep learning for phishing email detection: CNN and LSTM architectures achieve high precision, but at inference costs that rule them out for client-side deployment and the study also highlighted feature selection as the dominant per-formance bottleneck in browser-deployable systems.

C. Browser Extensions for Security

The defensive value of browser extensions comes from their position inside the browser process. Wu et al. [22] built a browser extension combining URL features with a lightweight RF classifier, hitting competitive accuracy at under 200 ms average latency. Thaqi et al. [8] introduced NoPhish, a Chrome extension using LightGBM over 87 URL-derived features, achieving 96.5% accuracy. The 87-feature overhead does impose non-trivial extraction cost on every page nav-igation. Wang et al. [23] studied the runtime cost of deep learning inference inside browsers and found local execution viable when feature extraction is carefully bounded. Dandotiya et al. [9] confirmed that browser-integrated RF classifiers can match server-side detectors when features are chosen to minimise extraction overhead. Fowdur and Hosenally [24] compared lightweight classifiers in a browser extension con-text and found RF offered the best accuracy-latency trade-off on resource-constrained hardware. Al-Sarem et al. [25] showed that optimised stacking ensembles improve over single RF models. Pantelaios and Kapravelos [10] analysed MV3's security implications: it hardens but does not fully close the extension attack surface. Li et al. [26] documented behavioural patterns in malicious extensions that overlap with the over-permissioning patterns NexoraScan targets. Singh et al. [27] established a baseline framework for detecting malicious ex-tensions using static and dynamic analysis.

D. Browser Permission Abuse

The W3C Permissions API provides standardised access to geolocation, camera, microphone, clipboard, and notifications. Large-scale studies confirm that notification-permission abuse is widespread: deceptive interfaces coerce users into granting notification access, after which spam or malicious payloads follow [11]. Previous studies shows that social engineering cues in deceptive interfaces predict subsequent malicious behaviour, and that the pattern of permission requests is statistically predictive of malicious labelling by automated classifiers. Choo et al. [28] measured the prevalence of overly broad permission requests across benign and malicious sites. To our knowledge, no prior

browser-extension ML system has fed browser permission request behaviour directly into a classification pipeline. Sumner et al. [29] showed that even trained users remain susceptible to social engineering via browser dialogs, confirming that technical detection must complement user education. This gap directly motivates the behaviour-layer feature design of NexoraScan.

III. System Architecture

NexoraScan has three integrated layers (Fig. 1):

1)Chrome Extension (Manifest V3): Extracts features from the active page in real-time, runs the embedded classification logic, and shows the result via a colour-coded popup.

2)Web Portal (nexorascan.com): A browser-based scan-ner for on-demand URL submission, sharing the ML inference endpoint with the extension.

3)Android APK: A companion app that exposes the same scanning capability on mobile.

A. Chrome Extension Design

The extension runs under Manifest V3 (MV3), which enforces stricter permission scoping, bans remotely-hosted code execution, and replaces persistent background pages with event-driven service workers [10]. Core files: manifest.json (metadata and permission declarations), background.js (service worker for redirect counting and classification dispatch), content.js (DOM-injected script for JavaScript feature extraction), popup.html/popup.js (user interface), and model_logic.js (embedded classifier decision logic).

