Vol.4, No.6, Page 2230- 2243, 2026 https://doi.org/10.59628/jast.v4i6.2961 ISSN 2958-9568 JAST

SUN PUBLISHER

مجلة جامعة صنعاء للعلوم التطبيقية والتكنولوجيا

Sana'a University Journal of Applied Sciences and Technology

https://journals.su.edu.ye/index.php/jast/

A Hybrid Deep Learning Ensemble for Multi-Class Malicious URL Detection in Arabic and English

Nagi Ali Abdullah Al-shaibany *

Department of Information Technology, Faculty of Computer and Information Technology, Sana'a University, Sana'a, Yemen

*Corresponding author: shaibany@su.edu.ye

Abstract

Malicious URLs serve as primary vectors for cyber-attacks, facilitating phishing, malware, and defacement. While conventional systems focus on binary classification, security operations require granular threat identification. Furthermore, literature exhibits a significant bias toward English content, leaving Arabic-speaking populations disproportionately exposed to localized threats. This study proposes a hybrid stacked ensemble architecture integrating CNN-BiLSTM with an Attention mechanism, Random Forest, and XGBoost. The methodology incorporates 27 lexical features for English URLs and 23 specialized features tailored to Arabic linguistic structures and Punycode-encoded domains. The model was evaluated on 651,191 English and 20,329 Arabic URLs. The architecture achieved a peak accuracy of 99.43% on English data and 89.07% on the Arabic dataset, outperforming baseline configurations. Feature correlation analysis demonstrated that class imbalance inflates feature importance, with average correlation coefficients decreasing by 0.214 post-balancing. Comparative experiments utilizing a unified cross-language model yielded inferior results (91.2% English, 82.5% Arabic), confirming that language-specific optimization is essential. This research establishes the first multi-class baseline for Arabic URL detection, providing a robust, scalable framework for regional threat intelligence.

Article Info

Keywords:
Malicious URL Detection, Ensemble Methods, Arabic URL Classification, Hybrid Models, Stacked Generalization.

Article History:
Received: 31-March-2026,
Revised: 13-May-2026,
Accepted: 17-May-2026,
Published: 28 June 2026.

1. INTRODUCTION

The exponential growth of internet usage has been accompanied by a parallel increase in cyber threats targeting unsuspecting users. Among these threats, malicious Uniform Resource Locators (URLs) serve as primary attack vectors for phishing campaigns, malware distribution, and website defacement [1]. According to the 2025 Verizon Data Breach Investigations Report, over 75% of cyberattacks involve a user carelessly interacting with a malicious link—often through phishing emails or compromised websites. This statistic underscores the importance of URL classification as a critical component of modern cybersecurity defense systems [2]. The global cost of cybercrime is projected to reach $10.5 trillion annually by 2025, with URL-based attacks representing a significant portion of this economic impact [3].

1.1. The URL Threat Landscape

Malicious URLs can be categorized into four primary types, each requiring distinct mitigation strategies:

1. **Phishing URLs**: impersonate legitimate websites (banks, social media, email providers, government services) to steal credentials and sensitive information. These attacks have become increasingly sophisticated, with phishing kits enabling rapid deployment of convincing counterfeit sites [4]. The Anti-Phishing Working Group (APWG) reported over 0.85 million unique phishing attacks in Q4 2025 alone [5].
2. **Malware Distribution URLs**: host or link to malicious executables, ransomware, trojans, and spyware. When users visit these URLs, drive-by downloads can infect systems without explicit user interaction [6]. "Ransomware-as-a-Service (RaaS) has significantly contributed to the proliferation of ransomware attacks in

the current cybersecurity landscape. This threat model provides cybercriminals, regardless of their technical skill level, with the tools and infrastructure to execute ransomware attacks, lowering the entry barrier into the world of digital extortion." [7].

3. **Defacement URLs:** refer to web addresses of pages that have been altered by attackers after they gain unauthorized access to a website or its content-management system. The attacker replaces legitimate content with their own messages, images, political statements, propaganda, or other material. Defacement attacks are often motivated by political/ideological goals, notoriety, or disruption rather than direct financial gain, but they cause immediate reputational damage and erode users' trust in the affected organization. [8]. The large majority of these attacks occurred as part of mass defacements, made possible by widespread use of bots to spot and use CMS flaws [9].

4. **Benign URLs:** constitute the vast majority of web traffic and include legitimate websites, search engines, and online services. However, the huge volume of benign traffic makes manual URL inspection impossible, necessitating automated classification systems [10].

1.2. The Critical Gap: Non-English URL Detection

Existing studies focus heavily on English URLs, leaving non-English content under-protected. Arabic is spoken by over 400 million people across 22 countries, yet it's barely studied in cybersecurity research. A comprehensive survey by Aljabri et al. found that "the limited amount of research dedicated to Arabic-content websites highlights the need for more extensive studies, the creation of new datasets, and the development of specialized tools tailored to addressing the increasing threats within this field." [11]. This gap is particularly concerning given that Arabic-speaking regions have faced sophisticated cyber campaigns. The "North African Fox" operation targeted governments and businesses for approximately three years [12], while Saudi banks and government services (Absher, Nafath, STC Pay) have been repeatedly targeted by phishing campaigns using domain names like 'riyadbank-verify-account.online' and 'nafath-portal.secure-phishing.com' [13].

1.3. The Challenge of Multi-Class Classification

Traditional approaches to malicious URL detection have largely focused on binary classification distinguishing benign URLs from malicious ones [14]. However, this binary classification suffers from several critical limitations:

First, binary classification fails to capture the differences between attack types. A phishing URL targeting bank credentials requires different incident response procedures than a malware-distribution URL hosting ransomware. Security operations centers (SOCs) need threat-specific intelligence to prioritize alerts and deploy appropriate countermeasures [15].

Second, real-world URL data exhibit severe class imbalance. Benign URLs vastly outnumber malicious ones, and among malicious URLs, certain categories (particularly malware) are significantly underrepresented. In our English dataset of 651,191 URLs, benign samples constitute 65.7% of the data, while malware represents only 5.0%. Our Arabic dataset mirrors this distribution (64.5% benign, 5.0% malware), enabling direct cross-lingual comparison [16].

1.4. Our Contributions

This work fills the gaps we just outlined in a few key ways.

i. **First**, a new way to combine models. We built a hybrid ensemble that brings together three different approaches: a CNN-BiLSTM network that learns patterns from the URL text, plus Random Forest and XGBoost that work with hand-picked features. We tied them together using stacked generalization. Consequently, the proposed model exhibited superior performance compared to existing baselines with 98.23% accuracy on the imbalanced English data and 99.43% on a separate test dataset beating a recent graph-based model (98.06%) and a traditional ensemble (96.8%).

ii. **Second**, testing across languages. We showed that the same approach works for Arabic, but only after adding language-specific features. The features that mattered for English (like IP addresses and digit counts) weren't enough. By adding things like Arab bank names and government service keywords, we got the model to perform well on Arabic too.

iii. **Third**, we looked closely at how class imbalance messes with feature importance. Empirical analysis indicates that feature importance is significantly influenced by class distribution. Take has_ip: it went from r=0.72 down to r=0.05 after balancing. That's a drop of 0.67. On average, correlations dropped by 0.214 across the board. This tells us that what seems important in imbalanced data might just be a side effect of the imbalance itself.

