IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 April 2024, accepted 3 June 2024, date of publication 11 June 2024, date of current version 24 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3412331

RESEARCH ARTICLE

Enhancing Malicious URL Detection: A Novel Framework Leveraging Priority Coefficient and Feature Evaluation

AHMAD SAHBAN RAFSANJANI[1], (Member, IEEE), NORSHALIZA BINTI KAMARUDDIN[2], (Member, IEEE), MEHRAN BEHJATI[1], SAAD ASLAM[1], AALIYA SARFARAZ[1], (Member, IEEE), AND ANGELA AMPHAWAN[1,3], (Senior Member, IEEE)

[1]School of Engineering and Technology, Sunway University, Bandar Sunway, Selangor Darul Ehsan 47500, Malaysia
[2]Faculty of Artificial Intelligence, Universiti Teknologi Malaysia, Kuala Lumpur 54100, Malaysia
[3]Smart Photonics Research Laboratory, School of Engineering and Technology, Sunway University, Selangor 47500 Malaysia

Corresponding author: Ahmad Sahban Rafsanjani (ahmadsahban@sunway.edu.my)

ABSTRACT Malicious Uniform Resource Locators (URLs) pose a significant cybersecurity threat by carrying out attacks such as phishing and malware propagation. Conventional malicious URL detection methods, relying on blacklists and heuristics, often struggle to identify new and obfuscated malicious URLs. To address this challenge, machine learning and deep learning have been leveraged to enhance detection capabilities, albeit relying heavily on large and frequently updated datasets. Furthermore, the efficacy of these methods is intrinsically tied to the quality of the training data, a requirement that becomes increasingly challenging to fulfill in real-world scenarios due to constraints such as data scarcity and the dynamic nature of evolving cyber threats. In this study, we introduce an innovative framework for malicious URL detection based on predefined static feature classification by allocating priority coefficients and feature evaluation methods. Our feature classification encompasses 42 classes, including blacklist, lexical, host-based, and content-based features. To validate our framework, we collected a dataset of 5000 real-world URLs from prominent phishing and malware websites, namely URLhaus and PhishTank. We assessed our framework's performance using three supervised machine learning methods: Support Vector Machine (SVM), Random Forest (RF), and Bayesian Network (BN). The results demonstrate that our framework outperforms these methods, achieving an impressive detection accuracy of 98.95% and a precision value of 98.60%. Furthermore, we conducted a benchmarking analysis against three comprehensive malicious URL detection methods (PDRCNN, the Li method, and URLNet), demonstrating that our proposed framework excels in terms of accuracy and precision. In conclusion, our novel malicious URL detection framework substantially enhances accuracy, significantly bolstering cybersecurity defenses against emerging threats.

INDEX TERMS Malicious URL detection, phishing, malware, network security, feature extraction, cyber threats, machine learning.

I. INTRODUCTION

The continuous evolution of Internet technology has spurred the proliferation of online services, including critical functions like electronic commerce and online banking, which involve the transmission of sensitive data such as credit card information and personal details. This surge in Internet usage has, in turn, led to a significant rise in diverse cyberattacks targeting unsuspecting users. Consequently, safeguarding the security of sensitive information during online transactions has become imperative. Among the myriads of cyber threats, malicious URLs stand out as one of the most prominent. Clicking on malicious websites is responsible for a substantial majority of cyberattacks [1], with these URLs hosting deceptive content that ensnares unsuspecting visitors, resulting in financial losses and data breaches. The rapid

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru.

proliferation of malicious URL tactics further compounds the challenge of detection, as cyberattacks employ sophisticated obfuscation techniques to cloak the true nature of these URLs [2]. The proliferation of phishing and malware websites serves as a stark illustration of this trend, with their numbers surging from a few thousand in January 2007 to over 2 million by January 2021 [3].

Malicious URL attacks encompass various categories, such as Cross-Site Scripting (XSS), malware (drive-by downloads), JavaScript obfuscation, SQL injection, clickjacking, and phishing [1]. Among these, phishing and malware stand out as the most prevalent malicious URL attacks, affecting millions of people daily and capable of targeting various operating systems [4], [5]. Phishing involves deceptive social engineering techniques used by attackers to trick unsuspecting victims into disclosing their personal information [6]. On the other hand, malware, a shortened form of malicious software, refers to code crafted by cyber attackers to cause extensive damage to data and systems or gain unauthorized access to a network [7].

Numerous scientific studies have presented a wide array of methods for detecting malicious URLs. Current solutions can generally be categorized into four main approaches: blacklist-based, heuristic-based, machine learning-based, and deep learning-based methods [8], [9]. An efficient method for detecting malicious URLs should adhere to specific criteria, ensuring a high level of accuracy. However, detecting malicious URLs faces a variety of challenges [10], [11], [12]. These challenges are:

- **Realtime Detection.** A robust method for detecting malicious URLs should promptly notify users of potentially harmful websites before they visit them, ensuring their protection. It should swiftly determine whether a URL is safe or malicious, minimizing delays and providing a seamless user experience.
- **Detect New URLs.** An effective method for detecting malicious URLs should have the capability to identify newly created websites and protect users from real-world cybercrime attacks and associated threats
- **Effective Detection.** It is crucial for an effective malicious URL detection method to accurately identify URLs while minimizing false positives and false negatives. Moreover, it must handle large-scale datasets and provide timely responses to match the dynamic and evolving cyber threat landscape.

Despite the numerous studies conducted on detecting malicious URLs, several fundamental issues still require attention. The primary concern lies in data dependence, which has garnered increased scrutiny. The reliability and accuracy of these methods heavily depend on the quality of the training dataset. The second prominent challenge that warrants consideration is the absence of prioritization for classes and features based on their importance within current detection methods. In reality, each class and feature hold distinct levels of significance in the detection of malicious URLs and facing them individually should yield distinct results. The third critical challenge that plays a vital role in detecting malicious URLs, albeit often given less attention, is feature evaluation. This process involves assessing the value of each feature, which is obtained from feature classification, to determine whether a URL is benign or malicious. In cases where the value of a feature is not available, this evaluation becomes particularly crucial.

To contribute to the research gap, we developed a novel framework to detect malicious URLs based on predefined static feature classifications by allocating priority coefficients and employing feature evaluation methods. This novel framework aims to enhance the accuracy and effectiveness of malicious URL detection. The key contributions of this work can be summarized as follows:

- **Heuristic-Based Real-Time Detection:** We have devised an innovative heuristic-based framework for the real-time detection of newly emerging phishing and malware URLs, achieving a remarkable level of accuracy.
- **Predefined Static Feature Classification:**Our work introduces a predefined static feature classification method explicitly tailored for malicious URL detection. This method assigns predefined values to various feature classes, encompassing 42 distinct categories, including blacklist, lexical, host-based, and content-based features. This comprehensive feature classification enhances detection accuracy and effectiveness.
- **Priority Coefficient Allocation:** To further enhance detection accuracy, we implement a priority coefficient mechanism that assigns weight to selected feature classes based on their significance. This prioritization amplifies the impact of crucial feature classes known for their effectiveness in identifying malicious URLs.
- **Meticulous Feature Evaluation:**Our research presents a meticulous feature evaluation method designed to assess the contributions of individual features comprehensively. It systematically determines the relevance of each feature in the final calculation. In cases where a feature lacks essential data, the method intelligently leverages the priority coefficient values of other features based on predefined conditions, ensuring robust decision-making.

The rest of the paper is structured as follows. Section II reviews the literature on malicious URL detection methods. Section III presents a comprehensive review of the URL feature classification. Section IV introduces the conceptualization and comprehensive framework for detecting malicious URLs. This section includes three phases: identification, feature classification and extraction, and feature evaluation and detection. Section V covers performance evaluation, including dataset description, evaluation metrics, and experimental results. Finally, Section VI concludes the paper and outlines potential future research directions.

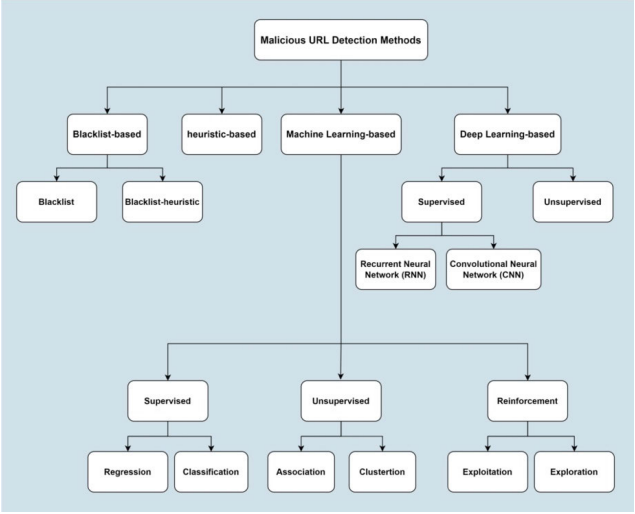

FIGURE 1. Malicious URL detection methods.

II. RELATED WORKS

Numerous scientific studies have explored various techniques for detecting malicious URLs, typically categorized into four types: blacklist-based, heuristic-based, machine learning-based, and deep learning-based methods [13], [14]. These methods are outlined in Figure. 1, and we will now briefly review the literature on malicious URL detection methods that are closely related to our work.

A. BLACKLIST METHOD

In previous years, the blacklist method stood out as the predominant method for detecting malicious URLs. This method relies on databases containing URLs previously identified as potentially harmful, such as phishing or malware, accumulated over time [15]. Whenever a new URL is requested, a database search is performed [4], [16]. If the URL is present in the blacklist database, it is deemed harmful, triggering a warning. Conversely, if there is no match in the database, the URL is considered benign [17].

Some of the researchers used this method for detecting malicious URLs. Reference [18] worked on an efficient approach for generating blacklist URLs that takes advantage of the current harmful URL search structure neighborhoods to discover and validate unknown malicious websites in order to grow the URL blacklist database. Reference [19] provided a strategy for detecting phishing based on the monitoring of URL alterations according to the blacklist method. They presented combinations of known phishing sites and an approximate matching method. Reference [20] suggested a systematic strategy for generating blacklist URLs that utilizes a search engine to locate URLs in the neighborhood of a malicious URL.

Blacklist heuristic approaches are a subset of blacklist techniques, with the goal of creating a blacklist of signatures. After identifying common attacks and associating them with a signature, the detection systems may examine URLs for these signatures and raise an alert if any unusual behaviour is detected [21]. These approaches outperform blacklisting in terms of generality since they can detect threats in fresh URLs as well [8]. In the other research [22], authors suggested a blacklist heuristic detection method according to the reputations of the Internet Protocol (IP) address block and registrars used by attackers. The proposal would create a combination of IP address blocks and registrars with a low reputation, which is widely used by attackers.

Nevertheless, such solutions can only be created for a limited number of typical threats, and obfuscation strategies may easily be passed through them [23]. Due to the inability of blacklist methods to detect newly created malicious websites, some researchers have proposed heuristic methods, which are discussed in the following section.

B. HEURISTIC METHOD

Heuristic (rule matching) detection methods depend on statistical similarities between phishing and malware URLs. These methods utilize extracted features, collect crucial information about a website, and incorporate expert knowledge to identify malicious URLs [24]. Detection of malicious URLs is executed based on features derived from numerous observations of known harmful webpages, which are then generalized into a specific set of heuristic rules [16].

Blacklist heuristics and heuristic methods differ in their approaches to detecting malicious URLs. Blacklist heuristic approaches typically rely on predefined lists of known malicious URLs or patterns to detect and block potential threats. These lists are compiled based on historical data or known signatures of malicious activity, such as phishing or malware distribution. On the other hand, heuristic methods employ a more dynamic and adaptive approach to detecting malicious URLs. Instead of relying solely on predefined lists, heuristic methods analyze the various features and characteristics of URLs to assess their likelihood of being malicious [23].

The researchers employed heuristics to condense an extensive array of online sites into a more manageable set of suspicious web pages. Reference [25] introduced the Carnegie Mellon Anti-phishing and Network Analysis Tool (CANTINA), a widely used heuristic-based phishing detection method. CANTINA utilizes the Term Frequency/Inverse Document Frequency (TF-IDF) algorithm to extract various components from a web page and determine whether the URL is benign or malicious based on information obtained from the search, along with additional heuristic features. Reference [26] proposed a heuristic malicious URL detection technique by using scraping and web crawling methods. PhishNet presented in [19], which proposed to detect phishing URLs based on a combination of five heuristic rules and a matching algorithm. Reference [27] described a heuristic technique for detecting phishing websites based on a set of 12 rules by evaluating the static features employed and observing the behaviour of the current phishing URLs.

Phishing Detection using Multi-filter Approach (PhiDMA) suggests phishing detection using a multi-filter approach based on multilayers, which is a heuristic-based method [28].

This approach consists of the following five layers: auto-upgrade whitelist, URL features, lexical signature, string matching layer, and accessibility score comparison. Each of the layer's functions as an especial filter to identify malicious activity along a certain metric. The experimental results demonstrate the method's efficacy in identifying malicious URLs with 92.72% accuracy in a dataset of 1662 malicious and benign URLs. Reference [29] suggested a heuristic technique for identifying harmful URLs by analyzing websites and discovering any direct and indirect links associated with the websites.

While the heuristic approach addresses the limitations of the blacklist method and eliminates the need for a large database of malicious URLs, most of the suggested methods are still incapable of achieving real-time detection of malicious websites. This is primarily because the rules are formulated based on the behavior of pre-existing malicious URLs. Furthermore, the analysis of harmful websites requires considerable subjective expertise. Currently, the behavior of phishing and malware websites is highly diverse, rendering rule-based techniques ineffective [30]. Recognizing the inadequacy of blacklist and heuristic-based methods in predicting new malicious URLs, researchers have turned to machine learning techniques in the past decade, achieving notable advancements.

C. MACHINE LEARNING METHOD

With the growing popularity of big data, ML techniques have emerged as the most widely employed and effective means of detecting malicious URLs. These techniques offer both generalizability and robustness, making them highly resistant to real-world attacks [16]. A huge number of machine learning methods have been utilized to learn the generalized and develop detection methods according to the existing URLs [4]. Furthermore, the identified sites are utilized to train and develop the detection algorithm method, while the unknown sites are classified via the already trained algorithm method. Following these procedures, the method will be equipped with special dynamic detection capabilities [24]. This method extracts features represented in a URL by using Application Programming Interfaces (APIs) and other components of a website, and then trains a prediction method on a dataset that includes both malicious and benign URLs [23].

Reference [31] proposed a machine learning approach for malicious URL detection by combining linear and non-linear space transformation approaches. The authors employed Singular Value Decomposition (SVD), Distance Metric Learning - Nyström techniques (DML-NYS) algorithms by using a dataset of 331,622 with 62 classes for training. These methods are effective and can significantly improve the performance of certain classifiers in identifying malicious URLs. Reference [32] suggested a novel and robust method for identifying harmful URLs. The features are gathered and utilized to evaluate the classification using the RF and Gradient Boosting Decision Tree (GBDT) machine learning methods. The findings demonstrate the suggested method's performance by reaching a superior accuracy of 96.4%. Reference [1] used a variety of URL-based feature classes to identify phishing, spam, and malware URLs. The performance of the proposed method was assessed using machine learning algorithms such as Extreme Gradient Boosting (XGBoost), Adaptive Boosting (AdaBoost), Light Gradient Boosting (LightGBM) and Categorical Boosting (CatBoost) with an accuracy rate above 95%. Reference [8] suggests a machine learning-based method for identifying malicious URLs. The authors gathered lexical, content-based, and host-based features from the website and trained it using machine learning approaches such as SVM and RF. A dataset of 470,000 URLs was used for training, and it had an accuracy of 92.174%.

The research [33], combined the usage of URL lexical features, payload size, and Python provided parameters to detect malicious URLs using machine learning algorithms. A SVM was employed in conjunction with a polynomial kernel and logistic regression to achieve a 98% level of accuracy. Reference [34] suggested a machine learning approach for developing and evaluating real-time malware detection for URLs. A combination of Naive Bayes (NB) and Logistic Regression is used to identify malicious URLs with an above 90% level of accuracy and report them to the site administrators. In the other study [4], the authors proposed a methodology to detect malicious URLs and the types of attacks based on multiclass classification. They utilized classification algorithms like One-Vs-All (OVA) L1-reg L2-loss SVM (OVA SVM), One-Vs-One (OVO) L1-reg L2-loss SVM (OVO SVM), and Multi-Class Confidence Weighted Learning (MC-CW). The dataset, which contains 49935 URLs, was collected from the Alexa top sites, PhishTank, MalwareDomainList, and jwSpamSpy. From a total of 117 features, they extracted 65 lexical, 34 content-based, and 18 host-based attributes. They have achieved the highest accuracy of 99.86% in the detection of malicious URLs using a binary setting and an average accuracy of 98.44% using CW.

