

Low-latency Malicious URL Detection based on Lexical and Statistical Features

Siyu Huang

Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China

Abstract: Malicious URLs remain a persistent threat because attackers can generate short-lived links faster than static blacklists can be updated. This paper presents a lightweight stateless detector that uses only URL lexical and statistical features, without webpage downloading, DNS probing, WHOIS lookup, or HTML rendering. Experiments on approximately 50,000 cleaned URLs assembled from PhishTank, CIC-Phishing2021/CIC-Bell-DNS2021, and Tranco show that the proposed LightGBM model achieves 98.64% Accuracy, an F1-Score of 0.9863, and 0.4165 ms feature-extraction-plus-inference latency on an Apple M4 device with 16 GB RAM. Under a stress test with 5% malicious traffic, it maintains 99.59% Precision, 96.60% Recall, 0.02% FPR, and PR-AUC of 0.9793. These results suggest that URL-only mathematical and linguistic signals can support low-latency browser, mobile-gateway, and edge-security screening, while domain-granularity and source-composition effects still require time-shifted and source-balanced deployment validation.

Keywords: Malicious URL Detection; Feature Engineering; Ensemble Learning; Random Forest; Lightweight Security.

1. Introduction

Malicious URLs are still a common way to start phishing and web-based fraud attacks because malicious domains and malicious links [1] can change more quickly than static reputation lists. Content-based detection and large neural models can increase coverage, but typically involve webpage retrieval, browser rendering, external queries, or high inference costs. This paper thus examines a stateless URL-only detector, which identifies simple lexical and statistical characteristics of the URL string such as length, depth of the URL path, morphology of the tokens, presence of sensitive words, character ratios, and entropy.

The study asks whether this URL-only pipeline can provide an effective first-line detector and whether RF and LightGBM offer a better accuracy-latency trade-off than a length-only baseline and a URLNet-lite CNN/RNN comparator under balanced, imbalanced, and source-bias control settings.

The contribution is intentional: the method is assessed as a lightweight plug-in, mobile gateway, or edge filter, but not as a replacement for webpage-content analysis, reputation systems, telemetry-based detection or production-grade defense-in-depth.

2. Related Work

Prior work on malicious URL and phishing detection spans blacklist defenses, webpage-content analysis, URL feature engineering, and neural representation learning. This section briefly positions the present study as a deployment-oriented URL-only method.

2.1. Phishing URL Detection and List-based Defenses

Blacklists and reputation systems are quick and easy to put in place but are unable to keep up with short-term phishing campaigns and constantly changing infrastructure. Survey and measurement studies demonstrate that one can identify suspicious sites using machine-learning URL features in addition to relying on a static list of sites [2,3], and phishing

systems and streaming analytics reveal the need to automatically classify phishing sites when the landing pages, hosts, and redirection structures continuously change during attacks [4–6]. Content-rich systems like CANTINA+ perform well if webpage evidence exists [7], but pre-click filtering of pages in browsers, mobile gateways, and edge devices is less appealing because of page fetching, rendering, and third-party queries.

2.2. Lexical/Statistical Features and Deep URL Representations

URL-only methods consider the URL string as the observable object. Previous studies harness URL length, token structure, special-character distribution, host morphology and online lexical features in order to quickly classify a phishing URL or webpage [8–11]. When benign samples are mostly domain-level records and malicious samples are full URLs, statistical descriptors like Shannon entropy compress evidence of character disorder into small numbers [12,13], but models may still overfit source artifacts. For example, deep URL models like URLNet or character-level CNN approaches leverage the representations of sequences directly from raw strings [14–16]. They capture some obfuscation motifs which may not be captured by handcrafted features but add training and inference complexity; therefore this paper uses a URLNet-lite CNN/RNN as a comparator, not as a full URLNet reproduction.