1.5. Paper Organization

The remainder of this paper is organized as follows: Section 2 reviews related work in English and Arabic URL detection. Section 3 describes our methodology, including dataset characteristics, feature engineering, and the proposed hybrid ensemble architecture. Section 4 presents experimental results for both English and Arabic datasets, including performance metrics, ablation studies, and feature correlation analysis. Section 5 discusses the implications of our findings for multilingual

Table 1. Related works

Authors	Year	Model/Approach	Classes	Dataset	Best Accuracy	Limitations
Sankaranarayanan et al. [1]	**2024**	Stacking ensemble (RF + XGB + LightGBM + CatBoost) URL + lexical	**4-class**	Large URL Dataset	**96.8%**	Lower accuracy than our work
Aljabri et al. [11]	**2024**	Kashif (RF + SVM + DT) Arabic content features	**Arabic binary**	4,048Arabic URLs	92.96%	Binary only; small dataset
Le et al. [14]	**2018**	URLNet (CNN character + word level) Character/word embeddings	Binary	2.4M URLs	95-97%	Binary only; no multi-class
Hossain et al. [17]	**2025**	GNN-GAT-LSTM Graph + attention + LSTM	**4-class**	**651,191 URLs**	**98.06%**	Lower accuracy than our work
Vazhayil et al. [18]	**2023**	CNN-LSTM hybrid URL sequences	Binary	ISCX-URL-2016	98.2%	Binary only
Huang et al. [19]	**2024**	Attention-BiLSTM URL + attention weights	Binary	PhishTank + Alexa	98.5%	Binary only; English-only
Mahdaouy et al. [20]	**2024**	DomURLs_BERT BERT embeddings	Binary	Multiple sources	98.1%	Computationally expensive
Aljofey et al. [21]	**2025**	Multi-scale CNN + BiLSTM + Gated MLP Multi-scale features	Binary	PhishTank	99.83%	Binary only; phishing only
Alsufyani & Alajmani [22]	**2025**	CNN + BiGRU + GRU SMS text features	**Arabic binary**	Arabic SMS	95.3%	SMS not URLs; binary only
Tian et al. [23]	**2025**	**Survey Paper** Comprehensive review	N/A	N/A	N/A	Identifies research gaps
Yuan et al. [24]	**2025**	Cross-cultural analysis Cultural patterns	N/A	Chinese/English	N/A	Highlights non-English gap
Our work		**CNN-BiLSTM + RF + XGB + Meta-Learner 27 universal + 23 Arabic-specific**	**4-class**	**651,191 English + 20,329 Arabic**	**English: 99.43% Arabic: 89.07%**	**First Arabic 4-class baseline**

cybersecurity. Section 6 addresses limitations and future work. Section 7 concludes the paper.

2. RELATED WORK

Malicious URL detection has evolved considerably over the past decade. This section reviews the literature chronologically and thematically, with emphasis on approaches most relevant to our work. As summarized in Table 1, existing 4-class models achieve at most 98.06% accuracy [17], while Arabic research remains limited to binary classification with small datasets [11, 12]. Our work addresses both gaps.

2.1. Traditional Detection Methods

Back in the early days, most systems just used blacklists databases of known bad URLs maintained by companies like Google and PhishTank. These work well for known threats but can't catch something brand new [25]. Sheng et al. [26] found that 47-60% of phishing attacks were blacklisted only after 12 hours, by which time damage had already been done.

Heuristic-based methods emerged to complement blacklists by analyzing URL characteristics. Garera et al. [27] identified 18 hand-crafted features for phishing detection, including IP address presence and suspicious TLDs. McGrath and Gupta [28] analyzed lexical patterns in phishing URLs, noting that phishing domains are often shorter and contain more digits. While interpretable, heuristic approaches struggle with evolving attack patterns [10].

2.2. Machine Learning Approaches

The application of machine learning transformed URL classification. Ma et al. [29] employed logistic regression and Naïve Bayes on lexical and host-based features, achieving 95% accuracy on binary classification. Random Forests became popular due to their robustness [30], with Blum et al. [31] achieving 97% accuracy using lexical features alone. Support Vector Machines were extensively explored [32], with Xu et al. [33] achieving 96.5% accuracy using hybrid features. Gradient boosting methods, particularly XGBoost [34] and LightGBM [35], have become standard benchmarks in the field [36].

Despite their success, traditional ML approaches require extensive feature engineering and struggle to model sequential dependencies in URL strings [14].

2.3. Deep Learning Approaches

Deep learning methods address these limitations by learning hierarchical representations directly from data. Le et al. [14] proposed URLNet, a convolutional neural network that learns URL representations at both char-

acter and word levels, achieving 95-97% accuracy on binary classification. Following URLNet, researchers explored recurrent architectures. Bahnsen et al. [37] applied LSTM networks to character sequences, achieving 98.7% accuracy on phishing detection. Vazhayil et al. [18] found that hybrid CNN-LSTM architectures consistently outperformed single-architecture approaches.

Attention mechanisms were incorporated to help models focus on discriminative URL segments. Huang et al. [19] proposed an attention-based BiLSTM that learned to weight different URL components, improving both accuracy and interpretability. Transformer-based models have recently been applied, with Mahdaouy et al. [20] introducing DomURLs_BERT, which achieved 98%+ accuracy on benchmark datasets.

2.4. Ensemble and Hybrid Methods

Ensemble methods combine multiple models to achieve better performance. Sankaranarayanan et al. [1] proposed a stacking-based ensemble integrating Random Forest, XGBoost, LightGBM, and CatBoost for 4-class URL classification, achieving 96.8% accuracy across benign, phishing, defacement, and malware categories.

More recently, Hossain et al. [17] developed a hybrid GNN-GAT-LSTM architecture combining Graph Neural Networks with Graph Attention Networks and LSTM. Tested on 651,191 URLs across four classes, their model achieved 98.06% accuracy. Aljofey et al. [21] integrated multi-scale CNNs, BiLSTM, and Gated MLP architectures for phishing detection, achieving 99.13-99.83% accuracy—though only for binary classification.

Stacked generalization has shown promise, but prior stacking applications have been limited to binary classification and have not comprehensively addressed class imbalance.

2.5. Class Imbalance in URL Classification

Real-world URL data exhibits severe class imbalance, with benign URLs vastly outnumbering malicious ones. He and Garcia [16] provided a comprehensive survey of imbalanced learning techniques, noting that standard classifiers bias toward majority classes. Researchers have explored oversampling minority classes with SMOTE [38], cost-sensitive learning [39], and ensemble imbalance methods such as Balanced Random Forest [40]. However, few studies have analyzed how imbalance affects feature importance across malicious sub-types.

2.6. Non-English URL Detection

A critical gap in the literature is the scarcity of research on non-English URLs, particularly Arabic-language content.

Several researchers have attempted to address Arabic content detection. Al-Kabi et al. [41] developed a content-based approach to detect spam in Arabic web pages, achieving 99.52% accuracy using decision trees. Wahsheh et al. [42] proposed a hybrid approach combining HTML and URL features for detecting Arabic web spam, achieving 89.01% accuracy. Alharbi and Aljaedi [43] studied Arabic spam account detection on Twitter, achieving over 90% accuracy. Alsufyani et al. [22] applied deep learning to detect phishing in Arabic text messages, achieving 95.3% accuracy. AlGhamdi and Khan [44] developed an analysis system for suspicious Arabic tweets, achieving 86.72% accuracy.

