URL-Based Malicious Content Detection: ML, DL, and Graph Techniquess

Meir Shuker
Department of Computer Science
Ariel University
meirshuker159@gmail.com

Noa Agasi
Department of Computer Science
Ariel University
noa0544569014@gmail.com

ABSTRACT

Malicious Uniform Resource Locators (URLs) are a major enabler of phishing, malware, and online fraud, posing a growing challenge to cybersecurity. While traditional blacklists and heuristic rules provide fast protection for known threats, they fail on zero-hour and obfuscated links. This work presents a systematic review and comparative analysis of modern detection, which relies on three paradigms: (i) classical Machine Learning (ML) with engineered lexical and host features, (ii) Deep Learning and NLP methods that learn URL representations from raw sequences, and (iii) Graph-based approaches that capture structural and relational patterns within and across URLs. Our synthesis highlights the strengths and limitations of these leading approaches and provides practical insights for improving malicious URL detection in real-world scenarios.

INTRODUCTION

Malicious URLs underpin phishing schemes, malware distribution, spam, and website defacement, costing organizations billions annually [1], [2]. Attackers rapidly mutate domains and obfuscate paths, rendering static blacklists or heuristic signatures ineffective against zero-day threats. The research community has responded with automated detection that analyses lexical, content, and structural signals to infer risk.

Traditional defences based on manually curated blacklists offer quick blocking of known bad sites but cannot cover the quasilimitless stream of new malicious URLs. ML, DL, and graph-centric approaches aim to generalise beyond specific signatures, learning discriminative patterns that distinguish malicious behaviour even when attackers alter superficial tokens.

This work surveys the state of the art across the three paradigms, consolidates recent datasets, and sketches a deployable architecture that combines graph neural networks with sequence modeling for practical, real-time protection.

TABLE I
REPRESENTATIVE MACHINE LEARNING MALICIOUS URL DETECTION STUDIES

Study	Models	Inputs / Features	Dataset	Key Result & Note
[8]	CatBoost, XGBoost, EBM , Random Forest (Explainable AI, SHAP)	URL & Domain Structure, Directory & File Characteristics, Time & Certificate Information, IP & Nameserver Details, the length_url is the most common feature in the data-sets so its important, time_domain_activation & qty_redirects are implying to suspect activity	Multiple: UCI Phishing, ISCX-URL-2016, Kaggle, Mendeley (18–112 features, hundreds of thousands of samples)	CatBoost highest accuracy (∼100%) with 10 key features in certain datasets, XGBoost fastest; EBM maximizes interpretability; note possible bias in single-feature reliance, and recommend hybridization for robustness
[7]	Self-Organizing Map combined with Radial Movement Optimization (SOM-RMO) — for feature extraction/dimensionality reduction. Radial Basis Function Network (RBFN) optimized using Tabu Search — for classification.	The study places greater emphasis on URL & Domain structure features (e.g., length_url, qty_slash_url) and Time & Certificate information features (e.g., time_domain_activation, time_domain_expiration). Additionally, it assigns significant weight to IP & Nameserver details (e.g., qty_ip_resolved, qty_nameservers) as key indicators for malicious URL detection.	The dataset comprises 651,191 URLs, categorized into four classes: benign (428,103 / 65.72 %), defacement (96,457 / 14.81 %), phishing (94,111 / 14.45 %) and malware (32,520 / 5.00 %). The URLs were collected from five public sources: ISCX-URL-2016, Malware Domain Blacklist, Faizan Git Repository, Phishtank and PhishStorm.	Accuracy: 96.5%, Precision: 95.2%, Recall: 94.8%, F1: 95%; robust multiclass capability (benign, phishing, malware, defacement); bested XGB, LRSVCDT, EnkNN; complex pipeline but strong defense against a diversity of attacks
[3]	EnBag (Ensemble Bagging Trees), EnkNN (Ensemble k-Nearest Neighbour), EnBos (Ensemble Boosted Trees), EnDsc (Ensemble Subspace Discriminator)	79 raw features (lexical tokens: query/domain/path; suspicious characters) → selected 59 via MRMR feature-selection;	dataset: ISCX-URL2016 (multiple classes: benign, spam, phishing, malware, defacement)	Accuracy 99.3% (binary), 97.9% (multi-class); EnBag top accuracy, EnkNN fastest; limitations: skewed class distribution, lesser sensitivity in phishing category

RELATED WORK

Malicious URL detection builds on three main paradigms: classical Machine Learning (ML), Deep Learning / Natural Language Processing (DL/NLP), and Graph-based approaches. Each offers distinct advantages and faces its own limitations.