Seize Malicious URL [30], proposed a novel approach to identifying harmful websites by leveraging a diverse set of machines learning techniques, including RF, Decision Trees (DT), k-Nearest Neighbors (k-NN), NB, and SVM. This approach involves the prediction of website classes, followed by the application of a threshold to refine the results. It then amalgamates the decisions based on associated class probabilities and utilizes the label with the highest-class probability to arrive at a comprehensive determination regarding unlabeled websites. This approach underwent rigorous testing across two distinct databases, an accuracy rate of 99.91% for the first dataset comprising 165,362 URLs and an impressive accuracy of 97.98% for the second dataset, which included 420,464 URLs.

Machine learning methods are increasingly popular for enhancing previous approaches to detecting malicious URLs. Although these techniques have proven effective, their widespread implementation in industry and in real time is yet to come. Certain limitations exist within machine

learning techniques for identifying malicious URLs. Their main weakness lies in their complete dependence on data. These methods often struggle due to the challenge of creating a comprehensive and generalized dataset. Malicious URL patterns and tactics continually change, making it difficult to keep datasets up to date. Additionally, assembling a dataset that accurately represents the full spectrum of malicious URLs across various contexts and languages can be a complex and resource-intensive task [35], [36]. Another significant weakness is the presence of bias in the training dataset. If the training data is biased towards certain types of malicious URLs or if it lacks diversity, the model may not perform well in detecting less common or evolving threats. Studies [37], [38] have demonstrated that methods constructed using a high accuracy machine learning method using a training dataset (such as Kaggle with over 400,000 websites) may not be effective when applied to another dataset.

A further limitation involves selecting and extracting relevant features from URL data. Determining which features are most informative for distinguishing between malicious and benign URLs can be a complex task, especially as attackers continuously modify their tactics. Inadequate feature selection and extraction can lead to suboptimal model performance, as important information may be overlooked or irrelevant features may introduce noise into the model. The last fundamental limitation lies in the delicate trade-off between overfitting and underfitting. Overfitting occurs when models become excessively specialized in recognizing known attack patterns present in the training data. While these models may accurately detect known threats, they often struggle with novel attack methods, failing to generalize effectively. On the other hand, underfitting is equally problematic, resulting in models that inadequately capture the complexities of the data, leading to poor detection capabilities. This highlights the importance of addressing data dependency issues and exploring ways to improve the generalizability and adaptability of malicious URL detection methods.

D. DEEP LEARNING METHOD

Deep learning is a subset of a larger family of machine learning approaches based on artificial neural networks and representation learning. Deep learning is based on statistics and predictive modelling, which is highly essential for data scientists and speeds up and simplifies the process of gathering, analyzing, and interpreting massive volumes of data [23].

In particular, it seeks to learn relevant features directly from a dataset and perform classification and clustering utilizing these features. This approach has recently demonstrated remarkable success in identifying malicious URLs, offering advancements in feature engineering and model development without necessitating domain-specific expertise, as evidenced by prior studies [6], [17], [39]. Deep learning eliminates the feature selection procedure of machine learning methods, which increases system performance and prevents the loss caused by the selection of incompatible features [24]. It does not require tedious feature extraction, which leads to a training method with minimal effort and results in an appropriate pattern for detecting malicious URLs [40].

A deep learning network is utilized to systematically extract features from a dataset of URLs, which is then used to identify harmful URLs. The static features (mostly lexical and HTML features) have been transformed into a matrix, and then the matrix is fed into a deep learning network. In the final step, the trained network can then return a float outcome (between 0 and 1) indicating whether the input URL is malicious or benign [41].

The authors in [42] focused on a deep learning neural network detection approach for detecting harmful URLs. The researchers conducted two separate datasets that utilized Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) methods, and a combination of them. The empirical findings were encouraging, with a precision of more than 96%. The research in [43] proposed a deep learning-based solution for identifying malicious URLs. It employs a convolutional Gated-Recurrent-Unit (GRU) neural network based on characters as text classification parameters, yielding an accuracy rate of more than 99.6% and making it ideal for high precision classification purposes.

In the other research, authors suggested URLNet, CNN-based deep neural network, to learn a nonlinear URL embedding for malicious URL detection directly from the URL [17]. The authors apply the method to both characters and words in the URL string to learn the URL embedding in a jointly optimized framework. This approach allows the method to capture several types of semantic information that were not possible with the existing methods. The bag-of-words approach was presented, which is a form of lexical feature, and jointly optimized the network using character and word CNNs. Furthermore, the research presented sophisticated word-embedding strategies that are especially suited to dealing with rare words, which are mostly used on malicious websites. The accuracy rate of 98.58% in a dataset of 10,000,000 URLs was achieved by the URL full method.

The paper [9] presented a Factorization Machine (FM), a form of deep learning algorithm for identifying malicious URLs. This method discovers the latent relationship between lexical characteristics. To minimize the ambiguity of URL tokens, position embedding is implemented for token vectorization. It means a Temporal Convolution Network (TCN) is employed to learn the long-distance dependence between URL characters.

Precise Phishing Detection with Recurrent Convolutional Neural Networks (PDRCNN) method presented in [24], suggests a rapid approach for detecting malicious URLs that relies solely on lexical features. It turns the URL's data into a two-dimensional tensor and feeds the tensor into a newly built neural network for classification. First, extract features from the built tensor and assign all string information to each character in the URL using a bidirectional LSTM network.

Second, employ CNN to automatically determine which characters are critical for detecting malicious URLs, extract the URL's major elements, and compress the collected classes into a fixed-length vector space. The PDRCNN achieves a detection accuracy of 97% on a dataset with 245,385 valid URLs.

Deep learning approaches have made great progress in detecting malicious URLs over the last few years. Many machine learning problems have been overcome, but there are still a number of major issues remaining. Massive volumes of URLs needed to be used for training to create a suitable detection method with acceptable levels of accuracy for deep learning [44], [45]. This problem becomes much worse when new URLs are available and the method need to retrained [46]. Furthermore, interpretability of method does not disclose the details and specifics of the method's prediction and feature selection, which often behave like black boxes. Interpretability can lead to some drawbacks, such as vulnerability to potential novel attacks. due to a lack of knowledge of rules developed by machines, which prevents upgrading and optimizing the rules by the developers [47]. Moreover, the detection method's reliability and level of accuracy are entirely dependent on the quality of the dataset [9], [48]. Lastly, an issue of note is the feature selection contradiction, with the majority of research still involving manual classification of features.

Table 1 concludes the limitations of current malicious URL detection methods and Table 2 illustrates a summary of the related works.

TABLE 1. The limitations of malicious URLs detection methods.

Method	Limitations
Blacklist	• Unable to detect new generated URLs • Unable to detect obfuscated URLs
Heuristic	• Limited generalization • Difficulty in adapting to evolving threats • Required comprehensive details to create rules
Machine Learning	• Require accurate labelled dataset for supervised ML • lack of generalized dataset • Performance depends on the quality of dataset • High retaining cost (computationally expensive) • Bias in the training dataset • Overfitting and underfitting of model • Difficulty of accurate feature selection and extraction
Deep Learning	• Require massive training time and costs • Interpretability of methods • Performance depends on the quality of dataset • Difficulty of precise feature selection/extraction

III. URL FEATURE CLASSIFICATION

Features play a significant role in detecting malicious URLs. However, each feature is not equally important and thus there are specific features that must be selected to have a successful and effective detection method. Each feature includes a number of classes which try to find out the characteristics of a URL to determine whether it is malicious or benign. These classes extract critical information from URLs and then assess them by comparing the specified values with the outcome values from the URL [2].

URL features analysis can be broadly be classified as static and dynamic [49], [50]. This classification is based on the types of information that is extracted from webpages. The static approach aims to detect malicious URLs by acquiring statistical analysis of the URL. This information may be

collected through invoking the APIs, downloading the entire webpage, or parsing and analyzing various components of the URL string [51]. The dynamic approach attempts to discover harmful URLs by evaluating runtime behaviors and monitoring various back-end components of websites with the use of analytic tools to detect suspicious activity [23]. The dynamic analysis approach only focuses on the content-based features where the scripting languages like HTML and JavaScript are located.

Researchers have proposed several types of features that can be used to extract valuable information to detect malicious URLs. The information gathered from the URL's features can be separated into various categories [52]. These features can provide information about a URL such as ranking, geolocation, traffic, WHOISE, HTML, JavaScript, URL string, bag of words, blacklist database, and certificates. According to [8] and [23], the features are generally categorized into four groups: blacklists, lexical, host-based, and content-based. Also, some research such as [4] added redirection (short URL feature) to this category. The following subsections provide a review of the relevant literature for each category.

A. BLACKLIST FEATURE

The blacklist feature is the most popular strategy for detecting malicious URLs in the past few years and is used by many researchers, IT security and antivirus applications [35]. Blacklists are just a database of URLs that have already been discovered to be harmful and have built up over the years [4], [16]. This technique is incredibly quick and simple to perform because of its low query overhead and very low percentage of false positives [17]. Blacklists are typically regarded as the first line of defense in protecting users against harmful URLs [53].

In the past decade, a number of significant studies have focused on blacklist methods to detect malicious URLs. The authors in [18] worked on an efficient approach for generating blacklist URLs that takes advantage of the current harmful URL search structure neighborhoods to discover and validate unknown malicious websites in order to grow the URL blacklist database. Reference [19] provides a strategy for detecting phishing based on the monitoring of URL alterations. They presented combinations of known phishing sites and an approximate matching method. The researchers suggest a systematic strategy for generating blacklist URLs that utilizes a search engine to locate URLs in the neighborhood of a malicious URL [20].

Heuristic blacklist approaches are a subset of blacklist techniques, with the goal of creating a blacklist of signatures.

TABLE 2. Summary of the most related and state-of-the-art research works.

Research	Detection Method	Features	Dataset	Accuracy
Li method [31]	Machine learning (SVD, DML-NYS)	Blacklist, lexical, host-based and content-based (Total 62 classes)	331,622 URLs	93.05%
PhiDMA [28]	Heuristic-based	Whitelist, URL features, lexical signature, string matching, accessibility score comparison	1,662 URLs	92.72%
[8]	Machine Learning (SVM, RF)	Lexical, host-based, and content-based (Total 54 classes)	470,000 URLs	92.174%
PDRCNN [24]	Deep Learning (CNN)	Lexical (Total 9 classes)	245,385 URLs	95.60%
[4]	Machine Learning (OVA SVM, OVO SVM, MC-CW)	lexical, host-based, redirection, and content-based (Total 117 static and dynamic classes)	49,935 URLs	98.44% accuracy in a multi-class and 99.86% in a binary scenario
Seize Malicious URL [30]	Machine Learning (RF, DT, k-NN, NB, SVM)	Lexical (token-based by counting the words)	Dataset 1 (165,362) Dataset 2 (420,464)	99.91% accuracy in the first and 97.98% in the second dataset.
URLNet [17]	Deep Learning (CNN)	Lexical (Bags of Words (BoW))	10,000,000 URLs	98.58%

After identifying common attacks and associating them with a signature, the detection systems may examine URLs for these signatures and raise an alert if any unusual behaviour is detected [8].

According to [54], there are over 40 cybersecurity organizations that provide services for determining the URL status. The majority are well-known and provide antivirus and IT security services; their techniques include evaluating historical data such as blacklists and whitelists, while others scan URLs in real time based on URL content and identify threats. The most well-known and extensively utilized tools are Google Safe Browsing, PhishTank [55], VirusTotal [56], Kaspersky, Norton Safe Web, McAfee, Trend Micro, and G Data.

B. LEXICAL FEATURE

The lexical (URL-based) feature is the textual properties of a URL and extracts various details from the URL string [9]. It is incredibly handy for acquiring additional information from the URL without digging too deep. The lexical feature has become one of the most popular sources of features for detecting malicious URLs, due to its lightweight computation, safety, speed, independence from other applications and high classification accuracy [57].

The lexical feature breaks the URL into multiple components, such as hostname, path, protocol, query, and Top-Level Domain (TLD), and each of them is inspected individually for analysis. Figure. 2 shows the components of a URL.

A lexical feature may be divided into a variety of classes, but based on the information retrieved from the components, it is generally classified into three categories of dictionary (binary), ratio, and numerical data [4]. The dictionary refers to searching for unusual terms and materials inside the URL name, such as illegitimate words, malicious TLDs, IPs, and ports. The ratio refers to the ratio of different existing types of digits or letters in a particular string. The numerical refers to counting the length and special symbols in the URL.

This feature is the most preferred for malicious URL detection due to its independence from other sources, and researchers have identified a range of classes. Reference [58] presents 76 lexical feature classes to determine if a URL

FIGURE 2. The component of a URL.

is benign or malicious. These classes are divided into four categories: length, counting feature, binary feature, and ratio feature. In the other research [59], authors mentions that after analyzing various URLs, they discovered 23 potential lexical classes to use to detect malicious and benign URLs. Also, [60] presents 106 lexical features for the URL classification. There are 41 word-based classes, 36 count-based classes that are the frequency of each alphanumeric character, and the remaining 29 features include special character count, URL entropy, domain, host, path, parameters, and query. This research [58] introduces new categories for lexical features classes and totally implemented 87 classes. It recommends N-grams, length, ratio, binary, counting, and pattern classes.

Multiple classifications were applied by [1] to identify harmful URLs. Lexical features such as Kullback–Leibler (KL) divergence and bag of words segmentation are gathered among the 17 lexical features. Reference [33] mentioned a machine learning technique to detect malicious URLs by combining URL lexical features, payload size, and Python supplied options. The system aims to train a detection model to identify both new and known malicious web links, prioritizing high detection rates and low false positives. The authors utilized SVM with a polynomial kernel and logistic regression and achieved an accuracy of 98%. Reference [34] develops a machine learning approach for real-time malware detection for URLs using lexical features. Researchers presents a deep learning algorithm for identifying malicious URLs [9]. This method discovers the latent relationship between lexical characteristics using FM. To minimize the ambiguity of URL tokens, position embedding is implemented for token vectorization. It means a TCN is employed to learn the long-distance dependence between URL characters.

This feature is separated into traditional and advanced lexical features according to the information extracted from the URL string [17], [23]. Traditional lexical features focus mostly on extracting common statistics about a URL string, such as the entire length of the website and the lengths of its individual components, as well as the quantity of special characters (e.g., "/", ".", "?", "=", etc.) [62]. A "bag of words" is a traditional type of lexical feature that is used to detect malicious URLs [63], [64]. This approach is implemented for detecting malicious URLs to showed how to use sophisticated word-embedding strategies that are good at dealing with rare words, which are mostly used on malicious websites [17] and [58].

The advanced lexical features are categorized into five feature types [23], which are: features associated with URLs (such as keywords, length, etc.); characteristics of the domain (such as length of domain name and presence of IP address); features associated with directories (such as directory length and number of subdirectory tokens); features associated with filenames (such as filename length and number of delimiters); and features associated with arguments (such as length of the argument and number of variables). In addition, some researchers employed sophisticated advanced lexical features, such as PhishDef [65], Kolmogorov Complexity [66], and PhishScore [67].

C. HOST-BASED FEATURE

The host-based (web-based) feature stands as a key factor in identifying malicious URLs, offering a preliminary insight into the attributes of the website's host [68]. This feature provides insights into the location of hosting for malicious sites, their ownership, and the way they are controlled. Furthermore, this feature facilitates the retrieval of webpage rankings and overall visit counts. The rationale behind leveraging this feature lies in the potential for harmful websites to be hosted in less reputable server facilities, on atypical machines not traditionally used as web hosts, or by using untrustworthy domain registrars [32].

This feature thoroughly examines a host's background activities in order to determine whether the website is harmful or safe. The malicious URLs are mostly young and frequently use new hosts and domain names in order to pass through detection methods, which this feature can help to detect the age and usage of a website [69]. Additionally, after detection and adding a URL to blacklist databases, the number of visitors and rank drop dramatically, which is detectable by traffic and web rank. Also, the hotbeds of malicious activity could be concentrated in specific regions using IP geolocation [70].

This feature is very popular and informatic, which is used by several researchers. An effort to identify malicious URLs [8] used machine learning methods by utilizing the Alexa API to extract several classes of host-based features. Reference [69] presents a unique deep reinforcement learning-based approach for detecting harmful websites by assessing the host-based features, and the WHOIS database was used for extracting domain classes.

This feature provides a variety of information about the host and its categories into three main features of WHOIS, Rank, and IP geolocation [31], [58]. The WHOIS feature provides information about the webpage such as: the creation date, the update date, and the expiration date, the owner's name, the address, and many more details [70]. The website rank feature shows the number of viewers, traffic, and popularity of a website, and has several components, such as global rank, country rank, category rank, total visits, average visit duration, and others [71]. The IP Geolocation feature provide the location based on an IP address that corresponds to the URL, such as the country, state, or city [72]. Numerous websites offer data for the host-based feature, with some of the most famous ones being Alexa, WHOIS Lookup, WhoXy, Similarweb, OpenPageRank, IPgeolocation.io, and IPWhoise.io.

