Machine Learning-Based Phishing Detection Using URL Features: A Comprehensive Review

Asif Uz Zaman Asif[1(✉)], Hossein Shirazi[2], and Indrakshi Ray[1]

[1] Colorado State University, Fort Collins, CO 80523, USA
{asif09,indrakshi.ray}@colostate.edu
[2] San Diego State University, San Diego, CA 92182, USA
hshirazi@sdsu.edu

Abstract. Phishing is a social engineering attack in which an attacker sends a fraudulent message to a user in the hope of obtaining sensitive confidential information. Machine learning appears to be a promising technique for phishing detection. Typically, website content and Unified Resource Locator (*URL*) based features are used. However, gathering website content features requires visiting malicious sites, and preparing the data is labor-intensive. Towards this end, researchers are investigating if *URL*-only information can be used for phishing detection. This approach is lightweight and can be installed at the client's end, they do not require data collection from malicious sites and can identify zero-day attacks. We conduct a systematic literature review on *URL*-based phishing detection. We selected recent papers (2018 –) or if they had a high citation count (50+ in Google Scholar) that appeared in top conferences and journals in cybersecurity. This survey will provide researchers and practitioners with information on the current state of research on *URL*-based website phishing attack detection methodologies. In this survey, we have seen that even though there is a lack of a centralized dataset, algorithms like Random Forest, and Long Short-Term Memory with appropriate lexical features can detect phishing *URL*s effectively.

Keywords: Phishing · social engineering · URL-based · survey · cybersecurity · machine learning · feature extraction · data repository

1 Introduction

Phishing is a social engineering attack intended to deceive the victim and attempt to obtain sensitive data with the ultimate goal of stealing the victim's valued possessions. Although phishing has persisted since the mid 90's [22], such attacks have escalated in recent times due to the increased use of online activities. According to reports provided by the Anti-Phishing Working Group (*APWG*), more than a million phishing attacks were recorded in the First Quarter (*Q*1) of 2022. With 23.6% of all attacks, the financial sector was the one most commonly

S. Dolev and B. Schieber (Eds.): SSS 2023, LNCS 14310, pp. 481–497, 2023.
https://doi.org/10.1007/978-3-031-44274-2_36

targeted by phishing in $Q1$ [9]. Attackers are constantly adapting their strategies which makes phishing detection particularly hard.

Typically, the attacker attempts to redirect users to a phishing site using a malicious *URL*. *URL* manipulation is often the first stage in building phishing websites. Attackers work on various means through which a malicious *URL* can be represented. Since the representation of *URL* keeps on changing, even professionals cannot correctly identify phishing *URL*s. Past approaches for phishing detection use signature-based and rule-based mechanisms. However, these approaches are ineffective against zero-day attacks which are referred to as vulnerabilities that are exploited as soon as they are discovered or even before anyone is aware of them.

Machine learning researchers have used *URL*-based features and content-based features (website images, HTML, and JavaScript code) to distinguish phishing from genuine websites. In this survey, we focussed only on *URL*-based features. A number of reasons motivated this choice. First, machine learning algorithms focusing on lexical characteristics of *URL* are lightweight and more efficient than those using both content-based and *URL*-based features. Second, this approach can thwart phishing attacks at the very initial stage when a user stumbles into a potentially harmful *URL* or phishing campaign. Third, the use of *URL* only features does not require one to visit malicious websites to download content-based features. Visiting malicious websites may cause malware to be loaded which may lead to future attacks. Fourth, *URL*-based classifiers can be installed on clients' mobile devices as they are lightweight – the clients' browsing habits are abstracted from the servers – making them more privacy-preserving.