A. Machine Learning Approaches

Classical ML relies on engineered features derived directly from URLs (length, tokens, reputation, etc.) and leverages algorithms suitable for real-time filtering and moderate datasets. ML models are valued for their interpretability and speed, but are challenged by concept drift, feature limitations, and class imbalance [3], [4], [5].

Key model examples:

- **Random Forest:** An ensemble of decision trees; excels in handling various types of features and robust to overfitting. Provides clear feature importances and strong detection accuracy for a range of feature-engineered datasets [3].
- **Support Vector Machine (SVM):** Classifies URLs by finding the decision boundary that maximizes class separation; effective for moderate-size, high-dimensional data [4].
- **Naïve Bayes:** Probabilistic model that assumes feature independence; simple, fast, and surprisingly effective when features are highly informative [5].
- **XGBoost/LightGBM:** Gradient boosting frameworks; combine weak decision trees for improved accuracy, especially in large feature spaces with complex interactions [6].
- **SOM-RMO + RBFN hybrid:** Unsupervised feature extraction (Self-Organizing Map + Radial Movement Optimization) fused with a Radial Basis Function neural network for classification; handles dimensionality reduction and complex decision surfaces [7].

B. Deep Learning and NLP Approaches

DL/NLP models learn feature representations directly from the raw URL string using neural architectures. These models are powerful against obfuscations and do not depend on manual feature engineering, though they require significant computational resources [9], [10], [11], [12].

Key model examples:

- **URLNet:** Dual-stream CNN architecture that learns both character-level and word-level URL patterns; improves detection of lexical tricks and generalizes better to unseen URLs [9].
- **LSTM-based detectors:** Sequential networks (Long Short-Term Memory) able to capture long-range dependencies in URL character sequences; effective at modeling prefix, suffix, and token-order patterns [12].

TABLE II
REPRESENTATIVE DEEP LEARNING / NLP MALICIOUS URL DETECTION STUDIES

Study	Models	Inputs / Features	Dataset	Key Result & Note
[9]	URLNet is dual-channel CNN that combine character-level + word-level	Raw URL string (char-level embedding + word-level embedding; minimal manual features). For the Word-CNN branch: Bag-of-Words on full URL, URL component tokenization (domain, path, TLD), position-sensitive bigrams, character trigrams, statistical expert features (URL length, hostname length, number of dots). For the Character-CNN branch: no manual feature extraction — directly embeds characters and learns sequential/structural patterns.	15M VirusTotal URLs (balanced: malicious/benign), also limited the frequency of any URL domain to be less than 5% and URL not in any bleack-list, (≈14M URLs: 4.68 M benign + 0.316 M malicious train; 9.37 M benign + 0.633 M malicious test)	Achieved an AUC of approximately 0.993 (training size 5 M) and demonstrated significant improvement over traditional feature-engineered methods; combines character- and word-level CNNs for end-to-end URL representation. Limitations: demands large training data and compute resources, and provides less interpretability compared to simpler models
[10]	DomURLs_BERT, a specialized BERT-based encoder pre-trained on a large-scale multilingual corpus of URLs, domain names, and DGA (Domain Generation Algorithms)datasets.	Raw URL/domain string tokenized via a dedicated SentencePiece tokenizer trained from scratch (no large hand-crafted feature set); special markers (e.g., [DOMAIN], [PATH], [IP]) annotate segments and the model is pre-trained on a large multilingual corpus of URLs/domains using MLM.	Large-scale pre-training corpus: approximately 375 million items (≈ 355 million URLs + ≈ 19.9 million domain names) using multilingual URLs, domain names and DGA data. Fine-tuning tasks employ multiple benchmark datasets for both binary and multi-class classification of URLs/domains (phishing, malware, DNS-tunnelling, DGA). All the used datasets have been divided into 60% for training, 20% for validation, and 20% for testing. Table 2 summarizes the characteristics of the evaluation datasets.	Achieved up to ≈ 99% accuracy in multiple URL & domain tasks, outperforming several character-based and BERT-based baselines; significant resource/compute demand remains a limitation. high compute demand
[11]	Transformer-based (multi-layer, custom architecture; not vanilla BERT)	Raw URL character sequence with multi-scale pyramid features (encodes substrings at multiple granularities); integrates spatial pyramid attention to fuse both local and global character-level patterns, enabling the model to capture hidden malicious cues across short, medium, and long URL segments. Structural representation supports adversarial robustness and hierarchical pattern discovery.	more than 2,000,000 URLs (aggregated from multiple public datasets); training included class balancing and adversarial example generation	Achieved up to 99.5% accuracy and F1-score of 0.997 on public datasets (OpenPhish, Mendeley, Kaggle), consistently outperforming URLNet , GramBeddings and classical Transformer baselines, especially for "hard" classes. Maintained robustness under adversarial evaluation: in targeted attack scenarios, detected 100% of malicious URLs (where previous models missed 4–7 cases). Performance attributed to pyramid features, character-level modeling, and large-scale adversarial training.