D. CONTENT-BASED FEATURE

Content-based (HTML) feature provides large amounts of information by downloading the whole webpage to analyze content and layout of a page. This feature is very "heavy-weight" in comparison to other features on structural information derived from parsing the HTML code [8]. In a case that other features are fail to detect a malicious URL, a more thorough analysis of the content-based feature may assist in the detection of threats [23].

Numerous researchers have invested significant effort on identifying harmful websites using a content-based feature. A comprehensive real-time phishing detection method is proposed using HTML features [73]. The authors proposed twelve host-based feature classes, six of which are new classes invented by the authors and help to improve the detection method's accuracy. Reference [74] presents WebMon were extracted 51 natives' keywords from JavaScript functions and HTML DOMs of content-based feature in order to identify harmful URLs.

The content-based feature of a webpage is primarily drawn from its HTML content and is divided into three categories: HTML, JavaScript, and Certificate Feature. The textual nature of a website's HTML content enables its utilization for lexical, statistical, and specific functional purposes [75]. In some instances, malicious code is encoded within HTML, which this feature can aid in identifying potentially harmful activities [76], [77]. JavaScript feature code is a rich source for detecting malware activity, where attackers try to propagate exploits over the Internet [78], [79]. JS obfuscation is a transformation aimed at generating JS code that is obscure to the human eye and undetectable to an online security tool [80]. The certificate feature assesses whether the digital certificate associated with the URL is issued by a trusted Certificate Authority (CA). This is crucial because URLs registered under such certificates enable the establishment of secure connections that authenticate and validate websites [81].

TABLE 3. URL feature classification.

Features	Category	Criteria					
		Collection Difficulty	External Dependency	Collect Time	Processing Time	Feature Sizes	Risk
Blacklist	Blacklist	Mod	Yes	Mod	Low	Low	Low
Lexical	Traditional	Easy	No	Low	Low	Very High	Low
	Advanced	Easy	No	Low	High	Low	Low
Host	WHOIS	Mod	Yes	High	Low	Mod	Low
	Rank and traffic	Mod	Yes	High	Low	Mod	Low
	Geographic	Mod	Yes	High	Low	Mod	Low
Content	HTML	Easy	No	Mod	High	High	High
	JavaScript	Easy	No	Mod	Mod	Mod	High
	Certificate	Easy	No	Low	Mod	Mod	Mod
Redirect	Short URL	Mod	No	No	Mod	No	High

E. URL REDIRECTION FEATURE

In most of the research redirection are sperate from the feature classification according to the fact that it does not extract information and only redirect to the original website. However, in this paper and some others such as [4], this feature is containing feature classification because it plays critical role on the other features to extract the right details about a website.

The URL redirection (shortened URL) feature involves detecting a shortening or redirection URL and transmitting it to the original website. This method is a type of obfuscating method that tricks users by displaying malicious URLs as legitimate ones and is widely used in phishing and malware attacks [70]. The URL shortening service has become popular and widespread in recent years, which allows the original URL to be represented by a shorter string and shared on the World Wide Web [82]. The most famous URL shortening service providers are bit.ly, rebrand.ly, ow.ly, tinyurl.com, tiny.cc, and cut.ly.

There are some studies that examined the number of short URLs (redirection) used in the URL datasets. Reference [83] observed that URL shortening services are an effective method for hiding harmful URLs. They were able to do so by looking at current detection techniques employed by well-known shortening providers. They examined 622 URL shortening providers and gathered 24,953,881 short URLs for over two years. Surprisingly, they discovered the use of short URLs for drive-by download attacks has risen, and only a limited range of visitors encountered harmful short URLs. In the other research, [84] examines the blacklisted Bitly URLs that had been shortened. The authors discovered various characteristics, including the time between the shortening and the formation of the domain, as well as the time between the shortening and the use of the link. The frequency of redirections has also been proven to be a characteristic of nested shortened URLs, with at least 1 redirection in 80% of phishing attacks.

F. OVERVIEW OF FEATURE CLASSIFICATION

This section provides an overview of the features for detecting malicious URLs. First, evaluate the criteria that assist in the collection and extraction of the features. Then, demonstrate the popularity of features that have been presented in multiple studies of malicious URL detection.

Table 3 illustrates a subjective evaluation of different features used in literature. The URL features are divided into several categories and then evaluated by multiple criteria such as collection difficulty, external dependency, collection time, processing time, feature size, and risk [23].

The term ''collection difficulty'' refers to the amount of technical effort necessary to obtain detailed information about the features. The blacklist and host-based features require additional dependencies and hence have a greater collection overhead than the other features, which are retrieved straight from the URL. The term ''risks'' refers to the various possible threats that might arise to a system. Content-based and redirection features have the highest threat, as harmful malware may be downloaded deliberately while attempting to obtain these features, whereas other features are not affected by similar issues. The term ''collect time'' refers to the amount of time needed to gather all the relevant information. Obtaining host-based and blacklist features is time-consuming due to calling APIs. The collection time of content-based features might be affected by the need to download the entire website. The lexical feature is highly fast because it is essentially derived directly from the URL string. The term ''processing'' refers to the comparison of the values that have already been collected. Most of the features are very fast. The term ''feature size'' refers to the size of the feature that is used to do comparison, search, and calculation. The lexical feature may be extremely high due to the usage of the bag of word features. Also, depending on the classes, host-based and content-based features may have a large size.

These features have been extensively employed for a long time to detect malicious URLs. The list of current research utilizing these features for the purpose of detecting malicious URLs is presented in Table 4.

There are some challenging issues in the feature classification for detecting malicious URLs. The first issue is that the detection methods for malicious URLs do not prioritize the classes and features according to their level of importance. Each class has a specific level of importance and confronting

TABLE 4. Features presentenced in recent research.

Features	Presentative Reference
Blacklist	[31, 70]
Lexical	[1, 4, 8-10, 17, 21, 24, 28, 30-34, 38, 44, 47, 52, 61, 68, 70, 73, 85-94]
Host	[4, 8, 10, 31-33, 44, 52, 68, 70, 74, 85, 87-90, 92-95]
Content	[4, 31-33, 70, 73, 74, 87, 88, 90-92, 94]
Redirect	[4, 70, 83, 84, 87, 88, 92]

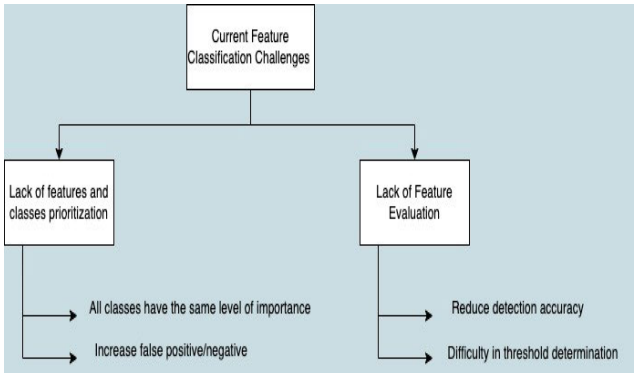

FIGURE 3. Feature classification challenges.

them ought to yield various outcomes. The second issue is the detection of malicious URLs when features fail to respond. This can happen if APIs do not respond (blacklist and host-based features) or if a server goes offline (content-based feature). Due to the fact that some of these features have external dependencies, they may not always provide information based on API-related issues. The other issue is difficulty in threshold determination. When some features do not return a value for the final calculation, determining an appropriate threshold value for classifying URLs as benign or malicious can become challenging. Therefore, the method should be able to detect malicious websites with a high level of accuracy when features do not respond. Figure. 3 illustrates the feature classification challenges.

IV. THE PROPOSED FRAMEWORK

The framework we proposed for detecting malicious URL is depicted in Figure. 4. This framework is composed of three phases. First, the framework starts with the identification phase, which includes the procedure for detecting obfuscated URLs. The second phase presents the predefined static features classification method by allocating the priority coefficients to the selected classes. The third phase presents the feature evaluation method, which determines if all the features deliver value for the final calculation and allocates feature priority coefficient. This phase also compares the final result to a threshold value in order to determine whether the URL is benign or malicious.

In the forthcoming sections, we will delve into the explanation of each phase within the framework. Also, we will introduce the metrics employed to assess the effectiveness of the framework.

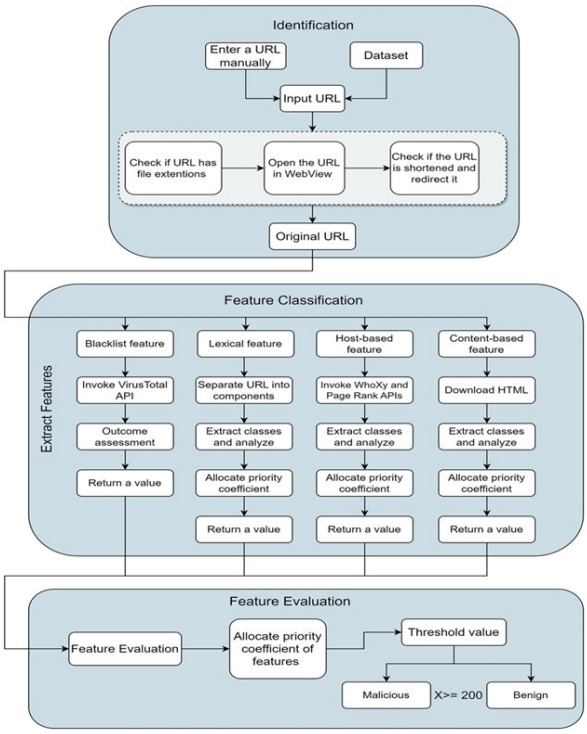

FIGURE 4. The proposed framework in detecting malicious URL.

A. IDENTIFICATION PHASE

The identification phase processes the input URL and applies an algorithm to detect redirections, focusing on dynamic features by examining the redirection chain and opening the website in an isolated environment to monitor its behavior. In this framework, five obfuscation strategies were detected in the experiment. These strategies are URL obfuscation with another domain, URL obfuscation with keywords, lengthy domains, URL obfuscation with IP address, and URL obfuscation with URL shortened [96], [97]. URL shorteners are detected and redirected in this phase, and the remaining methods will be detected in the feature classification phase. Short URLs have emerged as the most efficient obfuscation approach for duping users by displaying harmful URLs as legitimate and are extensively used in phishing and malware attacks. The proposed identification pseudocode is illustrated in Figure 5, this phase ensures that website behavior is monitored in a secure environment to enhance the detection accuracy of malicious activity.

The following are the steps taken by the proposed identification phase:

- **Step 1 Input URL:** The user may manually enter the URLs, or they can be read from the CSV file of a dataset.
- **Step 2 Analyze the URL:** Verify the URL in the list of data formats and file extensions (E) to determine their inclusion. If they are included, display the original URL (O) and ends (send the URL to the next phase). Also, it checks in an isolated environment for any malicious content. If not, open the website in the first WebView

Proposed Pseudocode for Identification

Input: The URL is entered manually or through a dataset
Output: Original URL (redirected URL)
Variables

U	URL
VW1	WebView number 1 (The Web browser open the URLs in Android)
VW2	WebView number2
E	Data formats and file extensions List
O	Original (redirected) URL
C	Count number of redirects

Begin

```
1: U ← Insert URL          // The URL is entered manually or through a dataset
2: If U = E                //verify the list of extension URLs to determine if it contains TLD.
3: VM1 ← U
4: Else
5: Return U                // If the URL contains data formats and file extensions, it analyzes in an isolated environment and then forwarded to the next phase.
6: End if
7: While VM1=O             // While WebView redirect to the original URL using redirection invoke function (It may take more than 10 times)
8:VM2←VM1
9: C← 0
10:C=C+1                   // Count number of redirect (use for lexical feature in next phase)
11: Else
12: Return U
13: Return VM2             // Display the redirected URL and send it to next phase (lexical feature)
14: End while
```

End

FIGURE 5. Pseudocode of identification phase.

(VW1). This occurs when a website includes an executive or data file (drive-by-download attack), and while these types of links are not being redirected, they will be sent to the next phase without checking for redirection. It results in an increase in the security of the device.

- **Step 3 Check the originality of the URL and open it in WebViews**: URLs are redirected using the Should Override URL Loading technique [98]. In this stage, two Android WebViews are used. By using the redirection function, the first WebView (VW1) determines whether the inserted URL is original or redirected. While this function is true, the webpage is redirected. Also, it counts the number of redirections that will be used later in the lexical features. The second WebView (VW2) will redirect to the original URL if it encounters a shortened URL. This method loads the URL in WebViews and sends it back to the original URL, repeating the process until the original URL is displayed. The VW2 displays the original website in the final stage.
- **Step 4 Transmit the original URL to feature classification phase:** The original URL appearing in VW2 will be transmitted to the feature classification phase, where each of them will be evaluated separately. In addition, the value of the first WebView will be sent to the class's lexical feature that counts redirections.

B. FEATURE CLASSIFICATION PHASE

The proposed framework for detecting malicious URLs relies on a predefined static feature classification method. This framework employed 42 classes that were selected according to the importance of extracting critical information from the website. The value of the classes is assigned by observing and analyzing several harmful and safe URL datasets. These classes are obtained by extracting and analyzing different URLs components, utilizing several APIs, and downloading the entire website to provide valuable information. The list of these classes is depicted in Table 5, along with their characteristics, reasons for selection, assigned values, and types.

Each feature includes several classes that extract critical details about a URL. The main reasons to separate the definitions of feature and class in this research are for calculation and final evaluation. The feature value is calculated by summing all the classes' values, and then this overall value will be sent to feature evaluation phase. In that phase, only the value of each feature is evaluated, not each class separately. Also, the other reason for this separation is allocating priority coefficients to the features. The priority coefficient is assigned to each feature according to its level of importance to boost the detection accuracy of malicious URLs. The priority coefficient assigns greater weight to the features, which increases the framework's detection accuracy.

Furthermore, Table 6 illustrates the selected classes for which coefficients are implemented, detailing their circumstances and reasons for selection. Additionally, the range of coefficients, spanning from 1 to 1.5, encapsulates a spectrum of importance levels crucial for effectively discerning the likelihood of maliciousness on a website. At its lower end, a coefficient of 1 signifies minimal significance, implying that the associated website does not show malicious activity and could be benign. Conversely, a coefficient of 1.5 denotes utmost importance, indicating that the website is highly suspect and demands thorough investigation.

This range is meticulously selected through iterative trial-and-error processes, where the efficacy of different coefficient values is rigorously tested and refined. By leveraging this range, the system can dynamically assign varying levels of weight to different websites based on their propensity for malicious behaviour. This strategic allocation of coefficient levels enhances the framework's ability to detect potential risks effectively.

The application of these coefficient values to classes is contingent upon specific circumstances and is not universally applicable. For example, when a URL is detected to include an IP address rather than a hostname, a coefficient value of 1.3 is assigned to elevate its priority relative to others in comparison to the threshold value.

In the subsequent section, the pseudocodesof blacklist, lexical, host-based, and content-based features are described. Also, it will present the procedure of assigning the range of predefined values, the methods for extracting the features, the process of allocating the priority coefficient to selected classes, and the calculation of the features.

Blacklist Feature Classification Pseudocode

Input: The URL is read from identification phase
Output: The total number of blacklist databases in which the URL has been detected as malicious (X).
Variable:
_U URL
_VT VirusTotal
_SID ScanID
_R Report
_X The number of websites that are detected the proposed URL
Begin
1: U← Read URL // Read URL that sent from identification phase
2: VT← U // Invoke VirusTotal API
3: Return SID (U) // A string and contains information include the number of websites that identified the proposed URL
4: R (SID) // Get a report from ScanID
5: Return X // This method reports the number of websites that are detected the URL, and the result is equal to X.
6: If X>5 // If 6 or more blacklists databases detected, it returns 5
7: Return 5
8: Else
9: If 0<X<6 // If 1 to 5 websites are detected, it returns 3
10: Return 3
11: Else
12: If X=0 // if no websites are detected, it returns 0
13: Return 0
14: Else X=-1 // if the API does not respond returns -1
15: End if
End

FIGURE 6. Pseudocode of blacklist feature.

1) BLACKLIST FEATURE

The blacklist feature offers a proactive line of defense and is highly efficient and quick, as it involves comparing URLs against an existing list of known malicious websites. In this paper, the blacklist is used as a feature to enhance the overall security posture. The blacklist feature classification pseudocode is demonstrated in Figure. 6.