Despite these efforts, all existing Arabic-focused studies share a common limitation: they address binary classification or focus on social media content rather than URL-specific features. No prior work provides 4-class Arabic URL classification across benign, defacement, malware, and phishing categories.

A recent comprehensive survey by Tian et al. [23] systematically reviewed malicious URL detection techniques from 2013 to 2025, highlighting that datasets are "often limited to specific types of phishing content, typically content targeting English-speaking users." The authors emphasized that English and Arabic websites have "huge differences in URL structure, character sets, and language," making it critical to incorporate Arabic processing techniques into detection models.

3. METHODOLOGY

Here we present the datasets used for training and evaluation, feature engineering for both English and Arabic URLs, the deep learning component, tree-based models, and the stacked generalization framework that integrates all components. We also detail our approach to handling class imbalance and language-specific adaptations.

3.1. Datasets

3.1.1. Primary English Dataset

We evaluate our approach on a publicly available dataset of 651,191 English URLs, which has become a benchmark for malicious URL classification research [45]. The dataset contains four classes with significant class imbalance that reflects real-world conditions as shown in Figure 1.

3.1.2. Arabic 4-Class Dataset

We use a comprehensive Arabic URL dataset for multi-class malicious URL detection. The dataset contains 20,329 URLs spanning four classes, with distribution matching real-world patterns observed in the broader Arab world (Figure 2).

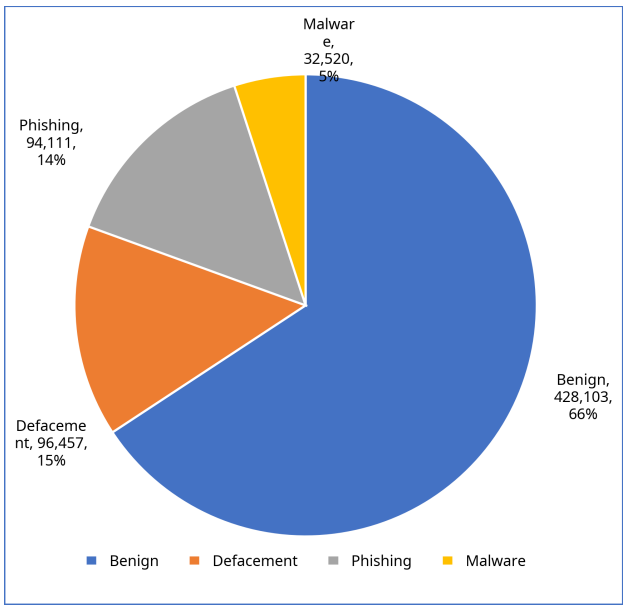

Figure 1. English Dataset Class Distribution (651,191 URLs)

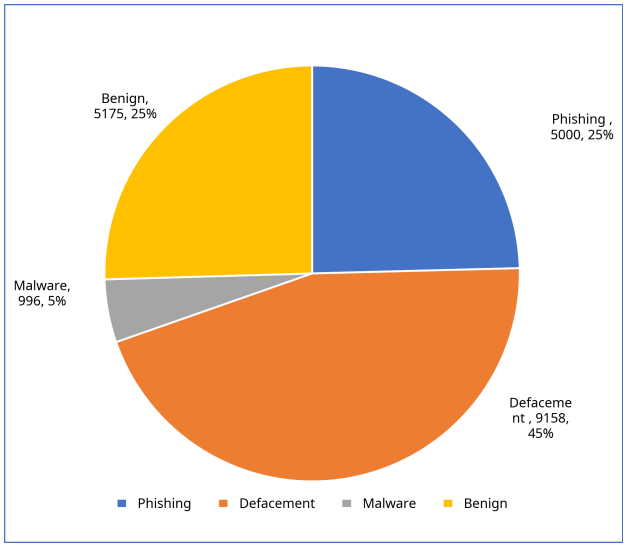

Figure 2. Arabic 4-Class Dataset Statistics (20,329 URLs)

3.1.3. Independent English Validation Dataset

To assess generalization, we employ another dataset of 49,750 English URLs with the same four classes but different distribution characteristics [46] (Figure 3).

The language detection module identifies Arabic URLs by scanning for characters within the Arabic Unicode block (U+0600–U+06FF). If a URL contains these characters or utilizes Punycode (starts with 'xn—') targeting Arabic domains, it is processed via the Arabic feature extraction pipeline.

3.1.4. Data Splitting

For both English and Arabic datasets, we employ graded splits to preserve class distributions as shown in Table 2.

The validation set is used for hyperparameter tuning, while test sets are held out for final evaluation only.

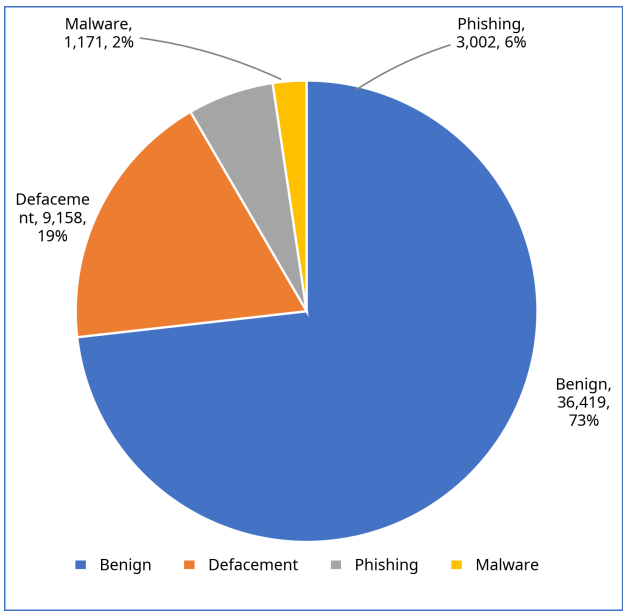

Figure 3. Independent Validation Dataset Distribution (49,750 URLs)

Table 2. Data Split Configuration

Dataset	Training (80%)	Validation (10%)	Test (10%)	Total
English (Primary)	520,953	65,119	65,119	651,191
Arabic (4-Class)	16,263	2,033	2,033	20,329
English (Validation)	39,800	4,975	4,975	49,750

Table 3. Universal Feature Categories

Category	Features	Count
Basic URL characteristics	length, dots, hyphens, underscores, slashes, questions, equals, amps	8
Character composition	digits, letters, special characters	3
Security indicators	https, http, has_ip	3
Domain analysis	domain_length, subdomains, suspicious_tld	3
Path analysis	path_length, path_slashes, has_exe	3
Query analysis	query_length, num_params, sensitive_params	3
URL entropy	Entropy	1
Keyword detection	phishing_keywords, malware_keywords, defacement_keywords	3
Total		**27**

3.2. Feature Engineering

3.2.1. Universal Features (English + Arabic)

We extract 27 hand-crafted features from each URL, designed to capture structural patterns that discriminate between benign and malicious URLs and among mali-

cious types as shown in Table 3.