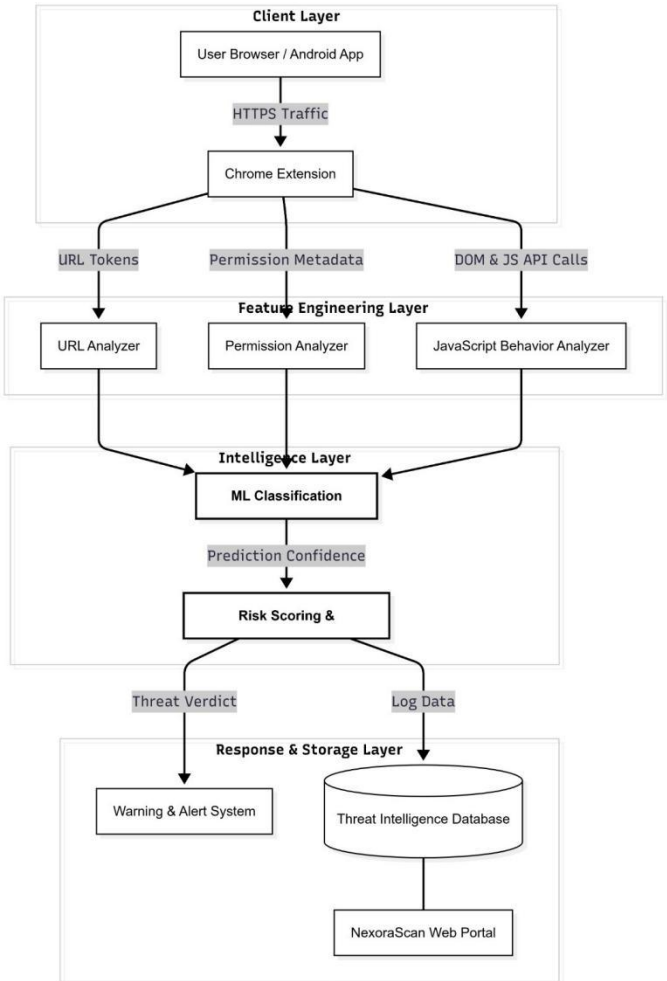

Fig. 1. Layered architecture of the NexoraScan framework. The system integrates URL analysis, SSL certificate inspection, JavaScript behaviour analysis, ML-based threat classification, and risk scoring to generate real-time warning alerts and maintain threat intelligence logs. All

classification runs locally inside the browser; no browsing data is transmitted externally.

The extension requests only the minimum permissions it needs: tabs, webRequest, scripting, storage, and activeTab. This least-privilege posture directly ad-dresses over-permissioning patterns common to malicious extensions [4]. All classification is local; no browsing data leaves the device, making the system privacy-preserving and compatible with GDPR data-minimisation obligations.

Classification Pipeline

The end-to-end pipeline runs in four stages (Fig. 2):

Feature Extraction: content.js scans all <script> elements and available external sources at document-idle, counting eval(and document.write(calls. background.js monitors webRequest events to count redirect hops and reads SSL certificate metadata from HTTP response headers.

Feature Scaling: The six-element feature vector is nor-malised using per-feature means and standard deviations

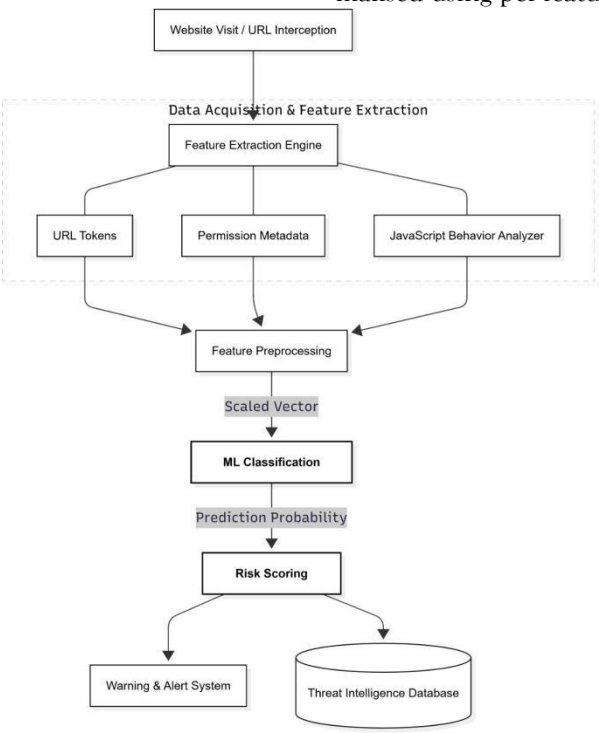

Fig. 2. End-to-end real-time classification pipeline of the NexoraScan Chrome extension. The pipeline performs local feature extraction, preprocessing, serialised RF inference, risk evaluation, and browser-based alert rendering. The entire pipeline executes inside the browser process with no external network dependency. computed offline during training and stored as constants in model_logic.js.

Table I: ***Feature Attributes, Data Types, and Security Rationale***

Feature	Type	Security Rationale
domain_age_days	Integer	Malicious domains are disproportionately newly registered (< 30 days) [6], [7]
ssl_expiry_days	Integer	Expired or near-zero SSL expiry values are predic-tive of short-lived malicious deploy-ments
ssl_valid	Binary	Absence of a valid TLS certificate is a high-risk indica-tor in modern web environments
http_redirects	Integer	Long redirect chains obscure malicious destinations; more than three redirects is Anomalous for benign sites [4]

js_eval_count	Integer	eval() is a common JavaScript obfuscation primitive [5], [23]
js_docwrite_count	Integer	document.write() enables hidden iframe injection and malicious redirect behaviour [4]

3) Inference: The trained Random Forest decision rules are serialised as nested conditional expressions in JavaScript, enabling fully offline inference at zero net-work latency.

4)Alert Rendering: The popup badge turns green (be-nign), amber ($0.60 < p < 0.80$, borderline), or red ($p \geq 0.80$, malicious), and a plain-language explanation of the flagged features is shown to the user.

IV. Feature Engineering

Feature selection was constrained to variables the Chrome WebExtensions API can extract in real-time without deep packet inspection, server-side access, or slow external API calls. The six selected features, their types, and security rationale are summarised in Table I.