The following are the steps taken to extract and classified blacklist feature:

- **Step 1 Read the URL from Identification Phase:** The original URL (U), which passed from the identification phase is used.
- **Step 2 Invoke VirusTotal API:** The VirusTotal (VT) [56] API will be called in this step. In order to do so, the API key is required, which has previously been obtained from the website, as well as the URL, which will be submitted with the request.
- **Step 3 ScanID Information in Response to the API Invoke:** The request that was sent to the API will now receive a response, and the information will be returned. Over 90 phishing and malware blacklist databases and online security scanners examined the URL, and the findings are reported by sending the ScanID. The ScanID (SID) is a lengthy string and contains various pieces of information, as well as the number of websites that identified the proposed URL throughout the investigation.
- **Step 4 VirusTotal Report:** The VirusTotal API retrieves valuable information through a method known as Report (R). This method provides a report on the number of websites that have identified the proposed URL and is equivalent to (X).
- **Step 5 Outcome Assessment:** It is necessary to evaluate the report in order to determine whether the URL is malicious or not. The greater the number of websites on which VirusTotal detects the URL, the more likely it is dangerous. The predefined values of classes are set here. The output of this feature should encompass four distinct values: 0, 3, 5, or -1, varying according to different circumstances. This range is meticulously determined through iterative trial-and-error processes. If 6 or more blacklists and security websites are detected, it returns 5, indicating that it is more likely malicious; if 1 to 5 websites are detected, it returns 3, indicating that it is possibly malicious; if no websites are detected, it returns 0, indicating that none of the websites contain the given URL, it is assumed that the URL is benign; and finally, if the API does not respond, it returns -1. The rank returned by this feature is used to assist in determining if a URL is malicious or benign and will be employed later in evaluation and detection phase.

2) LEXICAL FEATURE

The lexical feature has emerged as a prime choice for detecting malicious URLs due to its minimal computational load, security, rapidity, superior classification accuracy, and notable independence from alternative sources. The feature's pseudocode is represented in Figure. 7, with detailed extraction and classification procedures outlined herein. Additionally, Table 5 provides an encompassing overview of the lexical feature classes implemented throughout this research.

The following are the steps taken to extract and classified lexical feature:

- **Step 1 Read the URL from Identification Phase:**The original URL (U), which passed from identification phase is used.
- **Step 2 Parse URL into Various Components:**The lexical feature breaks the URL into multiple components, such as the hostname, entire URL, top-level domain (TLD), and path. Each of these parts is examined separately for analysis.
- **Step 3 Extraction Lexical Features Classes:**At this stage, the predefined lexical feature classes are extracted. This research presents 22 lexical feature classes (N) to determine if a URL is benign or malicious and are divided into two categories: dictionary (binary), and counting (numeric). These classes (C) are: counting the length of the entire URL, length of path, length of hostname, counting special symbols such as "-"; "_"; "@"; "? "; "="; "%"; "_"; "/"; "//"; "."; ":"; "∼"; in the path, counting digits and letters in the path, detecting suspicious words and TLDs, checking for file extensions, using IPs and ports inside the hostname and check if the URL is absolute, and finally counting the number of redirects from the identification phase. Based on comparison, each class may be given a value between 1 and 5, where 1 indicates the assumption that the URL is benign and 5 indicates it is more likely to be malicious.

TABLE 5. List of selected classes along with their characteristics, reasons for their selection, value, and type.

	No	Classes	Reason of Selection	Value	Type
Lexical	1	%	It is utilized to determine URL HTML encoding, the existence of IDs (such as affiliate, session, referrer, and tracking IDs), and time stamps. It can trigger URL encoding exploits, leading to as injection attacks or bypassing security filters.	X<=3→ X=1 X>3→X=5	Numeric
	2	//	The extra "//" symbol could trick the web server into interpreting the path differently than intended. This could potentially allow the attacker to access files or directories outside the web root, leading to unauthorized access or data leakage.	X<2 →X=1 2=<X<4→X=3 X>=4 →X=5	Numeric
	3	/	Having more than usual "/" symbols in a URL can pose security risks such as directory traversal attacks, injection vulnerabilities, and potential bypassing of security measures.	0<X<6 →X=1 6=<X<11→X=3 X=<11→X=5	Numeric
	4	.	It is employed to figure out the relative references in the domain name, SLD, TLD, and path hierarchy. Excessive occurrences of this symbol in a URL can potentially indicate an attempt to obfuscate or hide malicious content, making it more difficult for users and security systems to identify and block harmful websites.	X<4 →X=1 4=<X<7 →X=3 X>=7 →X=5	Numeric
	5	=	It is employed to determine the URL's value assignment. A high occurrence of the "=" symbol in a URL can potentially indicate a query string, which if manipulated maliciously, could lead to injection attacks or other security vulnerabilities.	X<4 →X=1 X>=4→ X=5	Numeric
	6	_	The number of word joiners is counted. The excessive appearance of "_" symbols in a URL can potentially lead to malicious exploits due to the risk of obfuscation and manipulation by attackers.	X<4 → X=1 X>=4→ X=5	Numeric
	7	-	The number of word separators is counted. It can create opportunities for attackers to obscure malicious content within the URL structure, making it harder for users and security systems to detect suspicious links or phishing attempts.	X<4 → X=1 X>=4 →X=5	Numeric
	8	@	It was employed to extract and examine whether user data was present in the URL. It can lead opportunities for attackers to disguise sensitive information, such as login credentials, within the URL itself.	X<4 →X=1 X>=4 → X=5	Numeric
	9	?	It's employed to determine whether the URL has a query component. Attackers can exploit this by crafting URLs with numerous query parameters, potentially hiding malicious code, attempting to manipulate server behavior, leading to security vulnerabilities.	X<2 →X=1 2=<X<4 →X=3 X>=4 →X=5	Numeric
	10	:	It's been applied to figure out if a port is in use. It could potentially allow attackers to exploit vulnerabilities with specific ports, bypass security measures, gain unauthorized access, or execute malicious actions on the targeted system.	X=1 →X=5	Numeric
	11	~	It has been employed to figure out whether the URL contains any references. It can potentially lead to malicious exploitation due to its association with user directory enumeration vulnerabilities.	X=1 →X=5	Numeric
	12	URL Length	It has been utilized to determine the obfuscation method most frequently used in phishing attacks. It can disguise deceptive domains or hide suspicious characters, increasing the likelihood of users falling victim to fraudulent websites.	X<51 →X=1 50<X<75→X=3 X=>75 →X=5	Numeric
	13	Hostname Length	It was used to determine which obfuscation method. It exploits vulnerabilities in parsing mechanisms, potentially leading to phishing or injection attacks.	X<20 →X=1 19<X<31 →X=3 X=>31 →X=5	Numeric
	14	Path Length	It is employed to determine whether an obfuscation technique is being used, typically in path length for phishing attacks.	X<30 →X=1 29<X<51 →X=3 X=>51 →X=5	Numeric
	15	Misspellings	Attackers create URLs that appear like legitimate ones with slight variations in spelling to trick victims into entering sensitive information, downloading malware, or falling for phishing scams.	If No →X=1 If Yes →X=5	Binary
	16	URL Shorteners	Attackers use URL shorteners to mask malicious links, making them appear legitimate and enticing victims to click, leading to potential phishing or malware propagation.	If No →X=1 If Yes →X=5	Binary
	17	Count Redirect	It was primarily utilized for passing the detection models and was used to determine the obfuscation approach. This class's data was acquired during the preparation stage.	X>=1 →X=1 X>1 →X=5	Numeric
	18	Count Digits	It may indicate the presence of encoded data, such as session identifiers or cryptographic keys, which could be used to exploit vulnerabilities in web applications or conduct phishing, injection, or session hijacking attacks.	X<31 →X=1 X>30 →X=5	Numeric
	19	Exist IP	It bypasses DNS-based security measures and facilitates direct access to servers, increasing the risk of targeted attacks or phishing schemes. It mostly used by Drive by download attack.	If no →X=1 If yes X=5	Binary
	20	Exist Port	When specific ports are used, a link is made between the web browser and the web servers, which exposes private user information to hackers and may lead to serious data exploitation.	If absence or present standard ports X=1 If present non-standard ports X=3 If present port 80 or 443 X=5	Binary
	21	Absolute URL	An absolute URL encompasses comprehensive information from the protocol to the top-level domain (TLD). Conversely, a relative URL omits the full web address and includes only the location following the domain name, typically employed for obfuscation purposes.	If yes →X=1 If no →X=5	Binary
	22	Suspicious TLD	The extensive list of suspicious top-level domains, which is divided into two categories: high risk and mid-risk, is primarily utilized for malware and phishing attacks.	If no →X=1 If mid risk→X=3 If High risk →X=5	Binary
	23	Suspicious Words	There are two categories of high risk and mid-risk words in the long list of suspicious terms; they are primarily utilized in malware and phishing attacks.	If no →X=1 If mid risk→X=3 If High risk →X=5	Binary
	24	File Extension	Attackers have mostly utilized it to download executive files to initiate a drive-by-download attack on their targets. All of the executive files in various operating systems are included in this list.	If no →X=1 If yes →X=5	Binary
Host	25	Create Date	Given that malicious websites are often temporary, the number shows here indicates the duration since its creation.	X>180 →X=1 90<X<181→X=3 X<=90 →X=5	Numeric
	26	Expiry Date	It is common for reputable domains to need payments in advance for a number of years. Malicious websites often expire and kindly do not persist over time.	X> 90 →X=1 0<X<91 →X=3 X=<0 →X=5	Numeric
	27	Update Date	It is based on the regular updates of information found on trusted websites.	X>730 →X=1 X<731 or if not provide →X=3	Numeric
	28	Owner Details	It verifies if the owner of the website submitted the information. The majority of forged URLs use rental domains and don't disclose any information.	If yes →X=1 If no →X=3	Numeric
	29	Global Rank	As malicious URLs typically have a short-lived existence, they often possess a low rank. Checking if a website ranks among the top 100,000 can help identify potential threats. However, it's noteworthy that approximately 25% of malicious URLs are hosted on reputable domains (This strategy reduces suspicion and enhances the difficulty of detection).	If top rank→X=1 If not rank→X=5	Numeric
	30	Total Visits	According to the ephemeral nature of phishing and malware websites, the metric used involves assessing the average monthly number of visitors to a website.	If X>= 1M→X=1 If 50K<X< 1M →X=3 If X<50K or if not provide→X=5	Numeric
	31	Country	It is crucial to examine the roster of countries where the predominant share of malicious domains is situated. Certain countries may host attackers due to factors such as lack cybersecurity regulations or a concentration of skilled tech professionals.	If not in list →X=1 If in the list or not provided →X=3	Binary
	32	City	It is crucial to examine the roster of cities where the predominant share of malicious domains is situated. Certain cities may host attackers due to factors such as lack cybersecurity regulations or a concentration of skilled tech professionals.	If not in list →X=1 If in the list or not provided →X=3	Binary
HTML	33	Iframe	Iframes serve as a common method to integrate dynamic content from an external website onto a page within your site. Attackers have exploited iframes within web pages to redirect visitors successfully. Consequently, hidden iframes are often employed by attackers to deceive users into accessing a malicious site.	If absence →X=1 If present →X=5	Binary
	34	Mailto	It enables automatic email composition, which can be exploited by attackers to initiate phishing attempts or distribute malware, which can lead to the unwitting disclosure of sensitive information or the execution of harmful scripts.	If absence →X=1 If present →X=5	Binary
	35	Webpage size-based	Malicious websites frequently consist of a single, extensive line of HTML code. As indicated by [71], 25% of malicious URLs with distinct domain names exhibited an identical total number of lines of code.	If X<500 Chars →X=1 If X=> 500Chars →X=5	Numeric
JavaScript	36	Popup windows	JavaScript's Window Open () pop-ups are commonly employed for advertisements and injecting exploits, primarily utilized by spammers.	If absence →X=1 If present →X=3	Binary
	37	JavaScript functions	Commonly, web-based malware distribution involves the utilization of native JavaScript functions such as escape (), eval (), unescape(), exec(), and search().	If absence →X=1 If present →X=5	Binary
	38	DOM functions	JavaScript enables attackers to manipulate the DOM structure of a webpage, providing them the means to gain unauthorized access to user data. In this study, an examination of the source code by checking for the presence of DOM functions like appendChild and createElement in a webpage.	If absence →X=1 If present →X=5	Binary
	39	JavaScript obfuscation, suspicious functions	To evade detection, attackers utilize obfuscation techniques that complicate code analysis. Several suspicious functions, including ActiveXObject, CreateObject, CreateTextFile, FileSystemObject, and FileExists, may be exploited to gain access to files and directories. Additionally, these functions can be used to create a backdoor for monitoring computer activity.	If absence →X=1 If present →X=5	Binary

TABLE 5. *(Continued.)* **List of selected classes along with their characteristics, reasons for their selection, value, and type.**

Certificate	40	Certificate	These classes are employed to verify and validate the existence of a certificate on the server side, ensuring a secure connection.	If absence →X=1 If present →X=5	Binary
	41	Certificate Analysis	Verifying the authenticity of SSL/TLS certificates to identify potential malicious websites. The lack of that may result in it compromising the security and integrity of data exchanged, leaving it vulnerable to interception by unauthorized parties	If absence →X=1 If present →X=5	Binary
Redirect	42	Redirect Chains (Dynamic)	Analyzing the sequence of redirects before landing on the final URL to identify malicious redirections. Counting the number of redirections can help detect malicious URLs by identifying unusually high redirection chains, which are commonly employed in phishing or malware distribution to obfuscate the true destination and evade detection.	X>=1 →X=1 X>1 →X=5	Numeric

Table 5 shows the classes of lexical features that are implemented in this research, the predefined values that are set for each class, and Table 6 illustrates the rules for allocating the priority coefficient.

- **Step 4 Analyze Results:**The selected classes search throughout the URL components, comparing the result to the specified predefined value and returning the outcomes. Multiple values are implemented in each class, and dealing with any of them leads to a particular outcome. These values are the consequence of extensive investigation into a variety of methods for detecting malicious links, as well as considering the behaviour of URLs in several datasets.
- **Step 5 Allocating Priority Coefficient on Selected Classes:**One of the significances of this research compared to others in terms of detecting the malicious URLs is that each class has a distinct level of importance. Some classes are more important than others in detecting malicious URLs, such as utilizing an IP address instead of a domain name or using a lengthy website name. This study highlights six classes of lexical features deemed of higher importance (high risk (HR)), necessitating coefficient implementation based on specific conditions. These classes are detailed in Table 6, with implementation guided by the conditions specified in the coefficient column. For example, if an IP presence is in the URL, it will return 5 (according to Table 5), and then the coefficient level 1.3 will be multiplied by it (1.3∗5), resulting in a higher value in the lexical feature.
- .As a result, if any of these classes is subjected to that comparison, the coefficient will be applied, and it will cause the detection framework to flag the URL as malicious or benign.
- **Step 6 Average and Outcome Assessment:**The last stage involves calculating the average of all the classes (X) (as well as the coefficient of the chosen classes). The average outcome should be between 1 and 5, which means 5 indicating that it is more likely malicious and 1 indicating that the URL does not have phishing and malware behaviour, and it is assumed that the URL is benign. The outcome of this feature will be utilized to assist in determining if the URL is malicious or benign in phase 3 and will always return the result.

3) HOST-BASED FEATURE

An essential factor in the detection of malicious URLs revolves around host-based features. These features offer a preliminary insight into the attributes and rank of the website host, enabling the discernment of the legitimacy or malicious intent of a URL. The pseudocode for the feature is shown in Figure. 8, and the extraction and classification processes are described. Table 5 also provides a comprehensive description of the host-based feature classes used in this research.

The following are the steps taken to extract and classified host-based feature:

- **Step 1 Read the URL from Identification Phase:**In this step, the original URL (U), which passed from identification phase, is used. It first checks the URL content, and if it contains an IP address, it returns -1 to phase 3. This is because the IP address lacks rank and WHOIS information.
- **Step 2 Invoke APIs:** The WhoXy (WXy) [99] and Open Page Rank (OPR) [100] APIs will be called in this step. WhoXy is a variant of the WHOIS API, functioning as a hosted web service that provides well-parsed WHOIS information in various formats. The Open Page Rank is designed to disseminate host ranks and visit metrics for the proposed platform. In order to do so, the API key is required, which has previously been obtained from the website, as well as the URL, which will be submitted with the request. If the APIs do not respond and return data, -1 sends to the feature evaluation and detection phase.
- **Step 3 Extraction Host-based Features Classes:** At this stage, the predefined host-based feature classes (C) are extracted. This research presents 9 classes (N) to determine if a URL is benign or malicious, and are divided into three categories: WHOIS, rank and IP geolocation. These classes are: created date, expiration date, updated date, owner details, global rank, total visits, country, city, and address. Each class may be assigned a value between 1 and 5 based on comparison, which means 5 indicates that it is more likely malicious and 1 indicates that it is assumed that the URL is benign. Table 5 shows the classes of host-based features that are implemented in this research, the predefined values that are set for each class, and the rules for allocating the priority coefficient.
- **Step 4 Analyze Results:**In this step, the selected classes are searching for the details from the APIs' responses, comparing the results to the defined value, and returning the outcome. Each class has several predefined values, and dealing with each of them gives a specific result.