2.3. Lightweight Tabular Models for Stateless URL Detection

Tree-based tabular models remain attractive for lightweight security screening because they combine nonlinear decision boundaries with compact, interpretable feature vectors. Random Forest and LightGBM are therefore used as deployment-aware classifiers for the engineered URL descriptors [17,18]. Prior malicious-URL studies also show that carefully engineered URL features can support competitive machine-learning performance [11,19,20]. The remaining gap is not whether richer content or deep models

can help, but whether a stateless URL-only pipeline can provide a sufficiently accurate, fast, and compact first-line detector when external retrieval and heavyweight inference are undesirable.

3. Methodology

The proposed methodology is designed around a stateless detection assumption: every prediction must be produced from the URL string alone, without webpage downloading, HTML parsing, DNS probing, WHOIS lookup, screenshot generation, or browser rendering. This assumption deliberately restricts the available signal but makes the system suitable for low-latency edge screening. The pipeline consists of dataset construction, URL canonicalization, feature extraction, class-balancing on the training split, classifier construction, and feature selection for model compression. This restriction also keeps decisions inspectable, because each prediction can be traced to explicit lexical or statistical descriptors rather than hidden webpage content.

3.1. Data Collection and Preprocessing

The malicious class is created from two sources that complement each other. The first consists of verified phishing URLs taken from the PhishTank downloadable feeds that offer community-verified phishing URLs and metadata including verification and online status [21]. The second is the phishing-labelled component of the CIC-Bell-DNS2021 resources, designed to simulate the behaviour of benign, phishing, spam and malware domains under realistic traffic imbalance [22]. The benign class is drawn from the Tranco List, where the top 100,000 ranked domains are considered high-reputation benign candidates as Tranco is constructed as a manipulation-resistant, research-focused ranking of popular websites [23].

All raw entries are imported into Pandas DataFrames, normalized, and processed for feature extraction. Exact duplicate URLs are first eliminated during preprocessing, and then the URLs are canonicalized: lowercasing scheme and host, stripping leading and trailing whitespace, decoding percent-encoded sequences where safe to do so, removing empty fragments, and standardizing redundant slashes. Entries that do not have a valid scheme or host, malformed strings, non-HTTP(S) resources, local addresses and source records that are inactive or invalid are discarded. This cleaning step does not contradict stateless detection: it is used only to construct a reliable experimental corpus and does not require the operational classifier to visit a webpage or download page content.

3.1.1. Training-set Balancing and Class-prior Stress Testing

The measured corpus is class-balanced at 25,000 benign and 25,000 malicious samples. SMOTE [24] is applied only to the training split after the 80/20 split to keep the training pipeline robust when class proportions vary. No held-out, imbalanced-stress, or domain-control test rows are synthetically resampled; operational imbalance is assessed separately through the 5% malicious stress setting.

3.2. Feature Engineering

Feature engineering is the central component of the proposed lightweight detector. Rather than extract dense representations from the raw string of characters, the method compresses each URL to a small set of numerical descriptors

that are easily interpretable. These descriptors are used to identify the linguistic, structural and statistical anomalies typically added by phishing kits, URL obfuscation, redirect chains, and automatically-generated domains.

3.2.1. Lexical Features

Lexical features include URL length and the number of characters in the host and path, the number of tokens and subdomains, ratios of digits to alphabetic characters and special characters, the presence of an IP-literal and of shortener patterns, and the frequency of sensitive tokens like login, verify, account, banking, wallet, and payment. The deterministic features can be obtained independently of any network request and thus are applicable to browser or gateway screening.