In this survey, we produced a comprehensive review of the research on *URL*-based phishing detectors using machine learning. We looked into the feature extraction procedure, the datasets, the algorithms, the experimental design, and the results for each work. We looked at the crucial steps in creating a phishing detector, and after analyzing several different approaches, we gave our conclusions regarding the features that may be used, the ideal algorithms, the dataset's current state, and some recommendations. We used two criteria for the paper selection process in this survey. First, we looked into the articles on *URL*-based phishing detection that has been published in the past five years (2018 onwards) in journals having an impact factor of 2.0 or higher and in conferences from Tier (1, 2, and 3)[1]. We also examined papers having at least 50 citations in Google Scholar. We found 26 papers satisfying our criteria.

The rest of the paper is organized as follows. The anatomy of an *URL* is explained in Sect. 2. Feature extraction techniques used by researchers are illustrated in Sect. 3. Section 4 of the paper discusses machine learning algorithms that are used. The numerous data sources that are used by researchers are covered in Sect. 5. Section 6 contains the experimental results and presents an overview of the survey findings. Finally, Sect. 7 concludes the paper.

[1] We used the following sources for conference rankings: https://people.engr.tamu.edu/guofei/sec_conf_stat.htm.

2 Malicious URLs

The anatomy of an *URL* is critical for understanding how attackers manipulate it for launching phishing attacks. An attacker may manipulate any segment of the URL to create a malicious link that can be used to launch a phishing attack.

The *URL* of a website is made up of three major components: scheme, domain, and path. The scheme specifies the protocol used by the *URL*. The domain name identifies a specific website on the internet. The paths are then used to identify the specific resource that a web client is attempting to access.

An attacker often uses social engineering to trick a victim so that the malicious *URL* goes undetected. To accomplish this goal, the attacker will employ various obfuscation techniques. In this case, the attacker may obfuscate the hostname with the IP address, and the phished domain name is placed in the path, for example, http://159.203.6.191/servicepaypal/. Furthermore, an attacker can obfuscate a domain name that is unknown or misspelled, such as http://paypa1.com, which is misspelled and unrelated to the actual domain.

The most important details in the above *URL*s are the techniques used to redirect a victim to a malicious site and entice them to provide sensitive information to the attacker. PayPal is incorporated in the malicious *URL* in all of these cases, creating a sense of urgency for the victim and making them vulnerable to judgemental errors. To prevent *URL*-based website phishing attacks, an automated approach is needed.

3 Feature Extraction

Manual feature extraction is required for *URL*-based website phishing attack detection when using machine-learning; this is generally known as using handcrafted features. However, when a deep learning approach is employed, the feature extraction procedure is done automatically and does not require domain expertise.

Researchers have often used *URL* lexical features alongside domain features to create a better ML model. Table 1 provides a list of features used by the algorithms.

URL Lexical Features: Information that is directly connected to a website's *URL* components is referred to as *URL* lexical features. *URL*-based characteristics include lexical features that keep track of the attributes of the *URL*, such as its length, domain, and subdomain. Popular lexical elements of *URL*s include the use of the Hypertext Transfer Protocol Secure (*HTTPS*) protocol, special characters and their counts (for a dot, a hyphen, and at symbol), numerical characters, and IP addresses.

Domain Features: Information about the domain on which a website is hosted is included in the domain features. The age of the domain and free hosting is generally included in this feature set as it is a crucial signal for distinguishing between a legitimate website and a phishing website. Typically, a newly hosted website serves as a warning sign for a phishing site.

Word Features: These prevent a typical user from becoming suspicious. Attackers utilize words like secure, support, safe, and authentic within the *URL* itself to make it appear real. To make the *URL* appear legitimate, they also include well-known brand names, such as PayPal and Amazon inside the *URL*.

Character Features: Phishing sites often use suspicious characters. The length of the *URL*, the usage of uncommon letters or symbols, and misspelled words are a few examples of character-based indicators that are frequently used to identify phishing websites.

Search Index Based Features: These include website page ranking, Google index, and website traffic information. The average lifespan of a phishing website is quite short, and it typically produces no statistics.