A. Domain Age

Domain age — the number of days since first registration, obtained via WHOIS through the webRequest API — is the hardest feature for attackers to fake. Forging it requires either letting real time pass or buying a pre-aged domain, both of which add cost and attribution risk. Malicious domains consistently cluster toward registrations under 30 days [6], [7], [12]. Kawaoka et al. [30] showed that malicious registrations exhibit distinct temporal burst patterns detectable even without URL-level features, reinforcing domain age as a robust signal. This feature is stored as a raw integer and z-score normalised during preprocessing.

B. SSL Certificate Features

ssl_valid (binary) and ssl_expiry_days (integer) jointly capture the TLS posture of a site. Certificate authorities like Let's Encrypt have made obtaining a cert trivial, but the combination of these two features with domain age remains predictive: a freshly registered site with a short-lived certificate fits the profile of a throwaway phishing deployment [9], [31].

C. HTTP Redirect Count

Legitimate sites use zero to two redirects (e.g., bare domain to www, or HTTP to HTTPS). Malicious sites frequently chain longer sequences to obscure the final destination, route traffic through affiliate networks, or inject tracking artefacts. Values above five are rare among benign sites and carry elevated risk signal in our dataset [4].

D. JavaScript Behavioural Features

js_eval_count and js_docwrite_count are counted by content.js, which parses all inline <script> blocks and available external sources at document-idle. eval() is the standard JavaScript obfuscation tool: it executes an arbitrary string as code at runtime, defeating static analysis [5], [23]. document.write() is regularly obfuscated payloads into the DOM.

Together these two features form a behaviour-layer signal that indirectly proxies for permission-abusing code. Malicious permission-requesting scripts often use dynamic code exe-cution to invoke the Permissions API only when the user appears susceptible, making eval() density a meaningful partial indicator for social-engineering-via-permission-dialog attacks [11], [32].

The prediction function encoding the directional risk of each feature is:

$$\hat{y} = f\Big(\underbrace{d_{age},\ s_{valid},\ s_{exp}}_{\text{infrastructure}},\ \underbrace{r_{http},\ e_{eval},\ e_{dw}}_{\text{behaviour}}\Big) \quad (1)$$

where $\hat{y} \in \{0, 1\}$ is the binary prediction (0 = benign, 1 = malicious).

E. Classifiers and Experimental Setup

A. Classifiers

Six supervised binary classifiers were implemented using scikit-learn 1.3 in Python 3.11.

Random Forest (RF): An ensemble of T decision trees, each trained on a bootstrapped sample and a random feature subset. The ensemble prediction is the majority vote:

$$\hat{y}_{RF} = \text{mode}\ \{h_t(\mathbf{x})\}_{t=1}^{T} \quad (2)$$

Variance reduction through decorrelated trees makes RF robust to noise and capable of capturing non-linear feature interactions [9], [12].

Support Vector Machine (SVM): Maximises the margin between classes in the kernel-transformed feature space. Decision function with RBF kernel K:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \quad (3)$$

K-Nearest Neighbours (KNN): A non-parametric instance-based learner; classification by majority vote among the k = 5 nearest training examples under Euclidean distance.

Decision Tree (DT): A single CART tree that recursively partitions the feature space by minimising Gini impurity. Interpretable but prone to overfitting; included as the RF base-learner baseline.

Logistic Regression (LR): Models the log-odds of the malicious class as a linear combination of features:

$$P(y=1 \mid \mathbf{x}) = \sigma(\mathbf{w}^{\top}\mathbf{x} + b) = \frac{1}{1+e^{-(\mathbf{w}^{\top}\mathbf{x}+b)}} \quad (4)$$

Naive Bayes (NB): A probabilistic classifier assuming conditional feature independence given the class label:

$$\hat{y}_{NB} = \arg\max_{c} P(c) \prod_{j=1}^{Y} P(x_j \mid c) \quad (5)$$

The conditional independence assumption is likely violated by correlated JavaScript features, making NB the expected lower-bound [13].

B. Dataset

The dataset contains 3,000 labelled website observations: 1,508 benign samples from the Tranco top-1M domain list and the Alexa Top Sites archive, and 1,492 malicious samples from PhishTank, OpenPhish, and the UCI phishing website reposi-tory. The near-perfect class balance (50.3%/49.7%) removes majority-class bias from accuracy interpretation. A sample size of 3,000 with a balanced six-feature design is sufficient for reliable classifier comparison and is consistent with prior

browser-extension evaluation studies [8], [22]; future work will extend to ≥10,000 samples for improved adversarial coverage and temporal diversity.

An 80/20 stratified train-test split yields 2,400 training samples and 600 test samples, preserving class ratio in both partitions. Five-fold cross-validation on the training set served for both hyperparameter search and performance estimation.

