From Past to Present: A Survey of Malicious URL Detection Techniques, Datasets and Code Repositories⋆

Ye Tian[a], Yanqiu Yu[a], Jianguo Sun[a] and Yanbin Wang[a,*]

[a]Hangzhou Institute of Technology,Xidian University,Hangzhou,310000,Zhejiang,China

ARTICLE INFO

Keywords:
Malicious URL detection
multimodality
data
code
machine learning

ABSTRACT

Malicious URLs persistently threaten the cybersecurity ecosystem, by either deceiving users into divulging private data or distributing harmful payloads to infiltrate host systems. The detection of malicious URLs is a protracted arms race between defenders and attackers. Gaining timely insights into the current state of this ongoing battle holds significant importance. However, existing reviews exhibit 4 critical gaps: 1) Their reliance on algorithm-centric taxonomies obscures understanding of how detection approaches exploit specific modal information channels; 2) They fail to incorporate pivotal LLM/Transformer-based defenses; 3) No open-source implementations are collected to facilitate benchmarking; 4) Insufficient dataset coverage.

This paper presents a comprehensive review of malicious URL detection technologies, systematically analyzing methods from traditional blacklisting to advanced deep learning approaches (e.g. Transformer, GNNs, and LLMs). Unlike prior surveys, we propose a novel modality-based taxonomy that categorizes existing works according to their primary data modalities (URL, HTML, Visual, etc.). This hierarchical classification enables both rigorous technical analysis and clear understanding of multimodal information utilization. Furthermore, to establish a profile of accessible datasets and address the lack of standardized benchmarking (where current studies often lack proper baseline comparisons), we curate and analyze: 1) publicly available datasets (2016-2024), and 2) open-source implementations from published works(2013-2025). Then, we outline essential design principles and architectural frameworks for product-level implementations. The review concludes by examining emerging challenges and proposing actionable directions for future research. We maintain a GitHub repository for ongoing curating datasets and open-source implementations: https://github.com/sevenolu7/Malicious-URL-Detection-Open-Source/tree/master.

1. Introduction

In the modern digital era, millions of people engage in global interactions through social networking sites. However, this widespread connectivity also brings concerns about privacy and security. With the proliferation of internet applications, cyber attacks are also on the rise, with attackers utilizing methods such as software distribution, spam emails, and phishing to exploit for personal gain. Unfortunately, as technology advances, these attacks are becoming more complex. They include creating fake websites to sell counterfeit goods, manipulating users to disclose sensitive information for financial fraud, and compromising systems through the installation of malicious software. Attackers employ various techniques such as hacking, drive-by downloads, social engineering, and phishing to pose serious threats to online security[1]. Users may receive deceptive emails with links disguised as legitimate websites, offering false information such as company details, job opportunities, or online sales. This can lead users to unknowingly access seemingly valuable content that is[2]. in fact, malicious. Malicious URLs are used to redirect users and compromise system security or gain access to personal information[3].

A Uniform Resource Locator (URL) is a web address used to indicate the location of a resource on the Internet. A URL has two main components: (1) a protocol identifier (indicating what protocol is used) and (2) a resource name (specifying the IP address or domain name where the resource is located). The protocol identifier and resource name are separated by a colon and two forward slashes, such as "https://www.google.com", as shown in Figure 1. On the other hand, malicious URLs are online addresses used to harm or exploit users. These URLs often point to websites designed to distribute malware, steal confidential data, or perform other destructive actions. Clicking on a malicious URL can lead to cyberattacks, data theft, and security vulnerabilities. Since they often disguise themselves

*Corresponding author at: Hangzhou Institute of Technology,Xidian University,Hangzhou,310000,Zhejiang,China
tianye@xidian.edu.cn (Y. Tian); 24241215222@stu.xidian.edu.cn (Y. Yu); jgsun@xidian.edu.cn (J. Sun); wangyanbin15@mails.ucas.ac.cn (Y. Wang)

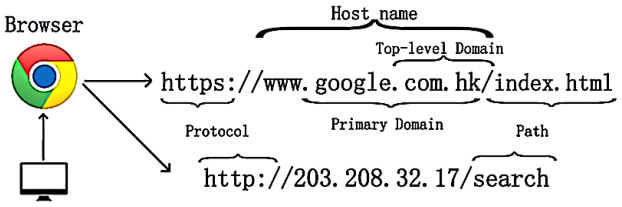

Figure 1: Example of URL

as trustworthy websites, they pose a threat to cautious users. According to data from the Anti-Phishing Working Group (APWG) [4], there is a clear upward trend in the number of phishing websites from Q1 2021 to Q1 2024. Although there are fluctuations in each quarter, the overall growth highlights the growing threat of phishing in recent years.

Malicious URL attacks can even cause billions of dollars in losses worldwide every year. Therefore, different methods have been developed around the world to detect such malicious URLs. Initially, people used the blacklist method, which involves listing known malicious URLs to easily identify harmful URLs. Due to its simplicity, it is considered a traditional method for URL detection. However, there are several problems with the blacklist method. Its lack of ability to detect newly emerging malicious URLs has led people to adopt heuristic methods, which are advanced techniques of the blacklist method and are designed to identify common attacks. Nevertheless, this method cannot defend against all types of attacks, resulting in limited use. With the accumulation of experience, researchers introduced machine learning techniques, which, through several learning stages, can accurately detect malicious URLs.

While current reviews cover traditional algorithmic developments, they lack investigation in several critical aspects:

1. Absence of Modality-Based Taxonomy: Existing reviews employ algorithm-focused taxonomy rather than systematically organizing works by their fundamental data modalities (e.g. URL strings, HTML structures, webpage visual). This obscures comprehensive awareness of available modalities and systematic understanding of how detection approaches utilize these distinct information.
2. Inadequate Coverage of Emerging Paradigms: Most surveys[1, 5, 6, 7, 8, 9] fail to address transformative advances in graph neural network(GNN), Transformer and Large Language model(LLM)-powered detection.
3. Lack of records for code repositories and available datasets:Existing reviews fail to survey code repositories. Additionally, their examination of available datasets is insufficient. This is a critical oversight that hinders benchmark development. Centralized references would enable proper performance comparisons and baseline standardization, essential for advancing detection research.

This survey provides a systematic examination of malicious URL detection methodologies, with dedicated analysis of understudied Transformer architectures, graph neural networks, and LLM-based approaches. Departing from conventional algorithm-centric classifications, we introduce a modality-driven taxonomy that explicitly maps detection techniques to their core data sources—URL lexemes, HTML DOM structures, JavaScript execution traces, visual renderings, and multimodal combinations—thereby elucidating both the technical implementations and operational contexts of each paradigm. Our analysis explicitly details how each modality is computationally exploited across different methods, revealing how specific data characteristics influence detection effectiveness. We subsequently curate and benchmark accessible code repositories (2013-2025) and public datasets to establish reproducible baselines. This consolidated resource framework enables: (1) meaningful cross-method performance evaluations, (2) tracking of genuine methodological advancements. The review concludes by identifying persistent challenges and proposing strategic research directions to overcome current limitations.

2. Background Study

This section mainly discusses the possible types of attacks that attackers may carry out through URLs, as well as the general workflow of malicious URL detection using machine learning and the machine learning algorithms commonly used today.

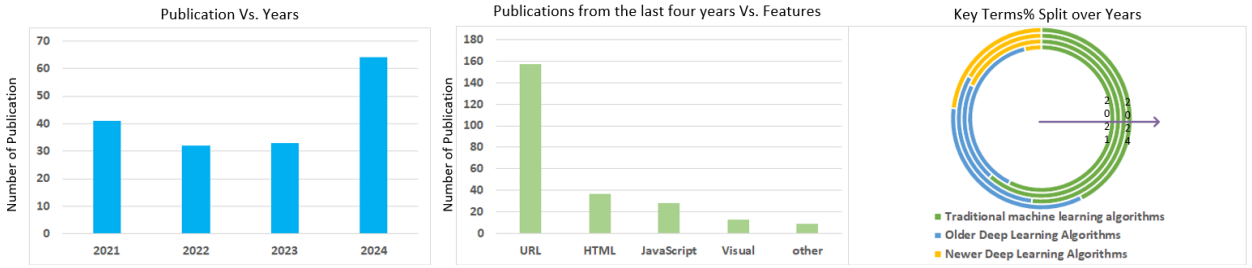

Figure 2: Count the number of papers published on the topic of malicious URL detection (Web of Science Core Collection) over the past four years, the features used in the publications, and the percentage of algorithms used

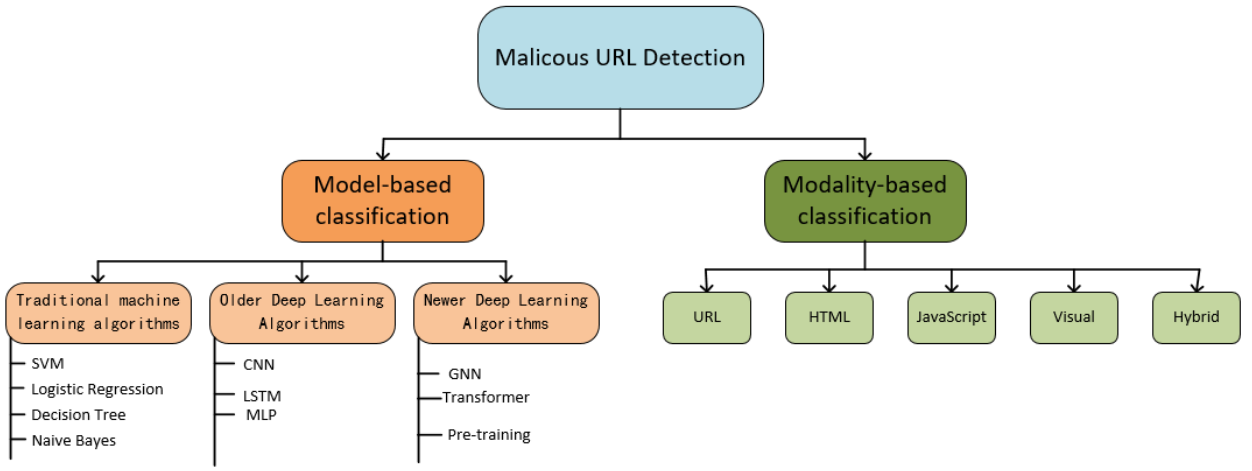

Figure 3: The proposed taxonomy of malicious url detection

2.1. URL Attack Types

2.1.1. Attacks via spam URL

Spam URL attacks involve the use of URLs in emails, forums, or websites to spread unwanted or unwelcome content, often with false or commercial intent. When this attack occurs, hackers design web pages with the intent of manipulating web browser engines to mistakenly identify these pages as legitimate pages when, in fact, they are not [5]. These transmissions are primarily emails and often contain links to websites under the attacker's control, with the intent of doing one of three things: impersonating well-known websites to obtain user credentials; implanting malware on the user's computer; or distributing spam to users [10].

2.1.2. Attacks via spam Malware

The main purpose of malware attacks is to steal sensitive user information or gain unauthorized access to a system. Malicious URL attacks direct users to harmful websites that install malware on their devices. This malware can facilitate behaviors such as file corruption, keylogging, and even identity theft. A common form of malware, known as a "drive-by download," occurs when users unknowingly download malware after visiting a deceptive website, which can cause significant damage to their computers and personal information[11].

2.1.3. Attacks via spam Phishing URL

Phishing is a social engineering attack method that tricks individuals into entering their login credentials through fake login forms, which then send the information to a malicious server[12]. These malicious URLs can be spread in both public and private environments. If steps are not taken to restrict or eliminate these malicious URLs, users' credentials may be stolen by attackers[13]. Attackers can use this information for financial gain, identity theft, or unauthorized access to accounts, resulting in financial losses, identity theft, and confidential information leakage.

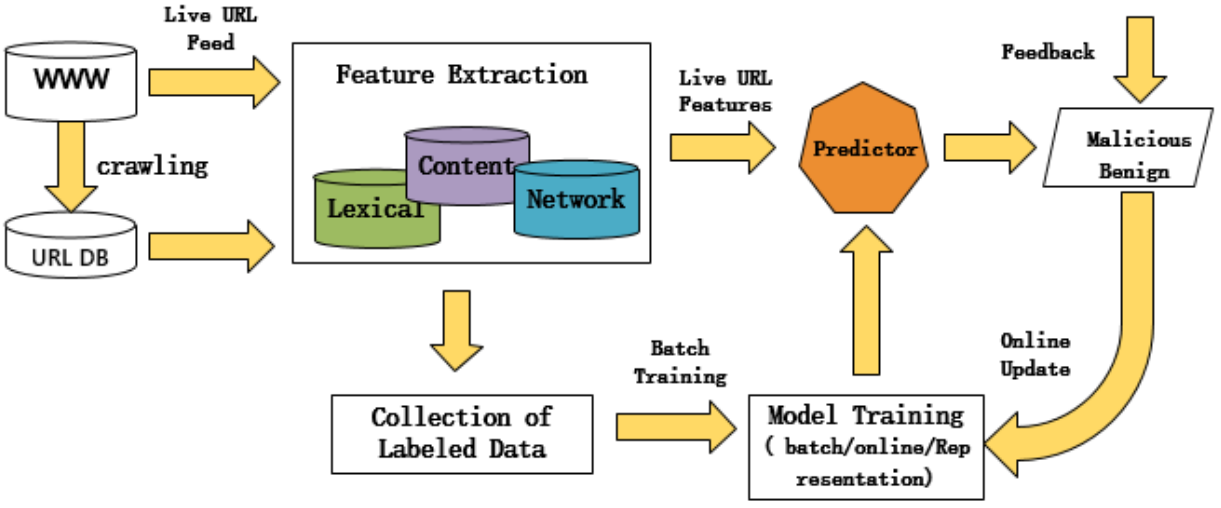

Figure 4: A general processing framework for Malicious URL Detection using Machine Learning

2.1.4. Attacks via spam Defacement URL

Website defacement attacks involve making unauthorized changes to a website's appearance or content, typically by replacing legitimate elements with the attacker's message or image. These attacks can be motivated by a variety of reasons, such as making a political statement, demonstrating hacker prowess, or personal vendettas. They can have serious consequences, including damage to an organization's reputation, loss of user trust, and possible disruption of online services.[5] Hacktivists often use website defacement as a tool to promote their sociopolitical and ideological goals. Samuel et al. claim that this requires hacking into a web server and replacing a page with one containing a statement reflecting those views. Many of the vandalism that occurred in 2004 may have been directed at specific organizations, often governments or companies, in an attempt to draw attention to and protest their actions.

2.2. Machine Learning in Malicious URL Detection

In this section, we will first explain the basic general process of using machine learning algorithms to classify malicious URLs, and then introduce some common machine learning algorithms that use URL feature classification.

Figure 4 shows the general workflow of malicious URL detection using ML. We formalize the malicious URL detection task as a binary classification problem, where the goal is to distinguish between "malicious" and "benign" URLs. Specifically, given a dataset containing T samples, each sample is represented as $\{(u_1, y_1), \ldots, (u_T, y_T)\}$, where u_T represents the URL and y_T is its corresponding label $y_t \in \{1, -1\}$, 1 represents a malicious URL, and -1 represents a benign URL. The key to automatic malicious URL detection lies in two aspects:

(I) Feature representation: extracting an appropriate feature representation: $u_T \rightarrow x_T$, where $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional feature vector representing a URL; and

(II) Machine learning: learning a prediction function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ that uses the appropriate feature representation to predict the class assignment of any URL instance x.

The first step involves two key methods: tokenization and vectorization, and lexical feature selection. Tokenization is a mechanism that breaks down the URL into meaningful substrings using specific characters such as slashes, dashes, and dots. After tokenization, the data can be converted into a sparse matrix vector using TfidfVectorizer [14], which fits well with ML frameworks. Meanwhile, lexical feature selection requires identifying relevant features based on the lexical properties of the URL, which cannot be directly calculated through mathematical functions (most of them cannot). It is necessary to use domain knowledge and relevant expertise to construct feature representations by grabbing all relevant information about the URL. These range from lexical information (length of URL, the words used in the URL, etc.) to network information (WHOIS info, IP address, location, etc.). Once the information is collected, it is

processed to be stored in a feature vector x. Numerical features can be stored in x as is, identity-related information or lexical features are usually stored through binarization or bag-of-words (BoW) methods.

After extracting the feature vector x of the training data, the process of learning the prediction function $f : \mathbb{R}^d \to \mathbb{R}$ is usually transformed into an optimization problem, whose goal is to minimize the loss function or maximize the detection accuracy. The prediction function f is usually parameterized by a d-dimensional weight vector w in the form of $f(\mathbf{x}) = (\mathbf{w}^\top \mathbf{x})$. For a given feature vector x_T, the predicted class label is expressed as $\hat{y}_t = \text{sign}(f(\mathbf{x}_t))$. The number of prediction errors on the entire training dataset can be calculated by the following formula: $\sum_{t=1}^{T} \mathbb{I}_{\hat{y}_t = y_t}$, Where $\mathbb{I}$ is an indicator function that takes a value of 1 when the condition is true and 0 otherwise. Since the indicator function is non-convex, it may be difficult to directly optimize this objective. Therefore, a convex loss function $\ell(f(x), y)$ is usually introduced to formulate the optimization problem as:

$$\min_{\mathbf{w}} \sum_{t=1}^{T} \ell(f(\mathbf{x}_t), y_t) \tag{1}$$

In order to adapt to different learning objectives and data characteristics, a variety of loss functions can be selected. For example, the commonly used hinge loss function is defined as $\ell(f(x), y) = \frac{1}{2}\max(1 - yf(x), 0)$, while the square loss function is expressed as $\ell(f(x), y) = \frac{1}{2}(f(x) - y)^2$. In some cases, regularization terms are added to the loss function to prevent model overfitting or learn sparse models. In addition, when there is class imbalance in the data or different threats have different costs, the loss function can also be adjusted for cost sensitivity.

Researchers often deploy a diverse array of ML models or hybrid approaches that amalgamate multiple classifiers to detect malicious URLs effectively.Notable classifiers used in this domain encompass Support Vector Machine (SVM), Random Forest (RF),Naïve Bayes (NB), Long Short-Term Memory(LSTM), Logistic Regression (LR), Gradient Boosting (GB), and Decision Tree (DT). In tandem, DL techniques, including Convolutional Neural Network (CNN), K-means clustering, Reinforcement Learning, K-Nearest Neighbors (KNN), Deep Q-Networks, Multi-Layer Perceptron (MLP), Natural Language Processing (NLP), and Bidirectional Encoder Representations and Transformers (BERT) are progressively being applied to bolster the detection capabilities for malicious URLs.

Several research endeavors have shed light on the efficacy of various ML algorithms in the context of identifying malicious URLs. For instance, Shayan Abad and team managed to attain an accuracy rate of 92.18% using Random Forest[15], showcasing its proficiency in this domain. Additionally, studies conducted by May et al. emphasized social semantic attacks through models like LSTM, CNN, and CharacterBERT[16], thereby highlighting the significance of character-aware language models in URL-based detection mechanisms. Malak et al. undertook comparative analyses utilizing algorithms such as CNN, LSTM, NB, and RF, where NB emerged as the frontrunner with an impressive 96.01% accuracy rate[17]. Furthermore, exploration into phishing URL detection strategies revealed the prowess of models like CS-XGBoost, which registered an exceptional accuracy rate of 99.05%[18].