Table 1. Combination of features used in the literature

Ref.	[19]	[37]	[11]	[24]	[21]	[2]	[35]	[13]	[6]	[34]	[14]	[17]	[42]	[40]	[7]	[39]	[45]	[41]	[44]	[4]	[3]	[5]	[8]	[29]	[12]	[20]
Automatic Features													✓	✓							✓					
Hand-Crafted Features	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓		✓				✓	✓	✓		✓
URL Lexical Features	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓						✓	✓	✓	✓	✓
Domain Features	✓	✓	✓		✓	✓					✓				✓								✓			
Word Features	✓	✓	✓		✓		✓	✓		✓							✓									
Character Features																	✓	✓	✓	✓						
Search Index Features						✓																				
Total Features	**17**	–	**46**	**51**	**14**	**35**	**104**	**12**	**42**	**93980**	**17**	–	–	–	**95+**	**87**	–	**9**	–	–	–	**111**	**30**	**30**	–	**48**

4 Algorithms

The parts that follow provide a description of the machine learning and deep learning algorithms used for *URL*-based phishing detection.

4.1 Classification Using Machine Learning

Logistic Regression (*LR*) is a common statistical machine-learning method for binary classification problems or for predicting an outcome with two possible values and this is specifically required for phishing detection because *URL* might either be legitimate or fraudulent [6,7,14,29,34,37,41]. *LR* can process a lot of *URL*s as it is a computationally efficient technique and can handle big datasets with high-dimensional feature spaces [7,29]. In order to select the most crucial aspects for phishing *URL* detection, feature selection can be done using *LR* models. In addition to increasing the model's effectiveness, this can decrease the input space's dimensionality and works with word-based features [37], character-based features [7] and bi-gram-based features which is, contiguous pairs of words [41]. Additionally, when given a balanced dataset, *LR* can learn the decision boundary that best discriminates between positive and negative samples without favoring either class [14,34]. However, in order to train the model, *LR* needs

labeled data. This can be a problem in phishing detection because acquiring labeled data can be challenging [6].

Decision Tree (*DT*) is a type of supervised machine learning method for classification and regression tasks. It works by iteratively segmenting the input data into subsets based on the values of the input attributes in order to discriminate between the various classes or forecast the target variable. *DT* is commonly used by researchers for phishing detection problems [5,6,13,20,29,34,35,40]. Phishing *URL*s frequently exhibit traits that set them apart from real *URL*s. *DT* algorithm learns to distinguish between legal and phishing *URL*s using these properties as input to the features needed to train the algorithm. For detecting *URL*-based phishing, *DT* is advantageous because it is a highly interpretable model that makes it possible for human specialists to determine the reasoning behind a choice. Given the large potential for feature density in *URL*s, the feature space is highly dimensional. Without suffering dimension problems, *DT* can handle this type of data [29]. Moreover, *DT* that use lexical features can produce a better result, it creates a set of rules based on lexical properties that are simple for human specialists to comprehend [34,35,40]. When working with massive datasets, decision trees offer outcomes with good performance [5,40]. However, overfitting is common in *DT*, especially in small or significantly unbalanced datasets. A model may as a result perform well on training data but badly on the newly collected information [13].

Random Forest (*RF*) is another machine learning method used for classification, regression, and feature selection tasks [4,6,7,11,13,20,24,29,34,35,39,40,42]. Because *RF* can manage large and complex datasets and has the capability to deal with noisy data it is well-adapted for *URL*-based phishing detection [40]. To make predictions, the ensemble learning method of *RF*, combines data from various decision trees, reducing the possibility of overfitting while improving the model's generalization capabilities [7,29,34,35,39]. A measure of feature importance can also be provided by *RF*, which means that this algorithm can be used to understand the key features that contribute to phishing detection, improving the algorithm's overall accuracy [11,24,29]. *RF* is better for the real-time detection of phishing *URL*s because it is computationally efficient and can be trained on big datasets rapidly as it requires minimal parameter tuning [11]. We also observed that using the lexical features of the *URL*, *RF* can produce good performance accuracy [35]. However, the *RF* algorithm may not work well with imbalanced datasets but it can be observed that on a balanced dataset, it gives better performance [4,39,42]. Another disadvantage of using *RF* is that the model produces better results at the cost of both training and prediction time [13].