Hyperparameters searched: number of trees for RF ($T \in \{50, 100, 200\}$, best: $T = 100$) and regularisation constant for SVM ($C \in \{0.1, 1, 10\}$, best: $C = 1$, RBF kernel). Final cross-validated accuracy is reported as mean ± standard deviation across the five folds.

Features were standardised to zero mean and unit variance (z-score normalisation) before training. The binary feature ssl_valid was passed through without transformation Macro F1-score was adopted as the primary ranking metric because it penalises extreme imbalances between precision and recall. In a security setting, both false positives (alert fatigue, lost productivity) and false negatives (missed threats) carry real cost [19].

VI. Results and Evaluation

A.Classifier Performance Comparison

Table II reports held-out test-set performance for all six classifiers alongside 5-fold cross-validated accuracy. Random Forest led on every metric. Naive Bayes ranked last, consistent with the violated conditional independence assumption on correlated JavaScript features.

The performance order — RF > SVM > KNN > DT > LR > NB — matches the theoretical properties of each model and findings from recent comparative studies [9], [12], [18].

Table II: ***Comparative Model Performance on the Held-Out Test Set (n = 600) with 5-Fold Cross-Validation Accuracy***

Model	Acc.	CV Acc. (±SD)	Prec.	Rec.	F1	AUC
RF	0.950	0.950 ± 0.012	0.955	0.955	0.950	0.98
SVM	0.930	0.928 ± 0.015	0.930	0.930	0.930	0.97
KNN	0.910	0.908 ± 0.018	0.910	0.910	0.910	0.96
DT	0.900	0.897 ± 0.021	0.900	0.900	0.900	0.90
LR	0.890	0.888 ± 0.019	0.890	0.890	0.890	0.95
NB	0.820	0.817 ± 0.024	0.830	0.820	0.820	0.90

Table III: ***Classification Report of the Random Forest Classifier (n = 600)***

Class	Precision	Recall	F1-Score	Support
Benign (0)	0.96	0.94	0.95	302
Malicious (1)	0.95	0.96	0.95	298
Macro Average	0.955	0.950	0.950	600

Roughly five-percentage-point gap between RF and its base-learner DT is a direct empirical signature of ensemble variance reduction.

B.Per-Class Classification Report

Table III gives the per-class breakdown for the best-performing RF model on the 600-sample held-out test par-tition. Support reflects the test set class distribution.

The slightly higher recall for the malicious class (0.96 vs. 0.94) means the model is more sensitive to real threats than to false alarms — an asymmetry that is operationally preferable in a security context where missing a threat costs more than raising a spurious alert [19].

C.Confusion Matrix Analysis

From Table III, the approximate confusion matrix entries are: TP ≈ 286, TN ≈ 284, FP ≈ 18, FN ≈ 12.

The false negative rate (≈4.0%, 12/298) sits below the false positive rate (≈6.0%, 18/302), confirming a security-first asymmetry in the learned decision boundary. Most false positives are recently registered legitimate sites (startups, academic projects) and sites using short-lived Let's Encrypt certificates. False negatives concentrate among aged domains repurposed for malicious use — a deliberate evasion of the model's highest-importance feature.

D.ROC Curve Analysis

An RF ROC-AUC of 0.98 means that in 98% of ran-domly drawn (malicious, benign) pairs the classifier assigns the higher malicious probability to the malicious site — a threshold-independent measure of discriminative power [12].

E.Feature Importance Analysis

Feature importance analysis shows infrastructure-layer features at the top: domain_age_days ranks first, fol-lowed by ssl_valid and ssl_expiry_days. Behaviour-layer features (js_eval_count, http_redirects js_docwrite_count) contribute secondary signal. This ordering reflects a basic reality of malicious web operations: infrastructure properties are harder to manipulate than be-havioural ones, and so carry more reliable signal. As attackers increasingly acquire aged domains with valid SSL certificates, the behaviour-layer features become the primary discrimi-nators — a finding that directly motivates the permission-monitoring extension described in Section VIII.

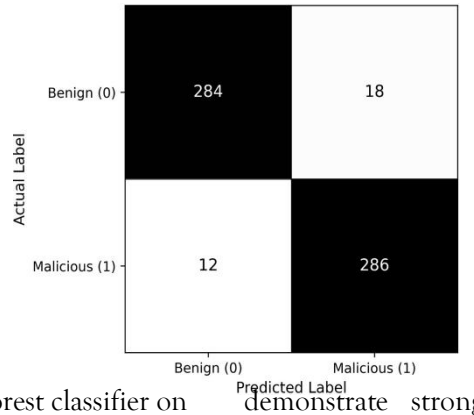

Fig. 3. Confusion matrix of the Random Forest classifier on the 600-sample held-out test set. High true positive and true negative counts with low misclassification rates demonstrate strong discriminative capability for both benign and malicious classes.