- **DomURLs_BERT:** Transformer-based model pre-trained on large-scale URL and domain corpora; leverages attention mechanisms to capture complex, context-dependent patterns in URLs [10].
- **TransURL:** Custom transformer with multi-scale encoders and spatial pyramid pooling; excels in capturing fine-grained local structures, and robust to adversarial manipulations [11].

C. Graph-Based Methods

Graph-based techniques represent URLs or infrastructures as structured graphs, enabling the model to utilize both internal structure and external relationships. They are unparalleled in detecting coordinated attacks but may require extensive context data and are computationally intensive [13], [14].

Key model examples:

- **GNN-GAT-LSTM:** Hybrid model that converts each URL to a character graph, applies Graph Neural Networks with attention (GAT) to highlight important substructures, and follows with LSTM for sequential dependencies; excels at integrating local and global cues [13].
- **LBP Graph Model:** Constructs a heterogeneous graph linking URLs, IPs, and name servers; uses Loopy Belief Propagation to spread risk information and label URLs based on both direct and indirect associations [14].
- **Hybrid structural + sequential graph models:** Combine GNN embeddings with classical or deep learning layers to enhance resilience to adversarial evasion and capture diverse pattern structures.

TABLE III
REPRESENTATIVE GRAPH-BASED MALICIOUS URL DETECTION STUDIES

Study	Models	Inputs / Features	Dataset	Key Result & Note
[13]	Hybrid GNN-GAT-LSTM (GNNs) Graph Attention Networks (GATs) and Long Short-Term Memory (LSTM) (structural + temporal; PyTorch Geometric)	Character graphs: each URL is modeled as a directed graph of characters with adjacency edges; sequential node encoding via LSTM to capture order; additional engineered features (packet size, repetition rate, symbol density); SMOTE for class balancing; attention layers for substructure saliency	Kaggle Malicious URLs (651,191 URLs: balanced across benign, phishing, defacement, malware); comprehensive ablation analysis; robust to class imbalance	Achieved 98.06% accuracy and weighted F1 of 98.04; excels at capturing both sequential and structural cues, outperforming CNN/LSTM baselines; shows strong robustness to obfuscated and adversarial URLs; recall for phishing class at 91.3%; scalable real-world deployment pipeline; interpretable attention to key substructures.
[14]	Heterogeneous Graph LBP (Loopy Belief Propagation), edge potential learning, node embedding	Graph: each URL node is linked to IP, authoritative name server, domain, and substrings; features include cosine/RBF similarity, prior risk from Random Forest, label/cosine edge potential, and meta-data; incorporates word2vec and locally linear embeddings for network entities	Multiple phishing and banking corpora: 53,000–306,000 mostly imbalanced URLs (benchmarks and newly collected); 5-fold cross-validation for stability	Achieved F1 up to 98.77% and high precision/recall on hard classes; robust to infrastructure churn and adversarial evasion; advanced convergence strategy handles graph cycles; outperformed RFC and classic ML in large-scale evaluation but requires external DNS/network data and complex graph construction.
[13]	Hybrid Graph Neural Network (GNN) framework integrating deep learning layers (e.g., multi-level attention, LSTM)	Each URL is represented as a graph: nodes correspond to characters, tokens, or URL segments; edges encode adjacency and hierarchy. Features combine structural URL properties, engineered statistical cues, and often include external network context (IP, nameserver, domain).	Public and custom benchmarks for multi-class URL classification (benign, phishing, malware, defacement); tens of thousands to hundreds of thousands of samples; class balancing applied.	Achieved accuracy often exceeding 98% and strong F1 for all malicious classes; robust to obfuscated and adversarial URLs; outperforms classic ML and sequential DL approaches; interpretable attention over suspicious substructures; moderate-to-high training compute, but deployable inference on standard hardware.