Naive Bayes (*NB*) algorithm is a probabilistic algorithm used in machine learning for classification purposes which is based on Bayes' theorem, to identify *URL*-based website phishing [7,11,13,14,20,21,29,35,39–41]. *NB* can manage high-dimensional data, which means the algorithm can handle a large number of features in the *URL* [7]. *NB* is susceptible to the model's feature selection,

though the model's accuracy may suffer if essential features are excluded [41]. It can therefore work better on small feature sets with important features [29]. Additionally, it was discovered that applying only word-based features to *NB* does not yield better results [35]. The *NB* algorithm also has the benefit of learning the underlying patterns in the data with a small quantity of labeled training data, given how difficult it can be to acquire labeled data, this is especially helpful for *URL*-based phishing detection [39,40]. When there are an uneven amount of samples in each class, *NB* may not perform well. This could lead to a model that is biased in support of the dominant class [21]. On a balanced sample, however, this algorithm performance improves [11].

Gradient Boosting (*GB*) is a machine learning technique that creates a sequence of decision trees, each of which aims to fix the flaws of the one previous to it. The combined forecasts of all the trees result in the final prediction [7,20,29,34,39]. Since *GB* can be used to train models on huge datasets, it is especially suitable for large-scale phishing attack detection [39]. Additionally, the balanced dataset makes sure that the accuracy of the model is not biased towards one class over another and forces the model to equally understand the underlying patterns of the data for each class. As a result, the model becomes more accurate and generalizable [20,34,39]. Moreover, *GB* is effective when more attributes are considered [7,34] as well as on character-based features [29,39]. The model may be less accurate or may not perform well on new, untested data if the training data is biased or insufficient. However, on a balanced dataset, the algorithm performs better [20]. To ensure that the model is able to extract the most informative features from the data, *GB* necessitates thorough feature engineering. When dealing with complicated and diverse information like *URL*s, this can be a time-consuming and difficult operation [7,34].

Adaptive Boosting (*AdaBoost*) is a machine learning algorithm that is a member of the ensemble learning technique family. This approach for supervised learning can be applied to classification and regression tasks and is also used for *URL*-based website phishing detection [29,35,39,40]. As an ensemble approach it combines several weak learners to provide a final prediction, *AdaBoost* is a powerful algorithm that can be a viable choice for *URL*-based phishing detection [29]. *AdaBoost* can predict outcomes more precisely when it has access to a larger training dataset. The algorithm can produce predictions that are more accurate by better capturing the underlying relationships and patterns in the data [39,40]. However, *AdaBoost* may not be the best option for datasets with a lot of irrelevant or redundant features because it does not directly do the feature selection. This may lead to longer training times and poor results [35].

K-Nearest Neighbour ($K-NN$) is an algorithm where a prediction is made based on the labels of the k data points that are closest to an input data point in the training set. In the context of *URL*-based phishing detection, this means that the algorithm may compare a new *URL* to a list of known phishing and legitimate *URL*s and find the ones that are most similar to the new *URL* and thus are used for *URL*-based website phishing detection [2,4,6,20,29,34,35,39]. High-

dimensional feature vectors, such as those found in *URL*s, might be challenging to process. However, the $K-NN$ technique can efficiently detect similarities across *URL*s and is well-suited to high-dimensional data [39]. Even with imbalanced datasets, where the proportion of samples in one class is significantly higher than the other, the $K-NN$ approach can perform well [20,34]. Additionally, $K-NN$ works well with word-based features [2,34,35]. In $K-NN$ when producing predictions, an algorithm that has a bigger value of k will take into account more neighbors and improves performance [4]. However, the number of nearest neighbors taken into account or the distance measure utilized can have an impact on how well the $K-NN$ method performs. These hyperparameters may need a lot of effort to be tuned [29]. The $K-NN$ method is susceptible to adversarial attacks, in which a perpetrator creates *URL*s on purpose to avoid being detected by the system [6,34].