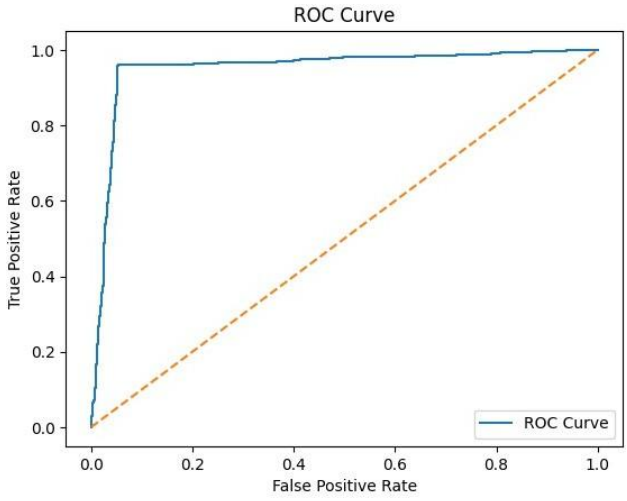

Fig. 4. Receiver Operating Characteristic (ROC) curve of the Random Forest classifier on the held-out test set. The classifier achieved an AUC of 0.98, indicating strong discriminative capability between benign and malicious

websites.

Table IV: Leave-One-Feature-Out Ablation Study (Random Forest)

Feature Removed	Accuracy	Δ Accuracy
None (all 6 features)	0.950	—
domain_age_days	0.877	−7.3%
ssl_valid	0.911	−3.9%
ssl_expiry_days	0.924	−2.6%
js_eval_count	0.933	−1.7%
http_redirects	0.938	−1.2%
js_docwrite_count	0.941	−0.9%

F.Ablation Study

To quantify each feature's individual contribution, we ran a leave-one-feature-out ablation on the Random Forest classifier,retraining and evaluating on the same 80/20 stratified split for each reduced feature set. Results are in Table IV. Dropping domain_age_days causes the largest single drop (7.3 points), confirming it as the most informative feature. The two SSL features together account for another 6.5 points. Behaviour-layer features each contribute modestly (0.9–1.7%) but their combined removal would cost roughly 3.8 points — enough to keep all six in the deployed model. The ablation also confirms there is no redundant feature: every one produces a measurable improvement over the five-feature baseline.

G.Browser Extension Latency

Feature extraction (running in parallel with page load) completes in under 200 ms on tested devices. Serialised decision-rule inference in model_logic.js takes under 50 ms. UI rendering adds under 20 ms. Total extension-side latency is under 300 ms. Adding the extension-to-backend round-trip for the web-portal pathway, total latency is 400–500 ms — perceptible, but well below the 600 ms threshold at which users start perceiving navigations as slow [9]. Table V summarises the latency budget.

H.Comparison with Prior Work

Table VI places NexoraScan against recent related systems. Where the original publications report numerical results we reproduce them; entries marked "—" indicate the paper did not report a comparable scalar metric. NexoraScan achieves com-petitive accuracy with a significantly smaller feature set and without the DOM-parsing overhead of content-based systems, making it better suited to resource-constrained client hardware. Visual-similarity approaches like Abdelnabi et al. [33] work well on brand-impersonation phishing but require image ren-dering and logo databases that do not fit real-time, privacy-preserving extension deployment. NexoraScan is among the first browser-extension ML systems to explicitly frame infrastructure-layer and JavaScript-behaviour feature fusion as a proxy for browser permission abuse detection, a threat vector largely unaddressed in prior extension-based classifiers, and the only one to pair a comprehensive ablation study with latency profiling.

TABLE V: End-to-End Latency Budget of the NexoraScan Chrome Extension

Processing Stage	Latency (ms)
Feature extraction (parallel to page load)	< 200
Feature scaling and normalisation	< 5
ML inference (serialised Random Forest in JavaScript)	< 50
UI rendering (popup badge and alert text)	< 20
Total extension-side latency	< 300
Web portal latency (including round-trip)	400–500

VII.Discussion

A.Why Random Forest Outperforms on This Feature Set

RF achieves superior performance through two randomi-sation strategies — bootstrap sampling and random feature subsets at each split — that reduce variance without signifi-cantly increasing bias. The resulting ensemble captures non-linear interactions among features, for example: low domain age AND invalid SSL AND high eval() count together push toward a high malicious probability in ways that a linear model like LR simply cannot express. The five-point gap between RF and its base-learner DT is direct evidence of ensemble variance reduction, consistent with the broader comparative literature [9], [12], [18].