3. Multimodal Classification of Malicious URL Detection

When classifying malicious URL detection methods, we mainly classify them into five categories based on data modality, that is, the category of extracted features: URL features, HTML features, JavaScript features, visual features, and mixed features. These classifications cover most of the existing research work. Of course, in addition to the above five categories of features, there are also researchers who use network traffic features and user behavior features to detect malicious URLs. However, there are relatively few such analysis methods, so we will briefly discuss these methods in the last part of this section. And we will explain each algorithm in detail for the first time.

3.1. URL-Based Detection

3.1.1. Technical Principles

The technical principle of using URL features to detect phishing websites mainly relies on analyzing the structure, content, and similarity of URLs with known legitimate and malicious URLs. Phishing website URLs usually imitate legitimate website URLs through various means to trick users into clicking, so the detection system will carefully check each component of the URL, including the protocol, domain name, path, query parameters, etc., to find abnormal or malicious patterns.

URL features mainly include:

- Lexical features (referring to static features extracted from URL strings): URL length, domain length, TLD (top-level domain), special characters, combinations of numbers and letters, number of subdomains, IP addresses in URLs, paths and parameters in URLs, keywords in URLs, double slashes (//) in URLs, redirects in URLs, and the use of HTTP and HTTPS in URLs. These features focus on the character composition, structural patterns, and differences from normal URLs without accessing the actual content to which the URL points. These features can be used alone or in combination.

- Network features: It is an important component of assessing the security of URLs, including domain name registration time, IP address reputation, and server location information. By analyzing WHOIS records, we can obtain information about the domain name owner, thereby determining the credibility and potential risks of the URL. These network features are crucial for identifying potentially dangerous websites. Specifically, network features include DNS resolution, network performance, and host quality. In threat assessment, indicators such as the number of resolved IP addresses, latency, number of redirects, domain name resolution time, number of DNS queries, connection speed, and open ports [5] are very useful. Next, we will introduce some URL-Based Detection methods.

3.1.2. Detection Methods

- Blacklist Approach:Currently, the most commonly used method for detecting malicious URLs is blacklisting technology. This is a classic detection method. According to the research by Jian Zhang et al. [19], blacklisting technology relies on a database that contains a collection of URLs that have been marked as malicious in the past. When a user accesses a new URL, the system searches the blacklisting database. If the URL is found in the blacklisting database, the system issues a warning indicating that the URL may have malicious behavior; if it is not found in the blacklisting database, it is considered safe. This method has a low false positive rate. However, attackers may take various measures to evade blacklisting detection.

  S.Sinha et al. proposed a reputation-based blacklist method that can identify compromised hosts as well as malicious content in URLs, networks, and hosts[3]. With this method, activities such as web browsing, email access, and other interactions within malicious networks or hosts can be prevented. Many organizations, such as intrusion detection and spam filtering, have adopted this approach. SpamAssassin and DSpam are two spam detectors used to identify malicious URLs in user emails. DSpam requires manual training for detection, while SpamAssassin utilizes multiple spam detectors and assigns scores to each detector. According to their perspective, blacklisted database searches are performed in reverse by IP addresses, where blacklisted regions can be appended, followed by DNS lookups. However, this method also faces several challenges, with the most significant drawback being the difficulty in maintaining a comprehensive list of malicious URLs.The blacklist method is ineffective in predicting newly generated URLs. According to the viewpoint of S. Sheng et al.[20], it is impossible to detect new threats in URLs generated daily. Another drawback of the blacklist method is that it hinders signature-based tools from detecting attacks by complicating the code. Attackers launch more attacks and modify attack signatures.

- Heuristic Approach: This method analyzes features observed in actual phishing attacks to detect zero-hour phishing threats. However, since these features may not always be present, it can result in a significant false positive rate for detection. Although this approach provides versatile protection against evolving threats, further improvements may be necessary to achieve greater accuracy[13, 21].

  Nguyen et al. introduce a heuristic-based detection technique that analyzes and extracts features specific to phishing sites[22]. By evaluating features of user-requested URLs, this method effectively identifies and mitigates potential phishing attacks, ultimately minimizing their impact.In [23], researchers analyzed the various components of the URL, including the protocol, subdomain, primary domain, top-level domain, and path, to extract relevant features and calculate the heuristic value of each feature. On this basis, combined with ranking information such as PageRank, AlexaRank, and AlexaReputation, a comprehensive evaluation is made on whether the website is a phishing website. However, these weights are based on a specific data set and may not be applicable to other data sets, and may also change over time. Reference [24] proposed a data-driven method for dynamically adjusting feature weights, such as automatically learning feature weights through machine learning algorithms, thereby improving the adaptability and accuracy of the model. In addition, the heuristic methods are divided into three categories: URL blacklist bypass, URL morphology, and user susceptibility, each of which

targets different URL features. The feature analysis method in [25] combines multi-dimensional features such as domain name creation date, domain name expiration date, special characters in the URL (such as "@", "-", the number of dots), and WHOIS information.

Through machine learning algorithms, features can be trained and classified to effectively detect and classify malicious URLs [26]. However, machine learning algorithms cannot use the original URL directly and must process the URL string to extract useful features. Sahingoz et al. [27] introduced a real-time anti-phishing system that applied seven different classification algorithms and natural language processing (NLP)-based features extracted from URLs. The system demonstrated advantages such as language independence, real-time execution, minimal dependence on external services, and high accuracy. The random forest classifier with only NLP features performed best and achieved very high accuracy in detecting phishing URLs in their experiments. To achieve this goal, many studies have used NLP techniques such as counting, hashing, and Bag-of-Words and classifiers such as random forests and neural networks to analyze URL datasets.

Bag-of-Words model is a common technique. The model represents the URL by splitting the URL string into words (with special characters as delimiters) and building a bag-of-words model. If a word appears in the URL, the corresponding feature value is marked as 1, otherwise it is 0 [28]. This method can effectively capture specific lexical patterns that appear in URLs, but it also has certain limitations. Specifically, it loses information about the order in which words appear in URLs, which may be important for URL classification in some cases. To solve this problem, some improvements were proposed in [29] and [30]. Reference [29] combines features such as WHOIS information, IP address attributes, and geographic location information with lexical features to further improve classification performance. Reference [30] distinguishes between different components of URLs, processes tokens belonging to host names, paths, top-level domains, and primary domain names separately, and provides a separate dictionary for each component. This distinction not only retains certain orders of occurrence of these words, but also enhances the model's understanding of URL structure. For example, it can distinguish "com" in the top-level domain from other parts of the URL, thereby more accurately capturing the characteristics of the URL. In addition, the study also introduced the concept of dynamic feature set, that is, as new malicious URLs appear, the model will introduce new feature words in real time to adapt to the changing distribution of malicious URL features. This method not only improves the adaptability of the model, but also enhances its ability to detect emerging threats.

In addition to the techniques mentioned above, there are several other methods that deserve attention. For example, the n-gram-based model considers word sequences (such as bigrams, trigrams, etc.) in feature extraction, rather than just single words. This method can better capture the contextual information of words in the URL, thereby significantly improving the expressiveness of the features [28]. Specifically, a fixed n value (such as 3, 4, or 5) can be selected to split each word or character sequence in the text into subsequences of length n, and these subsequences are used as features to construct feature vectors.

Furthermore, some researchers have proposed a feature extraction method based on Kolmogorov complexity [31]. The uniqueness of this method is that it does not require in-depth knowledge of the structural characteristics of the URL. Kolmogorov complexity is used to measure the complexity of a string, while conditional Kolmogorov complexity is a measure of the complexity of a string when another string is known. Specifically, this means that the existence of a known string does not increase the complexity of the original input string. Based on this principle, for a given URL, its conditional Kolmogorov complexity relative to a set of benign URLs and a set of malicious URLs can be calculated. By combining these metrics, it is possible to effectively determine the similarity of a given URL with a malicious URL database or a benign URL database. However, it is worth noting that although this feature is very useful in theory, it may face scalability challenges when processing large-scale URL data. To address this challenge, references [32] and [33] further defined a new concept of URL internal correlation to measure the relationship between different words in a URL, especially the association between the registered domain name and the rest of the URL. In addition, reference [21] also proposed new distance measurement methods, such as domain brand name distance and path brand name distance. These edit distance-based metrics are designed to detect malicious URLs that attempt to imitate well-known brands or websites. Next, I will introduce some machine learning algorithms for commonly used URL features.

- LR: LR is a famous discriminant model, which calculates the conditional probability that the feature vector x is classified as y = 1 by

$$P(y = 1 \mid \mathbf{x}; \mathbf{w}, b) = \sigma(\mathbf{w}^\top \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}} \tag{2}$$

Based on maximum likelihood estimation (equivalently defining the loss function as the negative log-likelihood), the optimization of LR can be stated as

$$(\mathbf{w}, b) \leftarrow \arg\min_{\mathbf{w},b} \frac{1}{T} \sum_{t=1}^{T} -\log P(y_t|\mathbf{x}_t; \mathbf{w}, b) + \lambda \mathcal{R}(\mathbf{w}) \tag{3}$$

The regularization term can be the L2-norm $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_2$ or the L1-norm $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_1$ to achieve a sparse model for high-dimensional data. LR has been a popular learning method for malicious URL detection [29, 34, 35, 36, 37, 38, 39]

- NB: It is a simple and effective classification algorithm based on Bayesian theorem, which is widely used in text classification, spam detection, and malicious URL detection. It assumes that the features are independent of each other (conditional independence assumption). For malicious URL detection, it assumes that all features of x are independent of each other. The posterior probability that the feature vector x is a malicious URL can be calculated by the following formula:

$$P(y = 1|\mathbf{x}) = \frac{P(\mathbf{x}|y = 1)}{P(\mathbf{x}|y = 1) + P(\mathbf{x}|y = -1)} \tag{4}$$

NB has been used for malicious URL detection [40, 35, 41, 42, 43, 44, 45, 46].

Based on NB, some researchers have proposed Laplace Smoothing, a technique for dealing with the "zero probability" problem in probability estimation [47]. And Gaussian NB, which is a NB classifier for dealing with continuous features [48]. It assumes that each feature follows a Gaussian distribution (normal distribution) under a given category. This assumption enables Gaussian NB to effectively deal with continuous features without discretizing them.

- Passive-Aggressive Algorithms (PA) :[49] are a class of online learning algorithms that are specifically designed to handle classification and regression tasks. The core idea is to balance the "passivity" and "aggressiveness" of model updates by dynamically adjusting the learning rate: when the model predicts the current sample well enough, keep the model stable (passive); when the prediction is wrong or the error is large, actively adjust the model to correct the error (aggressive). The optimization of PA learning can be cast as follows:

$$\mathbf{w}_{t+1} \leftarrow \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}_t - \mathbf{w}\|^2 \quad \text{subject to } y_t(\mathbf{w} \cdot \mathbf{x}_t) \geq 1 \tag{5}$$

The closed-form solution to the above can be derived as the following update rule:

$$w_{t+1} = w_t + \tau_t y_t x \tag{6}$$

where $\tau_t = \max\left(0, \frac{1-y_t(w_t^T x_t)}{\|x_t\|^2}\right)$.

The above models assume the existence of a hard margin, i.e., the data can be linearly separable, which may not always be true, especially when the data is noisy. To overcome this limitation, soft margin PA variants are often used, such as PA-I and PA-II [49].

- Confidence-Weighted Learning (CW):The CW algorithm [50] is similar to the PA learning algorithm in terms of the trade-off between dynamism and aggressiveness. By maintaining the mean and confidence of the weight vector (i.e., the covariance matrix), different update weights are assigned to each feature, so that weights with lower confidence are updated more aggressively than weights with higher confidence, making the model more robust when facing uncertain features. This method of utilizing second-order information can usually converge

faster and obtain better model performance than first-order algorithms. CW learning can be summarized as the following optimizations:

$$\left(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}\right) \leftarrow \arg\min_{\mu,\Sigma} \mathrm{KL}\left(\mathcal{N}(\mu_{t+1}, \Sigma_{t+1}) \| \mathcal{N}(\mu_t, \Sigma_t)\right) \tag{7}$$

$$\text{Subject to } y_t(\mu_{t+1}^{\top} \boldsymbol{x}_t) \geq \Phi^{-1}(\eta)\sqrt{\boldsymbol{x}_t^{\top} \Sigma_{t+1} \boldsymbol{x}_t} \tag{8}$$

CW algorithms have been applied for detecting malicious URLs by [30, 51].

In addition, the improved CW learning algorithm in Active Regularization of Weights (AROW) [52], which is used to learn inseparable data, is also used for malicious URL detection [53]. ROW enhances the robustness of the model to noise and abnormal data by introducing an adaptive regularization mechanism while maintaining efficient online learning capabilities. [54] adopts a hybrid online learning technique that combines CW and PA algorithms. Specifically, CW is used to learn from pure lexical features (e.g., bags of words), and PA is used to learn from descriptive features (e.g., statistical properties of lexical features). They believe that lexical features are more effective in detecting maliciousness, although they may change frequently (short-lived), while descriptive features are more stable and static.

- SVM:SVM is a classic supervised learning algorithm that is widely used in classification tasks, including malicious URL detection. The core idea of SVM is to find an optimal hyperplane by maximizing the classification interval to separate data points of different categories. SVM can be expressed as the following optimization:

$$(\mathbf{w}, b) \leftarrow \arg\min_{\mathbf{w},b} \frac{1}{T} \sum_{t=1}^{T} \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t + b)) + \lambda \|\mathbf{w}\|_2^2 \tag{9}$$

[29, 21, 55, 15, 56, 57, 58, 59] classify malicious websites by analyzing the text features of URLs (such as length, number of dots, keywords in the path) and host features (such as WHOIS information, IP address attributes, DNS records). [31] not only uses traditional URL features, but also introduces Kolmogorov complexity-based metrics and Huffman coding-based compression features. In addition, [32, 33] utilize URL internal correlation features (such as the Jaccard similarity index between different URL parts) and URL popularity features (such as the Alexa ranking of the domain name).

- CNN:The convolutional layer is the core component of the CNN [60]. Its main function is to extract the features of the input data through filters [61]. The size of the filter is smaller than the input data. It starts sliding from the starting position of the data and calculates the dot product of the input data and the filter one by one to generate a new feature matrix that highlights the key features of the input data. These features are then used for model training. The pooling layer reduces the feature dimension generated by the convolutional layer through pooling operations. Common pooling methods include maximum pooling, average pooling, etc. Finally, the fully connected layer (FCN) is a traditional neural network. The formula summarizes the FCN operation:

$$h_i = f\left(w_i^T x + b_i\right) \tag{10}$$

where f is the activation function, w_i^T is the weight, b_i is the bias belonging to the previous layer i , and x is the input matrix. [62, 63, 64] all use CNN to extract features, and all combine character-level and word-level CNN to extract multi-scale features. The difference is that [63] proposed a FastText model that can handle unseen words and generate word embeddings through character-level n-gram representation, while [62] directly regards each word in the URL as a feature. [64] only uses character-level features, but it uses convolutional layers and deconvolutional layers to build a convolutional autoencoder (CAE) to encode and decode character-level matrices. [65] splits the URL into n-grams of different lengths to capture patterns of different lengths in the URL, which helps the model better understand the structure of the URL.

- Transformer and its variants:Transformer is a neural network architecture based on the attention mechanism, which is mainly used to process sequence-to-sequence tasks [66]. Its core is the self-attention mechanism, which

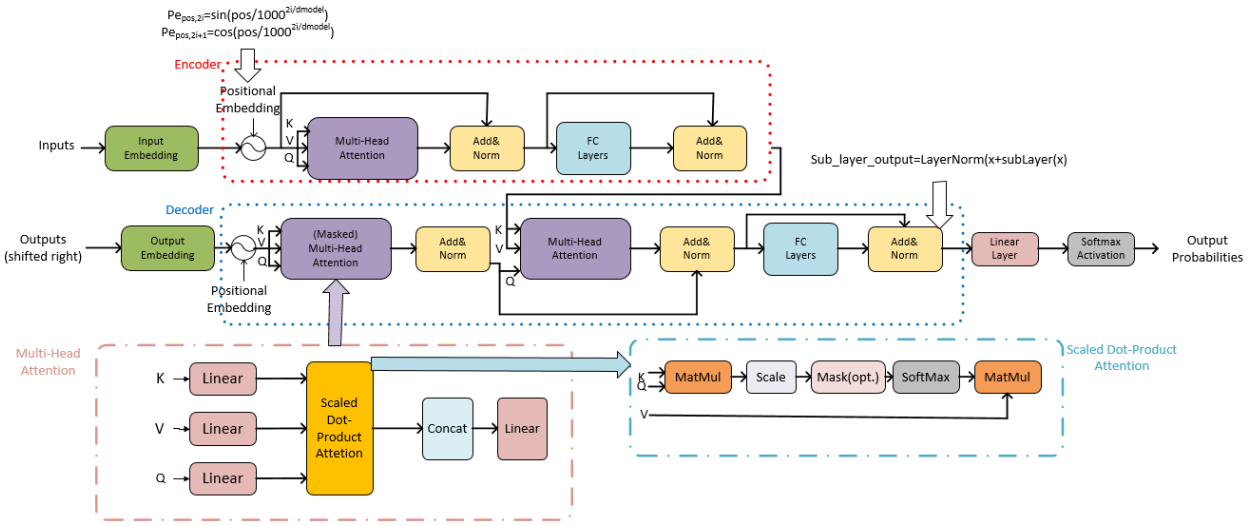

Figure 5: The Transformer model architecture

dynamically weights and sums the features of different positions by calculating the correlation (attention weight) between the input of each position in the sequence and the input of all other positions, thereby generating the output representation of each position. This mechanism can process all positions in the sequence at the same time, avoiding the position-by-position calculation limitation of RNN and greatly improving the training speed. The formula of scaled dot-product attention is as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{11}$$

Where Q,K,and V are the query, key, and value matrices, respectively, and d_k is the dimension of the key vector. The scaling factor $\frac{1}{\sqrt{d_k}}$ is used to avoid the problem of the softmax function gradient disappearing due to the dot product being too large.

The Transformer architecture includes an encoder and a decoder. The encoder consists of multiple stacked self-attention layers and a feedforward neural network, which is used to encode the input sequence into a contextual representation. The decoder also contains multiple stacked self-attention layers and a feedforward network, but also introduces an encoder-decoder attention layer to combine the encoder output with the decoder input to generate the target sequence. The main structure is shown in Figure 5.

Bidirectional Encoder Representations from Transformers(BERT) is a pre-trained language model based on the Transformer architecture, proposed by Google in 2018 [67]. Its core idea is to learn general language representations through large-scale unsupervised pre-training, and then fine-tune on specific downstream tasks, so as to achieve excellent performance in a variety of natural language processing tasks. [68, 69] directly use the BERT model to extract the semantic features of the URL, while [70] uses RoBERTa, an improved version of BERT, which focuses on the MLM (Masked Language Modeling) task and uses the LSTM in the classification stage. This combination can better handle the sequence features of the URL. [71, 72] Combined CNN and Transformer encoder, using CNN to extract local features of the URL, while the Transformer encoder is used to capture long-distance dependencies between URLs. [73] Combined multiple machine learning models (such as SVM, RF, XGBoost, etc.) and Transformer models (such as BERT), the prediction results of multiple models are fused through the stacking strategy, improving the accuracy and robustness of detection. In short, transformers and their variants are widely used in malicious URL detection [74, 75, 76, 77, 78]. In addition to the above algorithms, DT and LSTM are also often used to extract URL features for malicious URL classification [79, 59, 80, 81, 82, 83, 84, 85, 86].