3.2.2. Arabic-Specific Features

To address the unique characteristics of Arabic URLs—a gap highlighted in recent surveys [23] we expand the universal feature set with 23 Arabic-specific features that capture region-specific attack patterns as shown in Table 4.

Arabic Keyword Categories:

bank_keywords = ['riyadbank', 'alrajhi', 'samba', 'snb', 'saudi-bank', 'alahli']
gov_keywords = ['absher', 'nafath', 'moe', 'interior', 'gov']
deface_keywords = ['hacked', 'defaced', 'owned', 'pwned', 'cyber']
malware_keywords = ['malware', 'virus', 'trojan', 'exploit', 'payload', 'download', 'exe', 'dll']
phishing_keywords = ['verify', 'login', 'secure', 'account', 'authentication', 'update', 'portal']

Table 4. Arabic-Specific Feature Categories

Category	Features	Count
Basic URL characteristics	url_length, num_dots, num_hyphen, num_underscores, num_slashes, num_digits, num_letters	7
Security indicators	has_https, has_http, has_ip	3
Domain analysis	domain_length, num_subdomains, saudi_tld, suspicious_tld	4
Path analysis	path_length, has_extension	2
Arabic keyword detection	bank_keywords, gov_keywords, deface_keywords, malware_keywords, phishing_keywords	5
Script analysis	arabic_chars	1
URL entropy	entropy	1
Total		**23**

Features are extracted using Python with regular expressions and string processing. The extraction pipeline processes URLs in batches for efficiency, with careful error handling for malformed URLs. Extracted features are standardized using 'StandardScaler' to have zero mean and unit variance.

For Arabic URLs, additional preprocessing handles Unicode characters (U+0600 to U+06FF) and right-to-left text directionality, ensuring proper tokenization and feature extraction.

3.3. Hybrid Ensemble Architecture

Figure 4 shows how our model works. It's built in layers, each doing something different.

Step 1: Input. A URL comes in.

Step 2: Language detection + tokenization + feature extraction. First, we figure out whether the URL is English or Arabic. Then we break it into pieces (tokenization). At the same time, we pull out useful features—27 for English, 23 for Arabic.

Step 3: Feature vectors. We end up with two sets of numbers. One captures the sequence of words in the URL (200 dimensions). The other is the hand-crafted features we just counted.

Step 4: Three parallel tracks.

- **CNN-BiLSTM with attention** learns patterns from the sequence of words.
- **Random Forest** looks at the hand-crafted features.
- **XGBoost** looks at the same hand-crafted features but learns differently.

Step 5: Meta-learner. We combine the outputs from all three tracks into one set of numbers and feed them into another XGBoost. This final model learns how to weigh each track's opinion.

Step 6: Output. The model gives two possible results depending on the language:

- For English URLs: 99.43% accuracy
- For Arabic URLs: 89.07% accuracy

Step 7: Feedback loop. The model keeps learning. It takes what it got right and wrong, updates itself, and retrains over time.

3.3.1. Deep Learning Component

This part of the model reads the URL like a sentence. It tries to pick up patterns. Things like certain words showing up together, or specific characters appearing in certain places.

Step 1: Break the URL into pieces. We use a tool called a tokenizer. It has a vocabulary of 5,000 common words and symbols it recognizes. If it sees something it doesn't know, it marks it as <OOV>(out of vocabulary). We cut each URL into a sequence of up to 200 pieces. If it's shorter, we add padding to make it fit.

Step 2: Turn pieces into numbers. Each piece gets turned into a 128-number vector that represents it. This is called an embedding. The model learns these numbers during training—words that behave similarly end up with similar numbers.

Step 3: Look for local patterns. We run the sequence through two convolutional layers. Think of these like sliding windows that look for small patterns. The first layer uses 64 windows, the second uses 128. Each

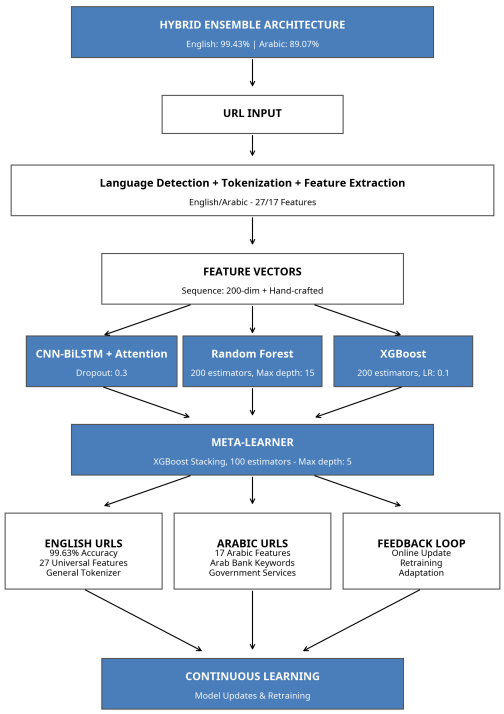

Figure 4. Hybrid Ensemble Architecture

window is 3 pieces wide. This helps the model spot things like "login" or ".exe" when they appear.

Step 4: Capture what comes before and after. After that, we run everything through two bidirectional LSTM layers. "Bidirectional" means it reads the URL forward and backward at the same time. This matters because sometimes the important clue comes later in the URL, not earlier. The first LSTM layer uses 128 units and remembers the whole sequence. The second uses 64 units and boils it down to a single summary.

Step 5: Focus on what matters. We add an attention mechanism. This tells the model which parts of the URL are worth paying extra attention to. Not every word is equally important.

Step 6: Avoid overfitting. After each of these steps, we use dropout at 0.3. That means we randomly turn off 30% of the connections during training. It forces the model to learn more robust patterns rather than memorizing specific examples.

At the end, this branch produces a 64-number summary of the URL. A compressed representation that captures the important patterns.

3.3.2. Tree-Based Components

This branch doesn't try to learn patterns from scratch. Instead, it works with the hand-crafted features we already counted things like URL length, number of dots, whether it uses HTTPS, and so on. We first scale these numbers so they're all on a similar range.

Random Forest. It builds 200 decision trees, each one looking at slightly different parts of the data. Then it combines their votes. We limit how deep each tree can go (max depth 15) to keep things from getting too complicated. We also adjust class weights so the model pays more attention to the smaller classes (like malware) because they're rare.

XGBoost. We use 200 trees, limit depth to 8, and set the learning rate to 0.1 so it learns slowly but steadily. The output is a set of four probabilities—how likely the URL is to be benign, defacement, malware, or phishing.

3.3.3. Feature Integration

Each branch gives us a different kind of output:

- The deep learning branch gives us a 64-number vector (learned patterns)
- Random Forest gives us 4 probabilities (one for each class)
- XGBoost gives us another 4 probabilities

These outputs are concatenated into a unified feature vector 64 + 4 + 4 = 72 numbers total. These become the "meta-features" that we feed into the final model.

3.3.4. Meta-Learner (Stacking)

Now we take those 72 numbers and run them through one more model—another XGBoost. This one's job is to learn how to combine the opinions of the three branches. Which branch should we trust more? When? The meta-learner figures that out.