TABLE 6. Selected classes for implementing coefficient levels and their circumstances, along with reasons for selection.

No	Class	Reason of Selection	Coefficient
1	%	This sign in URL encoding is exploited by attackers to obfuscate malicious payloads within URLs, enabling them to evade detection by security measures and manipulate query string parameters to execute various types of attacks.	If X=5 →1.2
2	Misspellings	Misspellings of website domains can be malicious as they are utilized in phishing scams, malware distribution, credential theft, and brand impersonation, deceiving users into visiting fraudulent sites and compromising their security.	If X=5 →1.2
3	URL Shorteners	Shortened URLs can pose a risk as they obscure the destination website, potentially leading users to malicious sites or phishing pages. Attackers leverage URL shortening services to disguise harmful links, making it challenging for users to assess their legitimacy.	If X=5 →1.3
4	Count Redirect	Redirecting to multiple websites can be malicious as it often indicates an attempt to deceive users or evade detection by security measures. Attackers may use redirection chains to obscure the true destination of a link, leading users to malicious or phishing websites.	If X=5 →1.3
5	Exist IP	The presence of an IP address instead of a hostname in a website's URL can signal potential malicious intent. Attackers often utilize IP addresses to obscure the true origin of a website, bypassing traditional domain-based security measures. Additionally, websites accessed solely through IP addresses may lack proper domain validation, increasing the risk of phishing scams or malware distribution.	If X=5 →1.3
6	Absolute URL	A website that lacks an Absolute URL may raise suspicion for malicious activity due to its potential for redirection to untrusted destinations. Without an Absolute URL specifying the complete web address, the website's links may lead users to unexpected or harmful locations. Attackers often exploit this ambiguity to trick users into visiting phishing sites or distributing malware, making it essential for users to exercise caution when encountering such URLs.	If X=5 →1.2
7	Suspicious TLD	A website with a suspicious top-level domain may pose a security risk due to factors such as unregulated or easily obtained TLDs commonly associated with malicious activities like phishing or malware distribution. Suspicious TLDs often lack the strict oversight and reputation of more established domains, making them attractive to cybercriminals seeking to deceive users or exploit vulnerabilities. (Two lists of suspicious TLDs are gathered and compared with the URL.)	If X=3 →1.2 If X=5 →1.3
8	Suspicious Words	A website containing suspicious words may signal potential maliciousness due to the presence of language commonly associated with scams, phishing, or malware. (Two lists of suspicious words are gathered and compared with the URL.)	If X=3 →1.2 If X=5 →1.3
9	File Extension	A file extension in a URL can be malicious as attackers often use it to initiate drive-by-download attacks, where unsuspecting users are tricked into downloading and executing harmful files. These files can include malicious executables targeting various operating systems, making it a common vector for spreading malware. (A list of suspicious file extension is gathered and compare with the URL.)	If X=5 →1.3
10	Create Date	Checking the creation date of a website is crucial as it helps identify potentially malicious URLs, particularly since many malicious websites are temporary. By analyzing the creation date, security measures can flag recently established websites, which are more likely to be associated with malicious activity.	If X=3 →1.2 If X=5 →1.3
11	Expiry Date	An expired domain may indicate abandonment or takeover by malicious actors who may repurpose it for illicit activities such as phishing or distributing malware.	If X=3 →1.2 If X=5 →1.3
12	Global Rank	Checking the global rank of a website provides insight into its popularity and reputation on the internet. Websites with low global ranks may indicate newer, which could be more prone to hosting malicious content or engaging in nefarious activities.	If X=5 →1.3
13	Iframe	It can be used to execute attacks such as phishing, malware distribution, or XSS. Attackers may use iframes to load malicious content from third-party websites, trick users into revealing sensitive information, or exploit vulnerabilities in the user's browser to execute malicious code.	If X=5 →1.2
14	JavaScript functions	These functions can manipulate URLs and execute arbitrary code, potentially leading to the generation and dissemination of malicious URLs. Attackers exploit these functions to obfuscate malicious payloads, evade detection by security measures, and ultimately compromise the security of web users.	If X=5 →1.2
15	DOM functions	The presence of DOM functions in URL introduces potential security risks as JavaScript allows attackers to manipulate the DOM structure, potentially compromising user data security. Attackers can dynamically alter webpage content or inject malicious scripts, leading to various types of attacks such as cross-site scripting (XSS) or data theft.	If X=5 →1.2
16	JavaScript obfuscation, suspicious functions	Attackers employ JavaScript obfuscation techniques, complicating code analysis and making it more challenging for security measures to detect malicious behavior. These functions serve as potential entry points for attackers to create backdoors and monitor computer activity, posing significant risks to website security and user privacy.	If X=5 →1.2
17	Suspicious functions	By examining digital certificates, security analysts can uncover indicators of compromised or fraudulent websites, such as mismatched domain names or expired certificates.	If X=5 →1.3

- **Step 5 Allocating Priority Coefficient on Selected Classes:**Created date, expiration date, and global rank classes have a higher level of importance, resulting in the allocation of a priority coefficient (CL) based on the conditions. The reasons for picking these classes are outlined in Table 5 and 6 and the primary one is that confronting them may provide us a better approach to determine if a URL is dangerous or safe.
- **Step 6 Average and Outcome Assessment:**The final step is to compute the average (X) of all the classes (as well as the coefficient of the chosen classes). The average outcome should be between 1 and 5, which means 5 indicating that it is more likely malicious and 1 indicating that the URL does not have malicious behaviour, and it is assumed that the URL is benign. The outcome of this feature will be utilized to assist in determining if the URL is malicious or benign in phase 3. This feature can return -1 in the case that the APIs do not respond.

4) CONTENT-BASED FEATURE

The content-based feature extracts statistics related to the HTML and JavaScript functions employed on the website. When other features fall short in identifying a URL, a more comprehensive analysis of content-based features aids in the detection of malicious websites. This feature is adept at uncovering potential threats originating from malicious websites.

Taking this into consideration, some other detection models do not employ this feature due to potential security vulnerabilities, often due to its significant computational load, which notably hampers the speed of identifying malicious URLs. Nonetheless, this research effectively addresses these challenges through two distinct strategies. First, the feature examines the URL input, and if the website contains an IP address, an executive file, or a data file, it does not download the URL, preventing a download-by-download attack and increasing security. Second, this feature concentrates solely on the essential classes that can identify malicious activity in HTML and JavaScript code, avoiding an exhaustive scan of the enormous data and reducing search time.

The pseudocode outlining the feature is presented in Figure. 9, accompanied by a breakdown of the extraction and classification procedures. Additionally, Table 5 furnishes a comprehensive overview of the content-based feature classes utilized within this study.

The following are the steps taken to extract and classified content-based feature:

- **Step 1 Read the URL from Identification Phase:**In the first step, the original URL (U), which passed through the identification phase, is used. If the URL contains a Top-Level Domain (TLD), it sends to the next step unless it returns -1 to the last phase. The reason is that some malware hosting websites lack domain names and are instead recognized by their IP addresses, which are frequently utilized in drive-by download

Lexical Feature Classification Pseudocode

Input: The URL is read from identification phase
Output: The lexical feature value (X) (The average outcome of classes)
Variable:

U	URL
N	Number of classes
i	Counter 1
C	Class
L	List of coefficients
CL	Coefficient Level
V	The comparison returns the value for each class
HR	The selected coefficient's value is met with a high-risk circumstance
j	Counter 2
X	The total value

Begin

```
1: U← Read URL          // Read URL that sent from identification phase
2: Parse U              // URL breaks into different components
3: For i=N Do           // Until all the classes covered
4: i=0
5: C insert             // Insert the class
6: Calculate C          // Calculate each class by comparing its predefined
                        value in the URL components (Table 5)
7: Return V             // Return the value between according to Table 5
8: If C=L               // If the class is included the coefficient list
                        according to Table 6
9: If V=HR              // If the selected classes' value is met with a high-
                        risk circumstance implement coefficient according
                        to Table 6
10: V=V*CL              // Implement coefficient according to Table 6
11: j=0
12: j=j+V               // Sum the classes values
13: End if
14: End if
15: i=i+1
16: End for
17: X=j/N               // Average the value of classes
18: Return X            // Return the lexical feature value and use for
                        feature evaluation and detection phase
```

End

FIGURE 7. Pseudocode of lexical feature.

attacks. Furthermore, it prevents a major vulnerability of content-based features, which is downloading malware content. It's also feasible that the website may supply a huge file to download that will require a lot of internet traffic and resources.

- **Step 2 Download the HTML Content:** At this point, the whole website's content will be downloaded for evaluation. If the server is not responding (which means that it is not possible to download website content), it will return -1 to the last phase from this feature.
- **Step 3 Extraction Content-based Features Classes:** At this stage, the selected classes for content-based features are extracted. This research introduces eight classes (N) for assessing the benign or malicious nature of a URL. The classes include mailTo, Iframe, webpage size, JavaScript functions, DOM functions, popup windows, JavaScript obfuscation, suspicious functions, and website certificates. Each class can be assigned a value ranging from 1 to 5 through comparison. A value of 5 suggests a higher likelihood of malicious intent, while a value of 1 implies an assumption that the URL is benign. Table 5 shows the classes of content-based features that are implemented in this research, the predefined values that are set for each class, and the rules for allocating the priority coefficient.
- **Step 4 Analyze Results:** In this part, the selected classes check for details in the downloaded HTML, compare the results according to the predefined, and return the outcome. Each class contains numerous values, and dealing with each one has a different result. These numbers were derived from extensive research into various ways of identifying malicious URLs and how URLs behaved in various datasets.
- **Step 5 Allocating Priority Coefficient on Selected Classes:** In terms of detecting malicious URLs, one of the differences between this research and others is that each class has its own importance level. Four classes of content-based features are prioritized according to Table 6. Consequently, if any of these classes is subjected to the comparison, the priority coefficient (CL) will be allocated, resulting in the detection framework identifying the URL as harmful.
- **Step 6 Average and Outcome Assessment:** The final step is to calculate the average (X) of all the classes (as well as the coefficient of the selected classes). The average outcome should be between 1 and 5, which means 5 indicating that it is more likely malicious and 1 indicating that the URL does not have malicious behaviour, and it is assumed that the URL is benign. The outcome of this feature will be utilized to assist in determining if the URL is malicious or benign in phase 3. This feature can return -1 in the case that the server is not responding.

C. ENHANCED MALICIOUS URLS DETECTION FRAMEWORK UTILIZING FEATURE EVALUATION METHOD

The Development of the Detection Framework (DF) was crafted the quantity of available data and the overall data volume. The suggested Malicious URL Detection Framework (DF) relies on a predefined static feature classification method, as outlined in equation 1.

$$DF = \sum_{i=1}^{i=n} (F_i \times CL_i \times 20) \tag{1}$$

Each feature, denoted as F_i, takes on a value (i = {1, 2, . . ., n}) indicating the number of features in total. The coefficient for each feature is represented as CL_i, and is detailed in the Table 7. The priority coefficient of a feature according to its level of importance lends greater weight to the essential classes that effectively detect malicious URLs. The cumulative value of each feature is then scaled by a factor of 20, resulting in a range of 0 to 100. This multiplication by 20 serves the purpose of distinguishing between the values assigned to classes and those attributed to features. Classes are assigned values from 1 to 5 according to a pre-defined static feature classification. Consequently, the value of each feature emerges from the sum of its constituent classes, which ranges from 1 to 5. This aggregated value is subsequently multiplied by 20, considering that the overall value of each feature is on a scale of 0 to 100. Subsequently, this value is multiplied by the priority coefficient of each feature, as per Table 7, enhancing the efficacy of malicious URL detection.

Host-based Feature Classification Pseudocode

Input: The URL is read from identification phase
Output: The host-based feature value (X) (The average outcome of classes)
Variable:

U	URL
WXy	WhoXy API
OPR	Open Page Rank API
N	Number of classes
i	Counter 1
C	Class
L	List of coefficients
CL	Coefficient Level
V	The comparison returns the value for each class
HR	The selected coefficient's value is met with a high-risk circumstance
j	Counter 2
X	The total value

Begin

```
1: U← Read URL          // Read URL that sent from identification phase
2: If U=IP              // If the URL is IP
3: Return -1            // The IP address lacks rank and WHOIS information.
4: Else
5: WXy← U               // Invoke WhoXy API
6: OPR← U               // Invoke Open Page Rank API
7: If WXy & OPR return  // Return information
8: For i=N Do           // Until all the classes covered
9: i=0
10: C insert            // Insert the class
11: Calculate C         // Calculate each class by comparing its predefined value in the URL components which specified in the Table 5
12: Return V            // Return the value between 1 to 5 according to Table 5
13: If C=L              // If the class is included the coefficient list according to Table 6
14: If V=HR             // If the selected classes' value is met with a high-risk circumstance implement coefficient according to Table 6
15: V=V*CL              // Implement coefficient according to Table 6
16: j=0
17: j=j+V               // Sum the classes values
18: End if
19: End if
20: i=i+1               // Next class
21: End for
22: Else
23: Return -1           // If WXy and OPR not responding return -1 (return -1 to phase 3, means this feature does not work)
24: X=j/N               // Average the value of classes
25: Return X            // Return the host-based feature value and use for phase 3
```

End

FIGURE 8. Pseudocode of host-based feature.

In the last phase, a comparison is conducted between DF and the predetermined threshold value of 200. If DF exceeds this threshold, the URL is categorized as malicious; otherwise, it is considered benign. The value of 200 was determined through testing and experimenting with various URLs. The assigned value is determined by the behavior of malicious URLs. When a feature identifies potential threats within a website, the corresponding feature value tends to be elevated. Through the allocation of a priority coefficient, this value experiences enhancement, leading to a more substantial figure. The cumulative effect of this augmented value, when combined with values from other features, is then assessed against the threshold value. The choice of 200 for the threshold has demonstrated effective performance given the predetermined feature values. The feature evaluation methodology employed in this research is encapsulated in the following rule:

$$\textit{If } F_i = -1 \textit{ and other } F_i \geq 3.5 \textit{ or blacklist} \geq 3 \textit{ then } CL_i \textit{ assign to the highest feature value}$$

This method evaluates the contribution of features in terms of value. It examines whether each feature s value to the overall computation, and if any feature falls short in delivering, the method will decide to use the other feature's coefficient value instead, based on specific conditions. The F_i represents the feature value, which could be obtained by using Equation 2.

$$F_i = \sum_{i=1}^{i=n} (C_i \times cL_i) \tag{2}$$

C_i indicates the class value, where $i = \{1, 2, \ldots, n\}$ represents the number of classes associated with a feature. Each class may assume a value of 1, 3, or 5, based on various comparisons and conditions specified in Table 7. The predefined values span this range to facilitate analysis, comparisons, and result derivation. cL_i represents the coefficient values corresponding to the classes detailed in Table 7. These coefficient values are assigned to selected classes based on the output of the classes and specific conditions.

Table 7 presents the priority coefficient of features. The priority coefficient is assigned to each feature according to its level of importance to boost the detection accuracy of malicious URLs. The range of coefficients, from 1 to 1.5, shows how important a website's potential maliciousness is. A coefficient of 1 means the website is likely not malicious, while 1.5 means it's highly suspicious. This range is meticulously selected through iterative trial-and-error processes, where the efficacy of different coefficient values is rigorously tested and refined.

Due to the fact that URLs that appear in many VirusTotal blacklist databases are more likely to be malicious, the blacklist feature has the highest priority coefficient value, which gives it more weight and makes it extremely useful for identifying harmful websites. The next feature is host-based, which can deliver significant information on website rank, host information (WHOIS), and location and is essential for detecting malicious URLs. The lexical and content-based features have a coefficient level of 1.2 and are of lower value for detecting malicious URLs, according to the statistical classes employed for them.

The pseudocode outlining an advanced malicious URL detection framework, integrating feature evaluation methods, is shown in Figure. 10 and explained step by step here.