Support Vector Machine (*SVM*), a form of supervised learning algorithm used in classification and regression analysis, was commonly used by researchers [2,4,6,11,13,14,20,21,24,29,34,39,42]. *SVM* is good for detecting *URL*-based website phishing because it can handle high-dimensional data and identify intricate connections between features [39]. Numerous characteristics, including the lexical features of the *URL*, and the existence of specific keywords, can be used to detect phishing when analyzing *URL*s. These characteristics can be used by *SVM* to recognize trends in phishing *URL*s and separate them from real *URL*s. It can be observed that only using the lexical features of the *URL* does not yield good results [34]. However, hybrid features like a combination of text, image, and web page content work better for *SVM* [2]. Hence to achieve optimum performance, *SVM* requires fine-tuning of several parameters, *SVM*s additionally can require a lot of computational power, especially when working with big datasets [13]. This may slow down training and prediction times and necessitate the use of powerful hardware [4]. Moreover, the ratio of legitimate *URL*s to phishing *URL*s is very uneven, which can result in unbalanced data that will degrade the performance of *SVM* [11]. However, on a balanced dataset, *SVM* performs better [42]. Additionally, if there is a lack of training data, *SVM*'s accuracy is likely to decline [21].

4.2 Classification Using Deep Learning

Neural Network (*NN*) uses complex patterns and correlations between input features can be learned. By finding patterns that are suggestive of phishing attempts, *NN* can learn to differentiate between legitimate and phishing *URL*s in the context of *URL*-based phishing detection [5,7,20]. The ability of *NN* to acquire intricate patterns and connections between the characters in a sequence makes them effective for character-based characteristics [7]. Additionally, because the algorithm can learn from the data and produce predictions for each class with nearly equal importance, neural networks can perform well on balanced datasets [20]. However, to perform well, *NN* needs a lot of high-quality

training data. Especially in rapidly changing phishing contexts, collecting and identifying a sufficiently large and diverse array of *URL*s might be difficult [5].

Multi-Layer Perceptron (*MLP*) is another type of *NN* that has been found to be successful in *URL*-based phishing detection [13,14,39]. A class imbalance may significantly affect several other algorithms, however, because *MLP*s employ numerous hidden layers and may thus identify more complex patterns in the data, they are less prone to this problem [13,14]. However, it can be computationally expensive to train *MLP*s, especially for larger datasets or more intricate network designs. Long training periods may result from this, which may slow down the deployment of phishing detection systems [39].

Convolutional Neural Network (*CNN*) is a class of neural networks that are frequently employed in computer vision, but recently it has emerged to be a great tool for phishing detection [3,4,7,8,12,40–42,45]. When labeled training data is limited, *CNN*s can benefit from pre-trained models and transfer learning to enhance performance in detecting phishing *URL*s. *CNN* is capable of handling variations in the input data, including changes to the *URL*'s length and the existence of unexpected letters or symbols. This is because the pooling layers can downsample the feature maps to lessen the influence of variances, while the convolutional filters used in *CNN* can recognize patterns in various regions of the *CNN* [7,45]. Without manual feature engineering, *CNN* can automatically extract high-level features from the data that comes in. This is because the filters in the convolutional layers are trained to identify the most important data patterns [4,7,8,41]. Additionally, the *CNN* performs well on a balanced dataset [12,42]. It is possible to train more sophisticated *CNN* architectures that can recognize subtler patterns and correlations in the data with a larger dataset which can increase the model's capacity to correctly categorize new phishing samples [3,4,40]. However, if a *CNN* model fits the training data too closely and cannot generalize to new, untested data, the problem of overfitting arises. This can be prevented by using batch normalization and dropout techniques [3]. Additionally, *CNN*s can require a lot of processing power, particularly when employing deep structures with numerous layers, therefore, this can need a lot of computing power and hardware resources [4,8,41].