TABLE IV
SUMMARY COMPARISON OF DETECTION APPROACHES WITH USE CASE/APPLICATION DOMAIN

Approach	Representative Methods	Inputs	Strengths	Weaknesses	Use Case / Application Domain
Machine Learning	Random Forest, SVM, XGBoost, LightGBM	Lexical, host-based, optional content features	Fast, interpretable, effective with engineered features	Relies on manual features, weaker generalization to novel attacks	Real-time systems, legacy/production security platforms
Deep Learning / NLP	Char-CNN, URLNet, BERT-family, LSTM/BiLSTM/GRU	Raw URL strings, learned embeddings	Automatic feature learning, strong against obfuscation and zero-day patterns	Compute-intensive, reduced interpretability, data-hungry	Large-scale analysis, modern phishing/URL detection, threat intelligence platforms
Graph-Based Methods	GNN, GAT, GCN, hybrids with sequence models/rules	Graphs over URL characters, domains, relational entities, optional network/DOM context	Captures structural/relational patterns, detects coordinated campaigns	Complex pipeline, higher latency, often needs extra context	Coordinated/advanced attack detection, infrastructure monitoring, forensic analysis

Summary

ML models offer speed and interpretability, DL/NLP models drive accuracy and adaptivity, and graph-based methods produce relational insight and resilience. The optimal approach hinges on the dataset, threat context, and operational requirements.

In recent years, deep learning and graph-based models have achieved significant advances and superior detection accuracy; however, these methods require substantial computational resources and specialized hardware for training and deployment. As a result, such models were not selected for our project, in order to ensure real-world feasibility on constrained or commodity hardware. Classical ML methods, therefore, remain preferable for scenarios where processing power is limited or real-time detection is required.

I. Methodology

This study employs a hybrid machine-learning architecture designed to improve malicious URL detection by combining feature-selection robustness with high-capacity gradient-boosting classification. Our methodology is organized into three main stages: data preprocessing, feature selection using Recursive Feature Elimination (RFE), and final classification using XGBoost.

A. Data Preprocessing

All URLs are converted into structured numerical representations that include lexical attributes (e.g., length, entropy, special-character ratios) and statistical character-level patterns extracted directly from the URL string. Missing values are imputed when needed, and all features are produced deterministically without relying on external sources such as WHOIS or DNS metadata.

B. RFE for Feature Selection

Recursive Feature Elimination (RFE) is utilized as the primary feature-selection mechanism. RFE iteratively trains a tree-based estimator and removes the least-informative features until a predefined number of features is retained. This allows it to identify discriminative attributes even in high-dimensional and noisy URL datasets.

We use RFE to rank features and retain only the top-k features according to the elimination procedure. This step reduces dimensionality, removes redundant or weak predictors, and mitigates noise—ultimately improving both performance and training stability for the downstream classifier. Importantly, feature selection is performed only on the training split as part of a pipeline to prevent information leakage.

C. XGBoost for Final URL Classification

Extreme Gradient Boosting (XGBoost) serves as the final prediction model. XGBoost is a powerful gradient-boosted tree framework optimized for both accuracy and computational efficiency. It incrementally constructs boosted trees that correct errors of previous trees through gradient-based optimization, enabling it to capture complex nonlinear interactions commonly present in malicious URL patterns. Hyperparameters such as learning rate, maximum tree depth, subsampling ratio, and column sampling are tuned to maximize detection performance while preventing overfitting.

D. Hybrid RFE–XGBoost Integration

The core contribution of this methodology lies in the integration of RFE and XGBoost into a two-phase hybrid model. RFE operates exclusively as a feature selector, ensuring that the classifier receives only the most informative attributes. This reduces feature noise, enhances model interpretability, and accelerates training.

XGBoost then operates on this refined feature subset, allowing it to focus its boosting capacity on high-quality signals rather than spending modeling capacity on irrelevant predictors. This combination leverages the strengths of both models: RFE provides a compact and robust feature set, while XGBoost delivers strong predictive accuracy.