The following are the steps taken by the proposed phase:

- **Step 1 Read Features Value from Feature Classification Phase:**The first step is reading the blacklist, lexical, host-based, and content-based values from previous phase.
- **Step 2 Calculate the Blacklist Feature Value:**The blacklist value (B_v) is read and will be computed here. If the API did not respond, -1 was returned. As a result, it would verify the other features' values, and if these

Content-based Feature Classification Pseudocode

Input: The URL is read from identification phase

Output: The content-based feature value (X) (The average outcome of classes)

Variable:

U	URL
N	Number of classes
i	Counter 1
C	Class
L	List of coefficients
CL	Coefficient Level
V	The comparison returns the value for each class
HR	The selected coefficient's value is met with a high-risk circumstance
j	Counter 2
X	The total value

Begin

```
1: U← Read URL          // Read URL that sent from identification phase
2: If U=IP              // If the URL is IP
3: Return -1
4: Else
5: Download U           // Download HTML content of website
6: If U respond         // If the server responds
7: For i=N Do           // Until all the classes covered
8: i=0
9: C insert             // Insert the class
10: Calculate C         // Calculate each class by comparing its value in the
                        URL components which specified in the Table 5
11: Return V            // Return the value between according to Table 5
12: If C=L              // If the class is included the coefficient list according
                        to Table 6
13: If V=HR             //If the selected classes' value is met with a high-risk
                        circumstance allocate priority coefficient based on
                        Table 6
14: V=V*CL              // Implement coefficient according to Table 6
15: j=0
16: j=j+V               // Sum the classes values
17: End if
18: End if
19: i=i+1               // Next class
20: End for
21: Else
22: Return -1           // If the server not reponed
23: X=j/N               // Average the value of classes
24: Return X            // Return the content feature value and use for feature
                        evaluation and detection phase
```

End

FIGURE 9. Pseudocode of content-based feature.

features were equal to or greater than 3.5, it would assign the coefficient level (CL) to the highest one, if not, it would return 0. If there is a value returned from the blacklist, it will be used in the evaluation. Then multiplying the value by 20 completes the process (the worth of each level is out of 100) and sums up with X. For example, if blacklist is returned with -1 and lexical feature's value is 3, host-based feature's value is 3.5 and content-base's value is 3.3, the coefficient of blacklist, which is 1.4 (according to Table 7), will be used for host-based feature. So, $3.5 \times 1.4 \times 1.3$ (host-based feature coefficient level) $\times 20 = 127.4$ is the total value of blacklist and host-based features. Also, if two features return -1 it proceeds in the same way. In the same example, if blacklist and content-based return -1, their coefficient value allocate to host-based feature and calculated by $3.5 \times 1.4 \times 1.2 \times 1.3 \times 20 = 152.88$. This is the total values of blacklist, host-based, and content-based features. In the other example, if blacklist is returned with -1 and lexical feature's value is 3, host-based feature's value is 3.2, and content-based feature's value is 3.4, the

TABLE 7. Coefficient level of features.

No	Feature	Coefficient Level
1	Blacklist	1.4
2	Lexical	1.2
3	Host-based	1.3
4	Content-based	1.2

priority coefficient of blacklist is not allocated to any other features.

- **Step 3 Calculate the Lexical Feature Value:** The main advantages of the lexical feature for identifying malicious URL models are data independent on other APIs and always returns the value. It may be useful, especially when other features do not respond. In this step, the lexical value (L_v) that was read from phase 1 will be multiplied by the lexical priority coefficient value, 20, and then summed up to X.
- **Step 4 Calculate the Host-based Feature Value:** The host-based value (H_v) read in phase 1 will be computed here. If the API did not respond, -1 was returned. As a result, it would verify the other features' values, and if the lexical and content-based values were equal to or greater than 3.5, or the blacklist value were equal to or greater than 3, it would assign the coefficient level (CL) to the highest one, if not, it would return 0. If there is a value returned from the host-based feature, it will be used in the evaluation. Multiplying the value by 20 completes the process (the worth of each level is out of 100) and sums up with X.
- **Step 5 Calculate the Content-based Feature Value:** The content-based value (C_v) read in phase 1 will be computed here. If the API did not respond, -1 was returned. As a result, it would verify the other features' values, and if the lexical and host-based values were equal to or greater than 3.5, or the blacklist value were equal to or greater than 3, it would assign the coefficient level (CL) to the highest one, if not, it would return 0. If there is a value returned from the host-based feature, it will be used in the evaluation. Multiplying the value by 20 completes the process (the worth of each level is out of 100) and sums up with X.
- **Step 6 Ranking and Detection:** The final stage will consist of analyzing X's value. It is computed by summing up all the features. The resulting sum is then compared to a threshold value of 200; if it surpasses the threshold, the URL is considered malicious; otherwise, it is deemed benign.

D. EVALUATION METRICS

We compared the results using the confusion matrix, a table designed to visualize the performance of malicious URL detection (Table 8). It includes the following prediction quality measures:

Enhanced Malicious URL Detection Framework (Feature evaluation)

Input: The value of each feature read from phase 2
Output: The outcome value (X) and compare with threshold value
Variable:

B_v	Blacklist feature value
L_v	Lexical feature value
H_v	Host-based feature value
C_v	Content-based feature value
CL	Coefficient Level of each feature
X	The total value

Begin

1: If B_v= -1	// If blacklist feature return -1
2: If L_v or H_v or C_v >= 5	// If another features value is equal or greater than 3.5
3: L_v or H_v or C_v ←B_vCL	// Coefficient level of blacklist feature will use for other feature with highest value (Table 7)
4: Else B_v=0	// If B_v= -1 and other features value are less than 3.5, the value of B_v is 0
5: Else X=X+ B_v*20	// Sum the value of features by multiplying to 20 (the worth of each level is out of 100)
6: X=X+L_v*20	// Lexical feature always returns a value
7: If H_v= -1	// If host-based feature return -1
8: If L_v or C_v>= 3.5 or B_v>=3	// If another features value is equal or greater than 3.5
9: L_v or B_v or C_v ← H_vCL	// Coefficient level of host-based feature use for other feature with highest value (Table 7)
10: Else H_v =0	// If other features value is less than 3.5, the value of H_v is 0
11: Else X=X+ H_v*20	// Sum the value of features by multiplying to 20 (the worth of each level is out of 100)
12: If C_v = -1	// If content-based feature return -1
13: If L_v or H_v>= 3.5 or B_v>=3	// If another features value is equal or greater than 3.5
14: L_v or H_v or B_v ← C_vCL	// Coefficient level of content-based feature use for other feature with highest value (Table 7)
15: Else C_v =0	// If other features value is less than 3.5, the value of C_v is 0
16: Else X=X+C_v*20	// Sum the value of features by multiplying to 20 (the worth of each level is out of 100)
17: If X>=200	// If the total value of the all features plus their coefficient is equal or greater than 200 (threshold value)
18: Return URL is malicious	
19: Else	// If the total value of all features plus their coefficient is smaller than 200
20: Return URL is benign	

End

FIGURE 10. The pseudocode of enhanced malicious URLs detection framework.

- **True Positive (TP):** the number of correctly detected malicious URLs.
- **True Negative (TN):** the number of accurately detected benign URLs.
- **False Positive (FP):** the number of benign URLs mistakenly detected as malicious.
- **False Negative (FN):** the number of malicious URLs incorrectly detected as benign.

In addition, to provide a comprehensive representation of the performance of the malicious URL detection framework, we assess it utilizing various metrics, including Accuracy (Acc), False Positive Rate (FPR), Precision (Pre), Recall (Rec), and F-1 score (F1). The evaluation metrics are presented in Table 9.

V. EXPERIMENTAL RESULTS AND DISCUSSION

To evaluate the efficacy of the proposed framework, multiple experiments were undertaken. In the initial experiment, the framework underwent evaluation through its three distinct phases to demonstrate the effectiveness of these methods in enhancing detection accuracy. Subsequently, experiments were carried out to compare the performance difference between the chosen supervised machine learning models and the proposed framework. In the final experiment, the proposed framework was benchmarked against three previous methods—PDRCNN, Li, and URLNet—to measure its performance. These comprehensive evolutions employed machine learning and deep learning, along with proficient feature classification, to detect malicious URLs.

We present an experimental setup that leverages the MVVM (Model-View-View Model) architecture implemented using the Kotlin programming language (version 1.6.0) on the Android platform. The main reason to implement our experimental setup in Android stems from a strategic choice to address real-world scenarios and deploy our innovative detection framework in a tangible and accessible manner. Android, being a ubiquitous mobile platform, not only offers a vast user base but also provides a familiar and user-friendly environment for evaluation of our detection framework's performance in diverse and real-world settings. This setup forms the foundation of our innovative mobile application, serving as a robust framework to seamlessly manage data flow and user interface interactions, ultimately enhancing the efficiency and reliability of our detection framework. Importantly, our framework is designed to be adaptable, as it can seamlessly implement across different platforms while maintaining optimal functionality. This adaptability underscores the versatility of our solution and its potential applicability across a range of technological ecosystems. To run the experiments, a macOS (10.15 Catalina) with the following setup was utilized: 2.7 GHz Dual-Core Intel Core i5, 8 GB 1867 MHz DDR3.

A. DATASET

The proposed framework for detecting malicious URLs is trained on a dataset including 6000 samples. In this research, it is referred to as dataset I and is available on GitHub [101]. The dataset includes 2000 malicious URLs obtained from a Kaggle dataset collected between 2020 and 2022, along with 4000 benign samples collected from the top 4000 site links of Alexa in 2022.

The experimental dataset, labeled dataset II [101], consists of 5000 real-world URLs recently gathered in 2022. This dataset comprises 2500 benign URLs and 2500 malicious URLs, including 1500 phishing and 1000 malware URLs. The malicious URLs were sourced from two prominent malware and phishing databases, URLhaus and PhishTank [55], [102]. Each URL in the dataset is labelled as either malicious or benign.

The collected dataset is comprised of active URLs hosted on responsive servers, enabling us to perform real-time assessments of our research. These URLs have undergone verification using a range of tools and have been classified as either benign or malicious. In assessing the framework's effectiveness, we've selected a diverse set of URL attributes

TABLE 8. Confusion matrix.

Actual Class		True Class	
		Malicious	Benign
Tested Class	Malicious	True Positives	False Positives
	Benign	False Negatives	True Negatives

TABLE 9. Evaluation metrics.

Metrics	Definition	Equation
ACC	Represents the proportion of correct classifications	$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$
Pre	Signifies the percentage of correctly detected malicious URLs out of all predicted malicious URLs.	$\text{PRECISION} = \frac{\text{TP}}{\text{TP} + \text{FP}}$
False Positive Rate	Indicates the percentage of predicted malicious URLs that are genuinely legitimate among all legitimate URLs.	$FPR = \frac{FP}{FP + TN}$
Rec	Denotes the percentage of successfully detected malicious URLs out of all malicious URLs.	$Recall = \frac{TP}{TP + FN}$
F-1 score	F1 is the harmonic average of the precision and recall rates.	$F1 = \frac{2 * (Precision * Recall)}{Precision + Recall}$

from this dataset, including shortened URLs, URLs with IP addresses, obfuscated URLs, excessively long and short URLs, as well as URLs that have undergone more than two redirections. Table 10 illustrates the types of challenging URLs used in the dataset.

B. EVALUATION OF THE PROPOSED FRAMEWORK WITH THREE PHASES

This experiment is focused on demonstrating the effectiveness of the employed methods in enhancing the detection accuracy of the framework. The proposed framework is evaluated in the three phases separately. The outcomes are presented in Table and Figure 11, utilizing dataset II.

The first evaluation presented is the absence of the identification phase. The primary objective of this method is to redirect shortened URLs to their original websites. This method significantly contributes to enhancing the detection accuracy of the framework by transmitting the original website to the feature classification phase. However, in cases where a URL does not undergo redirection and is directly processed by the features, certain features may inaccurately evaluate the website, leading to a high false positive rate. As can be seen in Table 11, the accuracy of the framework is enhanced by around 4% by implementing the proposed method for detecting obfuscated URLs.

The second evaluation involved the absence of priority coefficients and predefined value methods for the feature classification phase, which contributes significantly to accuracy and is attributed to two key factors. First, the predefined static feature classification method provides a range of values for each class, which provides an effective framework for malicious URL detection and overcomes the data-dependent drawbacks of learning methods. The range of values is independent of the training dataset and may be modified without retraining the entire method. Second, the priority coefficient is allocated to the selected classes based on their level of importance and lends greater weight to the essential classes that effectively detect malicious URLs. As illustrated in Table 11, the incorporation of feature classification methods results in an improvement of over 11% in the accuracy of the proposed framework.

The third evaluation examined the malicious URL detection framework and feature evaluation phase without the inclusion of priority coefficients for the features and feature evaluation methods. The main purpose is to illustrate the effectiveness of this phase in detecting websites. As shown in Table 11, the accuracy of the framework has remarkably improved by 17%, and it is because of two primary factors. In the initial step, this phase assesses the contributions made by features during the feature classification phase. It determines whether all features contribute value to the final calculation, and in case any feature fails to do so, the framework decides to utilize the coefficient value of another feature under various circumstances. Even when two features are unresponsive, this framework maintains the ability to accurately detect malicious URLs. In the subsequent step, this phase assigns a priority coefficient method to each feature based on its significance, providing greater weight to essential features that are more effective in detecting malicious URLs.

C. PERFORMANCE EVALUATION OF SUPERVISED MACHINE LEARNING MODELS

This experiment focused on comparing the accuracy of the proposed framework with three supervised machine learning approaches, namely RF, SVM, and Bayesian Network (BN). The proposed framework and supervised machine learning methods employ the same feature classification and dataset for the purposes of training, testing, and detection.

The split ratio of 70:30 is utilized for model training and testing. The main reason for selecting this ratio is grounded in several key considerations. Firstly, it strikes a balance between training and testing data, dedicating a substantial 70% of the dataset to training, which enables the model to effectively learn underlying patterns. This allocation also mitigates the risk of overfitting. Moreover, the 30% testing set provides a sufficiently large sample to accurately assess

TABLE 10. Types of challenging URLs in the dataset.

Types of Challenging URLs	Description
Shortened URLs	These are URLs employed by shortening services (e.g., Bitly) to reduce the character count.
URL redirection	URLs redirected to other domains; this dataset includes URLs redirected up to 5 times.
URLs with IP address	URLs using IP addresses instead of domain names, more prone to malicious use, and often exploited in drive-by download attacks.
Lengthy URLs	URLs with long domain names that are challenging to detect, primarily utilized in phishing attacks.
Obfuscated URLs	URLs obfuscated through various techniques.

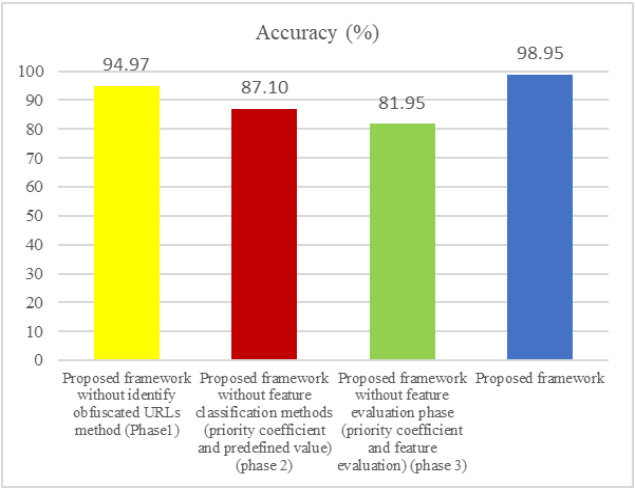

FIGURE 11. Evaluation accuracy of proposed framework with three phases.

the model's performance, ensuring statistically meaningful results. This split ratio aligns with established practices in malicious URL detection methods in the field of machine learning, contributing to the comparability of our research with existing studies [30], [68].

In the RF model, a maximum tree depth of 5 was selected as an essential hyperparameter tuning choice. This choice was driven by the objective of balancing model complexity and generalization performance, thus preventing the risk of overfitting by fitting the training data too closely. The SVM model was chosen for classification, employing a linear kernel with $C = 1$ to strike a balance between capturing decision boundary complexity and avoiding overfitting, thus rendering the model robust and interpretable. In the BN model, each of the 42 features is represented as an individual node within the network. Furthermore, there is a dedicated node for the classification target, distinguishing between malicious and benign URLs, denoted as node Y. Each of the 42 feature nodes establishes a probabilistic connection with the classification node Y via edges in the network structure.

The evaluation metrics that were utilized to assess these methods are accuracy, precision, and Receiver Operating Characteristics (ROC). The overall results of these comparisons are highlighted in Table and Figure 12. The outcome demonstrates that the proposed framework outperforms supervised machine learning approaches across all measures. The highest accuracy was obtained by the proposed framework, and surprisingly, SVM was located at the second stage by 98.95% and 93.33%, respectively. According to [74] and [96], RFs usually outperform other supervised machine learning approaches to detect malicious URLs, and it was expected that RF would have better accuracy than SVM, but it is in 3rd place by 92.86%. Furthermore, the Bayesian network shows unacceptable performance, with only 78.57% accuracy.

These are the two primary reasons for this evaluation. The first reason is to emphasize the primary shortcoming of supervised machine learning, which is its data dependency, and to demonstrate that these techniques perform poorly with a limited amount of data. The second reason is to demonstrate the effectiveness of allocating priority coefficients to the selected classes for detecting malicious URLs with the same feature classification.