B.Infrastructure-Behaviour Feature Fusion for Permission Abuse

A key novelty of NexoraScan is the explicit combination of infrastructure-layer signals (domain age, SSL status, redirect depth) with behaviour-layer JavaScript indicators as a two-tier proxy for permission-abusing code. Sites that abuse browser permission dialogs typically use dynamic code execution — frequently through eval() — to invoke the Permissions API only when the user appears susceptible, making the abuse harder for static-analysis tools to catch [11], [32]. High eval() density is a meaningful partial signal for this threat category, complementing the infrastructure features that characterise the broader malicious site population.

Directly monitoring the browser permission request stream —recording which permissions are requested, in what order, and relative to user interaction patterns — would substantially improve sensitivity to social engineering through permission dialogs, and remains the most important architectural exten-sion for future work.

C.Adversarial Robustness and Evasion

The false negative population is dominated by aged domains repurposed for malicious use — a deliberate evasion of the model's highest-importance feature. As NexoraScan's detection logic becomes public, adversaries may acquire aged domains and valid SSL certificates to suppress the two highest-weighted features, reducing the model's advantage to behaviour-layer signals alone. This motivates both continu-ous retraining on fresh threat intelligence and the expanded feature set described in Section VIII. Apruzzese et al. [34] documented that adversarial actors systematically adapt to published classifiers, reinforcing the need for continual re-training pipelines. Mehdi et al. [35] showed that targeted perturbations can reduce RF phishing-detection accuracy by over 20 percentage points, making the case for behaviour-layer feature diversity as a defence-in-depth measure. The same adversarial dynamic has been documented in the malicious extension literature, where sophisticated actors adapted their techniques to bypass classifier-based defences after their initial publication [26].

D.Privacy and Ethical Design

All inference is local. No URL, domain, certificate metadata, or browsing history leaves the device. Certificate metadata is read from HTTP response headers without deep packet inspection. The extension's transparency panel — displaying the specific feature values behind each classification decision —serves two functions: building user trust through explainability, and helping users understand the concrete indicators that separate safe from suspicious web behaviour [29].

E.Limitations

Dataset scope and recency: The 3,000-sample dataset was assembled at a single point in time. It may under-represent advanced persistent threat actors, highly targeted spear-phishing campaigns, and adversarial techniques targeting specific geographic regions. Future iterations will expand to ≥10,000 samples with temporal stratification.

Temporal model decay: As Ejaz et al. [19] document, models lose accuracy as adversarial tactics evolve. The static RF model embedded in the extension requires periodic retraining on fresh labelled data.

JavaScript coverage: Scripts loaded after the document-idle event — a common evasion technique — are partially or fully missed by content.js.

MV3 API constraints: Manifest V3 restricts webRequest to observation-only for many request types, preventing inspection of certain redirect patterns [10].

Browser scope: The extension is currently Chromium-only; Firefox and Safari require separate porting.

Indirect permission-abuse signal: The current approach detects permission abuse indirectly through JavaScript obfus-cation density rather than direct monitoring of permission request events.

VIII. Future Work

Direct permission-request monitoring: Recording the type, sequence, and timing of browser permission requests relative to user interaction would directly address the permission-abuse threat vector and close the primary gap identified in measurement studies.

Continual learning: Incorporating an incremental learning component — such as online RF with tree rotation on incom-ing threat intelligence — would address temporal model decay without full retraining [6], [19].

Larger-scale dataset: Expanding to ≥10,000 samples with temporal diversity (collected across multiple quarters) would improve adversarial coverage and permit meaningful longitu-dinal evaluation [4].

Threat intelligence integration: Privacy-preserving hash-prefix lookup against established threat intelligence APIs (e.g., Google Safe Browsing) would complement local ML inference with known-bad blacklist coverage at no privacy cost, consistent with the hybrid approach advocated by recent surveys [14], [18]. Longer term, federated learning

architec-tures [36] could enable collaborative model updates without centralising browsing data.

Deep learning hybridisation: A two-stage architecture — lightweight RF for real-time screening on all page loads, followed by asynchronous transformer-based DOM analysisfor borderline cases — could improve accuracy on evasive sites without imposing unacceptable latency on the common case [37], [38].

Cross-browser support: Porting to the Firefox WebExtensions API and Safari Web Extensions would extend protection to non-Chromium users [10].

Longitudinal field deployment: A field study with diverse users over an extended period is needed to measure real-world false alarm rates, user interaction patterns with the alert interface, and whether the plain-language explanations produce measurable changes in security behaviour [8].