E. Evaluation Protocol

To maintain methodological rigor, the entire pipeline—including RFE feature selection and XGBoost training—is encapsulated within a train/validation/test evaluation protocol implemented with scikit-learn. Feature selection is fitted only on the training data to avoid leakage. The validation split is used to select a decision threshold by scanning predicted probabilities to maximize recall while keeping the false positive rate at or below a strict policy constraint (default $\mathrm{FPR} \leq 0.01$). Performance is evaluated using standard metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and PR-AUC.

F. Comparison with Traditional Approaches

Compared to purely feature-engineered ML baselines, the proposed RFE–XGBoost hybrid reduces the dependence on noisy or redundant attributes by automatically selecting a compact subset of informative features, while still remaining lightweight enough for deployment on commodity hardware. Unlike deep neural models, it does not require GPUs or large-scale pre-training, yet it can approximate strong detection performance on structured URL features. This makes the approach suitable for security appliances and gateways with strict real-time and resource constraints.

Dataset

Primary dataset. We chose the Kaggle *"Benign and Malicious URLs"* dataset [15] as our primary benchmark because it is (i) **balanced** (50/50 benign vs. malicious), (ii) **large** enough to provide diverse URL patterns, and (iii) contains multiple malicious categories (e.g., phishing, malware, and defacement) that increase variability and reduce overfitting to a single attack type [15]. These properties make it a strong fit for training a robust lexical-only detector.

Why not use the exact datasets from the compared papers. We attempted to replicate the datasets used in the related works, but many of them are distributed only as **pre-extracted feature tables** rather than raw URLs. In addition, several studies rely on features that require external metadata sources such as WHOIS/DNS/certificates, which cannot be reproduced in our "Lexical-only" setting. Finally, many of those datasets are smaller than our primary benchmark, making them less suitable for training and stress-testing a generalized model.

Feature extraction and selection. In the first stage, we engineered **>120 candidate lexical features** from the raw URL string (length statistics, character/symbol counts, tokenization-based structure signals, entropy measures, and suspicious keyword indicators). Since prior work suggests that a larger number of features does not necessarily translate into better accuracy, we performed iterative pruning and optimization to obtain a compact, high-quality set of **24 lexical features** (Table VI). We then applied Recursive Feature Elimination

(RFE) to select the **top-k** features for the final model (we used $k = 20$ in the best configuration), improving robustness and reducing redundancy before training XGBoost.

Splits and threshold calibration. After preprocessing and label normalization, we obtained a balanced dataset of **632,508** samples: **316,254** benign and **316,254** malicious URLs. We used a stratified split of **64%** training, **16%** validation, and **20%** testing. The validation set was reserved for decision-threshold calibration: we first targeted a low false-positive regime and then optimized for a favorable trade-off between **low FPR** and **high recall**. We found that a threshold around **0.55** provides an effective balance in our setting.

External (real-world) validation on an imbalanced dataset. To better emulate real-world deployment, we additionally evaluated the trained model on an **unbalanced** external Kaggle dataset that contains raw URLs and labels (and was not analyzed or tuned during development): *"Phishing URL Dataset (URL and Label)"* [16]. Using the same feature extraction pipeline and threshold 0.55, the model was tested on **235,795** samples and achieved: Accuracy $= 79.66\%$, Recall $= 96.76\%$, Precision $= 90.15\%$, F1 $= 89.85\%$, and FPR $= 2.00\%$ (TN $= 80{,}411$, FP $= 2{,}691$, FN $= 13{,}215$, TP $= 139{,}478$). These results indicate that even under distribution shift and class imbalance, the model maintains very high recall while keeping the false-positive rate within a practical range.

As summarized in Table V, we compare our pipeline against the evaluated baselines, and Table VI reports the final lexical feature set.

II. Comparison with Related Work

To contextualize the performance and design choices of our proposed **RFE+XGBoost** pipeline, we compare it against two representative ensemble-based studies from our survey (results summarized in Table V). Specifically, we compare against: (1) [8], which evaluates **CatBoost**, **Random Forest (RF)**, and **XGBoost**; and (2) [3], which proposes ensemble methods such as **Ensemble Bagging Trees (EnBag)** and **Ensemble kNN (EnkNN)**. These works represent widely used, deployment-friendly strategies for URL classification and provide a strong reference point for assessing the added value of combining systematic feature elimination with gradient-boosted learning under a lexical-only constraint.

