COMPUTER SCIENCE DEPARTMENT

Cybersecurity and AI

Multi-Layered Phishing Detection Algorithm as a Browser Extension

---

Authors:
Almog Zemach 205789001
Tamir Houri 205668627

May. 2025

Contents

1 Introduction

Phishing attacks exploit users' trust to steal sensitive information through deceptive websites. As attackers adopt sophisticated evasion techniques—such as realistic URLs and dynamically loaded content—traditional detection methods often fail to provide timely protection.

This project introduces a *multi-layered phishing detection system* implemented as a *Chromium browser extension*, combining two client-side classifiers:
- **URL classifier** using a TensorFlow.js model trained on lexical features.
- **Content classifier** using Logistic Regression on static HTML/DOM features.

This layered detection strategy improves accuracy without compromising performance, offering fast response times and minimal memory overhead. The following sections detail the system's architecture, related methodologies, experimental results, and potential enhancements.

2 Related Work

This section surveys both academic and non-academic approaches to phishing detection, organized by detection source (URL vs. content/DOM). For each approach we describe two representative techniques and briefly reflect on their strengths and limitations.

2.1 URL Detection

Lexical-Feature Machine Learning

Extracts features directly from the URL string (length, number of dots, presence of suspicious tokens like "@" or IP addresses, character distribution) and feeds them into supervised classifiers (e.g., Random Forest, SVM) [1].

Reflection:
- *Pros:* Can detect zero-day phishing URLs by learning patterns of obfuscation; low overhead to compute features [2].
- *Cons:* Easily evaded by adversaries who tweak URL syntax; high feature correlation can lead to overfitting on training sets.

Host and WHOIS Feature Analysis

Augments lexical features with domain registration and hosting data (domain age, registrar reputation, DNS record TTL, geo-location). Models combine these with URL features in ensemble learners [3].

Reflection:
- *Pros:* Catches long-lived or suspiciously registered domains; raises the barrier for phishers who rapidly churn domains.
- *Cons:* Requires external lookups (WHOIS/DNS), adding latency; some hosting data can be spoofed or hidden via privacy services.

Blacklist Services

Queries a maintained database of known phishing URLs (e.g., PhishTank, Google Safe Browsing). If the URL appears, it is blocked or flagged [4, 5].

Reflection:
- *Pros:* Very low false-positive rate for entries in the list; simple to implement and update.
- *Cons:* Ineffective against new (unreported) phishing URLs; dependent on community reporting and update speed.

2.2 Content Detection

Content-Based TF-IDF Scoring

Computes TF-IDF scores for terms in the page body, queries a search en-

gine for each term, and interprets low search-volume terms as suspicious (phishing tends to use invented or obscure strings) [6].

Reflection:
- *Pros:* Leverages the "wisdom of the crowd" via search-engine backing; effective at spotting novel, content-light phishing pages.
- *Cons:* Relies on external search services (introduces latency); ineffective if page content is minimal or copied verbatim from the target site.

Heuristic DOM Analysis

Inspects the page's DOM tree for suspicious constructs (e.g., `<form>` elements posting to external domains, hidden iframes, excessive inline scripts) [7].

Reflection:
- *Pros:* Fast, runs entirely in-browser; no external services required.
- *Cons:* Phishers can obfuscate or dynamically inject scripts to evade static DOM checks; rules must be continuously updated.

Keyword-Based Content Scanning

Searches page text for phishing-related keywords ("login," "verify," "password," "bank," "secure"), often combined with scoring thresholds [8].

Reflection: - *Pros:* Simple to implement and tune; effective at catching naïve phishing pages that overtly solicit credentials.
- *Cons:* Generates false positives on legitimate pages using the same keywords; blind to sophisticated attacks that minimize textual cues.

3 Method

3.1 Architecture and Flow

The phishing detection system is implemented as a Chromium browser extension with a two-pronged detection approach: URL-based and static content-based analysis (See Fig. A.1). Both classifiers are executed simultaneously when the user opens the extension popup, allowing for interactive on-demand phishing evaluation.

The decision logic is structured as follows:

- If both classifiers return the same prediction, that result is shown to the user.
- If the classifiers disagree, a meta-decision layer—a logistic regression model trained on their outputs—is used to break the tie based on learned weights and a threshold.

3.2 URL-based Classifier

The URL-based classifier is a TensorFlow.js model converted from a Keras neural network trained in Python. It operates on 16 lexical and structural features extracted from the current page URL, including:

Length-based: Total URL length, subdomain length, main domain length, and path length.