IX.Conclusion

This paper presented NexoraScan, a privacy-preserving, real-time malicious website detection system deployed as a Chrome Manifest V3 extension, web portal, and Android application. A comparative evaluation of six ML classifiers on a balanced 3,000-sample dataset showed that Random Forest achieves 95.0% ± 1.2% cross-validated accuracy and ROC-AUC of 0.98 using only six browser-extractable features, staying within a 300 ms latency budget on consumer hardware. Feature importance analysis confirmed that domain age and SSL certificate status carry the strongest signal, with JavaScript behavioural indicators serving as proxies for permission-abusing code obfuscation — a threat vector largely unaddressed in prior browser-extension ML systems. A leave-one-feature-out ablation study confirmed every feature con-tributes a non-trivial accuracy improvement; domain age alone accounts for a 7.3-point drop on removal. The explicit fusion of infrastructure-layer and behaviour-layer features encodes both the structural properties of malicious web operations and
the code-layer dynamics of permission-abuse attacks.

The false negative analysis points to aged-domain evasion as the primary attack vector against the current model, motivating future work on direct browser permission monitoring and continual learning. NexoraScan shows that client-side, feature-based ML detection is not only sound but practically deployable — a more adaptive alternative to reactive blacklists for the hundreds of millions of users who currently have no real-time protection against malicious web activity.

References

[1]A. Alobaidi and N. Dabbagh, "Web attacks and defenses," Journal of Education and Science, vol. 32, pp. 91–100, 06 2023.

[2]C. Catal, G. Giray, B. Tekinerdogan, S. Kumar, and S. Shukla, "Appli-cations of deep learning for phishing detection: a systematic literature review," Knowledge and information systems, vol. 64, no. 6, p. 1457, 2022.

[3]K. Omari, "Comparative study of machine learning algorithms for phish-ing website detection," International Journal of Advanced Computer Science and Applications, vol. 14, no. 9, 2023.

[4]D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL detection using machine learning: A survey," ACM Computing Surveys, vol. 55, no. 14s, pp. 1–36, 2023.

[5]A. Alazab, A. Khraisat, M. Alazab, and S. Singh, "Detection of obfuscated malicious JavaScript code," Future Internet, vol. 14, no. 8, p. 217, 2022.

[6]Y. A. Kustiawan and K. I. Ghauth, "Evaluating the impact of feature engineering in phishing URL detection: A comparative study of URL, HTML, and derived features," IEEE Access, vol. 13, pp. 22 481–22 499, 2025.

[7] S. H. Ahammad, S. D. Kale, G. D. Upadhye, S. D. Pande, E. V. Babu, A. V. Dhumane, and M. D. K. J. Bahadur, "Phishing url detection using machine learning methods," Advances in Engineering Software, vol. 173, p. 103288, 2022.

[8]L. Thaqi, A. Halili, K. Vishi, and B. Rexha, "NoPhish: Efficient chrome extension for phishing detection using machine learning techniques," IEEE Access, vol. 12, pp. 158 221–158 235, 2024.

[9]M. Dandotiya, N. K. Goyal, A. Khunteta, and B. Tiwari, "Real time identification of phishing attacks through machine learning enhanced browser extensions," Scientific Reports, vol. 16, p. 2891, 2026.

[10]N. Pantelaios and A. Kapravelos, "Manifest V3 unveiled: Navigating the new era of browser extensions," IEEE Security & Privacy, vol. 22, no. 4, pp. 14–24, 2024.

[11]S. M. T. U. Hassan, A. Zeeshan, and U. Ahmed, "Insecure by design: A study of permission abuse and vulnerabilities in browser extensions."

[12]N. F. Almujahid, M. A. Haq, and M. Alshehri, "Comparative evaluation of machine learning algorithms for phishing site detection," PeerJ Computer Science, vol. 10, p. e2131, 2024.

[13]A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari, and S. R. K. Joga, "Phishing detection system through