3.2. HTML-Based Detection

3.2.1. Technical Principles

The technical principle of using HTML features to detect phishing websites is mainly based on analyzing the HTML structure and content of web pages to identify potential malicious patterns and abnormal behaviors. HTML is the basic framework of web pages, and its structure and content can reflect the layout, functions and design intentions of web pages. Phishing websites usually embed specific elements or attributes in HTML code to imitate the appearance and functions of legitimate websites, thereby deceiving users to enter sensitive information.

[40] proposed a method based on lexical features of HTML content of web pages to detect malicious web pages. These lexical features include words, tags, attributes, and the use of JavaScript functions and objects in HTML documents. By counting the frequency and distribution of these words and combining them with machine learning algorithms, the characteristic patterns of malicious web pages can be effectively identified. [40] The following features are recommended: document length, average word length, number of words, number of non-repeated words, number of words in a line, number of null characters, use of string concatenation, asymmetric HTML tags, links to remote script sources, and invisible objects. Malicious code is often encrypted in HTML, which is related to long word lengths or extensive use of string concatenation, so these features can help detect malicious activities.

Many subsequent researchers used similar features with minor changes, including [8] using the Term Frequency-Inverse Document Frequency (TF-IDF) technique to extract features from the plain text of a web page and the noisy parts of HTML (such as the attribute values of div,h1,h2, body, and form tags). [87] also used similar features and designed new HTML features that focus on capturing the differences in content and structure between phishing pages and legitimate pages. These features include: HTML tags that hide or restrict functionality (such as hidden ‘ ‘, disabled buttons or input boxes), consistency between title brands and URL brands, the relationship between the most frequently linked brands and URL brands, the ratio of internal to external resources, and the frequency of URL brands in HTML code. In addition, HTML string embedding technology was introduced to automatically extract features by learning vector representations of HTML content without relying on manually designed rules. These new features can more comprehensively reflect the abnormal behavior and structural characteristics of web pages, thereby effectively improving the accuracy and robustness of phishing page detection. [88] developed a delta method, which represents the changes between different versions of a website. They analyzed whether the changes were malicious or benign.

3.2.2. Detection Methods

- DT:This is a supervised learning algorithm based on a tree structure, which is widely used in classification and regression tasks. Recursively, it divides the data into subsets and then constructs a series of decision rules. Traditional DT algorithms (such as ID3, C4.5, CART) do not directly use optimization objectives, but in some modern DT variants such as XGBoost, their optimization objectives can be expressed as:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{12}$$

  Where $l(y_i, \hat{y}_i)$ is the loss function (such as hinge loss),$\Omega(f_k)$ is a regularization term used to limit the complexity of each tree. [89] uses the J4.8 algorithm to select the optimal split attribute by calculating the information gain ratio of each feature.[90] proposed a data mining method based on associative classification, Multi-label Classifier based Associative Classification (MCAC), which is specifically used to deal with multi-label classification problems.It can generate rules containing multiple categories (multi-label rules) by discovering frequent item sets in the training data set and generating association rules, thereby improving the accuracy and interpretability of classification. The rules generated by the MCAC algorithm are expressed in the form of "if-then", which is easy to understand and explain. In addition, the possibility of misclassification is reduced through multi-label rules, which improves the overall performance of the classifier. [91] A variety of DT algorithms (such as AdaBoost, Gradient Boosting, etc.) are used to improve detection performance.

- Recurrent Neural Networks (RNN) and its variants:RNN, Gated Recurrent Units (GRUs), and Long Short-Term Memory Networks LSTM are commonly used models for processing sequence data (such as text classification, video recognition, etc.). They assist in the classification of current input by memorizing previous input information. For example, phishing attacks are a type of social engineering attack that usually reaches users through social media, emails, etc., which contain text information (such as message content, URLs, etc.). These text information, as a data sequence, have a beginning, middle, and end that are crucial to understanding the

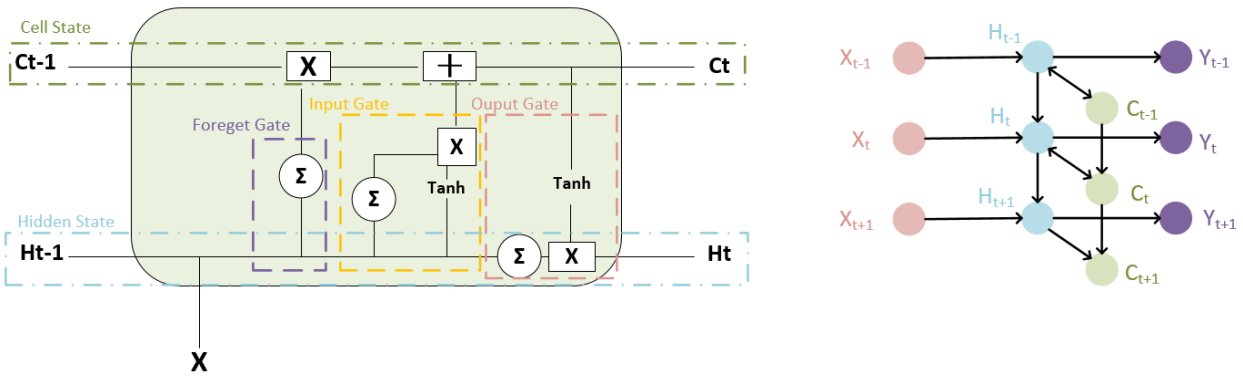

Figure 6: LSTM unit structure and sequence expansion diagram

overall meaning. Therefore, the model needs to be trained based on previous features and generate new features to pass to the next layer.

LSTMs [61] are the most commonly used sequence models. The left side of Figure 6 shows the internal structure of a single LSTM unit. LSTMs contain three gates (input gate, forget gate, output gate) and two states (hidden state and cell state). The hidden state stores the information of the previous layer, while the cell state is the main difference between LSTMs and RNNs. It passes information to the entire network chain with only minor changes at each layer. The model receives input from the feature extraction layer, the previous hidden state, and the cell state, and passes them to the forget gate and input gate. The forget gate uses formula (13) to determine which unit state values need to be forgotten (output 0 or 1), and the input gate updates the unit state through formulas (13) and (14). Finally, the output gate calculates the hidden state value of the next layer through formulas (13), (15) and (16).The right side of Figure 6 is an expanded view of the LSTM unit in the sequence, showing how the LSTM is expanded in the sequence. Each circle represents an LSTM unit, and the arrows represent the flow of information.

$$\begin{pmatrix} i_t \\ f_t \end{pmatrix} = \text{sigmoid}\left(\begin{pmatrix} W_i \\ W_f \\ W_o \end{pmatrix} h_{t-1} + \begin{pmatrix} U_i \\ U_f \\ U_o \end{pmatrix} X_t + \begin{pmatrix} b_i \\ b_f \\ b_o \end{pmatrix}\right) \tag{13}$$

$$\tilde{C} = \tanh(U_c \cdot X_t + W_c h_{t-1} + b_c) \tag{14}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{15}$$

$$h_t = O_t \cdot \tanh(C_t) \tag{16}$$

Where i is the input gate, f is the forget gate, and O is the output gate; $\tilde{C}$ is the current state of the cell memory, C_t is the memory cell state, and h_t is the hidden layer; W and U are the weighted matrixes of each gate, and b b is the bias value of each gate; X_t is the current input.

LSTM is often used in combination with CNN to detect malicious URLs. [92] Combining the advantages of CNN and BiLSTM, CNN is first used to extract features, and then BiLSTM is used to further extract time-dependent features based on the CNN layer. LSTM is widely used in malicious URL detection tasks in [93, 94, 95, 96, 97].

- CNN:[98] proposed HTMLPhish, which takes the HTML document content of a web page as input and uses a CNN to learn the semantic dependencies between characters and words in the HTML document. It uses character embedding and word embedding techniques to represent the HTML document as a feature matrix,

then concatenates the two embedding matrices and performs convolution operations on the CNN to extract semantic information that can reflect the characteristics of the web page. This feature information is passed to the subsequent pooling layer and dense layer, and finally outputs the probability of a web page being a phishing page or a legitimate page through the Sigmoid layer, thereby achieving automatic classification and detection of phishing pages.

- GCN:GCN is a deep learning model for graph-structured data, specially designed to handle complex relationships between nodes. It updates the feature representation of nodes by aggregating the neighborhood information of nodes, thereby capturing the structural information in the graph. Its propagation rules are as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right) \tag{17}$$

  Where $H^{(l)}$ is the node feature matrix of the lth layer, $\tilde{A}$ is the adjacency matrix after adding self-loops, $\tilde{D}$ is the degree matrix of $\tilde{A}$, whose diagonal elements are the degrees of each node. σ is a nonlinear activation function, such as ReLU.

  The tree structure of HTML is very suitable for building a graph structure, where nodes represent HTML tags and attributes, and edges represent parent-child relationships between nodes. [99] used GNN technology for the first time in the field of anti-phishing, using multiple graph convolutional skips (GCS) to process the graph structure of HTML content. GCN was widely used in malicious URL detection tasks in [100, 101].

- Hybrid:The phingNet model in [102] uses character-level CNN and attention mechanism to process HTML content. It first converts the HTML content into a character sequence and then uses character-level embedding to convert each character into a vector. Next, features are extracted through CNN and attention mechanism is used to highlight important features. [103] combines two pre-trained natural language processing (NLP) models (CANINE and RoBERTa) and a multi-layer perceptron (MLP) model, and inputs the embedding vectors of these models into a linear classifier for classification.

In addition to the above algorithms, LR is also often used to extract HTML features for malicious URL classification [37, 94].

3.3. JavaScript-Based Detection

3.3.1. Technical Principles

[40] They believed that hackers commonly use several JavaScript functions to encrypt malicious code or execute unwanted routines without the client's permission. For example, the widespread use of the functions eval() and unescape() may indicate the execution of encrypted code within HTML. Their goal was to use 154 native JavaScript functions as functions to identify malicious URLs. [104] A subset of these native JavaScript functions (seven) were identified that frequently appear in cross-site scripting and web-based malware distribution. These functions include: escape(), eval(), link(), unescape(), exec(), and search() functions.

The technical principle of using JavaScript features to detect phishing websites is mainly based on analyzing the behavior and characteristics of JavaScript code in web pages to identify potential malicious behaviors. Phishing websites usually use JavaScript to implement functions such as dynamic content loading, form validation, and user input collection, which are also common in legitimate websites. Therefore, the key to detection is to identify whether there are abnormal or malicious patterns in the JavaScript code, such as redirecting users to other websites, dynamically modifying web page content to imitate legitimate websites, or collecting user input information in the background.

The detection system extracts JavaScript code from web pages and performs static or dynamic analysis [35]. It can also use machine learning algorithms to identify phishing websites [105]. Static analysis checks malicious function calls, abnormal structures, and fragments similar to known phishing patterns in the code, and analyzes the execution logic to discover potential malicious behaviors; dynamic analysis runs the code in a sandbox environment to observe its network requests, DOM operations, and user input processing during runtime. In addition, by using machine learning algorithms to train a large number of code features, the model can learn to distinguish between normal and malicious behavior patterns, and use classification algorithms to determine whether the code has phishing characteristics, thereby effectively responding to ever-changing attack methods and improving the accuracy and robustness of detection.

3.3.2. Detection Methods

- Static Analysis:Static analysis mainly identifies potential malicious behavior by checking the syntax and structure of JavaScript code in web pages. This method does not require code execution, but directly analyzes the text content of the code to find out whether there are known malicious patterns or abnormal structures.

  [106] proposed a lexical analysis-based method to identify potential malicious behaviors by extracting JavaScript code from PDF documents and performing in-depth lexical analysis. However, it may face limitations in dealing with code obfuscation and dynamically generated malicious behaviors. [107] adopted a more comprehensive approach, not only focusing on the context information of function calls, including function definitions, parameter contents, and call paths, but also combining runtime inspection mechanisms. This effectively makes up for the shortcomings of pure static analysis. [108] focused on using the abstract syntax tree (AST) to extract context information from JavaScript code and classified this information through a Bayesian classifier. [109] proposed an innovative detection framework that combines flow graphs and regular path expressions, allowing users to customize queries to detect specific security vulnerabilities. This method not only provides a high degree of flexibility, but also can adapt to diverse security needs and is suitable for detecting complex and specific security issues. However, this flexibility also brings a higher technical threshold, requiring users to have certain expertise to define effective queries.

- Dynamic Analysis:Dynamic analysis detects potential malicious behavior by running JavaScript code in a controlled sandbox environment and observing its actual running behavior. This method can reveal JavaScript code that is not obvious in static code but exhibits malicious behavior at runtime. For example, dynamic analysis can detect whether the code sends data to a server address different from the legitimate website when the user submits a form, or whether the visual content of a web page is dynamically modified when the page loads to mimic a legitimate website.

  [110] proposed a dynamic analysis method to detect potential malicious behavior by loading malicious web pages in a real browser environment and monitoring their internal function call sequences. This method also involves parsing the HTML page structure to identify key locations where JavaScript code may be embedded, thereby more comprehensively capturing the characteristics of malicious behavior. However, it must rely on a specific browser. In contrast, the document [111] adopted a static analysis method based on information flow control. This method dynamically tracks the flow path of information in the code by extending the operational semantics of JavaScript to prevent sensitive information from being leaked to illegal channels. This method focuses on understanding and controlling the propagation of information at the semantic level, thereby protecting the integrity of data at runtime. In addition, [112, 113, 114, 115] also used dynamic analysis methods.

- SVM:[116, 117, 118] directly use the SVM algorithm for classification. [113] proposed a method called EarlyBird, which extends the SVM learning algorithm and optimizes the accuracy and time of detection by introducing time weights, encouraging the model to focus on malicious events that occur early, thereby achieving earlier detection. [119] uses a one-class support vector machine (One-Class SVM) to construct an initial classifier set, and in the detection stage, the outputs of these classifiers are combined through the majority voting rule to determine the final classification result.

- RNN and its variants:[120] used LSTM to classify malicious JavaScript bytecode sequences. [117, 121] both used bidirectional long short-term memory networks (Bidirectional LSTM) to extract features, but the difference is that [117] also combined the attention mechanism. [122] combined Taylor Series and Harris Hawks Optimization (HHO) to optimize the weights of the LSTM network, significantly improving the performance and generalization ability of the LSTM network in malicious JavaScript detection tasks.

- Hybrid:[115, 123] used a variety of machine learning algorithms. [115] mainly used DT algorithms for classification, while [123] used AdaBoostM1, Bagging and other methods to combine multiple weak classifiers for classification. [124, 125] combined static analysis and dynamic analysis methods to detect malicious URLs in malicious JavaScript code. [126, 127] both used GNN to extract features. The difference is that [126] extracted from the program dependency graph (PDG) and added an attention mechanism to process graph structure data, while [127] extracted structural features from the abstract syntax tree (AST) graph.

3.4. Visual-Based Detection

3.4.1. Technical Principles

Methods based on visual similarity detect phishing websites by comparing the visual features of suspicious websites with legitimate websites to identify phishing websites and non-phishing websites [128, 129]. Most of these methods focus on calculating the visual similarity with protected pages, where protected pages refer to real websites. If the visual similarity of suspected malicious URLs is high, it may indicate a phishing attempt. One of the earliest attempts to use visual features to accomplish this task was to calculate the Earth Mover's Distance (EMD) between two images [130]. [131, 132] addressed the same problem and developed a web page visual feature extraction system by decomposing a web page into multiple visually salient blocks and measuring visual similarity along three dimensions: block-level similarity (based on feature matching of text or image blocks), layout similarity (considering the spatial position relationship of blocks), and overall style similarity (based on the distribution of web page style feature values). [133] developed another method using visual features, in which OCR was used to read the text in web page images.

In recent years, deep learning methods have been widely used in this field. CNN are used to extract features of web page images and URLs and classify them. Through convolutional layers and pooling layers, CNNs can extract local features of images, while fully connected layers combine these features nonlinearly to generate the final classification results.

3.4.2. Detection Methods

- Scale-Invariant Feature Transform (SIFT):SIFT is a method for image feature extraction and matching [134] proposed by David G. Lowe. The SIFT method detects scale-invariant key points in an image and generates a descriptor for each key point. These descriptors are invariant to changes in image scale, rotation, and illumination. SIFT key point detection is based on scale space extremum detection and uses the Gaussian difference function (DoG) to identify potential key points. The position and scale of each key point are determined by fitting a model, and one or more directions are assigned to each key point. The descriptors of the key points are constructed by measuring the gradient direction histogram of the area around the key points. These descriptors can effectively represent the local image structure around the key points. Specifically, the detection of key points is achieved through the following formula:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{18}$$

  Among them, $G(x, y, \sigma)$ is the Gaussian kernel function, $I(x, y)$ is the input image, and k is the scale interval factor. The descriptor of the key point is constructed by calculating the gradient direction histogram of the area around the key point, and finally forms a 128-dimensional feature vector. The SIFT method has been widely used in computer vision tasks such as image matching, object recognition, and 3D reconstruction, and is highly praised for its robustness and discriminability.

  [135] SIFT is used to detect the overall visual similarity of phishing websites. It creates visual feature archives of trusted websites by extracting features from website content (including HTML files and logos, etc.), and then matches the newly loaded website content with these archives to determine whether it is a phishing website. [136] SIFT algorithm is used to identify key points and common image blocks are extracted by matching key points. [137] SIFT algorithm is used to identify logos and combined with DNS records or digital signatures to verify the legitimacy of the logo. [138] For "very similar" websites, wHash mechanism and color histogram are used. For "partially similar" websites, SIFT technology is used to extract features.

- Speeded-Up Robust Features (SURF):SURF is an improved image feature extraction and matching method [139] proposed by Herbert Bay et al. The SURF method aims to improve the speed of feature detection and description while maintaining robustness and discrimination comparable to SIFT. The core of the SURF method is to use an integral image to quickly calculate the convolution of the image, thereby accelerating the process of key point detection and descriptor generation. SURF's key point detection is based on the approximation of the Hessian matrix, and key points are identified by analyzing the second-order derivatives of the image at different scales. The generation of descriptors is based on Haar wavelet responses, and descriptors are constructed by calculating the Haar wavelet responses in the horizontal and vertical directions in the area around the key points. Specifically, the detection of key points is achieved through the following formula:

$$\det(H_{\text{approx}}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{19}$$

Among them, D_{xx}, D_{yy} and D_{xy} are approximations of the second-order derivatives calculated using the integral image, and w is a weight factor used to balance the determinant calculation of the Hessian matrix. The descriptor is generated by calculating the Haar wavelet response of the area around the key point, and finally forms a 64-dimensional or 128-dimensional feature vector. The SURF method significantly improves the computational efficiency while maintaining the advantages of the SIFT method, and is suitable for real-time or resource-constrained computer vision applications.

[140] The SURF detector is used to extract discriminative key point features from screenshots of legitimate and suspicious websites to calculate the similarity between legitimate and suspicious websites. If the similarity exceeds the set threshold, the website is considered to be a phishing website. [141] The SURF (Speeded Up Robust Features) algorithm is used to extract visual features of web page screenshots. These features are used to capture the visual similarity of web pages and help distinguish malicious web pages from safe web pages.