Character-based: Count of dots, hyphens, digits, and encoded characters.

Structural: URL path depth, token count, number of subdomains.

Semantic/Heuristic: Presence of HTTPS, use of URL shortening services, uncommon top-level domains, inclusion of suspicious keywords, and brand name indicators.

For our URL dataset features correlation, see A.2.

To train this model, we leveraged the DeepURLBench dataset introduced by Schvartzman *et al.* [9], which comprises over 10 million URLs labeled as benign and phishings. This dataset is particularly well-suited for our task because its large scale ensures exposure to diverse URL patterns, its balanced class distribution prevents bias toward the majority class, and its rigorous cleansing process guarantees high-quality labels—altogether enabling our model to generalize robustly to real-world phishing scenarios. The model architecture consists of:

$$\text{Input} \xrightarrow{\text{Norm}} D_{64}(\text{ReLU}) \rightarrow D_{32}(\text{ReLU}) \rightarrow D_{1}(\text{Sigmoid})$$

The final classification threshold was tuned using ROC curve analysis. The model has 3,234 total parameters and uses a validation split during training.

3.3 Content-based Classifier

The content-based classifier is a Logistic Regression model operating on static HTML and DOM features extracted after page load. The features include:

Suspicious Forms: Forms with phishing-related actions or cross-domain submissions.

Password Fields Without HTTPS: Password fields on non-secure pages.

Mismatched Link Text: Links with deceptive or misleading text.

Obfuscated JavaScript: Use of `eval()`, `document.write()`, or dynamic script injection.

External Logos: Images loaded from untrusted external domains.

Too Many Input Fields: Forms with an unusually high number of inputs.

For our content dataset feature correlation, see A.3.

To evaluate our content-based detector, we used the Website subset of the HuggingFace "Phishing Datasets" by Alvarado [10], which comprises approximately 80,000 static HTML pages (50,000 legitimate and 30,000 phishing). This subset offers a large, balanced, and well-labeled collection of real-world web pages—complete with forms, scripts, and link structures—making it an ideal benchmark for assessing content-based phishing detection.

Each content feature contributes a bounded score between 0 and 1, enabling stable probability estimation. The Logistic Regression model computes a sigmoid over a weighted sum of features and bias. Model thresholds were tuned using ROC analysis, and generalization performance was validated through 5-fold cross-validation.

3.4 Stacking Classifier

In cases where the two classifiers disagree, a lightweight logistic regression model aggregates their outputs using learned weights and a bias term. This model was trained on a separate validation set to minimize classification error and ensure more accurate decision.

4 Results

The performance of both detectors was measured on a test set, evaluating their accuracy, TPR, FPR, F1 score and real-time efficiency (See Table. A).

1. URL-Based Detector

The URL-based detector leverages features such as domain entropy, URL length, and special character counts to predict phishing behavior. These metrics

indicate a high detection rate with minimal false positives, confirming the effectiveness of URL-based features in identifying phishing patterns.

2. Content-Based Detector

The content-based detector analyzes static DOM structure features such as suspicious form actions, mismatched anchors, and inline scripts. After threshold tuning (optimal threshold = 0.357) performed in a Jupyter notebook, the model achieved reasonable precision and recall. Despite a higher false positive rate, it complements the URL-based model by detecting structurally deceptive websites.

4.1 System Performance

Detection Latency

From the `DOMContentLoaded` event to classification output: 15.20 ms

Fine-Grained performance metrics:
- **URL Detection Time**: 5.40 ms
- **Content Detection Time**: 2.70 ms

Both detectors demonstrated excellent responsiveness suitable for real-time browser execution.

Memory Footprint

We profiled the extension's runtime memory usage using Chrome's built-in Task Manager. During active phishing detection, the observed memory footprint was approximately **28.4 MB**. This level of consumption is well within acceptable bounds for real-time browser extensions.

5 Limitations

While the proposed multi-layered detection system provides effective real-time protection against a wide range of phishing attacks, several limitations remain:

Lack of Dynamic Analysis: The system performs only static URL and DOM inspection. It cannot detect phishing pages that rely on dynamic behavior, such as delayed script execution or content injection after DOM readiness.

Domain and Brand Generalization: The current URL classifier relies on known brand indicators and lexical patterns, which may not generalize well to novel phishing domains targeting lesser-known services.

Heuristic Sensitivity: Some content features rely on heuristic thresholds (e.g., number of input fields), which may produce false positives on complex legitimate sites.