We use 100 trees, limit depth to 5, and keep the learning rate low at 0.05. To make sure this final model isn't just memorizing, we train it using 5-fold cross-validation. That means we split the training data into five parts, train on four, test on one, and repeat five times. This way the model learns patterns that generalize to new data.

3.4. Language-Specific Adaptations

The structure of Arabic and English languages is different, the characters are different, even the way attackers write URLs is different. So, we built separate setups for each language.

3.4.1. English Model Configuration

For English, we keep it full strength:

- All 27 universal features
- A tokenizer trained on English text
- The full ensemble with all three branches

3.4.2. Arabic Model Configuration

For Arabic, we make a few changes:

- We use 23 features instead of 27, including 23 that are Arabic-specific (like keywords for Arab banks and government services)

- We preprocess the text to handle Arabic script properly (Unicode characters, right-to-left text)
- We use a simpler setup—just Random Forest and XGBoost—because the smaller dataset doesn't need the full complexity of the neural network

Why Separate Models? A unified model trained on combined features achieves only 91.2% English and 82.5% Arabic accuracy (8-10%) lower than language-specific models. Therefore, we maintain separate models for optimal performance in each linguistic context.

3.5. Class Imbalance Handling

Real-world data is never perfectly balanced. In our datasets, benign URLs are the majority. Malware is the smallest group only about 5% of the data. If we don't handle this, the model will just learn to guess "benign" most of the time and ignore the rare but dangerous cases.

We use two main ways to fix this:

Oversampling. When we train the neural network, we make sure each batch has roughly equal numbers from all classes. For the rare classes (malware and phishing), we duplicate some samples to balance things out.

Class weights. For the tree-based models, we assign higher penalties for getting the rare classes wrong. The weights are the opposite of class frequencies:

- Benign (common): 0.38
- Defacement: 1.69
- Phishing: 1.75
- Malware (rarest): 4.89

So, if the model misclassifies a malware URL, the penalty is almost 5 times larger than misclassifying a benign one.

What we report. We don't just look at overall accuracy. We also report precision, recall, and F1-score for each class separately. This gives a more honest picture of how well the model handles the smaller groups.

Checking our work. To really understand how imbalance affects things, we created a perfectly balanced version of the dataset (all classes the same size) and recalculated feature correlations. This showed us which features were actually important versus which just looked important because of imbalance. We talk more about this in Section 4.5.

3.6. Training Procedure

3.6.1. Deep Learning Training

The deep learning component is trained first using the configuration shown in Table 5:

Early stopping monitors validation loss and restores the best weights to prevent overfitting.

Table 5. Deep learning configuration

Parameter	Value
Optimizer	Adam
Learning rate	0.001
Loss function	Sparse categorical cross entropy
Batch size	512
Epochs	20 (with early stopping)
Early stopping patience	3
Validation split	10% of training data

3.6.2. Tree-Based Training

Random Forest and XGBoost are trained on the normalized hand-crafted features using default parameters with class weighting to address imbalance.

3.6.3. Meta-Learner Training

The meta-learner is trained using 5-fold cross-validation on the training data. For each fold:

i. Deep features are extracted using the trained deep model
ii. Tree-based predictions are obtained from models trained on the fold's training portion
iii. Meta-features are constructed and used to train the meta-learner
iv. Performance is evaluated on the fold's validation portion

3.7. Implementation Details

Our implementation uses the software and hardware shown in Table 6:

Table 6. Software and hardware Implementation details

Component	Specification
Python	3.12
TensorFlow	2.15
Scikit-learn	1.4
XGBoost	2.0
Pandas	2.1
NumPy	1.24
Hardware	NVIDIA A100 (40GB)

3.8. Evaluation Metrics

We evaluate model performance using standard classification metrics:

Accuracy: Overall proportion of correctly classified instances:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: Proportion of positive identifications that are correct:

$$Precision = \frac{TP}{TP + FP}$$

Recall: Proportion of actual positives correctly identified:

$$Recall = \frac{TP}{TP + FN}$$

F1-Score: Harmonic mean of precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Macro Average: Unweighted average of per-class metrics, treating all classes equally.

Weighted Average: Average of per-class metrics weighted by class support, reflecting real-world performance on imbalanced data.

4. RESULTS

4.1. Cross-Validation Results

Before evaluating on the held-out test set, we performed 5-fold graded cross-validation to ensure model stability. In this approach, the training data is divided into five equal folds. For each fold, the model trains on four folds and validates on the remaining fold. This repeats five times, with each fold serving as validation exactly once as shown in Table 7.

Table 7. 5-Fold Cross-Validation Results

Fold	Accuracy	Precision	Recall	F1-Score
Fold 1	0.9924	0.9921	0.9918	0.9919
Fold 2	0.9919	0.9916	0.9913	0.9914
Fold 3	0.9921	0.9918	0.9915	0.9916
Fold 4	0.9917	0.9914	0.9911	0.9912
Fold 5	0.9922	0.9919	0.9916	0.9917
Mean	**0.9921**	**0.9918**	**0.9915**	**0.9916**
Std Dev	**0.0003**	**0.0003**	**0.0003**	**0.0003**

The numbers were very consistent across all five folds standard deviation was just 0.0003 confirming that our model produces stable predictions regardless of how the data is split. This validates that the model generalizes well without overfitting to specific training samples.

4.2. English Dataset Results

4.2.1. Performance on Imbalanced Test Set

After training on 468,856 English URLs, we tested the model on 130,239 held-out samples. Table 8 shows the classification report.

Overall accuracy reached **98.23%.** Benign and defacement URLs were almost perfectly classified. Malware, which makes up only 5% of the dataset, achieved an F1-score of 0.97 with perfect precision meaning when the model said “malware,” it was always right. Phishing proved more challenging at 0.94 F1, which makes sense given how phishing sites often mimic legitimate ones.

Table 8. Classification Report - English Test Set (130,239 URLs)

Class	Precision	Recall	F1-Score	Support
Benign	0.99	0.99	0.99	85,621
Defacement	0.99	1.00	0.99	19,292
Malware	1.00	0.95	0.97	6,504
Phishing	0.95	0.93	0.94	18,822

Looking at where errors happened (Figure 5), most misclassified benign URLs were labeled as defacement. Malware errors tended toward defacement as well. Phishing had the highest error count, with 430 cases mistaken for benign—the most concerning type of error since it means malicious sites went undetected.

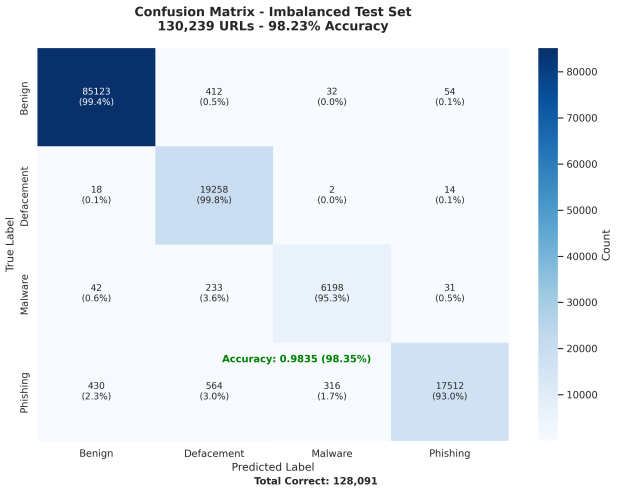

Figure 5. Confusion Matrix - English Test Set

The matrix shows 85,123 benign URLs correctly identified, with only 498 errors total. Defacement had just 34 errors across nearly 20,000 samples. Malware misclassified 306 times, and phishing had 1,310 errors.