Recurrent Neural Network (*RNN*) are a type of neural network that excels at processing sequential data such as text or time series data. Because *URL*s may be represented as a sequence of characters, and because *RNN*s can learn to recognize patterns and characteristics in this sequence, they can be utilized for *URL*-based phishing detection [40,41]. Each character or characteristic in a *URL* is built sequentially, depending on the ones that came before. These sequential relationships can be observed by *RNN*s, which can then utilize to forecast whether a *URL* is genuine or phishing. *RNN* performance on balanced datasets depends on the particular task at hand as well as the network's architecture. For tasks requiring capturing long-term dependencies and temporal correlations between the input features, *RNN*s are especially well-suited [41]. To properly learn to recognize patterns in sequential data, such as *URL*s, *RNN*s need a lot of training data. This implies that *RNN*s may not be used efficiently for phish-

ing detection for enterprises with limited access to training data [40]. *RNN*s can be challenging to understand, particularly when working with massive data sets. *RNN*s can only be as effective as the training set that they are given. The *RNN* may struggle to accurately identify new and emerging dangers if the training data is not representative of all the threats that an organization might encounter [40].

Long Short-Term Memory (*LSTM*) is a specific type of *RNN* that was developed to address the issue of vanishing gradients that *RNN* frequently encounter and thus this algorithm is used by researchers for phishing detection [3,8,14,19,40,42,44,45]. The long-term dependencies and sequential patterns in *URL*s can be captured by *LSTM*, making it a good choice for *URL*-based website phishing detection. In order to detect tiny variations and patterns in phishing *URL*s that could otherwise go undetected, *LSTM* networks are particularly good at identifying sequential data and hence is a good choice for *URL*-based website phishing attack [3,14]. *LSTM*s can function well even when trained on minimal amounts of data [40]. These models are perfect for dealing with imbalanced datasets because they can find long-term correlations in the data. For identifying trends in the minority class, these dependencies can be very important [44,45]. Additionally, *LSTM* performs poorly for small datasets [8] but performs well on large datasets [19]. However, particularly when using vast data sets, training *LSTM* models can be computationally and memory-intensive [42]. Overfitting is a possibility with *LSTM* models, especially when working with limited data. When a model develops a proficiency at recognizing trends in training data but is unable to generalize that skill to fresh, untried data, overfitting occurs. This issue can be solved by using dropout in *LSTM* [3]. *LSTM* is complex in nature but the number of parameters needed for an *LSTM* model can be decreased by using pre-trained word embeddings like Word2Vec [19].

Bidirectional Long Short-Term Memory (*BiLSTM*) is a form of machine learning-based *RNN* architecture that is used to detect *URL*-based website phishing attacks [12,19,41,44]. *BiLSTM* is a form of neural network design that is effective at detecting data's sequential patterns. The capacity of *BiLSTM* algorithms is to examine the complete *URL* string in both ways, i.e., from the beginning to the end and from the end to the beginning, which makes them particularly useful for *URL*-based phishing detection [12,19,41]. Positive instances are often more scarce in imbalanced datasets than negative examples. *BiLSTM* may simultaneously learn from both phishing and legitimate instances, which may aid in improving its ability to distinguish between the two classes [19,44]. It can be costly computationally to train *BiLSTM* networks, especially if the input sequences are large and complex. The algorithm's capacity to scale for very big datasets may be constrained by this [19,44].

Gated Recurrent Units (*GRU*) is a sort of recurrent neural network that has been found to be useful for *URL*-based phishing detection [19,44]. *GRU*s are more memory-efficient and require fewer parameters than other recurrent neural network types. They are thus well suited for use in contexts with limited

resources, such as those seen in cloud-based systems or on mobile devices [19]. Additionally, on imbalance datasets, *GRU*s can perform well [19,44].