- Contrast Context Histogram (CCH):CCH is an efficient local invariant descriptor for image matching and object recognition [142]. It constructs a descriptor by calculating the contrast value (i.e., grayscale difference) between each pixel in a local region of the image and the central keypoint. Specifically, for the central keypoint pc, the contrast value of point p in the local region R is defined as $C(p) = I(p)-$, where $I(p)$ and $I(p\_c)$ are the grayscale values of points p and pc, respectively. The CCH method divides the local region into multiple sub-regions using a logarithmic polar coordinate system and constructs positive and negative contrast histograms for each sub-region. The positive contrast histogram is defined as $H^+_{R_i}(p_c) = \sum\{C(p) \mid p \in R_i \text{ and } C(p) \geq 0\}$, and the negative contrast histogram is defined as $H^-_{R_i}(p_c) = \sum\{C(p) \mid$. Finally, the CCH descriptor is defined by combining the contrast histogram values of all sub-regions into a vector in the form of $\text{CCH}(p_c) = (H^+_{R_1}, H^-_{R_1}, H^+_{R_2}, H^-_{R_2}, \ldots, H^+_{R_t}, H^-_{R_t})$. In order to handle linear illumination changes, the CCH descriptor is normalized to a unit vector. The CCH method is not only robust to geometric transformations (such as rotation) and illumination changes, but also computationally efficient and has a low descriptor dimension, making it suitable for real-time applications.

  [143, 144, 145] proposed an image matching method based on CCH descriptor for detecting phishing web pages. It identifies key points in web page images through the Harris-Laplacian corner detection method and uses CCH descriptors to capture the invariant information around these key points. The CCH descriptor describes the features of key points by calculating the contrast values in the neighborhood of key points, and these feature vectors are used to match suspicious web pages and real web pages. If the similarity between two web pages exceeds a certain threshold, the suspicious web page is considered to be a phishing web page. However, [144, 145] also combined URL domain name authentication, first filtering out suspicious URLs through URL matching, and then further confirming the authenticity of the web page through image matching.

- CNN:[146] used CNN to extract visual features of web page images. [147] used triplet CNN to learn the visual similarity between web pages and detect phishing websites by comparing the visual similarity of the test web page with the trusted web pages in the training set.

- Hybrid and other methods:[44] converted web pages into images and used EMD to measure the visual similarity between two web page images. A fusion algorithm based on Bayesian theory was proposed to combine the results of text classifiers and image classifiers to improve the accuracy of detection. [148] used the Otsu threshold method [149] to convert web pages into black and white images, divide the images into non-overlapping layout blocks, and compare these blocks between two web pages. [150] proposed an algorithm to compare the CSS similarity between suspicious websites and legitimate websites. In addition to this method, Mishra and Gupta [151] also proposed a hybrid solution based on URL and CSS matching. Among them, CSS matching is borrowed from BaitAlarm. [152] proposed a phishing detection technology that maintains an image database to compare and determine the nature of the website. The image database contains images and corresponding domains of legitimate websites and phishing websites.

3.5. Hybrid Modality Detection

3.5.1. Technical Principles

Extracting only URL-based features will result in missing important features of phishing web pages, such as page titles and page codes [94]. It is also difficult to analyze tiny URLs using only URL-based methods. Content-based

methods extract information from web page content, such as images, JavaScript, text, Hypertext Markup Language (HTML) code, etc. However, content-based methods allow users or systems to open web pages and extract content, which may lead to attacks by downloading and installing malware. Next, the various algorithms we present use multiple features to improve detection accuracy, robustness, etc.

3.5.2. Detection Methods

- SVM:[28, 40, 36, 153, 154, 41] all use trained SVM models to classify web pages and determine whether they are malicious. The features they extract include URL features, HTML document-level features (such as the number of words, average word length, number of spaces, etc.), and the usage of JavaScript functions (such as the number of calls to functions such as eval and unescape). In addition, [153] also proposed target phishing brand name features (such as whether the URL contains the brand names of common phished websites, such as "eBay" and "PayPal", etc.). [154] combines the features of HTTP response header information (lexical+headers) and further combines the features of the last modification time of the homepage (lexical+headers+age). Experimental results show that this method has a high accuracy in detecting hidden phishing pages and hidden web tampering pages. [41] collected 19 features from the network layer, including the number of TCP session exchanges, the number of application layer bytes, and the number of DNS queries. These features are integrated into a cross-layer feature vector and input into the SVM model.

- NB:[40, 35, 41, 42, 43, 44] all use NB to evaluate text features, which determines whether a web page is phishing by calculating the conditional probability of each feature under different categories. The features they extract include the number of iframe tags in HTML documents, specific function calls in JavaScript code, and the length and structure of the URL. In addition, [41] network layer features include TCP traffic features, number of DNS queries, TTL value of IP address, etc. The PhishAri system proposed by [42] uses tweet content features, including tweet length, number of topic tags, number of mentioned users, etc. User features, such as user account age, number of tweets, followers-to-following ratio, etc. are used to analyze anomalies in tweet content and user behavior. [43] proposed a malicious URL detection method based on forwarding behavior, which combines URL features and graph features generated by forwarding behavior. Graph features include the number of forwarding times, the social relationship between the forwarder and the original author, etc. [44] uses the Earth Mover's Distance (Earth Mover's Distance, EMD) is used to evaluate the visual similarity between web page images. The Bayesian model is used to estimate the threshold used in the classifier to determine whether a web page is a phishing website, and the results of the text classifier and the image classifier are fused.

- CNN:[94, 80, 95] all use CNN to extract the number of hyperlinks, title tags, iframe existence, etc. from URL features and HTML features. [80] uses a multi-channel temporal convolutional network (TCN) to process URL character embedding features and hand-crafted features (including URL and HTML features) through two channels, and finally performs binary web page classification through global maximum pooling and late fusion. [95] additionally uses the HTML document object model (DOM) structure as the input of CNN, and applies CNN to different features to extract local features, and then extracts global semantic features through a bidirectional long short-term memory network (BiLSTM), and adds an attention mechanism to highlight important features to improve the classification performance of the model.

- DT and Variants:[33, 18, 155, 156, 157, 158, 159] all use DT algorithms or random forests to classify the extracted features and determine whether the URL is a malicious URL. The features they extract include URL features and domain name features (such as domain name length and domain name age). [33, 155] use majority voting technology to combine the results of multiple DT classifiers to improve the accuracy and reliability of classification. [157] also extracts some additional features related to HTML and JavaScript. [158] uses Wrapper-based Feature Selection (WFS) to train multiple different feature subsets and calculate the classification error rate to determine the score of each subset, and then find the optimal feature subset among all possible feature subsets. Then, based on the decision tree and combined with (WFS) to improve the performance of the model. [159] also extracts features such as the number of TCP session exchanges and the number of remote IPs. [160] conducted a comparative study on the performance of three DT-based ensemble learning algorithms, CatBoost, XGBoost and LightGBM, in URL phishing detection. [161] proposed a hybrid two-layer framework based on XGBoost to improve phishing website detection. The first layer of the framework uses an improved Diversity Oriented

Firefly Algorithm (DOFA) for feature selection, and the second layer is used to adjust the hyperparameters of XGBoost.

- LR:[36, 37, 38, 162, 163, 164, 165] use LR algorithms to classify URLs and determine whether they are malicious URLs. The features they extract include the length of the URL, the number of special characters, whether HTTPS is used, and domain name features. Most studies use L1 regularization for feature selection, which reduces the number of features and improves the efficiency and interpretability of the model. [36] also extracts color histograms, GIST features, and SIFT features from web page screenshots. [38] Use LR combined with Mutual Information (MI) for feature selection. [165] Combining TF-IDF and N-gram parameters for feature extraction can effectively capture vocabulary and sequence patterns in URLs.

- LLMs:LLMs are a type of artificial intelligence model built on deep learning technology, specifically designed to process and generate natural language text. These models are trained on massive amounts of text data to learn the patterns, structure, and semantics of language, enabling them to perform a variety of NLP tasks. As a result, LLMs are able to efficiently and accurately identify malicious URLs. Figure 7 gives a brief history of the development of LLMs.

  [166, 167, 168, 169, 170, 171] The extracted features all include text features of the URL, such as length, word count, special character distribution, etc., as well as features such as HTML tags and character distribution. [166, 167] use the reasoning ability of LLMs to predict the possibility of a given URL being benign or phishing through one-shot prompting and zero/few prompting methods respectively. [168] Use the ability of multimodal LLMs to identify web page brands by analyzing various aspects of the web page (such as logos, themes, favicons, etc.) and compare them with the domain name in the URL to detect phishing attacks. [169, 170] Detect malicious URLs based on GPT-4v and GPT-4o respectively, and the prompting engineering is multi-step. However, the former generates detailed explanatory information to help users understand why a web page is considered a phishing website, and combines visual and textual information to improve detection accuracy. The latter only generates a brief warning message. [171] explored the effectiveness of LLMs in detecting phishing URLs through prompt engineering and fine-tuning. The study compared the two methods in terms of performance, data privacy, resource requirements, and model maintenance, and compared them with existing state-of-the-art methods. [172] proposed the PMANet model, which leverages the powerful language understanding capabilities of the pre-trained language model CharBERT and enables it to better adapt to the malicious URL detection task through methods such as continued training, multi-level feature extraction, and attention mechanisms.

- Hybrid and other methods:[173] extracts text features (such as title tag text, DOM tree text, form fields, and images) and visual features (extracted from the visual representation of DOM elements through CNN) from the DOM tree, and uses heuristic rules to identify DOM element manipulation techniques and extract heuristic features. A Transformer-based encoder is proposed to jointly learn text, visual, and heuristic multimodal features. The Transformer module learns the joint features of feature mapping to brand identity through the self-attention mechanism. [174] A phishing attack detection framework based on continuous learning (CL) is proposed to address the problem that the performance of traditional machine learning models deteriorates over time. [175, 176] both use autoencoders as the basic model. The autoencoder compresses the input data into a low-dimensional representation through the encoder, and then tries to reconstruct the input data through the decoder, and uses the reconstruction error to identify abnormal samples. And the anomaly detection ability of the model is enhanced by introducing additional mechanisms (such as score-guided regularization and self-supervision tasks). However, [176] mainly focuses on the extraction of image features.

There are various ML algorithms in the literature that can be directly used for malicious URL detection. Our Table 1 categorizes the representative references according to the feature extraction classification of this paper. Table 2 summarizes some papers according to the publication year, URL classification, classifier, and the accuracy results obtained.

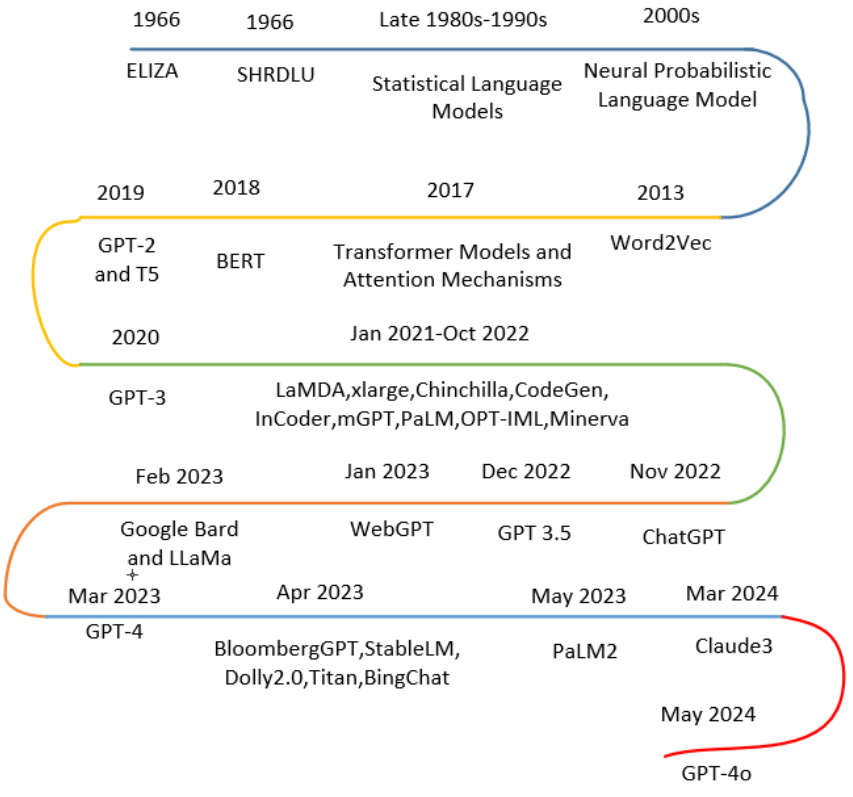

Figure 7: The brief history of Large Language Models

Feature	Representative References
URL	[28, 29, 30, 34, 35, 36, 44, 79, 177, 178, 51, 179], [53, 31, 21, 153, 154, 41, 180, 54, 181], [32, 33, 62, 182, 183, 184, 185, 186, 187, 188, 155, 27]
HTML	[40, 35, 89, 36, 189, 56, 180], [41, 190, 191, 186, 192]
JavaScript	[40, 35, 193, 105, 41, 180, 190]
Visual	[131, 132, 36, 44, 135, 143]

Table 1
Representative references of different types of features used by researchers in literature

Reference	Year	URL Classification	Classifier/Method	Result
[194]	2021	Malicious, Phishing and benign URLs	Attention-based bidirectional independent recurrent network (Bi-IndRNN), and capsule network (CapsNet)	99.89%
[26]	2019	Malicious and benign URLs	Random Forest, Gradient Boost, AdaBoost, Logistic Regression, Naive Bayes	92%, 90%, 90%, 87%, 70%

[14]	2022	Malicious or benign URLs	Logistic Regression, SVM, Random Forest, Gradient Boost, Bagging	92.80%, 97.32%, 97.35%, 96.27%, 97.35%
[15]	2023	Malicious and benign URLs	SVM, Random Forest, Decision Tree, K-Nearest Neighbors (KNN)	91.75%, 92.15%, 90.18%, 86.64%
[17]	2022	Malicious and benign URLs	CNN, LSTM, Naïve Bayes, Random Forest	95.13%, 95.14%, 96.01%, 95.15%
[18]	2021	Malicious and benign URLs	XGBoost, CS-XGBoost, SMOTE+XGBoost	97.83%, 99.05%, 98.43%
[195]	2023	Malicious URLs using unbalanced classification	DDQN-based classifier, Deep Reinforcement Learning	93.4%
[196]	2023	Phishing, benign, defacement and malware	Random Forest, LightGBM, XGBoost	96.6%, 95.6%, 93.2%
[197]	2020	Malicious and benign URLs	Random Forest, SVM	99.77%, 93.39%
[198]	2019	Good and bad URLs	Random Forest, SVM	92.38%, 87.93%
[199]	2023	Malicious website	MM-ComBERT-LMS	98.72%
[200]	2023	Phishing URLs through parallel processing	Naïve Bayes, CNN, Random Forest, LSTM	96%
[201]	2022	Malicious and benign URLs	Random Forest	96%
[202]	2022	Malware	Logistic Regression, SVM, ELM, ANN	89.99%, 96.49%, 98.17%, 97.20%
[203]	2022	Malicious and benign URLs	MLP	99.62%
[204]	2022	Phishing website	BERT, NLP, Deep CNN	96.6%
[205]	2023	Phishing and benign URLs	Random Forest, Gradient Boost, XGBoost	97.44%, 98.27%, 98.21%
[206]	2021	Malicious URLs using data mining approach	Classification Based on Association (CBA)	91.30%
[207]	2022	Phishing and legitimate URLs	LSTM, Bi-LSTM, GRU	97%, 99%, 97.5%

Table 2
Study of some malicious URLs detection based on machine learning

4. Malicious URL Detection on Arabic Studies

This section reviews and summarizes the related works on detecting malicious websites in Arabic using ML algorithms. These studies are divided into three main parts. The first part discusses the studies on the combination of url-based and html-based features. The second part studies the studies on content-based features. The third part studies the studies on url-based features.

4.1. URL and HTML-Based Features Studies

Al-Kabi et al. [208] studied a content analysis-based approach to detecting spam in Arabic web pages. By constructing a dataset of 15,000 web pages, they extracted a variety of HTML features, including the number of words in the page title (<title>), the content in the meta tag (<meta>), the number of popular words in the page content, and the length of the URL. Experimental results show that the DT algorithm performs best with an accuracy of 99.52%.

In a subsequent study [209], an online Arabic web spam detection system (OLAWSDS) was introduced, which combined URL, HTML features, and user feedback to extract features through a customized web crawler and web page analyzer, and used a DT classifier to detect spam web pages. The system also continuously optimized the detection performance through user feedback, with an accuracy of 99.96%.

EL-Mohdy et al. [210] proposed a system based on web mining technology to prevent the spread of spam web pages, which were manually collected using search engines. Experimental results show that the system has an accuracy of 97% in detecting spam web pages.

Wahsheh et al. [211] proposed a hybrid approach combining HTML and URL features for detecting Arabic web spam content. The goal of this study was to build the first web spam detection system for Arabic content or links using DT rules. The proposed system helps clean the search engine result pages (SERPs) of all URLs referencing Arabic spam web pages. The model achieved an accuracy of 90.11% (content features), 93.1034% (link features), and 89.01% (hybrid features).

4.2. Content-Based Features Studies

Alsaleh and Alarifi [212] showed how ineffective Google's anti-spam approach was against web spam pages containing non-English content. The researchers proposed a browser anti-spam plugin to detect Arabic spam web pages. The accuracy was 87.13%.

Another study, published by Al-Twairesh et al. [213] used rule-based methods and supervised learning methods (NB and SVM) to detect spam tweets. The researchers collected about 40,000 tweets and extracted a variety of content features (such as URL presence, phone number, number of tags in tweets, spam vocabulary, etc.), and experimental results showed that the average F1 score of the rule-based method was 85%, and the average F1 score of the supervised learning method was 91.6%.

Alkhair et al. [214] introduced the construction, analysis, and classification of an Arabic fake news corpus. The authors collected comment data related to the death of Arab celebrities through YouTube and performed data cleaning and analysis. The study used three machine learning classifiers, SVM, DT, and multinomial naive Bayes (MNB), to distinguish rumor from non-rumor comments. The results showed that SVM had the best accuracy of 95.35%.

Mataoui et al.[215] proposed a method for detecting spam in Arabic social media content. The authors collected 9997 comments from Facebook and extracted 9 features to characterize spam content. In the preprocessing step, they used standard NLP techniques to extract tags, such as tokenization, normalization, stop word removal, and stemming. Among them, the J48 classifier had the highest detection accuracy of 91.73%.

Najadat et al. [216] proposed a keyword-based spam detection method for Arabic social media comments. The authors collected 3,000 comments from Facebook using the Netvizz application and classified them based on content-based features. The DT classifier performed best with a detection accuracy of 92.63In addition, Mubarak et al. [217] proposed a model to detect Arabic spam tweets and identified different attributes of spam and ham tweets. They built their own dataset from Twitter and selected four content-based features. The highest result was achieved by the Arabic bidirectional encoder representation of transformers (AraBERT) with an accuracy of 99.7%.