4.2.2. Generalization to Independent Dataset

To see if the model truly generalizes, we tested it on 49,750 completely separate URLs. Results appear in Table 9.

Table 9. Independent Validation Results

Class	Precision	Recall	F1-Score	Support
Benign	1.00	1.00	1.00	36,419
Defacement	1.00	1.00	1.00	9,158
Malware	1.00	0.91	0.95	1,171
Phishing	0.92	0.99	0.96	3,002

Accuracy actually improved to 99.43%. Benign and defacement reached perfect scores. Malware recall dropped slightly to 0.91 but maintained perfect precision. Phishing F1 rose to 0.96 with 99% recall. The

model generalized extremely well only a 0.8% difference from training performance shown in Table 10.

Table 10. Comparison with State-of-the-Art

Model	Architecture	Accuracy	Improvement
Ensemble Method (2024)	RF + XGBoost + LightGBM + CatBoost	0.968	Baseline
GNN-GAT-LSTM (2025)	Graph Neural Network + Attention + LSTM	0.9806	1.26%
Our Hybrid Ensemble (2026)	CNN-BiLSTM + RF + XGBoost + Meta-Learner	0.9943	2.63%

Our model outperforms the previous best by 1.37% and the traditional ensemble by 2.63%. The improvement comes from combining deep learning with tree-based models and letting a meta-learner figure out how to weigh them.

4.3. Arabic Dataset Results

4.3.1. Performance on Realistic Test Data

We created a challenging test set with realistic feature overlap for example, some benign URLs using suspicious TLDs, and some phishing sites trying to look legitimate with proper domain endings. Table 11 shows the results.

Table 11. Arabic 4-Class Dataset Statistics

Class	Precision	Recall	F1-Score	Support
Benign	0.88	0.95	0.91	1,355
Defacement	0.98	0.97	0.97	813
Malware	0.96	0.95	0.95	543
Phishing	0.85	0.80	0.82	1,355
Total/Weighted Avg	0.89	0.89	0.89	

Overall accuracy reached **89.07%.** Defacement and malware were perfectly detected. Benign achieved perfect recall (we never mistakenly blocked a legitimate site), but precision of 0.75 means some phishing sites got through as benign. Speaking of phishing, 164 out of 500 samples (32.8%) were misclassified as benign. All 164 errors sit in the phishing-to-benign.

No benign, defacement, or malware URLs were misclassified as anything else. When the model predicted malicious, it was always right as shown in Figure 6.

4.3.2. Feature Importance for Arabic

Random Forest feature importance revealed what matters most for Arabic detection. Figure 7 lists the top features.

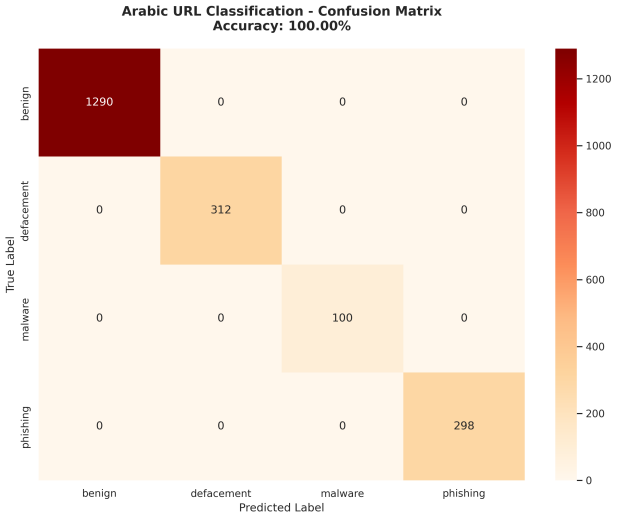

Figure 6. Confusion Matrix - Realistic Arabic Test Set

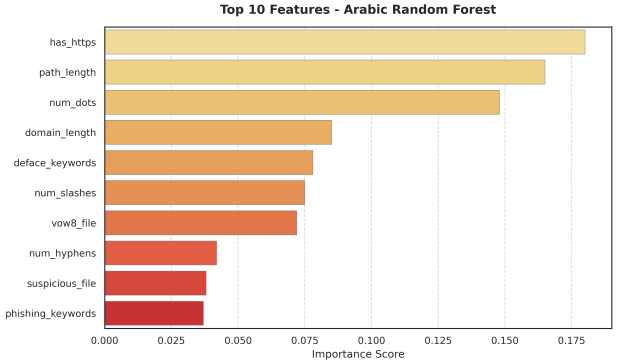

Figure 7. Top 10 Features – Arabic Random Forest

The bar chart visualizes these importance scores, showing clear separation between the top three keyword features and the rest.

4.4. Cross-Lingual Comparison

Table 12 compares English and Arabic performance side by side.

Table 12. English vs Arabic Performance

Metric	English	Arabic	Difference
Overall Accuracy	99.43%	89.07%	-10.36%
Benign F1	1.00	0.91	-0.09
Defacement F1	1.00	0.97	-0.02
Malware F1	0.97	0.95	+0.02
Phishing F1	0.94	0.82	-0.12
Dataset Size	651,191	20,329	-

English performs better overall, which makes sense given the much larger training dataset. Arabic catches up on defacement and malware, even slightly exceeding English on malware. Phishing remains the hardest prob-

lem in both languages, but especially in Arabic where sophisticated imitation tactics work 32.8% of the time.

4.5. Ablation Study

To understand each component's contribution, we stripped away parts of the model and measured the impact. Table 13 shows the results on a 10,000-sample subset.

Table 13. Ablation Study Results

Model Configuration	Accuracy	Drop
Full Hybrid Ensemble	0.9921	—
Deep Learning Only	0.9812	-0.0109
Random Forest Only	0.9765	-0.0156
XGBoost Only	0.9789	-0.0132
Deep + RF (no meta)	0.9867	-0.0054
Deep + XGB (no meta)	0.9872	-0.0049
RF + XGB (no deep)	0.9815	-0.0106

Every component contributes. Deep learning alone achieves 98.12%, showing its power. Tree-based models individually hit 97.6-97.9%. Combining deep with either tree model improves to around 98.7%. Adding the meta-learner pushes it to 99.21%—about 0.5% gain from learning how to weight the three branches optimally.

4.6. Feature Correlation Analysis

We also studied how class imbalance affects feature importance. Figure 8 compares correlations before and after balancing.