Study 1: CatBoost, Random Forest, XGBoost

- **Methodology:** This study evaluates strong tree-based learners for phishing URL detection using URL/domain-structure and lexical-style signals. The compared models include **CatBoost**, **Random Forest (RF)**, and **XGBoost**, aiming to achieve high accuracy while maintaining practical inference speed.
- **Key observation:** The study reports that **CatBoost** can achieve the highest accuracy in certain datasets (with a compact subset of highly informative features), while **XGBoost** is noted as a very fast and competitive alternative. **RF** provides a strong baseline due to its robustness on noisy feature sets.
- **Relation to our approach:** Our model adopts **XGBoost** as the high-capacity classifier but strengthens it via **explicit feature elimination (RFE)**. This directly targets redundancy and weak predictors before boosting, improving separability and generalization when working with lexical/URL-derived signals.

Study 2: Ensemble Bagging Trees

- **Methodology:** This approach aggregates multiple decision-tree learners trained on bootstrap samples (bagging) and merges predictions to reduce variance and improve stability, particularly when features contain noise or partial redundancy (typical in URL-lexical representations).
- **Limitation:** While bagging improves robustness, it does not explicitly optimize the feature space. As a result, correlated or weak predictors may persist, which can limit recall and overall discriminative power compared to pipelines that include systematic feature selection.
- **Our Improvement:** Our **RFE+XGBoost** pipeline removes weak/redundant predictors prior to training, then uses boosted trees to model nonlinear interactions efficiently. This combination tends to yield stronger ranking-based performance (ROC-AUC / PR-AUC) and recall, while maintaining a deployment-friendly inference profile.

III. Comparative Analysis

Overall, these studies demonstrate that **CatBoost/RF/XGBoost** and **bagging-based tree ensembles** are strong and widely used baselines for URL classification. However, our results support that integrating **explicit feature elimination (RFE)** with a **high-capacity gradient-boosted classifier (XGBoost)** provides a consistent advantage: it concentrates learning capacity on the most informative URL/lexical signals, reduces noise from redundant predictors, and preserves efficient inference suitable for security-oriented URL filtering at scale.

Conclusion

We surveyed ML, DL/NLP, and graph-based techniques for malicious URL detection and motivated a graph-centric hybrid architecture designed for deployment. By combining structural attention with sequential modeling and packaging the solution for real-time inference, we aim to advance both effectiveness and practicality. Future work includes exploring heterogeneous graph transformers, integrating DOM/network flow context, and pursuing continual learning to adapt to evolving adversarial tactics.

TABLE V
RESULTS COMPARISON ACROSS MODELS EVALUATED ON THE SAME DATASET: KAGGLE "BENIGN AND MALICIOUS URLS" [15]. THRESHOLD FOLLOWED THE SAME POLICY (0.55). BEST VALUES PER METRIC ARE HIGHLIGHTED (FOR FPR, LOWER IS BETTER).

Group (Citation)	Model	Acc.	Prec.	Rec.	F1	FPR (%)
This work (Ours)						
	`rfe_xgb`	**99.415**	**99.850**	**98.979**	**99.412**	**0.149**
[8]						
	CatBoost	99.263	99.811	98.713	99.259	0.187
	Random Forest (`rf`)	99.376	99.828	98.923	99.373	0.171
	XGBoost (`xgb`)	99.412	99.845	98.977	99.409	0.153
[3]						
	Ensemble kNN (`en_knn`)	99.308	99.778	98.835	99.304	0.220
	Ensemble Bagging Trees (`en_bag`)	98.689	99.827	97.548	98.674	0.169

TABLE VI
LEXICAL FEATURE SET USED BY `RFE_XGB` (24 FEATURES).