D. PERFORMANCE EVALUATION OF MALICIOUS URL DETECTION METHODS

In this section, the proposed framework is benchmarked with the other three prior research that have made major contributions to the detection of malicious URLs. These methods are PDRCNN [24], Li method [31], and URLNet [17], all of which are detailed in the related work section. These methods were chosen due to their significant contributions in the field of malicious URL detection, as outlined in the related work section. PDRCNN represents a convolutional neural network-based approach, which has gained attention for its potential in feature extraction from URL data. The Li method, on the other hand, utilized several machine learning methods that incorporates a diverse range of feature classes, including blacklist, lexical, host-based, and content-based features, making it a comprehensive solution for URL classification. URLNet, utilizing neural network architectures, offers an innovative approach to automatic feature classification. By benchmarking against these methods, we aim to showcase the advancements and superior performance of our proposed framework, which addresses the limitations and inefficiencies observed in existing methodologies.

Experiments were conducted using real-world URLs from dataset II to benchmark the performance of these methods and the proposed framework. The overall results of these benchmarks are highlighted in Table 13. Also, Figure. 13 demonstrates the accuracy of these methods. The highest accuracy was obtained by the proposed framework at 98.95%, and the Li method's performance was better than others at 86.37%. The evaluation involved subjecting the dataset to four classifiers from the Li method: 3-NN, LR, L-SVM, and

TABLE 11. Evaluation of the proposed malicious URL detection framework and phases.

Methods	Accuracy (%)	Precision (%)	F1 Score (%)	Recall (%)
Proposed framework without identification (Phase1)	94.97	96.30	94.97	93.68
Proposed framework without feature classification methods (priority coefficient and predefined value) (phase 2)	87.10	89.60	87.41	85.33
Proposed framework without feature evaluation phase (priority coefficient and feature evaluation) (phase 3)	81.95	83.80	82.28	80.81
Proposed framework	**98.95**	**98.60**	**98.95**	**99.30**

LDA. Among these, LR exhibited better performance and was chosen for this benchmarking, while the other classifiers displayed poor results. The URLNet (full) method followed with a performance rating of 76.27%, and the PDRCNN method lagged behind with a detection accuracy of 73.09%.

The Li method outperforms both PDRCNN and URLNet due to its effective feature classification. This method encompasses a wide range of classes extracted from various sources, including blacklist, lexical, host-based, and content-based features. However, the other methods only focus on the lexical feature that has the least level of importance in detecting malicious URLs.

Nevertheless, the Li method falls short in accuracy compared to the proposed framework due to several factors. To begin with, machine learning techniques often do not perform well with small datasets, such as the 5000 URLs in our case, and require a massive dataset for training to generate an acceptable prediction method. In contrast, the proposed framework excels across a range of dataset sizes according to its predefined static feature classification, consistently delivering robust performance. Furthermore, the inclusion of 62 selected classes in the Li method introduces a collection of irrelevant classes that do not contribute to the identification of malicious URLs, instead increasing the detection process. However, this particular limitation is effectively addressed in the proposed framework, where the emphasis is placed on the careful selection of correlated classes that are effective in detecting malicious URLs. Additionally, the framework allocates a priority coefficient for selected classes and features, further refining the accuracy of the detection process. The other significant limitation of the Li method is its incapacity to effectively handle short URLs, which consequently leads to reduced accuracy in overall detection. This limitation is effectively mitigated in the proposed framework through the implementation of an identification phase involving URL redirection. This phase acts by redirecting shortened URLs to their original form, thereby resulting in the accurate detection of malicious URLs by the features.

The PDRCNN and URLNet methods exhibit several weaknesses that contribute to a low level of accuracy. Firstly,

TABLE 12. Assessment of supervised machine learning methods and the proposed framework.

Methods	Accuracy (%)	Precision (%)	ROC (%)
RF	92.86	88.89	77.08
SVM	93.33	90.00	91.66
BN	78.57	77.78	78.17
Proposed Framework	**98.95**	**98.60**	**98.77**

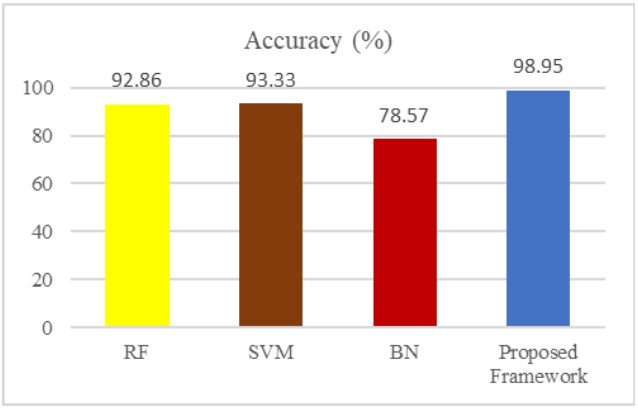

FIGURE 12. Evaluation accuracy of proposed framework with three supervised machine learning.

TABLE 13. Benchmark proposed framework and other malicious URL detection methods.

Methods	Accuracy (%)	Precision (%)	F1 Score (%)	Recall (%)
PDRCNN	73.09	74.22	73.39	72.58
Li method	86.37	87.44	86.51	85.61
URLNet (Full)	76.27	77.86	76.64	75.46
Proposed framework	**98.95**	**98.60**	**98.95**	**99.30**

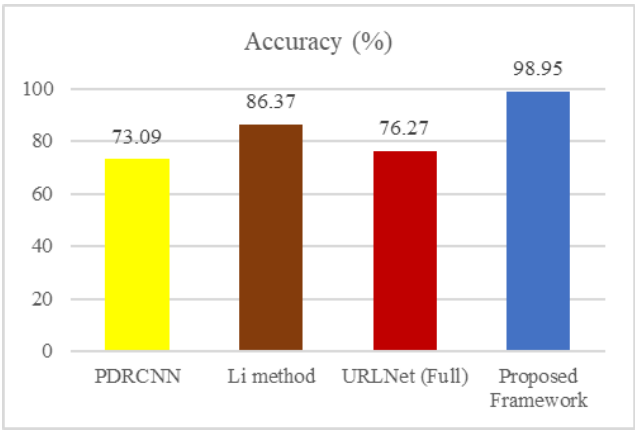

FIGURE 13. Evaluation accuracy of proposed framework with other malicious URL detection methods.

a significant motivation for researchers transitioning from machine learning to deep learning lies in the capacity for automatic feature classification from raw datasets, a task that

is inherently complex and requires expertise. However, these methods present a paradox by necessitating manual feature classification. Secondly, deep learning approaches demonstrate poor performance when confronted with limited data for training. Lastly, both methods lack mechanisms to detect short URLs. Consequently, when faced with datasets containing these types of challenging URLs, their performance is notably deficient.

Besides, the proposed framework sets itself apart from other methods through its novel feature evaluation method, which stands as the core innovation in this research. This method systematically evaluates the output of each feature in the ultimate computation. In cases where a feature lacks essential data, the method intelligently allocates the priority coefficient values of other features based on predefined conditions, ensuring robust decision-making that leads to accurate detection.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed an innovative framework for detecting malicious URLs. It is based on predefined static feature classification by allocating priority coefficients and feature evaluation methods. We utilized 42 classes of blacklist, lexical, host-based, and content-based features in the feature classification. The performance of the proposed framework was evaluated using a real-world dataset comprising 5000 URLs. In the experiments, we evaluated the framework with three supervised machine learning methods—SVM, RF, and BN—and benchmarked it against other malicious URL detection methods such as PDRCNN, the Li method, and URLNet, in terms of accuracy and precision metrics. The results demonstrated that our malicious URL detection framework outperforms others, achieving an accuracy of 98.95% and precision of 98.60%. The key factor contributing to this superior performance is the leveraging of priority coefficients assigned to classes and features, which lend greater weight, along with the implementation of a feature evaluation method systematically assessing the output of each feature.

In future work, there will be a concerted effort to enhance the proposed framework by delving deeper into dynamic feature classification and amalgamating it with current static features. Specifically, we plan to delve deeper into the behavior of dynamic features by executing them in an isolated environment, allowing for a more comprehensive understanding of their effectiveness in detecting malicious URLs. This approach will involve closely monitoring the behavior of dynamic features and analyzing their performance across various scenarios. For instance, features such as JavaScript execution behavior, URL redirection patterns, network traffic analysis, and content analysis responses will be scrutinized to determine their efficacy in flagging potential threats.

Moreover, the study will aim to optimize detection times by employing optimization techniques to streamline the detection process. These techniques may encompass algorithmic enhancements or parallel processing. By optimizing the detection pipeline, we aim to significantly reduce the time it takes to detect malicious URLs without compromising accuracy. Furthermore, we plan to provide a fair and rigorous experimental setup, ensuring a robust comparison of existing malicious URL detection methods in terms of detection time. By addressing these aspects, the future work aims to provide concrete strategies and preliminary results that validate the efficacy of the proposed enhancements, thereby bolstering the framework's robustness and practical applicability in combating evolving cyber threats.

REFERENCES

[1] T. Manyumwa, P. F. Chapita, H. Wu, and S. Ji, "Towards fighting cybercrime: Malicious URL attack type detection using multiclass classification," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 1813–1822.

[2] M. Alshehri, A. Abugabah, A. Algarni, and S. Almotairi, "Character-level word encoding deep learning model for combating cyber threats in phishing URL detection," *Comput. Electr. Eng.*, vol. 100, May 2022, Art. no. 107868.

[3] (2022). *Making the World's Information Safely Accessible*. [Online]. Available: https://safebrowsing.google.com/

[4] D. R. Patil and J. B. Patil, "Feature-based malicious URL and attack type detection using multi-class classification," *Int. J. Inf. Secur.*, vol. 10, no. 2, pp. 141–162, 2018.

[5] F. O. Catak, K. Sahinbas, and V. Dörtkardeş, "Malicious URL detection using machine learning," in *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems*. Hershey, PA, USA: IGI Global, 2021, pp. 160–180.

[6] E. Benavides, W. Fuertes, S. Sanchez, and M. Sanchez, "Classification of phishing attack solutions by employing deep learning techniques: A systematic literature review," in *Developments and Advances in Defense and Security* (Smart Innovation, Systems and Technologies), vol. 152, Á. Rocha and R. Pereira, Eds. Singapore: Springer, 2020, doi: 10.1007/978-981-13-9155-2_5.

[7] K. Krombholz, P. Frühwirt, P. Kieseberg, I. Kapsalis, M. Huber, and E. Weippl, "QR code security: A survey of attacks and challenges for usable security," in *Proc. Int. Conf. Hum. Aspects Inf. Secur., Privacy, Trust*. Cham, Switzerland: Springer, 2014, pp. 79–90.

[8] C. D. Xuan, H. Dinh, and T. Victor, "Malicious URL detection based on machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 1, pp. 148–153, 2020.

[9] Y. Liang, Q. Wang, K. Xiong, X. Zheng, Z. Yu, and D. Zeng, "Robust detection of malicious URLs with self-paced wide & deep learning," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 2, pp. 717–730, Mar. 2022.

[10] F. Sadique, R. Kaul, S. Badsha, and S. Sengupta, "An automated framework for real-time phishing URL detection," in *Proc. 10th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2020, pp. 335–341.

[11] D. Huang, K. Xu, and J. Pei, "Malicious URL detection by dynamically mining patterns without pre-defined elements," *World Wide Web*, vol. 17, no. 6, pp. 1375–1394, Nov. 2014.

[12] M. Alsaedi, F. Ghaleb, F. Saeed, J. Ahmad, and M. Alasli, "Cyber threat intelligence-based malicious URL detection model using ensemble learning," *Sensors*, vol. 22, no. 9, p. 3373, Apr. 2022.

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

[14] A. S. Rafsanjani, N. B. Kamaruddin, H. M. Rusli, and M. Dabbagh, "QsecR: Secure QR code scanner according to a novel malicious URL detection framework," *IEEE Access*, vol. 11, pp. 92523–92539, 2023.

[15] A. S. Rafsanjani, N. Kamaruddin, N. N. A. Sjariff, N. Firdaus, N. Maarop, and H. M. Rusli, "A evaluating security and privacy features of quick response code scanners: A comparative study," *Open Int. J. Informat.*, vol. 10, no. 2, pp. 197–207, 2022.

[16] J. Yuan, Y. Liu, and L. Yu, "A novel approach for malicious URL detection based on the joint model," *Secur. Commun. Netw.*, vol. 2021, pp. 1–12, Dec. 2021.

[17] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," 2018, *arXiv:1802.03162*.
[18] M. Akiyama, T. Yagi, and M. Itoh, "Searching structural neighborhood of malicious URLs to improve blacklisting," in *Proc. IEEE/IPSJ Int. Symp. Appl. Internet*, Jul. 2011, pp. 1–10.
[19] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "PhishNet: Predictive blacklisting to detect phishing attacks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–5.
[20] M. Akiyama, T. Yagi, and T. Hariu, "Improved blacklisting: Inspecting the structural neighborhood of malicious URLs," *IT Prof.*, vol. 15, no. 4, pp. 50–56, Jul. 2013.
[21] R. Bharadwaj, A. Bhatia, L. D. Chhibbar, K. Tiwari, and A. Agrawal, "Is this URL safe: Detection of malicious URLs using global vector for word representation," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2022, pp. 486–491.
[22] Y. Fukushima, Y. Hori, and K. Sakurai, "Proactive blacklisting for malicious Web sites by reputation evaluation based on domain and IP address registration," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Nov. 2011, pp. 352–361.
[23] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL detection using machine learning: A survey," 2017, *arXiv:1701.07179*.
[24] W. Wang, F. Zhang, X. Luo, and S. Zhang, "PDRCNN: Precise phishing detection with recurrent convolutional neural networks," *Secur. Commun. Netw.*, vol. 2019, pp. 1–15, Oct. 2019.
[25] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A content-based approach to detecting phishing Web sites," in *Proc. 16th Int. Conf. World Wide Web*, May 2007, pp. 639–648.
[26] R. Almeida and C. Westphall, "Heuristic phishing detection and URL checking methodology based on scraping and web crawling," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Nov. 2020, pp. 1–6.
[27] C. M. R. D. Silva, E. L. Feitosa, and V. C. Garcia, "Heuristic-based strategy for phishing prediction: A survey of URL-based approach," *Comput. Secur.*, vol. 88, Jan. 2020, Art. no. 101613.
[28] G. Sonowal and K. S. Kuppusamy, "PhiDMA—A phishing detection model with multi-filter approach," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 32, no. 1, pp. 99–112, Jan. 2020.
[29] G. Ramesh, I. Krishnamurthi, and K. S. S. Kumar, "An efficacious method for detecting phishing webpages through target domain identification," *Decis. Support Syst.*, vol. 61, pp. 12–22, May 2014.
[30] D. K. Mondal, B. C. Singh, H. Hu, S. Biswas, Z. Alom, and M. A. Azim, "SeizeMaliciousURL: A novel learning approach to detect malicious URLs," *J. Inf. Secur. Appl.*, vol. 62, Nov. 2021, Art. no. 102967.
[31] T. Li, G. Kou, and Y. Peng, "Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods," *Inf. Syst.*, vol. 91, Jul. 2020, Art. no. 101494.
[32] S. Singhal, U. Chawla, and R. Shorey, "Machine learning & concept drift based approach for malicious website detection," in *Proc. Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2020, pp. 582–585.
[33] R. Naresh, A. Gupta, and S. Giri, "Malicious URL detection system using combined SYM and logistic regression model," *Int. J. Adv. Res. Eng. Technol.*, vol. 11, no. 4, pp. 1–7, 2020.
[34] C. Ding, "Automatic detection of malicious URLs using fine-tuned classification model," in *Proc. 5th Int. Conf. Inf. Sci., Comput. Technol. Transp. (ISCTT)*, Nov. 2020, pp. 302–320.
[35] M. Al-Janabi, E. D. Quincey, and P. Andras, "Using supervised machine learning algorithms to detect suspicious URLs in online social networks," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Jul. 2017, pp. 1104–1111.
[36] K. Ramesh, M. A. Bennet, J. Veerappan, and P. Renjith, "Performance metric system for malicious URL data using revised random forest algorithm," in *Proc. 5th Int. Conf. Comput. Methodolog. Commun. (ICCMC)*, Apr. 2021, pp. 1188–1191.
[37] B. Janet and R. J. A. Kumar, "Malicious URL detection: A comparative study," in *Proc. Int. Conf. Artif. Intell. Smart Syst. (ICAIS)*, Mar. 2021, pp. 1147–1151.
[38] Y. Kumar and B. Subba, "A lightweight machine learning based security framework for detecting phishing attacks," in *Proc. Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2021, pp. 184–188.
[39] J. Saxe and K. Berlin, "EXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys," 2017, *arXiv:1702.08568*.
[40] S.-J. Bu and S.-B. Cho, "Deep character-level anomaly detection based on a convolutional autoencoder for zero-day phishing URL detection," *Electronics*, vol. 10, no. 12, p. 1492, Jun. 2021.
[41] X. Xiao, D. Zhang, G. Hu, Y. Jiang, and S. Xia, "CNN–MHSA: A convolutional neural network and multi-head self-attention combined approach for detecting phishing websites," *Neural Netw.*, vol. 125, pp. 303–312, May 2020.
[42] T. T. T. Pham, V. N. Hoang, and T. N. Ha, "Exploring efficiency of character-level convolution neuron network and long short term memory on malicious URL detection," in *Proc. 7th Int. Conf. Netw., Commun. Comput.*, Dec. 2018, pp. 82–86.
[43] W. Yang, W. Zuo, and B. Cui, "Detecting malicious URLs via a keyword-based convolutional Gated-Recurrent-Unit neural network," *IEEE Access*, vol. 7, pp. 29891–29900, 2019.
[44] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, "Classifying phishing URLs using recurrent neural networks," in *Proc. APWG Symp. Electron. Crime Res. (eCrime)*, Apr. 2017, pp. 1–8.
[45] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2091–2121, 4th Quart., 2013.
[46] N. A. ALfouzan and N. C, "A systematic approach for malware URL recognition," in *Proc. 2nd Int. Conf. Comput. Inf. Technol. (ICCIT)*, Jan. 2022, pp. 325–329.
[47] K. H. Park, H. M. Song, J. D. Yoo, S.-Y. Hong, B. Cho, K. Kim, and H. K. Kim, "Unsupervised malicious domain detection with less labeling effort," *Comput. Secur.*, vol. 116, May 2022, Art. no. 102662.
[48] S. Afzal, M. Asim, A. R. Javed, M. O. Beg, and T. Baker, "URLdeepDetect: A deep learning approach for detecting malicious URLs using semantic vector models," *J. Netw. Syst. Manage.*, vol. 29, no. 3, pp. 1–27, Jul. 2021.
[49] C.-M. Wu, L. Min, Y. Li, and X. Zou, and B. Qiang, "Malicious website detection based on urls static features," in *Proc. Int. Conf. Modeling, Simulation Optim.*, 2018, pp. 307–313.
[50] Z. Wang, S. Li, B. Wang, X. Ren, and T. Yang, "A malicious URL detection model based on convolutional neural network," in *Proc. Int. Symp. Secur. Privacy Social Netw. Big Data* Cham, Switzerland: Springer, 2020, pp. 34–40.
[51] L. Xu, Z. Zhan, S. Xu, and K. Ye, "Cross-layer detection of malicious websites," in *Proc. 3rd ACM Conf. Data Appl. Secur. privacy*, Feb. 2013, pp. 141–152.
[52] A. Rakotoasimbahoaka, I. Randria, and N. R. Razafindrakoto, "Malicious URL detection by combining machine learning and deep learning models," in *Emerging Trends in Artificial Intelligence for Internet of Things*, vol. 1. Department of Embedded Technology SENSE, Vellore Institute of Technology.
[53] V. K. Nadar, B. Patel, V. Devmane, and U. Bhave, "Detection of phishing websites using machine learning approach," in *Proc. 2nd Global Conf. Advancement Technol. (GCAT)*, Oct. 2021, pp. 1–8.
[54] L. Zeltser. (2021). *Free Online Tools for Looking Up Potentially Malicious Websites*. [Online]. Available: https://zeltser.com/lookup-malicious-websites/
[55] D. Ulevitch. *PhishTank*. Cisco Talos Intell. Group (Talos). [Online]. Available: https://www.phishtank.com/
[56] D. Ulevitch. (2023). PhishTank. Cisco Talos Intelligence Group (Talos). [Online]. Available: https://www.phishtank.com/
[57] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, "Detecting malicious URLs using lexical analysis," in *Proc. Int. Conf. Netw. Syst. Secur.* Cham, Switzerland: Springer, 2016, pp. 467–482.
[58] M. Darling, G. Heileman, G. Gressel, A. Ashok, and P. Poornachandran, "A lexical approach for classifying malicious URLs," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Jul. 2015, pp. 195–202.
[59] A. Joshi, L. Lloyd, P. Westin, and S. Seethapathy, "Using lexical features for malicious URL detection—A machine learning approach," 2019, *arXiv:1910.06277*.
[60] H. M. Junaid Khan, Q. Niyaz, V. K. Devabhaktuni, S. Guo, and U. Shaikh, "Identifying generic features for malicious URL detection system," in *Proc. IEEE 10th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2019, pp. 347–352.
[61] M. Darling, G. Heileman, G. Gressel, A. Ashok, and P. Poornachandran, "A lexical approach for classifying malicious URLs," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Jul. 2015, pp. 195–202.