3.2.2. Statistical Features and Shannon Entropy.

The statistical characteristics transform the unstructured text of the URL into numerical measurements of disorder, repetition and symbol distribution. Shannon entropy is the most notable feature in this group, and was initially formulated to measure the information uncertainty in a discrete source [12]. In a string that is a URL, the sample space is the alphabet of characters. If c_i is the count of character x_i and n is the total number of characters in the analyzed string, then $p(x_i) = c_i/n$ is the empirical probability of that character. The Shannon entropy of the URL is computed as follows:

$$H(X) = -\sum_i p(x_i) \log_2 p(x_i)$$

From a modelling perspective, it is a formula that turns any unstructured URL string into a scalar feature that quantifies the distribution of characters. Benign URLs typically include meaningful words, brand names or directory names, while malicious URLs can include DGA-generated domain names, long encoded payloads, and excessive amounts of hexadecimal characters. Thus, higher entropy means higher disorder at the character level and can be combined with other lexical features to identify suspicious URLs [13].

The entropy is calculated at a variety of granularities: the complete URL, the registered domain, the subdomain sequence, the path and the query string. Together with digit ratio, special-character ratio, vowel-consonant balance, repeated-character length, and token-length statistics, entropy forms a small vector that captures the mathematical texture of the original URL without using a sequence model.

3.3. Model Construction and Feature Selection

Two kinds of tree-based classifiers are built, namely Random Forest (RF) and LightGBM. RF is chosen due to the fact that bagged decision trees are robust to nonlinear interactions between different lexical elements, and also provide intrinsic scores for the importance of each feature [17]. LightGBM also has an efficient implementation of gradient-boosted decision trees with histogram-based learning, Gradient-based One-Side Sampling and Exclusive Feature Bundling, making it well suited to fast training and small model sizes on tabular features [18].

To ensure a fair comparison, the same preprocessed feature matrix is used for training the models. Hyperparameters, such as the depth of the tree, number of estimators, minimum leaf size, and learning rate, are not optimized indefinitely but are selected with a deployment-driven objective that prioritizes speed of inference, memory usage, and recall for malicious URLs. Hence, the classifier is not just a predictive model, but also a tool to identify which URL statistics are meaningful.

3.3.1. Correlation Matrix Pruning

A Pearson correlation matrix is computed among all numeric features prior to final training. Two features that have an absolute correlation higher than a threshold value (e.g., $|\rho| > 0.90$) are considered redundant. To remove repeated signals, for each pair of highly correlated features, the one with lower interpretability, higher extraction cost or lower validation importance is removed, which reduces the number of features in the model.

3.3.2. Tree-model Importance Interpretation and Ablation

Feature selection in the implemented pipeline is limited to Pearson correlation pruning. Tree-model feature importance is used only as post-hoc interpretability evidence, while the marginal value of feature groups is tested through the LightGBM ablation analysis reported in Section 4.3.

4. Experimental Setup and Results Analysis

This section reports the experimental protocol and measured results used to evaluate the proposed detector. The final dataset contains approximately 50,000 cleaned URL samples after integrating PhishTank, CIC-Phishing2021/CIC-Bell-DNS2021, and Tranco sources. All numerical values in the following tables were obtained from the local experimental run rather than estimated or illustrative values.

4.1. Experimental Environment and Datasets

The experiment was designed to run on a consumer edge-class device equipped with an Apple M4 chip and 16 GB RAM. Approximately 50,000 URL samples were selected after cleaning and deduplication, combining malicious samples from PhishTank and CIC-Phishing2021/CIC-Bell-DNS2021 with benign samples from the Tranco Top 100,000 list. The dataset was divided by stratified sampling into 80% training data and 20% testing data. SMOTE was applied only to the training set after feature extraction, while the test set remained untouched to preserve a within-corpus held-out estimate rather than a synthetic oversampling artifact.

The implementation stack consists of Python, Pandas for data cleaning, numerical feature extraction utilities for URL parsing and entropy calculation, and machine learning libraries for RF, LightGBM, and baseline deep neural models. The primary evaluation metrics are Accuracy, Recall, Precision, F1-Score, prediction latency per URL in milliseconds, and serialized model size in megabytes. Prediction latency is measured under a single-URL inference protocol to reflect real-time deployment rather than high-throughput batch scoring.