Feature	Description	Feature	Description
`url_len`	Full URL length	`digit_count`	Number of digits in the URL
`host_len`	Hostname length	`alpha_count`	Number of alphabetic characters
`path_len`	Path length	`has_https`	1 if scheme is HTTPS, else 0
`query_len`	Query-string length	`has_ip_host`	1 if host is an IP address, else 0
`dot_count`	Number of '.' characters	`port_present`	1 if an explicit port exists, else 0
`dash_count`	Number of '-' characters	`subdomain_count`	Number of subdomains (host dots minus 1)
`underscore_count`	Number of '_' characters	`token_count`	Number of tokens across host and path
`slash_count`	Number of '/' characters	`avg_token_len`	Average token length
`at_count`	Number of '@' characters	`max_token_len`	Maximum token length
`qmark_count`	Number of '?' characters	`shannon_entropy`	Shannon entropy of the URL string
`eq_count`	Number of '=' characters	`suspicious_keyword_count`	Count of hits in a suspicious keyword list
`amp_count`	Number of '&' characters	`percent_count`	Number of '%' characters

REFERENCES

[1] APWG, "Phishing activity trends report q1 2025," APWG, Tech. Rep., 2025. [Online]. Available: https://apwg.org/trendsreports

[2] IBM Security, "Ibm security x-force threat intelligence index 2025," IBM Security, Tech. Rep., 2025. [Online]. Available: https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/2025-threat-intelligence-index

[3] Q. A. Al-Haija and M. Al-Fayoumi, "An intelligent identification and classification system for malicious uniform resource locators (urls)," *Neural Computing and Applications*, 2023. [Online]. Available: https://doi.org/10.1007/s00521-023-08592-z

[4] F. Abad and [OtherAuthors], "Machine learning approaches for malicious url detection," *[JournalName]*, 2023. [Online]. Available: [URL]

[5] F. Aljabri and [OtherAuthors], "An assessment of lexical, network, and content-based features for detecting malicious urls," *Security and Communication Networks*, vol. 2022, pp. 1–15, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1155/2022/3241216

[6] F. Shetty and [OtherAuthors], "Lexical features for url-based phishing detection," *[JournalName]*, 2023. [Online]. Available: [URL]

[7] T. Swetha, M. Seshaiah, K. L. Hemalatha, B. H. ManjunathaKumar, and S. V. N. Murthy, "Hybrid machine learning approach for real-time malicious url detection using som-rmo and rbfn with tabu search optimization," *IJCSCE*, 2023, hybrid SOM-RMO feature extraction, RBFN + Tabu Search classification. [Online]. Available: https://arxiv.org/pdf/2407.06221

[8] A. Fajar, S. Yazid, and I. Budi, "Enhancing phishing detection through feature importance analysis and explainable ai," *Preprint*, 2024, comparative study of CatBoost, XGBoost, EBM with SHAP/XAI feature selection. [Online]. Available: https://arxiv.org/pdf/2411.06860

[9] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," *arXiv preprint arXiv:1802.03162*, 2018.

[10] A. El Mahdaouy, S. Lamsiyah, M. Janati Idrissi, H. Alami, Z. Yartaoui, and I. Berrada, "Domurls_bert: Pre-trained bert-based model for malicious domains and urls detection and classification," *arXiv preprint arXiv:2409.09143*, 2024. [Online]. Available: https://arxiv.org/abs/2409.09143

[11] Z. Wang, X. Chen, M. Li, Y. Zhou, L. Huang, and H. Zhang, "Transurl: Improving malicious url detection with multi-layer transformer representations," *Computer Security and Reliability (CSR)*, 2025, transformer-based hierarchical URL embedding with multi-layer attention aggregation.

[12] B. Kim and H. Lee, "Character-level LSTM for malicious URL detection," in *Proceedings of the International Conference on Information and Communication Technology Convergence*. IEEE, 2018, pp. 153–158.

[13] M. Shafiq, A. Javed, S. Mumtaz, A. Alzahrani, A. Alghamdi, A. Ghafoor, A. Alfakeeh, and A. Sadiq, "A graph-attentive lstm model for malicious url detection," *IEEE Access*, vol. 9, pp. 170 208–170 221, 2025.

[14] Y. Liu, J. Wang, X. Chen, T. Zhou, L. He, and Y. Li, "Efficient phishing url detection using graph-based methods," *IEEE Transactions*

on Information Forensics and Security, vol. 18, pp. 4025–4038, 2023.

[15] S. Malibari, "Benign and malicious urls," Kaggle dataset, 2023, accessed: 2026-01-18. [Online]. Available: https://www.kaggle.com/datasets/samahsadiq/benign-and-malicious-urls

[16] M. Janety, "Phishing url dataset (url and label)," Kaggle dataset, accessed: 2026-01-18. [Online]. Available: https://www.kaggle.com/datasets/marryjanety/phishing-url-dataset-url-and-label