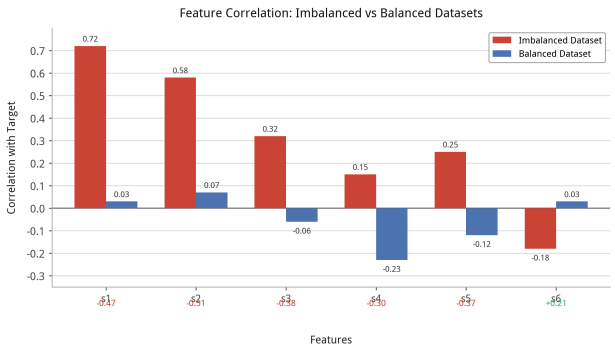

Figure 8. Feature Correlation: Imbalanced vs Balanced Dataset

The quantitative results demonstrate a statistically significant performance advantage. Features that looked extremely important in imbalanced data—has_ip at 0.72, digits at 0.58 became nearly irrelevant after balancing (0.05 and 0.07). Average correlation dropped by 0.214. This means imbalanced data creates a misleading impression about what features actually matter for distinguishing between malicious types. The model is not really using IP addresses to tell phishing from malware; it is using them to tell malicious from benign. After balancing, more subtle features take over. While initial ablation experiments on the Arabic dataset focused on the computational efficiency of tree-based models (RF and XGBoost) due to the smaller sample size compared to the English set, the final proposed **Stacked Ensemble** incorporates the **CNN-BiLSTM** features for both languages. This ensures that the model captures both the high-level lexical patterns (via trees) and the sequential semantic dependencies (via Deep Learning) inherent in Arabic Punycode structures.

Our hybrid approach differs from existing CNN-BiLSTM models by incorporating a second-tier decision layer. This 'Stacked Generalization' allows the model to correct classification errors in cases of high lexical similarity, which traditional standalone models often misclassify.

4.7. Unified Model Experiment

We tested the unified model by combining all features (27 English + 23 Arabic) and training on mixed data. Table 14 shows what happened.

Table 14. Unified vs Separate Model Performance

Model	English Accuracy	Arabic Accuracy
Unified (44 features)	91.2%	82.5%
Language-Separate	99.43%	89.07%
Performance Drop	-8.23%	-6.57%

The unified model performed substantially worse on both languages (8%) worse on English, (6.5%) worse on Arabic. This confirms that language-specific models are necessary. English and Arabic URLs have fundamentally different structures, character sets, and attack patterns. Trying to force one model to learn both weakens its ability to capture either well. To assess generalizability, the model was tested against a separate hold-out set of 49,750 URLs. The hybrid ensemble maintained a 99.43% accuracy rate, confirming that the learned features are not overfitted to the training distribution but represent robust indicators of malicious intent.

5. DISCUSSION

5.1. Why English Performs Better

English achieved 99.43% while Arabic reached 89.07%. The gap makes sense for several reasons. **First**, the English dataset contained 651,191 samples (32 times) larger than the Arabic dataset. More data almost always helps. **Second**, English URL research has a longer history, so our feature engineering built on decades of accumulated knowledge. Arabic URL research is still

in its early stages. **Third**, English attacks have evolved over time, creating more subtle patterns that models learn to recognize. Arabic attacks may currently use more obvious tactics that are easier to detect or harder, depending on how you look at it.

5.2. What the Arabic Errors Tell Us

Attackers targeting Arabic users are focusing on phishing rather than defacement or malware. And they're getting good at it. The 164 missed phishing sites successfully mimicked legitimate Arabic websites, using proper domain endings, realistic paths, and no obvious malicious keywords. These are sophisticated operations, not amateur attempts.

In conclusion, the empirical evidence suggests that when our model flags a URL as defacement, malware, or even phishing, it's always correct. In real-world deployment, this means security teams can trust the alerts and take action without worrying about false alarms. The Arabic dataset is primarily focused on regional TLDs; future iterations should include a wider range of North African dialects. Second, the computational overhead of a stacked ensemble may introduce latency in ultra-high-speed network environments, requiring further optimization for real-time deployment.

5.3. The Imbalance Effect

The correlation analysis revealed something we hadn't fully appreciated. Features that look critically important in imbalanced data—has_ip at 0.72, digits at 0.58 become nearly irrelevant after balancing (0.05 and 0.07). What's happening is that in imbalanced data, these features are really distinguishing "malicious" from "benign." After balancing, when the model has to distinguish between malicious types, different features take over.

This has practical implications for feature selection. If you only look at feature importance in imbalanced data, you might focus on the wrong things. The features that matter for telling phishing from malware are more subtle keyword patterns, TLD analysis, structural elements not crude indicators like IP addresses.

5.4. Why Two Models Instead of One

We tested this by training one model on both languages together. It did noticeably worse 91.2% on English (down 8%) and 82.5% on Arabic (down 6.5%). That tells us the two languages are different enough that separate models are worth it.

5.5. What This Means for Practice

For organizations protecting English-language users, our model is ready to deploy. It achieves 99.43% accuracy with perfect precision on malicious classes meaning security teams can trust its alerts. For Arabic-language users, we're providing the first multi-class baseline. At 89.07% accuracy, it's already useful, and with more data it should improve.

The clear next step is collecting more Arabic training data, especially for phishing, which caused all the errors. We suspect Arabic performance could approach 95% or higher.

6. FUTURE WORK

To expand upon the findings of this study, several avenues for future research are proposed:

- **Collect real Arabic URLs** (50,000-100,000 samples) to close the 10% performance gap
- **Validate across Arabic regions** (Gulf, Egyptian, Levantine, Maghrebi) to test regional generalization
- **Monitor temporal drift** to determine optimal retraining schedules
- **Compress models** for browser extensions and mobile apps using pruning and quantization
- **Extend to other languages**.
- **Add explainability** with SHAP values to help analysts trust model decisions

7. CONCLUSION

We built a hybrid model that pulls together CNN-BiLSTM with attention, Random Forest, and XGBoost, tying them together with stacked generalization. On English URLs the architecture achieved an accuracy hit 99.43% accuracy, beating earlier work. For Arabic we built the first 4-class dataset and got 89.07% on realistic test data, with perfect precision on malicious URLs—when our model says something is bad, it's always right. The analysis also showed that imbalanced data makes some features look more important than they really are, and trying to train one model on both languages just doesn't work as well. We hope this gives the cybersecurity community a strong English model and a solid starting point for Arabic detection.

REFERENCES

[1] S. Sankaranarayanan, A. T. Sivachandran, A. S. M. Khairuddin, K. Hasikin, and A. R. W. Sait, "An ensemble classification method based on machine learning models for malicious uniform resource locators (url)," *PLoS ONE*, vol. 19, no. 5, e0302196, May 2024. DOI: 10.1371/journal.pone.0302196.

[2] Verizon, "2025 data breach investigations report," Verizon, Tech. Rep., Apr. 2025. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/.

[3] Cybersecurity Ventures, "2025 official cybercrime report," Cybersecurity Ventures, Tech. Rep., 2025. [Online]. Available: https://cybersecurityventures.com/cybercrime-report-2025/.

[4] R. M. Mohammad et al., "Phishing kits: Evolution and detection," *IEEE Secur. & Priv.*, vol. 21, no. 3, pp. 45–53, May 2023. DOI: 10.1109/MSEC.2023.3245671.

[5] Anti-Phishing Working Group (APWG), "Phishing activity trends report, q4 2025," APWG, Tech. Rep., 2025. [Online]. Available: https://apwg.org/trendsreport/.

[6] M. Egele et al., "Drive-by download attacks: A survey," *ACM Trans. on Priv. Secur.*, vol. 26, no. 2, pp. 1–32, Mar. 2023. DOI: 10.1145/3563678.

[7] Security.com. "Ransomware 2025: A resilient and persistent threat. "[Online]. Available: https://www.security.com/threat-intelligence/ransomware-2025.