4.1.1. Dataset Composition and Source-Bias Considerations

After normalization and deduplication, the corpus contains 25,000 benign and 25,000 malicious URLs. Its source composition is uneven: PhishTank provides 24,405 malicious full URLs; CIC-Phishing2021/CIC-Bell-DNS2021 provides 595 phishing domains; CIC benign contributes 20,406 benign domains; and Tranco contributes 4,594 benign domains. Because benign records are often domain-level while PhishTank samples often include paths and queries, mean URL length differs substantially (21.09 benign versus 55.48 malicious characters). The length-only baseline is therefore also a source/granularity-bias warning, and the domain-only control in Section 4.3 is a bias check rather than proof of source invariance.

4.1.2. Reproducibility Configuration

Sampling uses `random_state = 42` and a stratified 80/20 train-test split. The extractor produces 35 descriptors, and Pearson pruning at $|\rho| > 0.90$ leaves 30 selected features. SMOTE is applied only to the training split. Baseline A uses four length features; RF, LightGBM, and URLNet-lite are trained under fixed deployment-aware settings. Latency is reported as model-side inference and, separately, single-URL preparation-plus-inference cost. Additional checks include a 5% malicious stress test, a domain-only source-bias control, five repeated stratified splits, and ROC-AUC/PR-AUC from continuous model scores. The accompanying reproduction environment specifies pandas 2.3.3, numpy 2.3.5, scikit-learn 1.8.0, imbalanced-learn 0.14.1, LightGBM 4.6.0, joblib 1.5.2, tldextract 5.3.0, and torch 2.9.1. Model artifacts, selected-feature lists, and generated CSV tables are retained as outputs, which reduces manual transcription error during manuscript updates.

4.2. Baseline Models

Two baseline groups are used to isolate the contribution of the proposed feature set and lightweight model design. Baseline A is a length-only model that uses only basic structural features, including URL length, hostname length, path length, and query-string length. This baseline tests whether simple length descriptors are sufficient for real-time detection or whether entropy and lexical-semantic indicators add measurable value.

Baseline B is a URLNet-lite CNN/RNN baseline inspired by URLNet-style sequence learning [14]. It is included as a compact high-capacity comparator rather than a full reproduction of the original URLNet system, allowing the experiment to test whether sequence learning improves detection quality enough to justify additional training and inference cost.

4.3. Results and Evaluation

Table 1. Classification performance comparison for measured experimental runs.

Model	Feature / Representation	Accuracy (%)	Recall (%)	F1-Score
Baseline A Length-only	Basic URL length features	96.80	94.78	0.9673
Proposed RF	Selected lexical-statistical URL features	98.74	97.68	0.9873
Proposed LightGBM	Selected lexical-statistical URL features	98.64	97.62	0.9863
Baseline B URLNet-lite CNN/RNN	Character-sequence CNN/RNN representation	98.75	97.88	0.9874

The evaluation results are reported in terms of Accuracy, Precision, Recall, F1-Score, false-positive rate (FPR), single-URL prediction latency, serialized model size, ROC-AUC, and PR-AUC. Accuracy measures global correctness, whereas Recall is particularly important because false negatives may expose users to credential theft or malware delivery. Precision, FPR, and PR-AUC are particularly important under class imbalance, where a small false-positive fraction can still produce many user-facing warnings.

Table 1 shows that the URLNet-lite CNN/RNN baseline has the highest balanced Accuracy, Recall, and F1-Score, but

its margin over LightGBM is small. The result suggests an accuracy ceiling for a compact sequence baseline, whereas later deployment-cost evidence favors LightGBM for latency and model size.

To inspect error types beyond aggregate scores, the balanced held-out test was also evaluated using a confusion matrix and false-positive indicators. TN and TP denote correctly classified benign and malicious URLs, respectively, whereas FP and FN denote false alarms and missed malicious URLs.

Table 2. Balanced held-out confusion matrix and false-positive indicators.