Alsulami and Yousef [218] proposed a personalized filtering model based on sentiment and behavior analysis, SentiFilter, for detecting Arabic semi-spam content. It aims to provide a personalized level of protection for each user. SentiFilter combines the sentiment polarity of user comments and user like behavior to detect semi-spam content. Experimental results show that comment behavior is more effective in detecting semi-spam than like behavior. The SVM classifier achieved the best classification results with an average accuracy of 90.89%.

Alkadri et al. [219] proposed an integrated framework for detecting spam on Arabic social media. The framework combines data augmentation, natural language processing, and supervised machine learning algorithms to improve the performance of spam detection by increasing data diversity and improving feature extraction. Experiments show that after using data augmentation, the model's macro F1 score is significantly improved and the accuracy reaches 92%, which is significantly better than existing methods.

Likewise, Ezzat et al. [220] proposed a real-time framework (RTAOSD) for detecting opinion spam in Arabic social media and classifying non-spam content based on topic relevance. The framework combines advanced language models (such as AraBERT), data augmentation techniques, and real-time processing techniques. Experimental results show that the framework performs well in spam detection and topic relevance classification, and both macro F1 scores and accuracy are better than existing methods.

Furthermore,Kihal, M. and Hamza, L. [221] proposed a deep learning method based on Transformer and 2D CNN for detecting spam content in Arabic and English social media. The method represents text features as 2D curve images and uses CNN for classification. Experimental results show that the method achieves high accuracy (98.68% and 93.67% respectively) on both Arabic and English datasets, significantly outperforming traditional machine learning methods.

Radwa et al. [222] proposed an ensemble method for detecting spam comments in Arabic opinion texts. The method combines rule-based classifiers with multiple machine learning techniques and uses N-gram features and negation processing to improve detection performance. Experimental results show that ensemble methods (especially stacked ensembles) perform well in spam comment detection, with an accuracy of up to 99.98%, significantly outperforming existing methods.

4.3. URL-Based Features Studies

Alorini [223] studied how to use machine learning to detect malicious content in the Gulf Arabic dialect on social media, specifically Twitter. The authors used the Twitter streaming API and Python's Twestern package to construct a dataset of 2,000 Gulf Arabic sentences from Twitter and translated them into English. The study used NB and Support SVM classification methods to detect spam, and the results showed that NB was more accurate in detecting Arabic spam.

Another study by Wahsheh et al. [224] explored how to detect Arabic spam web pages using link-based techniques. The authors collected a dataset of 3,000 Arabic spam web pages and used two classifiers, DT and NB, to evaluate link similarity. The DT classifier produced the highest accuracy of 91.4706%.

Alharbi and Aljaedi [225] studied how to predict and detect malicious content and Arabic spam accounts on Twitter. The authors collected nearly 3 million Arabic tweets, analyzed the generated 47 features, and selected the best features. Experimental results show that the random forest classifier with 16 features performs best with an accuracy of over 90%.

Alsufyani et al. [226] proposed a deep learning-based model for detecting phishing messages in Arabic text messages. Three deep learning models, CNN, BiGRU, and GRU, were used and evaluated on an Arabic text message dataset containing URLs. Experimental results show that the GRU model performs best with an accuracy of 95.3%.

Finally, AlGhamdi and Khan [227] introduced an intelligent analysis system for detecting suspicious information in Arabic tweets. The authors collected Arabic tweet data through the Twitter Streaming API and used six supervised learning algorithms (DT, k-nearest neighbor, linear discriminant analysis, support vector machine, artificial neural network, and long short-term memory network) for classification. SVM performed best with an average accuracy of 86.72%.

5. Published Algorithms and Datasets

5.1. Open Source Implementation

Public implementations of algorithms and models facilitate baseline experiments. Table 3 provides a summary of published implementations, outlining when they were published and the URL of the code repository.

5.2. Datasets Used

Researchers have utilized a variety of datasets, including PhishTank, Kaggle, CommonCrawl, GitHub, Phishstorm, Malcode, and DomainTools, to evaluate the effectiveness of network detection and classification models. This approach ensures that their findings are robust and relevant in real-world scenarios.

Reference	Time	Code Repository
[62]	2018	https://github.com/Antimalweb/URLNet
[182]	2017	https://github.com/MjafarMashhadi/Haplophysh
[228]	2020	https://github.com/Microsoft/CNTK
[147]	2019	https://github.com/S-Abdelnabi/VisualPhishNet
[74]	2023	https://github.com/vul-det/transurl
[69]	2022	https://github.com/GregaVrbancic/Phishing-Dataset
[175]	2021	https://github.com/urbanmobility/SGM
[172]	2023	https://github.com/alixyttte/malicious-url-detection-pmnet
[229]	2021	https://github.com/SharifAmit/Semi-supervised-Phishing-Detection-GAN
[230]	2022	https://github.com/ehsannowroozi/sec_classifying_url_detection
[231]	2024	https://github.com/AbdelkaderMH/DomURLs_BERT
[232]	2025	https://github.com/venyeguo/lbp
[233]	2022	https://github.com/qalateawang/tsgn-master
[234]	2024	https://github.com/Davidup1/URLBERT
[235]	2023	https://github.com/Davidup1/FedURLBERT

Table 3
Published Algorithms

In research focused on detecting malicious websites, researchers manually extracted HTML and JavaScript code, WHOIS host information, and Web URL features, which were then integrated into ML or heuristic systems to enhance detection efficiency[194]. For instance, a training dataset for a classification model included 5 million URLs sourced from Openphish, Alexa's whitelist, and internal FireEye sources, maintaining a balanced distribution of 60% benign URLs and 40% malicious URLs[26].

Furthermore, a 2020 study utilized the ISCX-URL-2016 dataset to extract 78 lexical variables, enabling the categorization of URLs into five distinct categories: benign, malware, phishing, spam, and website tampering[53]. Notably, PhishTank is frequently cited as a primary source of malicious URL datasets in various research studies.

We summarize most of the datasets collected in the study. The datasets used to train and test the detection models come from various sources, including open sources, datasets created by the study authors, datasets adapted from other authors, or a combination of these datasets. The most common dataset sources are PhishTank and Alexa, as well as datasets collected by the study authors. The specific creation time, creator, and link information are shown in Table 3.

5.3. Evaluation indicators

To date, the most widely used metrics for evaluating malicious URL detection performance include accuracy, precision, recall, F1 score, Receiver Operating Characteristic Curve (ROC), and Area Under the ROC Curve (AUC). The formulas/explanations are as follows:

1. accuracy:Accuracy refers to the proportion of samples correctly classified by the model to the total number of samples. In the case of data imbalance (for example, the number of malicious URLs is far less than that of normal URLs), it may not accurately reflect the performance of the model.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{20}$$

2. precision:Precision refers to the proportion of samples predicted to be malicious URLs that are actually malicious URLs.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{21}$$

3. recall:Recall rate refers to the proportion of samples that are correctly predicted to be malicious URLs among all samples that are actually malicious URLs.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{22}$$

Dataset	Source	Time	Size	Access Method
ISCX-URL2016	Canadian Institute for Cybersecurity	2016	114,250	https://www.unb.ca/cic/datasets/url-2016.html
malicious_phish	Manu Siddhartha	2021	351,191	https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset
Phishing URL dataset	JISHNU K S KATHOLIKKAL, Arthi B	2021	450,176	https://data.mendeley.com/datasets/vfszbj9b36/1
Pristine and Malicious URLs	Ehsan Nowroozi	2023	3,980,870	https://ieee-dataport.org/documents/pristine-and-malicious-urls
URLhaus data	URLhaus	2025	3,344,703	https://urlhaus.abuse.ch/browse/
PhiUSIIL Phishing URL	Prasad, A. & Chandra, S.	2023	235,795	https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset
Dataset of suspicious phishing URL detection	Tamal MA, Islam MK, Bhuiyan T, Sattar A	2024	247,950	https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2024.1308634/full
Malicious URLs dataset	Gigasheet	2024	651,192	https://app.gigasheet.com/spreadsheet/malicious-urls-dataset/4f2bbbf5_fd2f_47a5_bbd9_60a9efcf43da
URL Reputation	UCI Machine Learning Repository	2009	239,6130	https://archive.ics.uci.edu/dataset/187/url+reputation
PhishStorm-phishing/legitimate URL dataset	Aalto University	2014	96,018	phishstorm-phishing–legitimate-url-dataset
CyberSecurit: BookMyShow ads URL Analysis	Shibe Mohapatra	2022	110,000	https://www.kaggle.com/datasets/shibumohapatra/book-my-show
Benign and Malicious URLs	Samah Malhr	2024	632,508	https://www.kaggle.com/datasets/samahsadiq/benign-and-malicious-urls
Phishing Site URLs	Tarun Tiwari	2020	549,346	https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls
GramBeddings	AHMET SELMAN BOZKIR, FIRAT C.	2022	800,000	https://web.cs.hacettepe.edu.tr/~selman/grambeddings-dataset/#
Mendeley	AK Singh	2020	156,1934	https://data.mendeley.com/datasets/gdx3pkwp47/2

Table 4
Summary of Some Datasets Available for Malicious URL Detection Research

4. F1 score:The F1 score is the harmonic mean of precision and recall, which is used to comprehensively evaluate the performance of the model and is suitable for situations with unbalanced data.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{23}$$

5. roc and auc:The ROC curve is a two-dimensional curve, with the horizontal axis representing the false positive rate (FPR) and the vertical axis representing the true positive rate (TPR). The AUC value represents the area under the ROC curve. An AUC value of 1 indicates a perfect model, and an AUC value of 0.5 indicates that the

model has no distinguishing ability.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{24}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{25}$$

where,TP (True Positive) are the actual URL is malicious and predicted to be malicious. FP (False Positive) are the actual URL is normal and predicted to be malicious. TN (True Negative) are the actual URL is normal and predicted to be normal. FN (False Negative) are the actual URL is malicious and predicted to be normal.

6. Malicious URL Detection as a Service

ML has broad application prospects in the field of malicious URL detection, but it faces many challenges in actual deployment. To address these challenges, some researchers have proposed a complete malicious URL detection system architecture and applied it to practical scenarios. Among them, many studies focus on online social networking platforms such as Twitter, where users frequently share a large number of URLs, providing rich application scenarios for malicious URL detection ([55, 236, 180, 237], etc.).

6.1. Design Principles

When designing and building a real-world malicious URL detection system using ML techniques, we aim to achieve several goals. There are many goals and parameters that need to be weighed to achieve the desired results. We briefly discuss the most important considerations below:

(i) Accuracy: This is usually one of the most important goals to be achieved in any malicious URL detection. Ideally, the system should identify all malicious URLs as comprehensively as possible (i.e., achieve a high "true positive" rate) while minimizing the number of cases where normal URLs are misclassified as malicious URLs (i.e., "false positives"). However, given that a perfect detection system does not exist in reality, malicious URL detection systems in actual applications often need to make a reasonable trade-off between false positive rate and false negative rate by adjusting the detection threshold based on the needs of specific scenarios (e.g., the balance between security requirements and user experience).

(ii) Detection speed: In actual malicious URL detection systems, especially in online systems and network security applications, detection speed is a key consideration. Take the deployment of malicious URL detection services in online social networks such as Twitter as an example. When a user posts a new URL, the ideal system needs to complete the entire process from receiving the URL to outputting the detection result in a very short time to achieve instant detection of the malicious URL. This "quick response" capability can promptly block the spread of malicious URLs and their related tweets, thereby avoiding potential threats and harm to users. In some network security scenarios, the requirements for detection speed are more stringent, and detection may need to be completed within milliseconds so that malicious URL requests can be immediately intercepted at the moment the user clicks.

(iii) Scalability: This is one of the important features that malicious URL detection systems must have when facing the ever-increasing amount of data and users. Ideally, the system should be able to smoothly expand its processing capacity to meet the needs of large-scale data detection while maintaining performance stability. To achieve this goal, researchers mainly start from two directions. On the one hand, develop more efficient and scalable learning algorithms, such as online learning algorithms or efficient random optimization methods; on the other hand, use distributed computing environments to build scalable learning systems, such as using emerging distributed frameworks (such as Apache Hadoop, Spark, Apache Flink, etc.) to achieve large-scale data processing on clusters.

(iv) Adaptability: Real-world malicious URL detection systems must deal with a variety of practical complexities, including but not limited to adversarial behaviors, such as dynamic changes in the distribution of malicious URLs over time (i.e., concept drift), and adversarial strategies adopted by malicious users to bypass detection. In addition, the system needs to deal with missing values in the data (such as certain features are unavailable or too expensive to compute), as well as the continuous emergence of new features. In addition, given that malicious users may evade detection by changing URL features, the system needs to be able to not only identify and resist such adversarial attacks, but also be able to self-optimize based on the latest threat intelligence and data dynamics (e.g., by regularly updating

models or feature libraries). Therefore, the malicious URL detection system must be highly adaptable to ensure efficient and robust operation in a variety of situations.

(v) Flexibility: Given the high complexity of malicious URL detection, a real-world malicious URL detection system with ML should be designed with good flexibility and be easy to improve and expand. Specifically, the system should have the following capabilities: first, it should be able to quickly update the prediction model based on new training data; second, it should be able to easily replace the training algorithm and model architecture when necessary; third, it should be able to flexibly expand the training model to respond to emerging attacks and threats; fourth, it should be able to interact with humans when necessary, such as through active learning or crowdsourcing to improve the performance of the system. These features will ensure that the malicious URL detection system always maintains efficient and accurate detection capabilities in the face of evolving network threats.

6.2. Design Frameworks

Below, we discuss some real-world implementations, heuristics, and systems that attempt to provide malicious URL detection as a service. [180] designed a framework called Monarch to provide malicious URL detection as a service. Monarch can crawl URLs in web services in real time and determine whether they are malicious. The study also explored the differences in the distribution of malicious URLs in Twitter and spam, and established different detection models based on this. Monarch processes up to 15 million URLs per day, and the operating cost is less than $800 per day. The implementation of the system includes a URL aggregator to collect URLs from various data sources such as Twitter or email. Subsequently, the features of these URLs are extracted and processed into sparse feature vectors in the feature extractor. Finally, a classifier is trained based on these processed data to detect malicious URLs. The collected features cover URL-based attributes and content-based attributes, as well as initial URLs, login URLs, and redirection information. On a distributed computing architecture, the researchers used a linear classifier based on LR combined with an L1 regularizer for training to achieve fast classification training from the perspective of memory usage and algorithm update efficiency [238, 239]. This combined approach can induce a sparse model, thereby improving training efficiency. In actual applications, the system takes an average of 5.5 seconds to complete the processing of a single URL, of which feature extraction takes up most of the time. In contrast, the prediction process is relatively efficient.

[240] explored the use of ML techniques to predict malicious URLs and attempted to use these predictions to maintain a blacklist of malicious URLs. Prophiler [35] proposed a two-stage URL classification process. In the first stage, the system quickly screens out URLs for which the classifier has high confidence by analyzing lightweight URL features. For predictions with lower confidence, further intensive content-based analysis is performed. WarningBird [236] is similar to Monarch in that it focuses on detecting suspicious URLs in Twitter streams. However, WarningBird uses the SVM model of LIBLINEAR [241] to acquire new features instead of training on a distributed architecture. In addition, BINSPECT [242] is a system that uses ensemble classification, and its final prediction is based on a confidence-weighted majority voting mechanism.

7. Discussion

7.1. Challenges

7.1.1. Resilience Against Cloaking Attacks

In recent years, phishing attackers have been using more sophisticated techniques, one of which is called "cloaking", which aims to evade the detection of phishing detection systems [404]. Many studies have shown that existing anti-phishing systems are vulnerable to server-side and client-side cloaking attacks. Specifically, in server-side cloaking attacks, attackers use various attributes that phishing detection systems rely on, such as IP addresses, domain names, and host names, to identify and filter out requests that may trigger detection mechanisms [243, 244, 245]. With these strategies, attackers can effectively prevent detection systems from accessing phishing websites. In client-side cloaking attacks, attackers are committed to verifying whether the person interacting with the website is a real human user. To this end, they often set up mechanisms such as pop-up alerts or verification code challenges, which often make it difficult for phishing detection systems to intervene, allowing phishing websites to evade detection [246, 247, 248]. In addition, recent research has revealed a new type of hiding technology that further conceals the true intentions of phishing websites by exploiting the differences in behavior patterns between legitimate users and anti-phishing entities [249]. Although current phishing website detection mechanisms continue to introduce more advanced and powerful

models, most mechanisms still seem powerless when faced with the above-mentioned disguised attacks and find it difficult to effectively deal with the increasingly complex and evolving threats posed by phishing websites.

7.1.2. Datasets

Datasets play a key role in developing and evaluating phishing detection systems, but they have several limitations that may affect the accuracy and generalizability of detection systems.

These datasets or data sources may be biased in different ways. First, they are often limited to specific types of phishing content, typically content targeting English-speaking users or using common platforms. This limitation means that phishing attempts in other languages or targeting localized platforms remain scarce [250]. As a result, models trained on these datasets may perform poorly when exposed to linguistically or culturally different phishing attacks, as discussed earlier. In addition, these datasets rely on historical records, making them outdated and unable to cover new and emerging phishing techniques as attackers continue to evolve their methods, especially with the advent of artificial intelligence. However, many datasets fail to be updated quickly enough to include these new patterns. In addition, the features and indicators used in existing datasets may not cover all techniques. They often focus on well-known features, such as specific domain patterns or visual similarities. However, they may ignore or under-reflect new indicators.

In the process of dataset construction, how to efficiently and accurately collect and annotate a large amount of URL data while ensuring the diversity and representativeness of the data poses a very challenging technical problem. This process not only needs to consider the breadth and depth of data collection, but also needs to ensure the accuracy and consistency of data annotation to meet the needs of model training and verification.

7.1.3. Models

As the network environment becomes increasingly complex, the forms and attack methods of malicious URLs are constantly evolving, which makes traditional detection methods gradually unable to cope with the situation. Although machine learning-based detection algorithms can identify malicious URLs by learning the inherent laws and patterns of data, their generalization and recognition capabilities when facing unseen malicious URLs still need to be further verified and improved. In addition, due to the lack of sufficient historical data, machine learning models may encounter difficulties in dealing with emerging threats or zero-day attacks. Therefore, it is particularly important to develop adaptive models that can quickly adapt to changing trends. At the same time, malicious actors may evade detection by regularly changing the URL structure, which requires machine learning models to have the ability to resist such polymorphic attacks.

7.1.4. Multi-language Environment

In the context of English and Arabic learning, it is critical to recognize the challenges of detecting malicious URLs in a multilingual environment. English and Arabic websites have huge differences in URL structure, character sets, and language, which pose unique challenges to detection algorithms. With the growth of Arabic Internet users, it becomes critical to incorporate Arabic processing and detection techniques into malicious URL detection models. Compared to English URLs, Arabic URL structures may contain unique characters and different syntax, requiring specialized preprocessing and feature extraction methods.

To address these challenges, researchers must develop models that can handle multiple languages, including English and Arabic. Machine learning models should be trained on datasets containing representations of multiple languages, which can help detect malicious URLs in various language environments. In addition, in order to achieve accurate detection, it is necessary to adapt to different cultural and language environments, as malicious actors may target specific language groups to launch culturally specific attacks.