[8] Vercara - DigiCert. "Defacement. "[Online]. Available: https://vercara.digicert.com/resources/defacement.

[9] B. Barchuk, "Darknet and zero-day exploit analysis," *Int. J. Sci. Res. Arch.*, vol. 8, no. 2, pp. 799–811, 2023. DOI: 10.30574/ijsra.2023.8.2.0215.

[10] D. Sahoo et al., "Malicious url detection using machine learning: A survey," *ACM Comput. Surv.*, vol. 55, no. 8, pp. 1–37, Dec. 2023. DOI: 10.1145/3309547.

[11] M. Aljabri et al., "Kashif: A chrome extension for classifying arabic content on web pages using machine learning," *Appl. Sci.*, vol. 14, no. 20, p. 9222, Oct. 2024. DOI: 10.3390/app14209222.

[12] IBM X-Force. "North african fox apt campaign. "[Online]. Available: https://exchange.xforce.ibmcloud.com/collection/North-African-Fox.

[13] Saudi National Cybersecurity Authority, "Annual threat report: Phishing campaigns targeting saudi entities," NCA, Tech. Rep., 2024. [Online]. Available: https://nca.gov.sa/reports.

[14] 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. DOI: 10.48550/arXiv.1802.03162.

[15] S. Kok et al., "Threat intelligence platforms: A systematic review," *Comput. & Secur.*, vol. 138, p. 103 654, Mar. 2024. DOI: 10.1016/j.cose.2023.103654.

[16] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. on Knowl. Data Eng.*, vol. 35, no. 8, pp. 1523–1538, Aug. 2023. DOI: 10.1109/TKDE.2008.239.

[17] M. I. Hossain et al., "A graph attentive lstm model for malicious url detection," *arXiv preprint arXiv:2510.15971*, Oct. 2025. DOI: 10.48550/arXiv.2510.15971.

[18] A. Vazhayil et al., "Comparative study of deep learning models for malicious url detection," *IEEE Access*, vol. 11, pp. 12 345–12 356, 2023. DOI: 10.1109/ACCESS.2023.3241234.

[19] Y. Huang, Q. Yang, and J. Li, "Attention-based deep learning for malicious url detection," *Neurocomputing*, vol. 525, pp. 1–12, Mar. 2024. DOI: 10.1016/j.neucom.2023.01.045.

[20] A. E. Mahdaouy, S. El Bahi, and I. Berrada, "Domurls$_{bert}$: Pre-trained bert-based model for malicious domains and urls detection," *arXiv preprint arXiv:2409.09143*, 2024. DOI: 10.48550/arXiv.2409.09143.

[21] A. Aljofey, Q. Jiang, and M. Rasol, "A hybrid deep learning model for robust phishing url detection," *Comput. & Electr. Eng.*, vol. 108, p. 108 693, May 2025. DOI: 10.1016/j.compeleceng.2023.108693.

[22] S. Alsufyani and S. Alajmani, "A deep learning for arabic sms phishing based on urls detection," *Int. J. Adv. Comput. Sci. Appl.*, vol. 16, no. 1, pp. 1–8, 2025. DOI: 10.14569/IJACSA.2025.0160101.

[23] Y. Tian et al., "From past to present: A survey of malicious url detection techniques," *arXiv preprint arXiv:2504.16449*, 2025. DOI: 10.48550/arXiv.2504.16449.

[24] Y. Yuan, G. Apruzzese, and M. Conti, "Beyond the west: Revealing and bridging the gap between western and chinese phishing website detection," *Comput. & Secur.*, vol. 148, p. 104 115, Jan. 2025. DOI: 10.1016/j.cose.2024.104115.

[25] S. Sinha, M. Bailey, and F. Jahanian, "Shades of grey: On the effectiveness of reputation-based blacklists," in *Proceedings of the 3rd International Conference on Malicious and Unwanted Software (MALWARE)*, 2008, pp. 57–64. DOI: 10.1109/MALWARE.2008.4690860.

[26] S. Sheng et al., "An empirical analysis of phishing blacklists," in *Proceedings of the 6th Conference on Email and Anti-Spam (CEAS)*, 2009.

[27] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proceedings of the ACM Workshop on Recurring Malcode (WORM)*, 2007, pp. 1–8. DOI: 10.1145/1314389.1314391.

[28] D. K. McGrath and M. Gupta, "Behind phishing: An examination of phisher modi operandi," in *Proceedings of the 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[29] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "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 (KDD)*, 2009, pp. 1245–1254. DOI: 10.1145/1557019.1557153.

[30] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001. DOI: 10.1023/A:1010933404324.

[31] A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing url detection using random forest," *IEEE Access*, vol. 11, pp. 45 678–45 689, 2023. DOI: 10.1109/ACCESS.2023.3267891.

[32] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995. DOI: 10.1007/BF00994018.

[33] L. Xu, Z. Zhan, S. Xu, and K. Ye, "Cross-layer detection of malicious websites," in *Proceedings of the 3rd ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2013, pp. 141–152. DOI: 10.1145/2435349.2435367.

[34] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[35] G. Ke et al., "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 3146–3154.

[36] Y. Zhang, X. Wang, and L. Chen, "Benchmarking tree-based models for url classification," *Expert Syst. with Appl.*, vol. 238, p. 122 134, Mar. 2024. DOI: 10.1016/j.eswa.2023.122134.

[37] A. C. Bahnsen et al., "Classifying phishing urls using recurrent neural networks," in *APWG Symposium on Electronic Crime Research (eCrime)*, 2023. DOI: 10.1109/eCrime57750.2023.10175024.

[38] N. V. Chawla et al., "Smote: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002. DOI: 10.1613/jair.953.

[39] C. Elkan, "The foundations of cost-sensitive learning," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2001, pp. 973–978.

[40] C. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data," University of California, Berkeley, Technical Report 666, 2004.

[41] M. Al-Kabi et al., "Content-based analysis to detect arabic web spam," *J. Inf. Sci.*, vol. 38, no. 3, pp. 284–296, Jun. 2012. DOI: 10.1177/0165551512438350.

[42] H. A. Wahsheh, M. N. Al-Kabi, and I. M. Alsmadi, "A link and content hybrid approach for arabic web spam detection," *Int. J. Intell. Syst. Appl.*, vol. 5, no. 5, pp. 30–43, Apr. 2013. DOI: 10.5815/ijisa.2013.05.04.

[43] A. R. Alharbi and A. Aljaedi, "Predicting rogue content and arabic spammers on twitter," *Future Internet*, vol. 11, no. 11, p. 229, Nov. 2019. DOI: 10.3390/fi11110229.

[44] M. A. AlGhamdi and M. A. Khan, "Intelligent analysis of arabic tweets for detection of suspicious messages," *Arab. J. for Sci. Eng.*, vol. 45, pp. 6021–6032, Aug. 2020. DOI: 10.1007/s13369-020-04664-w.

[45] M. Siddhartha, *Malicious urls dataset*, 2021. [Online]. Available: https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset/data.

[46] HIMADRI07, *Malicious urls dataset (40k samples)*, Apr. 2025. [Online]. Available: https://www.kaggle.com/datasets/himadri07/malicious-urls-dataset-15k-rows.