Model	TN	FP	FN	TP	Precision (%)	FPR (%)
Baseline A Length-only	4941	59	261	4739	98.77	1.18
Proposed RF	4990	10	116	4884	99.80	0.20
Proposed LightGBM	4983	17	119	4881	99.65	0.34
Baseline B URLNet-lite CNN/RNN	4981	19	106	4894	99.61	0.38

Table 2 shows that the tree-based models substantially reduce false positives compared with the length-only baseline. RF records only 10 false positives, LightGBM records 17, and the URLNet-lite CNN/RNN baseline records 19. LightGBM therefore keeps FPR at 0.34% while preserving 99.65% Precision, which is a more deployment-relevant signal than Accuracy alone.

Deployment cost is evaluated independently from classification quality because real-time URL filtering must make a decision before the browser initiates or completes a risky request. Table 3 preserves the model-side inference latency used in the original comparison; Table 6 later adds input-preparation time to prevent ambiguity about end-to-end URL processing cost.

Table 3. Deployment-cost comparison for measured experimental runs.

Model	Feature / Representation	Model-side Latency (ms)	Model Size (MB)	Deployment Interpretation
Baseline A Length-only	Length-only features	0.2209	0.0017	Minimal but weak feature scope
Proposed RF	Selected URL features	14.4515	17.6629	High accuracy with interpretable trees
Proposed LightGBM	Selected URL features	0.3995	1.7641	Best latency-accuracy trade-off
Baseline B URLNet-lite CNN/RNN	Character-sequence CNN/RNN	1.1939	0.1860	Accurate but slower sequence inference

The deployment distinction is shown in Table 3; RF is accurate, but it is much slower, and it takes 17.6629 MB of storage, while LightGBM takes only 0.3995 ms per URL and 1.7641 MB of storage. In this CPU-only scenario, LightGBM runs, on average, 2.99 times faster than URLNet-lite and over 36 times faster than RF.

To account for operational traffic imbalance, a 5% malicious stress setting was also evaluated using 9,500 unused Tranco benign URLs and 500 malicious URLs from the held-out split. The accuracies of RF and LightGBM were both 99.81%, and both achieved an FPR of 0.02%. The highest recall was 96.80% for URLNet-lite with 0.06% FPR. Although this is a positive outcome for a lightweight screening test, it is still a controlled stress test and not time-shifted production validation.

Lastly, a domain-only control experiment was conducted to check the source-bias threat. Before retraining, each URL was reduced to its scheme and host, duplicate domains were grouped together with a malicious-if-any labelling rule, and a balanced domain-level dataset was created. This control removes most evidence from paths and queries; therefore, any performance degradation indicates the degree of reliance on the full-URL structure and source composition.

Table 4 shows that performance remains above 95% after converting records to domain-level form, but the decline is meaningful: LightGBM drops from 98.64% to 96.52% Accuracy, and RF drops from 98.74% to 96.65%. The result supports the usefulness of full-URL signals while reinforcing the need for benign full-URL and time-shifted validation.

Threshold-independent evaluation further clarifies

performance. ROC-AUC measures global ranking quality, whereas PR-AUC is more informative for security screening

under imbalanced traffic because it directly reflects the precision-recall trade-off for the minority malicious class.

Table 4. Domain-only source-bias control experiment.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score	Accuracy Drop (pp)
Baseline A Length-only	95.37	97.83	92.79	0.9524	1.43
Proposed RF	96.65	99.00	94.26	0.9657	2.09
Proposed LightGBM	96.52	98.84	94.15	0.9644	2.12

Table 5. ROC-AUC and PR-AUC on balanced and imbalanced test settings.