[62] A. Aljofey, Q. Jiang, Q. Qu, M. Huang, and J.-P. Niyigena, "An effective phishing detection model based on character level convolutional neural network from URL," *Electronics*, vol. 9, no. 9, p. 1514, Sep. 2020.
[63] T. Shibahara, Y. Takata, M. Akiyama, T. Yagi, and T. Yada, "Detecting malicious websites by integrating malicious, benign, and compromised redirection subgraph similarities," in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Jul. 2017, pp. 655–664.
[64] T. Shibahara, K. Yamanishi, Y. Takata, D. Chiba, M. Akiyama, T. Yagi, Y. Ohsita, and M. Murata, "Malicious URL sequence detection using event de-noising convolutional neural network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7.
[65] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL names say it all," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 191–195.
[66] H.-K. Pao, Y.-L. Chou, and Y.-J. Lee, "Malicious URL detection based on Kolmogorov complexity estimation," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, vol. 1, Dec. 2012, pp. 380–387.
[67] S. Marchal, J. François, R. State, and T. Engel, "PhishScore: Hacking phishers' minds," in *Proc. 10th Int. Conf. Netw. Service Manage. (CNSM) Workshop*, Nov. 2014, pp. 46–54.
[68] R. Patgiri, H. Katari, R. Kumar, and D. Sharma, "Empirical study on malicious URL detection using machine learning," in *Proc. 15th Int. Conf. Distrib. Comput. Internet Technol. (ICDCIT)*, Bhubaneswar, India. Cham, Switzerland: Springer, Jan. 2019, pp. 380–388.
[69] M. Chatterjee and A.-S. Namin, "Detecting phishing websites through deep reinforcement learning," in *Proc. IEEE 43rd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jul. 2019, pp. 227–232.
[70] K. Althobaiti, G. Rummani, and K. Vaniea, "A review of human- and computer-facing URL phishing features," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, Jun. 2019, pp. 182–191.
[71] G. Palaniappan, S. Sangeetha, B. Rajendran, Sanjay, S. Goyal, and B. S. Bindhumadhava, "Malicious domain detection using machine learning on domain name features, host-based features and web-based features," *Proc. Comput. Sci.*, vol. 171, pp. 654–661, Jan. 2020.
[72] R. Madhubala, N. Rajesh, L. Shaheetha, and N. Arulkumar, "Survey on malicious URL detection techniques," in *Proc. 6th Int. Conf. Trends Electron. Informat. (ICOEI)*, Apr. 2022, pp. 778–781.
[73] Y. Li, Z. Yang, X. Chen, H. Yuan, and W. Liu, "A stacking model using URL and HTML features for phishing webpage detection," *Future Gener. Comput. Syst.*, vol. 94, pp. 27–39, May 2019.
[74] S. Kim, J. Kim, S. Nam, and D. Kim, "WebMon: ML- and YARA-based malicious webpage detection," *Comput. Netw.*, vol. 137, pp. 119–131, Jun. 2018.
[75] V. Arceri and I. Mastroeni, "Analyzing dynamic code: A sound abstract interpreter for evil eval," *ACM Trans. Privacy Secur.*, vol. 24, no. 2, pp. 1–38, May 2021.
[76] A. Lemay and S. P. Leblanc, "Is eval () evil : A study of Javascript in PDF malware," in *Proc. 13th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2018, pp. 1–10.
[77] S. Kumi, C. Lim, and S.-G. Lee, "Malicious URL detection based on associative classification," *Entropy*, vol. 23, no. 2, p. 182, Jan. 2021.
[78] S. Morishige, S. Haruta, H. Asahina, and I. Sasase, "Obfuscated malicious Javascript detection scheme using the feature based on divided URL," in *Proc. 23rd Asia–Pacific Conf. Commun. (APCC)*, Dec. 2017, pp. 1–6.
[79] J. McGahagan, D. Bhansali, C. Pinto-Coelho, and M. Cukier, "A comprehensive evaluation of webpage content features for detecting malicious websites," in *Proc. 9th Latin-American Symp. Dependable Comput. (LADC)*, Nov. 2019, pp. 1–10.
[80] S. Ndichu, S. Kim, and S. Ozawa, "Deobfuscation, unpacking, and decoding of obfuscated malicious Javascript for machine learning models detection performance improvement," *CAAI Trans. Intell. Technol.*, vol. 5, no. 3, pp. 184–192, Sep. 2020.
[81] M. Talal, A. A. Zaidan, B. B. Zaidan, O. S. Albahri, M. A. Alsalem, A. S. Albahri, A. H. Alamoodi, M. L. M. Kiah, F. M. Jumaah, and M. Alaa, "Comprehensive review and analysis of anti-malware apps for smartphones," *Telecommun. Syst.*, vol. 72, no. 2, pp. 285–337, Oct. 2019.
[82] Y. Mourtaji, M. Bouhorma, D. Alghazzawi, G. Aldabbagh, and A. Alghamdi, "Hybrid rule-based solution for phishing URL detection using convolutional neural network," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–24, Sep. 2021.
[83] F. Maggi, A. Frossi, S. Zanero, G. Stringhini, B. Stone-Gross, C. Kruegel, and G. Vigna, "Two years of short URLs internet measurement: Security threats and countermeasures," in *Proc. 22nd Int. Conf. World Wide Web*, May 2013, pp. 861–872.
[84] N. Gupta, A. Aggarwal, and P. Kumaraguru, "Bit.Ly/malicious: Deep dive into short URL based e-crime detection," in *Proc. APWG Symp. Electron. Crime Res. (eCrime)*, Sep. 2014, pp. 14–24.
[85] S. Selvaganapathy, M. Nivaashini, and H. Natarajan, "Deep belief network based detection and categorization of malicious URLs," *Inf. Secur. J., Global Perspective*, vol. 27, no. 3, pp. 145–161, May 2018.
[86] M. Sameen, K. Han, and S. O. Hwang, "PhishHaven—An efficient real-time AI phishing URLs detection system," *IEEE Access*, vol. 8, pp. 83425–83443, 2020.
[87] S. Y. Yerima and M. K. Alzaylaee, "High accuracy phishing detection based on convolutional neural networks," in *Proc. 3rd Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, Mar. 2020, pp. 1–6.
[88] N. Al-Milli and B. H. Hammo, "A convolutional neural network model to detect illegitimate URLs," in *Proc. 11th Int. Conf. Inf. Commun. Syst. (ICICS)*, Apr. 2020, pp. 220–225.
[89] C. Rupa, G. Srivastava, S. Bhattacharya, P. Reddy, and T. R. Gadekallu, "A machine learning driven threat intelligence system for malicious URL detection," presented at the 16th Int. Conf. Availability, Rel. Secur., Aug. 2021, doi: 10.1145/3465481.3470029.
[90] G. Wejinya and S. Bhatia, "Machine learning for malicious URL detection," in *ICT Systems and Sustainability*. Cham, Switzerland: Springer, 2021, pp. 463–472.
[91] X. Wan, P. Li, Y. Wang, W. Wei, and L. Xiao, "Reinforcement learning based accurate detection of malicious URLs with multi-feature analysis," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Jul. 2021, pp. 17–22.
[92] S. D. V. Prasad and K. R. Rao, "A novel framework for malicious URL detection using hybrid model," *Turkish J. Comput. Math. Educ. (TURCOMAT)*, vol. 12, no. 7, pp. 68–76, 2021.
[93] S. He, B. Li, H. Peng, J. Xin, and E. Zhang, "An effective cost-sensitive XGBoost method for malicious URLs detection in imbalanced dataset," *IEEE Access*, vol. 9, pp. 93089–93096, 2021.
[94] M. Chatterjee and A. S. Namin, "Deep reinforcement learning for detecting malicious websites," 2019, *arXiv:1905.09207*.
[95] G. Tan, P. Zhang, Q. Liu, X. Liu, C. Zhu, and F. Dou, "Adaptive malicious URL detection: Learning in the presence of concept drifts," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2018, pp. 737–743.
[96] I. N. V. D. Naveen, K. Manamohana, and R. Verma, "Detection of malicious URLs using machine learning techniques," *Int. J. Innov. Technol. Exploring Eng.*, vol. 8, no. 4S2, pp. 389–393, 2019.
[97] V. Vundavalli, F. Barsha, M. Masum, H. Shahriar, and H. Haddad, "Malicious URL detection using supervised machine learning techniques," in *Proc. 13th Int. Conf. Secur. Inf. Netw.*, Nov. 2020, pp. 1–6.
[98] P. Mutchler, A. Doupe, J. Mitchell, C. Kruegel, and G. Vigna, "A large-scale study of mobile web app security," in *Proc. Mobile Secur. Technol. Workshop (MoST)*, vol. 50, 2015, pp. 1–11.
[99] (2023). *WhoXy*. [Online]. Available: https://www.whoxy.com/
[100] (2023). *OpenPageRank*. [Online]. Available: https://www.domcop.com/openpagerank/
[101] A. S. Rafsanjani. (2023). *Enhancing-Malicious-URL-Detection*. [Online]. Available: https://github.com/ahmadsa63/Enhancing-Malicious-URL-Detection
[102] (2023). *URLhaus*. [Online]. Available: https://www.urlhaus.abuse.ch

AHMAD SAHBAN RAFSANJANI (Member, IEEE) received the master's degree in information security and the Ph.D. degree in network security from Universiti Teknologi Malaysia (UTM), Malaysia. He is currently a Program Leader and a Lecturer with the Department of Computing and Information Systems, School of Engineering and Technology, Sunway University, Malaysia. He has published several research papers in journals and conference proceedings. His research interests include network security, data hiding, cryptography, the IoT security, and malware analysis. He is intending to explore his research interests in deep learning and machine learning. He is also a member of the IEEE Computer Society.

NORSHALIZA BINTI KAMARUDDIN (Member, IEEE) received the B.S. degree in information technology from Universiti Utara Malaysia, and the M.S. degree in computer science and the Ph.D. degree in image processing from the University of Malaya, in 2003 and 2016, respectively. From 2001 to 2018, she was a Lecturer at a private university in Malaysia before she joined Universiti Technologi Malaysia, as a Senior Lecturer, in 2019. Her research interests include image processing, machine learning, and artificial intelligence. She is also actively doing research on mental health issues based on text analysis and real-time series data with machine learning techniques.

MEHRAN BEHJATI received the B.Sc. degree in electrical and electronic engineering from the Azad University of Iran, in 2009, and the M.Sc. and Ph.D. degrees in communication and computer engineering from UKM, in 2013 and 2017, respectively. He was a Postdoctoral Researcher with UKM and a Researcher with Iran's National Elites Foundation. He is currently a Lecturer with Sunway University. His research interests include aerial wireless communications, wireless networks, the IoT, edge intelligence, federated machine learning, and embedded machine learning.

SAAD ASLAM received the Ph.D. degree in electrical and electronic engineering from Massey University, New Zealand. He is currently the Head of the Department with the School of Engineering and Technology, Sunway University, Malaysia. He has over 12 years of experience in academia blended with industry exposure. His current research interests include exploiting machine learning for optimizing wireless networks, D2D communication, clustering algorithms, distributed systems, and game theory optimization.

AALIYA SARFARAZ (Member, IEEE) received the master's degree in computer science from the University of Seoul, South Korea, in 2014, and the Ph.D. degree from the University of New South Wales, Australia, in 2023. Since April 2023, she has been with Sunway University, Malaysia, where she is currently a Lecturer. Her research interests include blockchain, supply chain, data privacy, and security, including the adoption and use of these technologies.

ANGELA AMPHAWAN (Senior Member, IEEE) received the Ph.D. degree in optical communications from the University of Oxford, U.K. She currently leads the Smart Photonics Research Laboratory, Sunway University. Before joining Sunway University, she was the Deputy Vice Chancellor of the University Malaysia of Computer Science and Engineering. Prior to this, she was Director of the Optical Technology Research Laboratory at Universiti Utara Malaysia. Earlier on, she was awarded the prestigious Fulbright Award to work on optical devices and networks at the Research Laboratory of Electronics and the MIT Media Laboratory, Massachusetts Institute of Technology, USA. Earlier in her career, she lectured at Multimedia University and was a Computing Officer at the University of Oxford. She is currently on the National 5G Task Force for development of 5G infrastructure. Her research has been funded by the Fulbright Foundation, Telekom Malaysia, and the Ministry of Higher Education. She also serves on the IEEE Joint Sensor and Nanotechnology Councils. She has served as Co-chair, technical program committee member, and international scientific committee member for numerous international conferences. In addition, she is on the editorial board of the APL Photonics under the American Institute of Physics and several other international journals. She has given keynote addresses at several Fulbright and IEEE events. She has won the international academic award, several excellent service awards, teaching awards, best paper awards, and exhibition medals.