6 Future Work

Incorporating Dynamic Behavior Analysis: Future versions could monitor runtime behaviors such as redirects, DOM mutations, and asynchronous JavaScript execution to detect evasive threats.

Model Enhancement with Sequence Learning: Replacing the current URL classifier with an LSTM or Transformer-based model could improve detection of sophisticated domain name patterns and sequential structures.

Improved Feature Robustness: Feature engineering can be expanded to include DOM tree structure patterns, script complexity metrics, and visual similarity comparisons to known brands.

User Feedback Loop: Integrating user feedback on false positives/negatives could help refine feature weights and personalize detection to user behavior over time.

A Appendix

System design

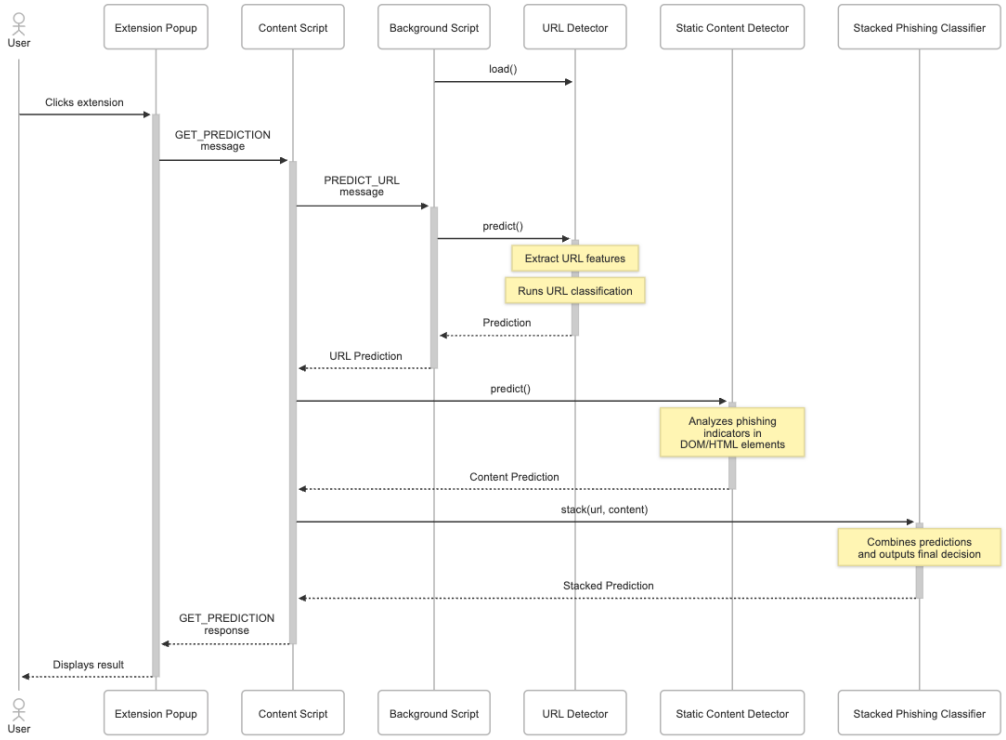

Figure A.1: Sequence diagram of the phishing detection extension.

Detectors Performance

Table A.1: Performance metrics for each detector

Detector	Accuracy	TPR	FPR	F1 Score
URL-based	0.91	0.87	0.08	0.87
Content-based	0.82	0.81	0.18	0.75
Stacked	0.85	0.86	0.11	0.82