Model	Balanced ROC-AUC	Balanced PR-AUC	5% ROC-AUC	5% PR-AUC
Baseline A Length-only	0.9896	0.9922	0.9886	0.9747
Proposed RF	0.9923	0.9949	0.9923	0.9797
Proposed LightGBM	0.9915	0.9945	0.9924	0.9793
Baseline B URLNet-lite CNN/RNN	0.9929	0.9952	0.9921	0.9792

Table 5 shows that all models maintain high ranking quality. The URLNet-lite CNN/RNN baseline obtains the highest balanced PR-AUC at 0.9952, while LightGBM remains close at 0.9945 and reaches 0.9793 PR-AUC under the 5% malicious stress test. The PR-AUC gap under imbalance is small, so the deployment decision should rely on the combined evidence from PR-AUC, FPR, latency, and model complexity rather than Accuracy alone.

The repeated-split results indicate that the main conclusion is stable rather than dependent on a single favorable random_state = 42 split. LightGBM obtains $98.60\% \pm 0.14\%$ Accuracy and PR-AUC of 0.9941 ± 0.0008 across five stratified splits. RF is similarly stable, while the URLNet-lite CNN/RNN baseline remains marginally higher in mean Accuracy but does not overturn the latency-centered deployment conclusion.

LightGBM ablation indicates that the full selected feature set improves Accuracy by 0.65 percentage points over length-only features. Removing only entropy features changes Accuracy by -0.01 percentage points, and removing entropy and statistical distribution features also changes Accuracy by only -0.01 percentage points. Thus, entropy is useful as part of the broader lexical-statistical feature set, but it is not the sole driver of detection performance.

Latency is finally decomposed into input preparation and model inference. For the engineered-feature models, input preparation refers to the current Python URL parsing and lexical-statistical feature extraction implementation; for the URLNet-lite CNN/RNN baseline, it refers to character encoding. These values are not optimized lower bounds, but they clarify what the reported latency does and does not include.

Table 6. Single-URL latency decomposition for the current Python implementation.

Model	Input Prep. (ms)	Model Inference (ms)	Total (ms)	Latency Scope
Baseline A Length-only	0.0033	0.2209	0.2242	minimal parser for four length features
Proposed RF	0.0170	14.4515	14.4685	full lexical-statistical feature extractor
Proposed LightGBM	0.0170	0.3995	0.4165	full lexical-statistical feature extractor
Baseline B URLNet-lite CNN/RNN	0.0019	1.1939	1.1957	character encoding plus neural forward pass

Table 6 shows that feature extraction adds only 0.0170 ms per URL for RF and LightGBM in the current implementation. Therefore, LightGBM’s total measured single-URL cost is 0.4165 ms, not merely the 0.3995 ms model-side inference time. This distinction should be preserved in deployment claims: the result supports sub-millisecond URL processing in the current Python implementation, while production browser or gateway implementations should still benchmark parsing, feature extraction, model loading, and runtime

integration together.

An anonymized error-pattern analysis was also conducted. The categories in Table 7 are diagnostic and may overlap; the table reports both URLs misclassified by any evaluated model and URLs misclassified by the final LightGBM model. Sanitized examples use placeholders such as <brand> and <domain> rather than real user-facing URLs.

The error analysis shows that the most persistent LightGBM failures are short phishing URLs and random or

high-entropy host patterns. HTTPS phishing and brand-token mimicry were misclassified by at least one evaluated model, but the final LightGBM model corrected these cases in the balanced held-out split. This supports the need for continued

adversarial evaluation: attackers can deliberately shorten malicious URLs, obtain HTTPS certificates, inject brand-like tokens, or use generated subdomains to move toward the benign side of the learned decision boundary.

Table 7. Anonymized diagnostic categories for misclassified URL patterns.

Pattern	Dom. Error	Any-model Count	LGBM Count	Interpretation
Short phishing URL	FN	261	116	Compact phishing domains or landing pages resemble benign domains and weaken length-based separation.
HTTPS phishing URL	FN	95	0	TLS presence is no longer sufficient evidence of benignness; models must avoid treating HTTPS as a trust signal.
Brand/account-token mimicry	FN	3	0	Authentication and finance words can make malicious URLs look like legitimate user workflows.
Random subdomain or high-entropy host	FP/FN	222	76	Generated service labels, CDNs, and malicious randomization create overlapping character-distribution signals.
Long benign URL	FP	4	2	Legitimate documentation, search, and application URLs can resemble long phishing paths.