7.2. Future Direction

Phishing techniques are constantly evolving, and attackers frequently update their strategies to circumvent detection systems. To stay ahead of these threats, future phishing detection models must become more adaptive and self-learning, able to detect new phishing attempts without the need for frequent manual updates or retraining. This can be achieved through several advanced learning techniques, including reinforcement learning (RL) [251], unsupervised learning [252], and online learning [253]. In the context of phishing detection, RL can be used so that the system dynamically adjusts its strategy based on changes in the environment and receives feedback based on whether its detection decisions are correct. Over time, the RL model adjusts its decision-making process to minimize errors and improve accuracy. In addition, unsupervised learning techniques can be used to detect new, unseen phishing attempts. Being able to discover

hidden patterns and anomalies in the data can help identify new malicious URL attacks. In addition, online learning involves constantly updating available data. Being able to process new URL data in real time and update the model based on the latest data improves the detection capabilities of new malicious URLs.

In order to design a robust and accurate phishing detection system, it is important to integrate multimodal information in the detection tool. Although using a single modality can simplify analysis in some cases, its detection capability is limited by the characteristics of the selected modality and cannot fully capture the multi-dimensional characteristics of malicious behavior, thus affecting detection efficiency and real-time performance. Combining information from multiple modalities such as URL, HTML, JavaScript, and vision can enhance the system's adaptability to new attacks, because the characteristics of different modalities can complement each other and reduce the detection blind spots that may exist in a single modality. In addition, multimodal methods are also more advantageous in processing the dynamics and complexity of data, and can better cope with the dynamic changes and disguise techniques of malicious URLs.

It is also possible to combine cross-domain knowledge such as psychology and sociology to improve the detection effect of malicious URLs. For example, by analyzing user behavior patterns (such as the speed of clicking links, the dwell time, etc.), abnormal behaviors can be identified to detect potential malicious URLs. In addition, by integrating computer science and network engineering, network traffic analysis technology is used to detect abnormal network requests and data transmission patterns and identify malicious URLs.

8. Conclusion

This survey provides a modality-driven perspective on malicious URL detection, systematically analyzing how different techniques exploit URL text, HTML structures, JavaScript behaviors, and visual features—a dimension critically missing in prior algorithm-centric reviews. By consolidating datasets and open-source implementations, we establish the first unified benchmark for reproducible evaluation, addressing the field's longstanding baseline scarcity. We rigorous coverage of Transformer, GNNs and LLMs—technologies underrepresented in prior reviews. For practitioners, we distill design principles for real-world deployment; for researchers, we pinpoint three high-impact directions.

References

[1] Sahoo, D., Liu, C., Hoi, S.C., 2017. Malicious url detection using machine learning: A survey. arXiv preprint arXiv:1701.07179 .

[2] Aalla, H.V.S., Dumpala, N.R., Eliazer, M., 2021. Malicious url prediction using machine learning techniques. Annals of the Romanian Society for Cell Biology 25, 2170–2176.

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

[4] Anti-Phishing Working Group, 2023. Apwg. https://apwg.org/. [Online; accessed 10-April-2025].

[5] Aljabri, M., Altamimi, H.S., Albelali, S.A., Al-Harbi, M., Alhuraib, H.T., Alotaibi, N.K., Alahmadi, A.A., Alhaidari, F., Mohammad, R.M.A., Salah, K., 2022. Detecting malicious urls using machine learning techniques: review and research directions. IEEE Access 10, 121395–121417.

[6] Reyes-Dorta, N., Caballero-Gil, P., Rosa-Remedios, C., 2024. Detection of malicious urls using machine learning. Wireless Networks , 1–18.

[7] Aung, E.S., Yamana, H., 2020. Malicious url detection: a survey, in: DEIM Forum F6–3.

[8] Aljofey, A., Jiang, Q., Rasool, A., Chen, H., Liu, W., Qu, Q., Wang, Y., 2022. An effective detection approach for phishing websites using url and html features. Scientific Reports 12, 8842.

[9] Asiri, S., Xiao, Y., Alzahrani, S., Li, S., Li, T., 2023. A survey of intelligent detection designs of html url phishing attacks. IEEE Access 11, 6421–6443. doi:10.1109/ACCESS.2023.3237798.

[10] Johnson, C., Khadka, B., Basnet, R.B., Doleck, T., 2020. Towards detecting and classifying malicious urls using deep learning. J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl. 11, 31–48.

[11] Cova, M., Kruegel, C., Vigna, G., 2010. Detection and analysis of drive-by-download attacks and malicious javascript code, in: Proceedings of the 19th international conference on World wide web, pp. 281–290.

[12] Sánchez-Paniagua, M., Fernández, E.F., Alegre, E., Al-Nabki, W., González-Castro, V., 2022. Phishing url detection: A real-case scenario through login urls. IEEe Access 10, 42949–42960.

[13] Pandey, A., Chadawar, J., 2022. Phishing url detection using hybrid ensemble model. INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) 11.

[14] Wang, Y., 2022. Malicious url detection an evaluation of feature extraction and machine learning algorithm. Highlights in Science, Engineering and Technology 23, 117–123.

[15] Abad, S., Gholamy, H., Aslani, M., 2023. Classification of malicious urls using machine learning. Sensors 23, 7760.

[16] Almousa, M., Anwar, M., 2023. A url-based social semantic attacks detection with character-aware language model. IEEE Access 11, 10654–10663.

[17] Aljabri, M., Alhaidari, F., Mohammad, R.M.A., Mirza, S., Alhamed, D.H., Altamimi, H.S., Chrouf, S.M.B., 2022. An assessment of lexical, network, and content-based features for detecting malicious urls using machine learning and deep learning models. Computational Intelligence and Neuroscience 2022, 3241216.
[18] He, S., Li, B., Peng, H., Xin, J., Zhang, E., 2021. An effective cost-sensitive xgboost method for malicious urls detection in imbalanced dataset. IEEE Access 9, 93089–93096.
[19] Zhang, J., Porras, P.A., Ullrich, J., 2008. Highly predictive blacklisting., in: USENIX security symposium, pp. 107–122.
[20] Sheng, S., Wardman, B., Warner, G., Cranor, L., Hong, J., Zhang, C., 2009. An empirical analysis of phishing blacklists. Proceedings of Sixth Conference on Email and AntiSpam (CEAS) .
[21] Chu, W., Zhu, B.B., Xue, F., Guan, X., Cai, Z., 2013. Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing urls, in: 2013 IEEE international conference on communications (ICC), IEEE. pp. 1990–1994.
[22] Nguyen, L.A.T., To, B.L., Nguyen, H.K., Nguyen, M.H., 2014. A novel approach for phishing detection using url-based heuristic, in: 2014 international conference on computing, management and telecommunications (ComManTel), IEEE. pp. 298–303.
[23] Nguyen, L.A.T., To, B.L., Nguyen, H.K., Nguyen, M.H., 2013. Detecting phishing web sites: A heuristic url-based approach, in: 2013 International Conference on Advanced Technologies for Communications (ATC 2013), IEEE. pp. 597–602.
[24] da Silva, C.M.R., Feitosa, E.L., Garcia, V.C., 2020. Heuristic-based strategy for phishing prediction: A survey of url-based approach. Computers & Security 88, 101613.
[25] Salihu, S.A., Oladipo, I.D., Wojuade, A.A., Abdulraheem, M., Babatunde, A.O., Ajiboye, A.R., Balogun, G.B., 2022. Detection of phishing urls using heuristics-based approach, in: 2022 5th Information Technology for Education and Development (ITED), pp. 1–7. doi:10.1109/ITED56637.2022.10051199.
[26] Joshi, A., Lloyd, L., Westin, P., Seethapathy, S., 2019. Using lexical features for malicious url detection–a machine learning approach. arXiv preprint arXiv:1910.06277 .
[27] Sahingoz, O.K., Buber, E., Demir, O., Diri, B., 2019. Machine learning based phishing detection from urls. Expert Systems with Applications 117, 345–357.
[28] Kolari, P., Finin, T., Joshi, A., et al., 2006. Svms for the blogosphere: Blog identification and splog detection, in: AAAI spring symposium on computational approaches to analysing weblogs.
[29] Ma, J., Saul, L.K., Savage, S., Voelker, G.M., 2009a. 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.
[30] Ma, J., Saul, L.K., Savage, S., Voelker, G.M., 2009b. Identifying suspicious urls: an application of large-scale online learning, in: Proceedings of the 26th annual international conference on machine learning, pp. 681–688.
[31] Pao, H.K., Chou, Y.L., Lee, Y.J., 2012. Malicious url detection based on kolmogorov complexity estimation, in: 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pp. 380–387. doi:10.1109/WI-IAT.2012.258.
[32] Marchal, S., François, J., State, R., Engel, T., 2014a. Phishscore: Hacking phishers' minds, in: 10th international conference on network and service management (CNSM) and workshop, IEEE. pp. 46–54.
[33] Marchal, S., François, J., State, R., Engel, T., 2014b. Phishstorm: Detecting phishing with streaming analytics. IEEE Transactions on Network and Service Management 11, 458–471.
[34] 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.
[35] Canali, D., Cova, M., Vigna, G., Kruegel, C., 2011. Prophiler: a fast filter for the large-scale detection of malicious web pages, in: Proceedings of the 20th international conference on World wide web, pp. 197–206.
[36] Bannur, S.N., Saul, L.K., Savage, S., 2011. Judging a site by its content: learning the textual, structural, and visual features of malicious web pages, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, pp. 1–10.
[37] Rayala, R., Kuppa, R., Pasumarthi, S., Karthik, S., 2023. Malicious url detection using logistic regression. Authorea Preprints .
[38] Vajrobol, V., Gupta, B.B., Gaurav, A., 2024. Mutual information based logistic regression for phishing url detection. Cyber Security and Applications 2, 100044.
[39] Yan, X., Xu, Y., Cui, B., Zhang, S., Guo, T., Li, C., 2020. Learning url embedding for malicious website detection. IEEE Transactions on Industrial Informatics 16, 6673–6681. doi:10.1109/TII.2020.2977886.
[40] Hou, Y.T., Chang, Y., Chen, T., Laih, C.S., Chen, C.M., 2010. Malicious web content detection by machine learning. expert systems with applications 37, 55–60.
[41] Xu, L., Zhan, Z., Xu, S., Ye, K., 2013. Cross-layer detection of malicious websites, in: Proceedings of the third ACM conference on Data and application security and privacy, pp. 141–152.
[42] Aggarwal, A., Rajadesingan, A., Kumaraguru, P., 2012. Phishari: Automatic realtime phishing detection on twitter, in: 2012 eCrime Researchers Summit, IEEE. pp. 1–12.
[43] Cao, J., Li, Q., Ji, Y., He, Y., Guo, D., 2016. Detection of forwarding-based malicious urls in online social networks. International Journal of Parallel Programming 44, 163–180.
[44] Zhang, H., Liu, G., Chow, T.W., Liu, W., 2011. Textual and visual content-based anti-phishing: a bayesian approach. IEEE transactions on neural networks 22, 1532–1546.
[45] Magdacy Jerjes, A.Z.A., Dawod, A.Y., Abdulqader, M.F., 2023. Detect malicious web pages using naive bayesian algorithm to detect cyber threats. Wireless Personal Communications , 1–13.
[46] Koca, M., Avcı, İ., Al-hayani, M.A.S., 2023. Classification of malicious urls using naive bayes and genetic algorithm. Sakarya University Journal of Computer and Information Sciences 6, 80–90.
[47] Vundavalli, V., Barsha, F., Masum, M., Shahriar, H., Haddad, H., 2020. Malicious url detection using supervised machine learning techniques, in: 13th International Conference on Security of Information and Networks, pp. 1–6.

[48] Mankar, N.P., Sakunde, P.E., Zurange, S., Date, A., Borate, V., Mali, Y.K., 2024. Comparative evaluation of machine learning models for malicious url detection, in: 2024 MIT Art, Design and Technology School of Computing International Conference (MITADTSoCiCon), IEEE. pp. 1–7.
[49] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y., 2006. Online passive-aggressive algorithms. Journal of Machine Learning Research 7, 551–585.
[50] Dredze, M., Crammer, K., Pereira, F., 2008. Confidence-weighted linear classification, in: Proceedings of the 25th international conference on Machine learning, pp. 264–271.
[51] 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.
[52] Crammer, K., Kulesza, A., Dredze, M., 2009. Adaptive regularization of weight vectors. Advances in neural information processing systems 22.
[53] Le, A., Markopoulou, A., Faloutsos, M., 2011. Phishdef: Url names say it all, in: 2011 Proceedings IEEE INFOCOM, IEEE. pp. 191–195.
[54] Lin, M.S., Chiu, C.Y., Lee, Y.J., Pao, H.K., 2013. Malicious url filtering—a big data application, in: 2013 IEEE international conference on big data, IEEE. pp. 589–596.
[55] Alshboul, Y., Nepali, R., Wang, Y., 2015. Detecting malicious short urls on twitter. AMCIS 2015 Proceedings Search .
[56] Pan, Y., Ding, X., 2006. Anomaly based web phishing page detection, in: 2006 22nd Annual Computer Security Applications Conference (ACSAC'06), IEEE. pp. 381–392.
[57] Wang, D., Navathe, S.B., Liu, L., Irani, D., Tamersoy, A., Pu, C., 2013. Click traffic analysis of short url spam on twitter, in: 9th IEEE international conference on collaborative computing: networking, applications and worksharing, IEEE. pp. 250–259.
[58] Hamza, A., Hammam, F., Abouzeid, M., Ahmed, M.A., Dhou, S., Aloul, F., 2024. Malicious url and intrusion detection using machine learning, in: 2024 International Conference on Information Networking (ICOIN), IEEE. pp. 795–800.
[59] Chiba, D., Tobe, K., Mori, T., Goto, S., 2012. Detecting malicious websites by learning ip address features, in: 2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet, IEEE. pp. 29–39.
[60] Zhu, Y., Zuo, Y., Li, T., 2021. Modeling of ship fuel consumption based on multisource and heterogeneous data: Case study of passenger ship. Journal of Marine Science and Engineering 9, 273.
[61] Géron, A., 2022. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.".
[62] Le, H., Pham, Q., Sahoo, D., Hoi, S.C., 2018. Urlnet: Learning a url representation with deep learning for malicious url detection. arXiv preprint arXiv:1802.03162 .
[63] Tajaddodianfar, F., Stokes, J.W., Gururajan, A., 2020. Texception: A character/word-level deep learning model for phishing url detection, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2857–2861. doi:10.1109/ICASSP40776.2020.9053670.
[64] Bu, S.J., Cho, S.B., 2021. Deep character-level anomaly detection based on a convolutional autoencoder for zero-day phishing url detection. Electronics 10, 1492.
[65] Bozkir, A.S., Dalgic, F.C., Aydos, M., 2023. Grambeddings: a new neural network for url based identification of phishing web pages through n-gram embeddings. Computers & Security 124, 102964.
[66] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2023. Attention is all you need. URL: https://arxiv.org/abs/1706.03762, arXiv:1706.03762.
[67] Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. URL: https://arxiv.org/abs/1810.04805, arXiv:1810.04805.
[68] Otieno, D.O., Abri, F., Namin, A.S., Jones, K.S., 2023. Detecting phishing urls using the bert transformer model, in: 2023 IEEE International Conference on Big Data (BigData), pp. 2483–2492. doi:10.1109/BigData59044.2023.10386782.
[69] Su, M.Y., Su, K.L., 2023. Bert-based approaches to identifying malicious urls. Sensors 23, 8499.
[70] S, J.K., B, A., 2023. Phishing url detection by leveraging roberta for feature extraction and lstm for classification, in: 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS), pp. 972–977. doi:10.1109/ICAISS58487.2023.10250684.
[71] Asiri, S., Xiao, Y., Li, T., 2024. Phishtransformer: A novel approach to detect phishing attacks using url collection and transformer. Electronics 13. URL: https://www.mdpi.com/2079-9292/13/1/30, doi:10.3390/electronics13010030.
[72] Wang, C., Chen, Y., 2022. Tcurl: Exploring hybrid transformer and convolutional neural network on phishing url detection. Knowledge-Based Systems 258, 109955. URL: https://www.sciencedirect.com/science/article/pii/S0950705122010486, doi:https://doi.org/10.1016/j.knosys.2022.109955.
[73] Mandapati, K.S., Meesala, S., Maddela, D., Ponnada, K., Neyyala, H., Shaik, E.A., 2023. A hybrid transformer ensemble approach for phishing website detection, in: 2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS), pp. 1–8. doi:10.1109/ICSSAS57918.2023.10331880.
[74] Liu, R., Wang, Y., Guo, Z., Xu, H., Qin, Z., Ma, W., Zhang, F., 2024. Transurl: Improving malicious url detection with multi-layer transformer encoding and multi-scale pyramid features. Computer Networks 253, 110707. URL: https://www.sciencedirect.com/science/article/pii/S1389128624005395, doi:https://doi.org/10.1016/j.comnet.2024.110707.
[75] Do, N.Q., Selamat, A., Fujita, H., Krejcar, O., 2024. An integrated model based on deep learning classifiers and pre-trained transformer for phishing url detection. Future Generation Computer Systems 161, 269–285. URL: https://www.sciencedirect.com/science/article/pii/S0167739X24003315, doi:https://doi.org/10.1016/j.future.2024.06.031.
[76] Mehak, G., Muneer, I., Nawab, R.M.A., 2023. Urdu text reuse detection at phrasal level using sentence transformer-based approach. Expert Systems with Applications 234, 121063. URL: https://www.sciencedirect.com/science/article/pii/S0957417423015658, doi:https://doi.org/10.1016/j.eswa.2023.121063.