Feature Engineering

URL-Detector

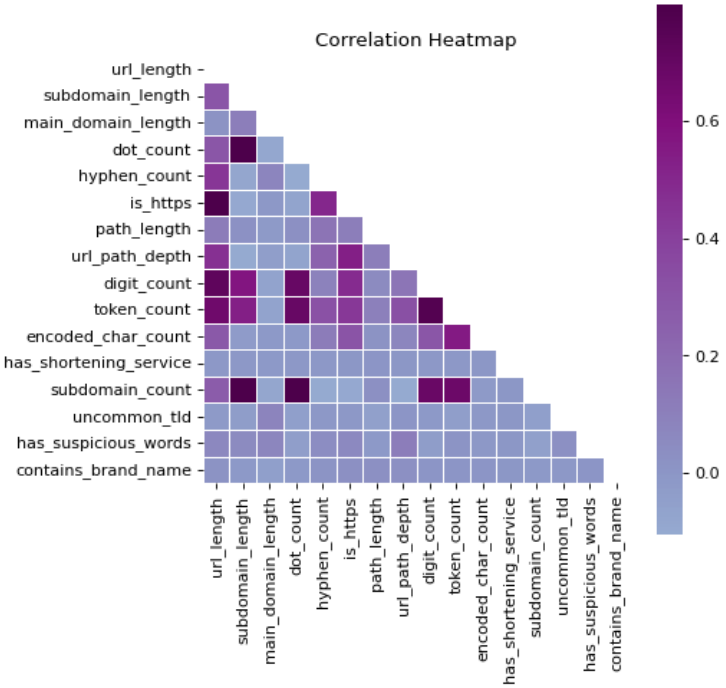

Figure A.2: Feature correlation heatmap for URL-based classifier.

Content-Detector

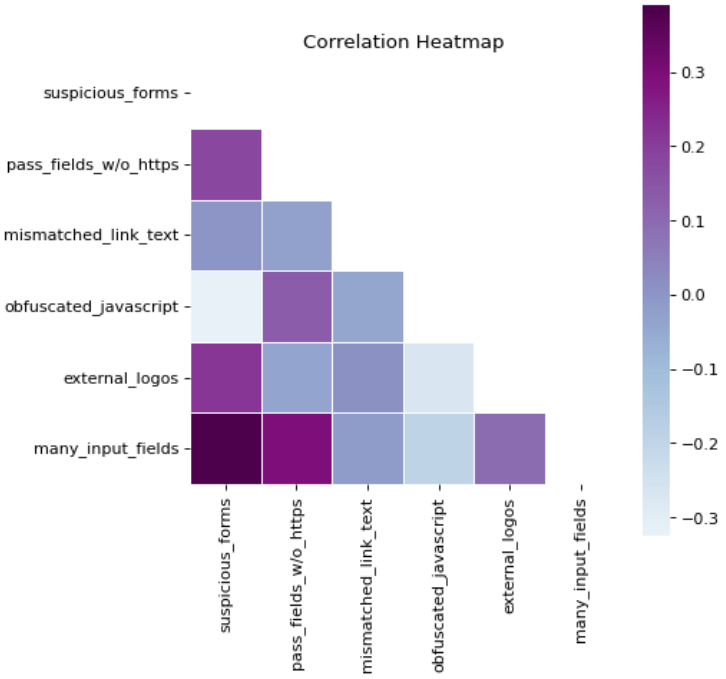

Figure A.3: Feature correlation heatmap for content-based classifier

References

[1] S. Mohammad, A. Thabtah, and A. McCluskey, "Predicting Phishing Websites Based on Self-Structuring Neural Network," *Knowledge-Based Systems*, vol. 108, pp. 72–83, 2016.

[2] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A Framework for Detection and Measurement of Phishing Attacks," in *Proceedings of the 2007 ACM Workshop on Recurring Malcode*, 2007.

[3] M. Alsharnouby, G. Toterosius, and F. Thieltges, "User Awareness of Phishing Attacks: A Literature Review," *International Journal of Computer Applications*, vol. 116, no. 14, 2015.

[4] Google, "Google Safe Browsing API," 2025. [Online]. Available: https://developers.google.com/safe-browsing/.

[5] OpenDNS, "PhishTank," 2025. [Online]. Available: https://www.phishtank.com/.

[6] I. Fette, N. Sadeh, and A. Tomasic, "Learning to Detect Phishing Emails," in *Proceedings of the 16th International Conference on World Wide Web*, 2007.

[7] Y. Zhang, S. Egelman, L. Cranor, and J. Hong, "Phinding Phish: Evaluating Anti-Phishing Tools," in *Proceedings of the 14th International Conference on Financial Cryptography and Data Security*, 2011.

[8] R. Kumaraguru, Y. Rhee, and A. Acquisti, "Protecting Users Against Phishing Attacks with Anti-Phishing Workbench," in *Proceedings of the 2009 IEEE Symposium on Security and Privacy*, 2009.

[9] I. Schvartzman, R. Sarussi, M. Ashkenazi, I. Kringel, Y. Tocker, and T. Furman Shohet, "A New Dataset and Methodology for Malicious URL Classification," *arXiv preprint arXiv:2501.00356*, 2024. [Online]. Available: https://arxiv.org/abs/2501.00356.

[10] E.Alvarado, "Phishing Datasets," HuggingFace, 2024. [Online]. Available: https://huggingface.co/datasets/ealvaradob/phishing-dataset.