Overall, URLNet-lite is slightly stronger on balanced quality metrics, RF gives the fewest false positives but is heavier, and LightGBM offers the best overall deployment trade-off: near-best accuracy and PR-AUC, 0.34% balanced FPR, 0.4165 ms total single-URL cost, and 1.7641 MB size. These findings support LightGBM as a first-line stateless screening candidate within this dataset, not as a replacement for content, reputation, telemetry, or time-shifted validation.

5. Threats to Validity and Limitations

The main validity threat is source composition: benign records are often CIC/Tranco domains, while many PhishTank samples are full URLs. The domain-only control reduces this mismatch, and accuracy remains above 95%, but the decline shows that source and URL-granularity effects remain. The balanced corpus and 5% malicious stress test also do not constitute production prevalence validation.

Other limitations include the absence of a strict time-shifted malicious split, limited benign full URLs from comparable channels, URLNet-lite rather than full URLNet, Python single-URL latency rather than integrated browser/gateway latency, and no systematic adversarial URL manipulation study. These limits mean the results should be read as laboratory evidence for a candidate screening layer, not as deployment proof.

6. Conclusion

This paper has investigated a lightweight stateless framework for low-latency malicious URL screening based solely on statistical and lexical URL features. The design avoids webpage downloading, external network queries, and heavyweight sequence inference when a compact URL-level decision is appropriate for preliminary screening. In the final measured experiment, the LightGBM model achieved 98.64% Accuracy, 97.62% Recall, an F1-Score of 0.9863, balanced-test PR-AUC of 0.9945, and 5% imbalanced-stress PR-AUC of 0.9793. Its model-side latency was 0.3995 ms, and its measured feature-extraction-plus-inference latency was

0.4165 ms.

The conclusion is qualified by feature-importance, ablation, and domain-only control analyses. URL length, entropy, host randomness, token morphology, and character-distribution patterns provide valuable evidence, although some signals are partially redundant with entropy and other statistical features. The domain-only control also demonstrates that measured performance still depends on source composition and URL granularity. Therefore, the approach should be viewed as a way to filter candidates and should not be used as a substitute for more content-rich defenses, reputation, or telemetry-based defenses in browsers, mobile gateways, or edge filters.

Future work will include expanding the experiment to larger time-shifted phishing corpora, obtaining benign full URLs from similar operational channels, reporting fully integrated browser or gateway latency, comparing with full-scale deep URL architectures, testing for adversarial URL manipulation, and validating the detector under real traffic before making claims of production-level deployment. Continuous model refresh may be possible through privacy-preserving deployment telemetry, when available from the field.

Acknowledgments

The author thanks the academic supervisors and peers who provided constructive guidance on the research direction, experimental design, and manuscript preparation. The author also acknowledges the public security datasets and research communities that support reproducible studies in malicious URL detection.

References

- [1] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., & Feamster, N. (2012). From throw-away traffic to bots: Detecting the rise of DGA-based malware. In Proceedings of the 21st USENIX Security Symposium (pp. 491–506). Bellevue.