[77] Wang, Y., Zhu, W., Xu, H., Qin, Z., Ren, K., Ma, W., 2023a. A large-scale pretrained deep model for phishing url detection, in: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 1–5.
[78] Wang, Y., Ma, W., Xu, H., Liu, Y., Yin, P., 2023b. A lightweight multi-view learning approach for phishing attack detection using transformer with mixture of experts. Applied Sciences 13, 7429.
[79] Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M., 2011. Exposure: Finding malicious domains using passive dns analysis., in: Ndss, pp. 1–17.
[80] Aljofey, A., Bello, S.A., Lu, J., Xu, C., 2025. Comprehensive phishing detection: A multi-channel approach with variants tcn fusion leveraging url and html features. Journal of Network and Computer Applications , 104170.
[81] Alshingiti, Z., Alaqel, R., Al-Muhtadi, J., Haq, Q.E.U., Saleem, K., Faheem, M.H., 2023. A deep learning-based phishing detection system using cnn, lstm, and lstm-cnn. Electronics 12, 232.
[82] Su, Y., 2020. Research on website phishing detection based on lstm rnn, in: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), pp. 284–288. doi:10.1109/ITNEC48623.2020.9084799.
[83] Arivukarasi, M., Antonidoss, A., 2020. Performance analysis of malicious url detection by using rnn and lstm, in: 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), pp. 454–458. doi:10.1109/ICCMC48092.2020.ICCMC-00085.
[84] Aslam, S., Aslam, H., Manzoor, A., Chen, H., Rasool, A., 2024. Antiphishstack: Lstm-based stacked generalization model for optimized phishing url detection. Symmetry 16, 248.
[85] Gupta, N., Thapliyal, S., Sharma, A., Sheladia, J., Wazid, M., Giri, D., 2024. Deep learning approach for malicious url detection using cnn, rnn, lstm and bi-lstm models, in: 2024 6th International Conference on Computational Intelligence and Networks (CINE), IEEE. pp. 1–5.
[86] Ozcan, A., Catal, C., Donmez, E., Senturk, B., 2023. A hybrid dnn–lstm model for detecting phishing urls. Neural Computing and Applications , 1–17.
[87] Li, Y., Yang, Z., Chen, X., Yuan, H., Liu, W., 2019. A stacking model using url and html features for phishing webpage detection. Future Generation Computer Systems 94, 27–39.
[88] Borgolte, K., Kruegel, C., Vigna, G., 2013. Delta: automatic identification of unknown web-based infection campaigns, in: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 109–120.
[89] Seifert, C., Welch, I., Komisarczuk, P., 2008. Identification of malicious web pages with static heuristics, in: 2008 Australasian Telecommunication Networks and Applications Conference, IEEE. pp. 91–96.
[90] Abdelhamid, N., Ayesh, A., Thabtah, F., 2014. Phishing detection based associative classification data mining. Expert Systems with Applications 41, 5948–5959.
[91] Sameen, M., Han, K., Hwang, S.O., 2020. Phishhaven—an efficient real-time ai phishing urls detection system. Ieee Access 8, 83425–83443.
[92] Pooja, A.S.S.V.L., Sridhar, M., 2020. Analysis of phishing website detection using cnn and bidirectional lstm, in: 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), pp. 1620–1629. doi:10.1109/ICECA49313.2020.9297395.
[93] Pham, T.T.T., Hoang, V.N., Ha, T.N., 2018. Exploring efficiency of character-level convolution neuron network and long short term memory on malicious url detection, in: Proceedings of the 2018 VII International Conference on Network, Communication and Computing, pp. 82–86.
[94] Opara, C., Chen, Y., Wei, B., 2024. Look before you leap: Detecting phishing web pages by exploiting raw url and html characteristics. Expert Systems with Applications 236, 121183.
[95] Feng, J., Zou, L., Ye, O., Han, J., 2020. Web2vec: Phishing webpage detection method based on multidimensional features driven by deep learning. IEEE Access 8, 221214–221224. doi:10.1109/ACCESS.2020.3043188.
[96] Ariyadasa, S., Fernando, S., Fernando, S., 2020. Detecting phishing attacks using a combined model of lstm and cnn. International Journal of ADVANCED AND APPLIED SCIENCES 7, 56–67. doi:10.21833/ijaas.2020.07.007.
[97] Manjula, M., Venkatesh, Kenchamma, R.H., Basapur, S.B., 2024. Pd-uhd features: Phishing detection approach using uncooked url, html content and domain name features, in: 2024 Second International Conference on Networks, Multimedia and Information Technology (NMITCON), pp. 1–8. doi:10.1109/NMITCON62075.2024.10699168.
[98] Opara, C., Wei, B., Chen, Y., 2020. Htmlphish: Enabling phishing web page detection by applying deep learning techniques on html analysis, in: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. doi:10.1109/IJCNN48605.2020.9207707.
[99] Ariyadasa, S., Fernando, S., Fernando, S., 2022. Combining long-term recurrent convolutional and graph convolutional networks to detect phishing sites using url and html. IEEE Access 10, 82355–82375.
[100] Ouyang, L., Zhang, Y., 2021. Phishing web page detection with html-level graph neural network, in: 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 952–958. doi:10.1109/TrustCom53373.2021.00133.
[101] Yoon, J.H., Buu, S.J., Kim, H.J., 2025. Reinforced disentangled html representation learning with hard-sample mining for phishing webpage detection. Electronics 14, 1080.
[102] Huang, Y., Yang, Q., Qin, J., Wen, W., 2019. Phishing url detection via cnn and attention-based hierarchical rnn, in: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 112–119. doi:10.1109/TrustCom/BigDataSE.2019.00024.
[103] Çolhak, F., Ecevit, M.İ., Uçar, B.E., Creutzburg, R., Dağ, H., 2024. Phishing website detection through multi-model analysis of html content, in: International Conference on Theoretical and Applied Computing, Springer. pp. 171–184.
[104] Choi, H., Zhu, B.B., Lee, H., 2011. Detecting malicious web links and identifying their attack types, in: 2nd USENIX Conference on Web Application Development (WebApps 11).
[105] Wang, Y., Cai, W.d., Wei, P.c., 2016. A deep learning approach for detecting malicious javascript code. Security and Communication Networks 9, 1520–1534.
[106] Laskov, P., Šrndić, N., 2011. Static detection of malicious javascript-bearing pdf documents, in: Proceedings of the 27th Annual Computer Security Applications Conference, Association for Computing Machinery, New York, NY, USA. p. 373–382. URL: https://doi.org/10.1145/2076732.2076785, doi:10.1145/2076732.2076785.

[107] Xu, W., Zhang, F., Zhu, S., 2013. Jstill: mostly static detection of obfuscated malicious javascript code, in: Proceedings of the Third ACM Conference on Data and Application Security and Privacy, Association for Computing Machinery, New York, NY, USA. p. 117–128. URL: https://doi.org/10.1145/2435349.2435364, doi:10.1145/2435349.2435364.

[108] AL-Taharwa, I.A., Lee, H.M., Jeng, A.B., Wu, K.P., Ho, C.S., Chen, S.M., 2015. Jsod: Javascript obfuscation detector. Security and Communication Networks 8, 1092–1107. URL: https://doi.org/10.1002/sec.1064, doi:10.1002/sec.1064.

[109] Nicolay, J., Spruyt, V., De Roover, C., 2016. Static detection of user-specified security vulnerabilities in client-side javascript, in: Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, Association for Computing Machinery, New York, NY, USA. p. 3–13. URL: https://doi.org/10.1145/2993600.2993612, doi:10.1145/2993600.2993612.

[110] Gorji, A., Abadi, M., 2014. Detecting obfuscated javascript malware using sequences of internal function calls, in: Proceedings of the 2014 ACM Southeast Conference, Association for Computing Machinery, New York, NY, USA. URL: https://doi.org/10.1145/2638404.2737181, doi:10.1145/2638404.2737181.

[111] Sayed, B., Traoré, I., Abdelhalim, A., 2014. Detection and mitigation of malicious javascript using information flow control, in: 2014 Twelfth Annual International Conference on Privacy, Security and Trust, pp. 264–273. doi:10.1109/PST.2014.6890948.

[112] Xue, Y., Wang, J., Liu, Y., Xiao, H., Sun, J., Chandramohan, M., 2015. Detection and classification of malicious javascript via attack behavior modelling, in: Proceedings of the 2015 International Symposium on Software Testing and Analysis, Association for Computing Machinery, New York, NY, USA. p. 48–59. URL: https://doi.org/10.1145/2771783.2771814, doi:10.1145/2771783.2771814.

[113] Schütt, K., Kloft, M., Bikadorov, A., Rieck, K., 2012. Early detection of malicious behavior in javascript code, in: Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence, Association for Computing Machinery, New York, NY, USA. p. 15–24. URL: https://doi.org/10.1145/2381896.2381901, doi:10.1145/2381896.2381901.

[114] Corona, I., Maiorca, D., Ariu, D., Giacinto, G., 2014. Lux0r: Detection of malicious pdf-embedded javascript code through discriminant analysis of api references, in: Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, Association for Computing Machinery, New York, NY, USA. p. 47–57. URL: https://doi.org/10.1145/2666652.2666657, doi:10.1145/2666652.2666657.

[115] Wang, J., Xue, Y., Liu, Y., Tan, T.H., 2015. Jsdc: A hybrid approach for javascript malware detection and classification, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, Association for Computing Machinery, New York, NY, USA. p. 109–120. URL: https://doi.org/10.1145/2714576.2714620, doi:10.1145/2714576.2714620.

[116] Morishige, S., Haruta, S., Asahina, H., Sasase, I., 2017. Obfuscated malicious javascript detection scheme using the feature based on divided url, in: 2017 23rd Asia-Pacific Conference on Communications (APCC), pp. 1–6. doi:10.23919/APCC.2017.8303992.

[117] Phung, N.M., Mimura, M., 2021. Detection of malicious javascript on an imbalanced dataset. Internet of Things 13, 100357. URL: https://www.sciencedirect.com/science/article/pii/S2542660521000019, doi:https://doi.org/10.1016/j.iot.2021.100357.

[118] Ndichu, S., Ozawa, S., Misu, T., Okada, K., 2018. A machine learning approach to malicious javascript detection using fixed length vector representation, in: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. doi:10.1109/IJCNN.2018.8489414.

[119] Jodavi, M., Abadi, M., Parhizkar, E., 2015. Jsobfusdetector: A binary pso-based one-class classifier ensemble to detect obfuscated javascript code, in: 2015 The International Symposium on Artificial Intelligence and Signal Processing (AISP), pp. 322–327. doi:10.1109/AISP.2015.7123508.

[120] Fang, Y., Huang, C., Liu, L., Xue, M., 2018. Research on malicious javascript detection technology based on lstm. IEEE Access 6, 59118–59125. doi:10.1109/ACCESS.2018.2874098.

[121] Song, X., Chen, C., Cui, B., Fu, J., 2020. Malicious javascript detection based on bidirectional lstm model. Applied Sciences 10. URL: https://www.mdpi.com/2076-3417/10/10/3440, doi:10.3390/app10103440.

[122] Alex, S., Dhiliphan Rajkumar, T., 2021. Taylor–hho algorithm: A hybrid optimization algorithm with deep long short-term for malicious javascript detection. International Journal of Intelligent Systems 36, 7153–7176.

[123] Dabral, S., Agarwal, A., Mahajan, M., Kumar, S., 2017. Malicious pdf files detection using structural and javascript based features, in: Information, Communication and Computing Technology: Second International Conference, ICICCT 2017, New Delhi, India, May 13, 2017, Revised Selected Papers 2, Springer. pp. 137–147.

[124] Wang, Q., Zhou, J., Chen, Y., Zhang, Y., Zhao, J., 2013. Extracting urls from javascript via program analysis, in: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, Association for Computing Machinery, New York, NY, USA. p. 627–630. URL: https://doi.org/10.1145/2491411.2494583, doi:10.1145/2491411.2494583.

[125] He, X., Xu, L., Cha, C., 2018. Malicious javascript code detection based on hybrid analysis, in: 2018 25th Asia-Pacific Software Engineering Conference (APSEC), IEEE. pp. 365–374.

[126] Fang, Y., Huang, C., Zeng, M., Zhao, Z., Huang, C., 2022. Jstrong: Malicious javascript detection based on code semantic representation and graph neural network. Computers & Security 118, 102715. URL: https://www.sciencedirect.com/science/article/pii/S0167404822001110, doi:https://doi.org/10.1016/j.cose.2022.102715.

[127] Rozi, M.F., Ban, T., Ozawa, S., Yamada, A., Takahashi, T., Kim, S., Inoue, D., 2023. Detecting malicious javascript using structure-based analysis of graph representation. IEEE Access 11, 102727–102745. doi:10.1109/ACCESS.2023.3317266.

[128] Varshney, G., Misra, M., Atrey, P.K., 2016. A survey and classification of web phishing detection schemes. Security and Communication Networks 9, 6266–6284.

[129] Aung, E.S., Zan, C.T., Yamana, H., 2019. A survey of url-based phishing detection, in: DEIM forum, pp. G2–3.

[130] Fu, A.Y., Wenyin, L., Deng, X., 2006. Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd). IEEE transactions on dependable and secure computing 3, 301–311.

[131] Liu, W., Deng, X., Huang, G., Fu, A.Y., 2006. An antiphishing strategy based on visual similarity assessment. IEEE Internet Computing 10, 58–65.

[132] Wenyin, L., Huang, G., Xiaoyue, L., Min, Z., Deng, X., 2005. Detection of phishing webpages based on visual similarity, in: Special interest tracks and posters of the 14th international conference on World Wide Web, pp. 1060–1061.

[133] Dunlop, M., Groat, S., Shelly, D., 2010. Goldphish: Using images for content-based phishing analysis, in: 2010 Fifth international conference on internet monitoring and protection, IEEE. pp. 123–128.
[134] Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. International journal of computer vision 60, 91–110.
[135] Afroz, S., Greenstadt, R., 2011. Phishzoo: Detecting phishing websites by looking at them, in: 2011 IEEE fifth international conference on semantic computing, IEEE. pp. 368–375.
[136] Huang, C.Y., Ma, S.P., Yeh, W.L., Lin, C.Y., Liu, C.T., 2010. Mitigate web phishing using site signatures, in: TENCON 2010 - 2010 IEEE Region 10 Conference, pp. 803–808. doi:10.1109/TENCON.2010.5686582.
[137] Wang, G., Liu, H., Becerra, S., Wang, K., Belongie, S., Shacham, H., Savage, S., 2011. Verilogo: Proactive phishing detection via logo recognition. Department of Computer Science main & Engineering .
[138] Chen, J.L., Ma, Y.W., Huang, K.L., 2020. Intelligent visual similarity-based phishing websites detection. Symmetry 12. URL: https://www.mdpi.com/2073-8994/12/10/1681.
[139] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., 2008. Speeded-up robust features (surf). Computer vision and image understanding 110, 346–359.
[140] Rao, R.S., Ali, S.T., 2015. A computer vision technique to detect phishing attacks, in: 2015 Fifth International Conference on Communication Systems and Network Technologies, pp. 596–601. doi:10.1109/CSNT.2015.68.
[141] Kazemian, H., Ahmed, S., 2015. Comparisons of machine learning techniques for detecting malicious webpages. Expert Systems with Applications 42, 1166–1177. URL: https://www.sciencedirect.com/science/article/pii/S0957417414005284, doi:https://doi.org/10.1016/j.eswa.2014.08.046.
[142] Huang, C.R., Chen, C.S., Chung, P.C., 2008. Contrast context histogram—an efficient discriminating local descriptor for object recognition and image matching. Pattern Recognition 41, 3071–3077. URL: https://www.sciencedirect.com/science/article/pii/S0031320308000988, doi:https://doi.org/10.1016/j.patcog.2008.03.013.
[143] Chen, K.T., Chen, J.Y., Huang, C.R., Chen, C.S., 2009. Fighting phishing with discriminative keypoint features. IEEE Internet Computing 13, 56–63.
[144] Arade, M.S., Bhaskar, P., Kamat, R., 2011. Antiphishing model with url & image based webpage matching. Int. J. Comput. Sci. Technol. IJCST 2, 282–286.
[145] Balamuralikrishna, T., Raghavendrasai, N., Sukumar, M.S., 2012. Mitigating online fraud by ant phishing model with url & image based webpage matching. International Journal of Scientific & Engineering Research 3, 1–6.
[146] Al-Ahmadi, S., 2020. A deep learning technique for web phishing detection combined url features and visual similarity. International Journal of Computer Networks & Communications (IJCNC) Vol 12.
[147] Abdelnabi, S., Krombholz, K., Fritz, M., 2020. Visualphishnet: Zero-day phishing website detection by visual similarity, in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Association for Computing Machinery, New York, NY, USA. p. 1681–1698. URL: https://doi.org/10.1145/3372297.3417233, doi:10.1145/3372297.3417233.
[148] Lam, I.F., Xiao, W.C., Wang, S.C., Chen, K.T., 2009. Counteracting phishing page polymorphism: An image layout analysis approach, in: Advances in Information Security and Assurance, Third International Conference and Workshops, ISA 2009, pp. 270–279. doi:10.1007/978-3-642-02617-1_28.
[149] Otsu, N., 1979. A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics 9, 62–66. doi:10.1109/TSMC.1979.4310076.
[150] Mao, J., Li, P., Li, K., Wei, T., Liang, Z., 2013. Baitalarm: Detecting phishing sites using similarity in fundamental visual features, in: 2013 5th International Conference on Intelligent Networking and Collaborative Systems, pp. 790–795. doi:10.1109/INCoS.2013.151.
[151] Gupta, B.B., Mishra, A., 2014. Hybrid solution to detect and filter zero-day phishing attacks, in: International Conference on Emerging Research in Computing, Information, Communication and Applications.
[152] Hara, M., Yamada, A., Miyake, Y., 2009. Visual similarity-based phishing detection without victim site information, in: 2009 IEEE Symposium on Computational Intelligence in Cyber Security, pp. 30–36. doi:10.1109/CICYBS.2009.4925087.
[153] Huang, H., Qian, L., Wang, Y., 2012. A svm-based technique to detect phishing urls. Information Technology Journal 11, 921–925.
[154] Sorio, E., Bartoli, A., Medvet, E., 2013. Detection of hidden fraudulent urls within trusted sites using lexical features, in: 2013 International Conference on Availability, Reliability and Security, IEEE. pp. 242–247.
[155] Patil, D.R., Patil, J.B., et al., 2018. Malicious urls detection using decision tree classifiers and majority voting technique. Cybernetics and Information Technologies 18, 11–29.
[156] Machado, L., Gadge, J., 2017. Phishing sites detection based on c4.5 decision tree algorithm, in: 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), pp. 1–5. doi:10.1109/ICCUBEA.2017.8463818.
[157] Sangra, E., Agrawal, R., Gundalwar, P.R., Sharma, K., Bangri, D., Nandi, D., 2024. Malicious website detection using random forest and pearson correlation for effective feature selection. International Journal of Advanced Computer Science and Applications 15. URL: http://dx.doi.org/10.14569/IJACSA.2024.0150876, doi:10.14569/IJACSA.2024.0150876.
[158] Zhu, E., Ju, Y., Chen, Z., Liu, F., Fang, X., 2020. Dtof-ann: An artificial neural network phishing detection model based on decision tree and optimal features. Applied Soft Computing 95.
[159] Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation 9, 1735–1780. doi:10.1162/neco.1997.9.8.1735.
[160] Odeh, A., Al-Haija, Q.A., Aref, A., Taleb, A.A., 2023. Comparative study of catboost, xgboost, and lightgbm for enhanced url phishing detection: a performance assessment. Journal of Internet Services and Information Security 13, 1–11.
[161] Jovanovic, L., Jovanovic, D., Antonijevic, M., Nikolic, B., Bacanin, N., Zivkovic, M., Strumberger, I., 2023. Improving phishing website detection using a hybrid two-level framework for feature selection and xgboost tuning. Journal of Web Engineering 22, 543–574. doi:10.13052/jwe1540-9589.2237.
[162] Vanitha, N., Vinodhini, V., 2019. Malicious-url detection using logistic regression technique. International Journal of Engineering and Management Research (IJEMR) 9, 108–113.