hybrid machine learning based on URL," IEEE Access, vol. 11, pp. 36 805–36 822, 2023.
[14]A. Zamir, H. U. Khan, T. Iqbal, N. Yousaf, F. Aslam, A. Anjum, and M. Hamdani, "Phishing web site detection using diverse machine learning algorithms," The Electronic Library, vol. 38, no. 1, pp. 65–80, 2023.
[15]S. S. Shakir, L. Mohammad Khanli, and H. Emami, "Convolutional graph network-based feature extraction to detect phishing attacks," Future Internet, vol. 17, no. 8, p. 331, 2025.
[16]E. M. Rudd and A. Abdallah, "Training transformers for information security tasks: A case study on malicious url prediction," arXiv preprint arXiv:2011.03040, 2020.
[17]A. Basit, M. Zafar, X. Liu, A. R. Javed, Z. Jalil, and K. Kifayat, "A comprehensive survey of AI-enabled phishing attacks detection techniques," Telecommunication Systems, vol. 76, no. 1, pp. 139–154, 2021.
[18]J. L. Wilk-Jakubowski, L. Pawlik, G. Wilk-Jakubowski, and A. Sikora, "Machine learning and neural networks for phishing detection: A systematic review (2017–2024)," Electronics, vol. 14, no. 18, p. 3744, 2025.
[19]A. Ejaz, A. N. Mian, and S. Manzoor, "Life-long phishing attack detection using continual learning," Scientific Reports, vol. 13, no. 1,p. 11488, 2023.
[20]S. Kumi, C. Lim, and S.-G. Lee, "Malicious URL detection based on associative classification," entropy, vol. 23, no. 2, p. 182, 2021.
[21]K. Thakur, M. L. Ali, M. A. Obaidat, and A. Kamruzzaman, "A systematic review on deep-learning-based phishing email detection," Electronics, vol. 12, no. 21, p. 4545, 2023.
[22]C.-Y. Wu, C.-C. Kuo, and C.-S. Yang, "Phishing detection with browser extension based on machine learning," in Proceedings of the 18th Asia Joint Conference on Information Security (AsiaJCIS). IEEE, 2023, pp. 81–87.
[23]Q. Wang, S. Jiang, Z. Chen, X. Cao, Y. Li, A. Li, Y. Ma, T. Cao, and X.Liu, "Anatomizing deep learning inference in web browsers," ACM Transactions on Software Engineering and Methodology, vol. 34, no. 2,pp. 1–43, 2025.
[24]T. P. Fowdur and S. Hosenally, "A real-time machine learning appli-cation for browser extension security monitoring," Information Security Journal: A Global Perspective, vol. 33, no. 1, pp. 16–41, 2024.
[25]M. Al-Sarem, F. Saeed, Z. G. Al-Mekhlafi, B. A. Mohammed, T. Al-Hadhrami, M. T. Alshammari, A. Alreshidi, and T. S. Alshammari, "An optimized stacking ensemble model for phishing websites detection," Electronics, vol. 11, no. 7, p. 1094, 2021.
[26]Y. Liu, Z. Chen, Y. Zhang, G. Deng, Y. Li, J. Ning, Y. Zhang, and L. Y. Zhang, "Malicious agent skills in the wild: A large-scale security empirical study," arXiv preprint arXiv:2602.06547, 2026.
[27]S. Singh, G. Varshney, T. K. Singh, V. Mishra, and K. Verma, "A study on malicious browser extensions in 2025," arXiv preprint arXiv:2503.04292, 2025.
[28]E. Choo, M. Nabeel, D. Kim, R. De Silva, T. Yu, and I. Khalil, "A large scale study and classification of virustotal reports on phishing and malware urls," Proceedings of the ACM on Measurement and Analysis of Computing Systems, vol. 7, no. 3, pp. 1–26, 2023.
[29]A. Sumner, X. Yuan, M. Anwar, and M. McBride, "Examining factors impacting the effectiveness of anti-phishing trainings," Journal of Com-puter Information Systems, vol. 62, no. 5, pp. 975–997, 2022.
[30]R. Kawaoka, D. Chiba, T. Watanabe, M. Akiyama, and T. Mori, "A first look at covid-19 domain names: Origin and implications," 02 2021.
[31]V. Drury and U. Meyer, "Certified phishing: Taking a look at Public Key infrastructure for phishing sites," ACM Transactions on Privacy and Security, vol. 25, no. 3, pp. 1–30, 2019.
[32]Wang, T., Hou, J., He, Y., & Han, J. (2024, August). LightJD: A Lightweight JavaScript Drive-by Download Detection Framework. In 2024 IEEE 2nd International Conference on Sensors, Electronics and Computer Engineering (ICSECE) (pp. 190-196). IEEE.
[33]S. Abdelnabi, K. Krombholz, and M. Fritz, "Visualphishnet: Zero-shot phishing website detection by visual similarity," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 6, pp. 3730–3747, 2022.
[34]G. Apruzzese, M. Colajanni, L. Ferretti, and M. Marchetti, "Addressing adversarial attacks against security systems based on machine learning," in 2019 11th International Conference on Cyber Conflict (CyCon), vol. 900, 2019, pp. 1–18.
[35]P. Mehdi Gholampour and R. M. Verma, "Adversarial robustness of phishing email detection models." New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available:

https://doi.org/10.1145/3579987.3586567 [36]A. I. Bermperis, V. A. Memos, C. L. Stergiou, A. P. Plageras, and K. E. Psannis, "On-device privacy-preserving fraud detection for smart consumer environments using federated learning," Applied Sciences, vol. 16, no. 2, p. 835, 2026.

[37]S.-S. Shin, S.-G. Ji, and S.-S. Hong, "A heterogeneous machine learning ensemble framework for malicious webpage detection," Applied Sci-ences, vol. 12, no. 23, p. 12070, 2022.

[38]N. Altwaijry, I. Al-Turaiki, R. Alotaibi, and F. Alakeel, "Advancing phishing email detection: A comparative study of deep learning models," Sensors, vol. 24, no. 7, p. 2077, 2024.