- [2] Sahoo, D., Liu, C., & Hoi, S. C. H. (2017). Malicious URL detection using machine learning: A survey. *arXiv preprint arXiv:1701.07179*. <https://doi.org/10.48550/arXiv.1701.07179>.
- [3] Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1245–1254). Paris. <https://doi.org/10.1145/1557019.1557153>.
- [4] Garera, S., Provos, N., Chew, M., & Rubin, A. D. (2007). A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM Workshop on Recurring Malcode* (pp. 1–8). Alexandria. <https://doi.org/10.1145/1313848.1313850>.
- [5] Whittaker, C., Ryner, B., & Nazif, M. (2010). Large-scale automatic classification of phishing pages. In *Proceedings of the Network and Distributed System Security Symposium* (pp. 1–14). San Diego.
- [6] Marchal, S., Francois, J., State, R., & Engel, T. (2014). PhishStorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4), 458–471. <https://doi.org/10.1109/TNSM.2014.2377295>.
- [7] Xiang, G., Hong, J., Rose, C. P., & Cranor, L. (2011). CANTINA+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security*, 14(2), 21. <https://doi.org/10.1145/1987346.1987348>.
- [8] Blum, A., Wardman, B., Solorio, T., & Warner, G. (2010). Lexical feature based phishing URL detection using online learning. In *Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security* (pp. 54–60). Chicago. <https://doi.org/10.1145/1866642.1866651>.
- [9] Kan, M.-Y., & Thi, H. O. N. (2005). Fast webpage classification using URL features. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management* (pp. 325–326). Bremen. <https://doi.org/10.1145/1099554.1099631>.
- [10] Shirazi, H., Bezawada, B., & Ray, I. (2018). Kn0w Thy Domain Name: Unbiased phishing detection using domain name based features. In *Proceedings of the 23rd ACM Symposium on Access Control Models and Technologies* (pp. 69–75). Indianapolis. <https://doi.org/10.1145/3205977.3205981>.
- [11] Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications*, 117, 345–357. <https://doi.org/10.1016/j.eswa.2018.09.029>.
- [12] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27, 379–423, 623–656. <https://doi.org/10.1002/j.1538-7305.1948.tb01338.x>.
- [13] Kumi, S., Lim, C., & Lee, S. G. (2021). Malicious URL detection based on associative classification. *Entropy*, 23(2), 182. <https://doi.org/10.3390/e23020182>.
- [14] Le, H., Pham, Q., Sahoo, D., & Hoi, S. C. H. (2018). URLNet: Learning a URL representation with deep learning for malicious URL detection. *arXiv preprint arXiv:1802.03162*. <https://doi.org/10.48550/arXiv.1802.03162>.
- [15] Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2018). Evaluating deep learning approaches to characterize and classify malicious URLs. *Journal of Intelligent & Fuzzy Systems*, 34(3), 1333–1343. <https://doi.org/10.3233/JIFS-171439>.
- [16] Saxe, J., & Berlin, K. (2017). eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. *arXiv preprint arXiv:1702.08568*. <https://doi.org/10.48550/arXiv.1702.08568>.
- [17] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>.
- [18] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*. Long Beach.
- [19] Alqahtani, H., & Abu-Khadrah, A. (2024). Enhance the accuracy of malicious uniform resource locator detection based on effective machine learning approach. *Bulletin of Electrical Engineering and Informatics*, 13(6), 4422–4429. <https://doi.org/10.11591/eei.v13i6.7121>.
- [20] Butnaru, A., Mylonas, A., & Pitropakis, N. (2021). Towards lightweight URL-based phishing detection. *Future Internet*, 13(7), 154. <https://doi.org/10.3390/fi13070154>.
- [21] Cisco Talos. (2026). PhishTank developer information. https://phishtank.org/developer_info.php. Accessed 30 May 2026.
- [22] MahdaviFar, S., Maleki, N., Lashkari, A. H., Broda, M., & Razavi, A. H. (2021). Classifying malicious domains using DNS traffic analysis. In *Proceedings of the IEEE International Conference on Dependable, Autonomic and Secure Computing* (pp. 60–67). Calgary. <https://doi.org/10.1109/DASC.2021.9566527>.
- [23] Le Pochat, V., Van Goethem, T., Tajalizadehkhoob, S., Korczynski, M., & Joosen, W. (2019). Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the Network and Distributed System Security Symposium*. San Diego. <https://doi.org/10.14722/ndss.2019.23106>.
- [24] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357. <https://doi.org/10.1613/jair.953>.