[163] Gonaygunta, H., 2023. Machine learning algorithms for detection of cyber threats using logistic regression. Department of Information Technology, University of the Cumberlands .
[164] Pastika, P.B., Alamsyah, A., 2024. Machine learning-based malicious website detection using logistic regression algorithm. Engineering, MAthematics and Computer Science Journal (EMACS) 6, 207–213.
[165] Thakur, I., Panda, K., Kumar, S., 2022. Deep learning methods for malicious url detection using embedding techniques as logistic regression with lasso penalty and random forest, in: 2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC), pp. 181–186. doi:10.1109/PDGC56933.2022.10053199.
[166] Rashid, F., Ranaweera, N., Doyle, B., Seneviratne, S., 2025. Llms are one-shot url classifiers and explainers. Computer Networks 258, 111004.
[167] Li, L., Gong, B., 2023. Prompting large language models for malicious webpage detection, in: 2023 IEEE 4th International Conference on Pattern Recognition and Machine Learning (PRML), pp. 393–400. doi:10.1109/PRML59573.2023.10348229.
[168] Lee, J., Lim, P., Hooi, B., Divakaran, D.M., 2024. Multimodal large language models for phishing webpage detection and identification. arXiv preprint arXiv:2408.05941 .
[169] Koide, T., Nakano, H., Chiba, D., 2024. Chatphishdetector: Detecting phishing sites using large language models. IEEE Access .
[170] Desolda, G., Greco, F., Viganò, L., 2024. Apollo: A gpt-based tool to detect phishing emails and generate explanations that warn users. arXiv preprint arXiv:2410.07997 .
[171] Trad, F., Chehab, A., 2024. Prompt engineering or fine-tuning? a case study on phishing detection with large language models. Machine Learning and Knowledge Extraction 6, 367–384.
[172] Liu, R., Wang, Y., Xu, H., Qin, Z., Zhang, F., Liu, Y., Cao, Z., 2025. Pmanet: Malicious url detection via post-trained language model guided multi-level feature attention network. Information Fusion 113, 102638. URL: https://www.sciencedirect.com/science/article/pii/S1566253524004160, doi:https://doi.org/10.1016/j.inffus.2024.102638.
[173] Memon, A., Manjotho, A.A., 2024. Apformer: Anti-phishing transformer for website-phishing detection via joint feature learning, in: 2024 International Conference on Engineering & Computing Technologies (ICECT), IEEE. pp. 1–5.
[174] Ejaz, A., Mian, A.N., Manzoor, S., 2023. Life-long phishing attack detection using continual learning. Scientific reports 13, 11488.
[175] Huang, Z., Zhang, B., Hu, G., Li, L., Xu, Y., Jin, Y., 2023. Enhancing unsupervised anomaly detection with score-guided network. IEEE Transactions on Neural Networks and Learning Systems .
[176] Huang, C., Yang, Z., Wen, J., Xu, Y., Jiang, Q., Yang, J., Wang, Y., 2021. Self-supervision-augmented deep autoencoder for unsupervised visual anomaly detection. IEEE Transactions on Cybernetics 52, 13834–13847.
[177] Ma, J., Saul, L.K., Savage, S., Voelker, G.M., 2011. Learning to detect malicious urls. ACM Transactions on Intelligent Systems and Technology (TIST) 2, 1–24.
[178] Zhang, W., Ding, Y.X., Tang, Y., Zhao, B., 2011. Malicious web page detection based on on-line learning algorithm, in: 2011 International Conference on Machine Learning and Cybernetics, IEEE. pp. 1914–1919.
[179] Ma, J., Kulesza, A., Dredze, M., Crammer, K., Saul, L., Pereira, F., 2010. Exploiting feature covariance in high-dimensional online learning, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings. pp. 493–500.
[180] Thomas, K., Grier, C., Ma, J., Paxson, V., Song, D., 2011. Design and evaluation of a real-time url spam filtering service, in: 2011 IEEE symposium on security and privacy, IEEE. pp. 447–462.
[181] Huang, D., Xu, K., Pei, J., 2014. Malicious url detection by dynamically mining patterns without pre-defined elements. World Wide Web 17, 1375–1394.
[182] 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 .
[183] Shima, K., Miyamoto, D., Abe, H., Ishihara, T., Okada, K., Sekiya, Y., Asai, H., Doi, Y., 2018. Classification of url bitstreams using bag of bytes, in: 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), IEEE. pp. 1–5.
[184] Yu, B., Pan, J., Hu, J., Nascimento, A., De Cock, M., 2018. Character level based detection of dga domain names, in: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–8.
[185] Rajitha, K., Vijayalakshmi, D., 2018. Suspicious urls filtering using optimal rt-pfl: A novel feature selection based web url detection, in: Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016, Volume 2, Springer. pp. 227–235.
[186] Jain, A.K., Gupta, B.B., 2018. Towards detection of phishing websites on client-side using machine learning based approach. Telecommunication Systems 68, 687–700.
[187] Verma, R., Das, A., 2017. What's in a url: Fast feature extraction and malicious url detection, in: Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics, pp. 55–63.
[188] Buber, E., Diri, B., Sahingoz, O.K., 2017. Nlp based phishing attack detection from urls, in: International Conference on Intelligent Systems Design and Applications, Springer. pp. 608–618.
[189] He, M., Horng, S.J., Fan, P., Khan, M.K., Run, R.S., Lai, J.L., Chen, R.J., Sutanto, A., 2011. An efficient phishing webpage detector. Expert systems with applications 38, 12018–12027.
[190] Yoo, S., Kim, S., Choudhary, A., Roy, O., Tuithung, T., 2014. Two-phase malicious web page detection scheme using misuse and anomaly detection. International Journal of Reliable Information and Assurance 2, 1–9.
[191] Altay, B., Dokeroglu, T., Cosar, A., 2019. Context-sensitive and keyword density-based supervised machine learning techniques for malicious webpage detection. Soft Computing 23, 4177–4191.
[192] Jain, A.K., Gupta, B.B., 2019. A machine learning based approach for phishing detection using hyperlinks information. Journal of Ambient Intelligence and Humanized Computing 10, 2015–2028.
[193] Choi, Y., Kim, T., Choi, S., Lee, C., 2009. Automatic detection for javascript obfuscation attacks in web pages through string pattern analysis, in: Future Generation Information Technology: First International Conference, FGIT 2009, Jeju Island, Korea, December 10-12,

2009. Proceedings 1, Springer. pp. 160–172.
[194] Yuan, J., Liu, Y., Yu, L., 2021. A novel approach for malicious url detection based on the joint model. Security and Communication Networks 2021, 4917016.
[195] Maci, A., Santorsola, A., Coscia, A., Iannacone, A., 2023. Unbalanced web phishing classification through deep reinforcement learning. Computers 12, 118.
[196] DR, U.S., Patil, A., et al., 2023. Malicious url detection and classification analysis using machine learning models, in: 2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), IEEE. pp. 470–476.
[197] Do Xuan, C., Nguyen, H.D., Tisenko, V.N., 2020. Malicious url detection based on machine learning. International Journal of Advanced Computer Science and Applications 11.
[198] Patgiri, R., Katari, H., Kumar, R., Sharma, D., 2019. Empirical study on malicious url detection using machine learning, in: Distributed Computing and Internet Technology: 15th International Conference, ICDCIT 2019, Bhubaneswar, India, January 10–13, 2019, Proceedings 15, Springer. pp. 380–388.
[199] Tong, X., Jin, B., Wang, J., Yang, Y., Suo, Q., Wu, Y., 2023. Mm-convbert-lms: detecting malicious web pages via multi-modal learning and pre-trained model. Applied Sciences 13, 3327.
[200] Nagy, N., Aljabri, M., Shaahid, A., Ahmed, A.A., Alnasser, F., Almakramy, L., Alhadab, M., Alfaddagh, S., 2023. Phishing urls detection using sequential and parallel ml techniques: comparative analysis. Sensors 23, 3467.
[201] Alsaedi, M., Ghaleb, F.A., Saeed, F., Ahmad, J., Alasli, M., 2022. Cyber threat intelligence-based malicious url detection model using ensemble learning. Sensors 22, 3373.
[202] Hajaj, C., Hason, N., Dvir, A., 2022. Less is more: Robust and novel features for malicious domain detection. Electronics 11, 969.
[203] Umer, M., Sadiq, S., Karamti, H., Alhebshi, R.M., Alnowaiser, K., Eshmawi, A., Song, H., Ashraf, I., 2022. Deep learning-based intrusion detection methods in cyber-physical systems: Challenges and future trends. Electronics 11, 3326.
[204] Elsadig, M., Ibrahim, A.O., Basheer, S., Alohali, M.A., Alshunaifi, S., Alqahtani, H., Alharbi, N., Nagmeldin, W., 2022. Intelligent deep machine learning cyber phishing url detection based on bert features extraction. Electronics 11, 3647.
[205] Abdul Samad, S.R., Balasubaramanian, S., Al-Kaabi, A.S., Sharma, B., Chowdhury, S., Mehbodniya, A., Webber, J.L., Bostani, A., 2023. Analysis of the performance impact of fine-tuned machine learning model for phishing url detection. Electronics 12, 1642.
[206] Kumi, S., Lim, C., Lee, S.G., 2021. Malicious url detection based on associative classification. Entropy 23, 182.
[207] Roy, S.S., Awad, A.I., Amare, L.A., Erkihun, M.T., Anas, M., 2022. Multimodel phishing url detection using lstm, bidirectional lstm, and gru models. Future Internet 14, 340.
[208] Al-Kabi, M., Wahsheh, H., Alsmadi, I., Al-Shawakfa, E., Wahbeh, A., Al-Hmoud, A., 2012. Content-based analysis to detect arabic web spam. Journal of Information Science 38, 284–296.
[209] Al-Kabi, M.N., Wahsheh, H.A., Alsmadi, I.M., 2014. Olawsds: an online arabic web spam detection system. Int J Adv Comput Sci Appl 5, 105–110.
[210] El-Mohdy, E.M., El-Gamal, A., Elrefaey, H., 2018. Web mining techniques to block spam web sites. International Journal of Computer Applications 975, 8887.
[211] Wahsheh, H.A., Al-Kabi, M.N., Alsmadi, I.M., 2013. A link and content hybrid approach for arabic web spam detection. International Journal of Intelligent Systems and Applications (IJISA) 5, 30–43.
[212] Alsaleh, M., Alarifi, A., 2016. Analysis of web spam for non-english content: toward more effective language-based classifiers. PloS one 11, e0164383.
[213] Al Twairesh, N., Al Tuwaijri, M., Al Moammar, A., Al Humoud, S., 2016. Arabic spam detection in twitter, in: The 2nd Workshop on Arabic Corpora and Processing Tools 2016 Theme: Social Media, p. 38.
[214] Alkhair, M., Meftouh, K., Smaïli, K., Othman, N., 2019. An arabic corpus of fake news: Collection, analysis and classification, in: Arabic Language Processing: From Theory to Practice: 7th International Conference, ICALP 2019, Nancy, France, October 16–17, 2019, Proceedings 7, Springer. pp. 292–302.
[215] Mataoui, M., Zelmati, O., Boughaci, D., Chaouche, M., Lagoug, F., 2017. A proposed spam detection approach for arabic social networks content, in: 2017 International Conference on Mathematics and Information Technology (ICMIT), IEEE. pp. 222–226.
[216] Najadat, H., Alzubaidi, M.A., Qarqaz, I., 2021. Detecting arabic spam reviews in social networks based on classification algorithms. Transactions on Asian and Low-Resource Language Information Processing 21, 1–13.
[217] Mubarak, H., Abdelali, A., Hassan, S., Darwish, K., 2020. Spam detection on arabic twitter, in: Social Informatics: 12th International Conference, SocInfo 2020, Pisa, Italy, October 6–9, 2020, Proceedings 12, Springer. pp. 237–251.
[218] Alsulami, M.M., Al-Aama, A.Y., 2020. Sentifilter: A personalized filtering model for arabic semi-spam content based on sentimental and behavioral analysis. Int. J. Adv. Comput. Sci. Appl 11.
[219] Alkadri, A.M., Elkorany, A., Ahmed, C., 2022. Enhancing detection of arabic social spam using data augmentation and machine learning. Applied Sciences 12. URL: https://www.mdpi.com/2076-3417/12/22/11388, doi:10.3390/app122211388.
[220] Ezzat, C.A., Alkadri, A.M., Elkorany, A., 2025. A real-time framework for opinion spam detection in arabic social networks. Egyptian Informatics Journal 29, 100626.
[221] Kihal, M., Hamza, L., 2025. Efficient arabic and english social spam detection using a transformer and 2d convolutional neural network-based deep learning filter. International Journal of Information Security 24, 56. URL: https://doi.org/10.1007/s10207-024-00975-0, doi:10.1007/s10207-024-00975-0.
[222] Saeed, R.M., Rady, S., Gharib, T.F., 2022. An ensemble approach for spam detection in arabic opinion texts. Journal of King Saud University - Computer and Information Sciences 34, 1407–1416. URL: https://www.sciencedirect.com/science/article/pii/S1319157819307414, doi:https://doi.org/10.1016/j.jksuci.2019.10.002.
[223] Alorini, D., 2018. Towards Machine Learning for Gulf Dialectical Arabic Malicious Content Detection in Social Media. Ph.D. thesis. Howard University.

[224] Wahsheh, H.A., Al-kabi, M.N., Alsmadi, I.M., 2012. Evaluating arabic spam classifiers using link analysis, in: Proceedings of the 3rd international conference on information and communication systems, pp. 1–5.
[225] Alharbi, A.R., Aljaedi, A., 2019. Predicting rogue content and arabic spammers on twitter. Future Internet 11, 229.
[226] Alsufyani, S., Alajmani, S., 2025. A deep learning for arabic sms phishing based on urls detection. International Journal of Advanced Computer Science & Applications 16.
[227] AlGhamdi, M.A., Khan, M.A., 2020. Intelligent analysis of arabic tweets for detection of suspicious messages. Arabian Journal for Science and Engineering 45, 6021–6032.
[228] Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., Woźniak, M., 2020. Accurate and fast url phishing detector: a convolutional neural network approach. Computer Networks 178, 107275.
[229] Kamran, S.A., Sengupta, S., Tavakkoli, A., 2021. Semi-supervised conditional gan for simultaneous generation and detection of phishing urls: A game theoretic perspective. arXiv preprint arXiv:2108.01852 .
[230] Nowroozi, E., Abhishek, Mohammadi, M., Conti, M., 2023. An adversarial attack analysis on malicious advertisement url detection framework. IEEE Transactions on Network and Service Management 20, 1332–1344. doi:10.1109/TNSM.2022.3225217.
[231] Mahdaouy, A.E., Lamsiyah, S., Idrissi, M.J., Alami, H., Yartaoui, Z., Berrada, I., 2024. Domurls_bert: Pre-trained bert-based model for malicious domains and urls detection and classification. arXiv preprint arXiv:2409.09143 .
[232] Guo, W., Wang, Q., Yue, H., Sun, H., Hu, R.Q., 2025. Efficient phishing url detection using graph-based machine learning and loopy belief propagation. URL: https://arxiv.org/abs/2501.06912, arXiv:2501.06912.
[233] Wang, J., Chen, P., Xu, X., Wu, J., Shen, M., Xuan, Q., Yang, X., 2022. Tsgn: Transaction subgraph networks assisting phishing detection in ethereum. URL: https://arxiv.org/abs/2208.12938, arXiv:2208.12938.
[234] Li, Y., Wang, Y., Xu, H., Guo, Z., Cao, Z., Zhang, L., 2024. Urlbert: A contrastive and adversarial pre-trained model for url classification. arXiv preprint arXiv:2402.11495 .
[235] Li, Y., Wang, Y., Xu, H., Guo, Z., Zhang, F., Liu, R., Ma, W., 2023. Fed-urlbert: Client-side lightweight federated transformers for url threat analysis. arXiv preprint arXiv:2312.03636 .
[236] Lee, S., Kim, J., 2012. Warningbird: Detecting suspicious urls in twitter stream., in: Ndss, pp. 1–13.
[237] Cao, C., Caverlee, J., 2015. Detecting spam urls in social media via behavioral analysis, in: Advances in Information Retrieval: 37th European Conference on IR Research, ECIR 2015, Vienna, Austria, March 29-April 2, 2015. Proceedings 37, Springer. pp. 703–714.
[238] McDonald, R., Hall, K., Mann, G., 2010. Distributed training strategies for the structured perceptron, in: Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics, pp. 456–464.
[239] Singer, Y., Duchi, J.C., 2009. Efficient learning using forward-backward splitting. Advances in Neural Information Processing Systems 22.
[240] Whittaker, C., Ryner, B., Nazif, M., 2010. Large-scale automatic classification of phishing pages., in: Ndss, p. 2010.
[241] Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J., 2008. Liblinear: A library for large linear classification. the Journal of machine Learning research 9, 1871–1874.
[242] Eshete, B., Villafiorita, A., Weldemariam, K., 2013. Binspect: Holistic analysis and detection of malicious web pages, in: Security and Privacy in Communication Networks: 8th International ICST Conference, SecureComm 2012, Padua, Italy, September 3-5, 2012. Revised Selected Papers 8, Springer. pp. 149–166.
[243] Oest, A., Safaei, Y., Doupé, A., Ahn, G.J., Wardman, B., Tyers, K., 2019. Phishfarm: A scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists, in: 2019 IEEE Symposium on Security and Privacy (SP), pp. 1344–1361. doi:10.1109/SP.2019.00049.
[244] Oest, A., Safaei, Y., Zhang, P., Wardman, B., Tyers, K., Shoshitaishvili, Y., Doupé, A., 2020. PhishTime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists, in: 29th USENIX Security Symposium (USENIX Security 20), USENIX Association. pp. 379–396. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/oest-phishtime.
[245] Oest, A., Safei, Y., Doupé, A., Ahn, G.J., Wardman, B., Warner, G., 2018. Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis, in: 2018 APWG Symposium on Electronic Crime Research (eCrime), pp. 1–12. doi:10.1109/ECRIME.2018.8376206.
[246] Zhang, P., Oest, A., Cho, H., Sun, Z., Johnson, R., Wardman, B., Sarker, S., Kapravelos, A., Bao, T., Wang, R., Shoshitaishvili, Y., Doupé, A., Ahn, G.J., 2021. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing, in: 2021 IEEE Symposium on Security and Privacy (SP), pp. 1109–1124. doi:10.1109/SP40001.2021.00021.
[247] Oest, A., Zhang, P., Wardman, B., Nunes, E., Burgis, J., Zand, A., Thomas, K., Doupé, A., Ahn, G.J., 2020. Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale, in: 29th USENIX Security Symposium (USENIX Security 20), USENIX Association. pp. 361–377. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/oest-sunrise.
[248] Maroofi, S., Korczyński, M., Duda, A., 2020. Are you human? resilience of phishing detection to evasion techniques based on human verification, in: Proceedings of the ACM Internet Measurement Conference, Association for Computing Machinery, New York, NY, USA. p. 78–86. URL: https://doi.org/10.1145/3419394.3423632, doi:10.1145/3419394.3423632.
[249] Li, W., He, Y., Wang, Z., Alqahtani, S., Nanda, P., 2023. Uncovering flaws in anti-phishing blacklists for phishing websites using novel cloaking techniques, in: Proceedings of the 20th International Conference on Security and Cryptography - Volume 1: SECRYPT, INSTICC. SciTePress. pp. 813–821. doi:10.5220/0012135600003555.
[250] Yuan, Y., Apruzzese, G., Conti, M., 2025. Beyond the west: Revealing and bridging the gap between western and chinese phishing website detection. Computers & Security 148, 104115. URL: https://www.sciencedirect.com/science/article/pii/S0167404824004206, doi:https://doi.org/10.1016/j.cose.2024.104115.
[251] Kaelbling, L.P., Littman, M.L., Moore, A.W., 1996. Reinforcement learning: a survey. J. Artif. Int. Res. 4, 237–285.
[252] Ngo Bibinbe, A.M.S., Mbouopda, M.F., Mbiadou Saleu, G.R., Mephu Nguifo, E., 2022. A survey on unsupervised learning algorithms for detecting abnormal points in streaming data, in: 2022 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. doi:10.1109/IJCNN55064.2022.9892195.

[253] Hoi, S.C., Sahoo, D., Lu, J., Zhao, P., 2021. Online learning: A comprehensive survey. Neurocomputing 459, 249–289.