Bidirectional Gated Recurrent Units (*BiGRU*) is a *GRU* version that captures sequential dependencies in both forward and backward directions. *BiGRU* is useful for detecting *URL*-based phishing [19,44]. There are two layers in *BiGRU*, one of which moves the input sequence forward and the other which moves it backward. This gives the network the ability to record dependencies that happen both before and after a certain input feature, which is helpful for identifying intricate patterns in *URL*s. Additionally, on imbalance datasets, *BiGRU*s can perform well [19,44].

5 Dataset

The availability and quality of data are essential for the performance of machine learning-based phishing detection algorithms. To detect phishing attacks, algorithms need to be trained on large and diverse datasets. It is also important to keep the data up-to-date to reflect the latest trends and techniques used by attackers. This section will explore various data sources available for both phishing and legitimate websites and the detailed overview is shown in Table 3.

Phishing data sources are collections of *URL*s used to identify and block phishing websites and train a machine-learning model to detect new samples of phishing websites.

PhishTank.com is a community-based repository where contributors work to sanitize data and information pertaining to online phishing. The data is available in CSV or XML formats. In addition, an Application programming interface (*API*) is also available for research purposes [32].

OpenPhish.com is a live repository of phishing *URL*s, obtained from security researchers, government agencies, and other organizations. It uses automated and manual verification methods to ensure the sites are phishing sites [31].

Researchers also use websites like **MalwareUrl** [26], **MalwareDomain** [33], and **MalwareDomainList** [25] to collect malicious *URL*s. These community-driven tools are used to combat cyber threats.

Researchers collect legitimate *URL*s by compiling a list of popular websites, using web crawling sources, and online directories.

Common Crawl is a large-scale web crawl that is made up of petabytes of data that have been collected since 2008. It includes raw web page data, extracted metadata, and text extractions. This repository's material is maintained in Web ARChive (*WARC*) format, which contains *URL*-related data [15].

DMOZ.org was a large, open directory of the web, created and maintained by a volunteer editor community. It was one of the largest and most comprehensive directories on the web, with millions of websites listed and organized into thousands of categories. However, the project was discontinued in 2017 due to a decline in editor participation and the dominance of search engines [16].

Yandex.XML as a search engine provides API to submit queries and receive answers in XML format [43].

Alexa Web Crawl. Alexa is used to collect authentic *URL*s through the Internet Archive starting from 1996 [10].

In addition to data sources of phishing and legitimate *URL*s, there are existing ready-to-use datasets.

ISCXURL2016 is a dataset that includes both authentic and phishing *URL*s. There are 35,300 benign *URL*s in this dataset that was gathered from the top Alexa websites using the Heritrix web crawler. For phishing, this dataset also contains 12,000 *URL*s from the WEBSPAM-UK2007 dataset, 10,000 *URL*s from OpenPhish, 11,500 *URL*s from DNS-BH, and 45,450 *URL*s from Defacement *URL*s; a total of more than 78,000 *URL*s [38].

MillerSmiles Archives is a collection of phishing emails compiled by security researcher Paul Miller. The archives have not been updated since 2013 and the domain name millersmiles.co.uk is inactive [28].

Phishstorm is a dataset that contains both legitimate and phishing *URL*s. 48,009 legitimate *URL*s and 48,009 phishing *URL*s are included in this dataset's total of 96,018 *URL*s [27].

Ebbu2017 dataset comprises 36,400 valid *URL*s and 37,175 phishing *URL*s. The legitimate *URL*s were collected from Yandex.XML and the phishing data was collected from PhishTank [18].

UCI-15 dataset defined 30 different attributes for phishing *URL*s and extracted values of those attributes for each phishing URL. Data were collected mainly from PhishTank, MillerSmiles, and from Google search operator and the total number of instances in this dataset is 2456 [30].

UCI-16 dataset containing 1353 examples of both legitimate and phishing *URL*s, is also used by researchers. It comprises 10 distinct features. Phishing *URL* data are gathered from PhishTank and legitimate *URL*s as collected from Yahoo and using a crawler [1].

MDP-2018 dataset, which was downloaded between January and May 2015 and May and June 2017, has 48 features that were taken from 5000 legitimate *URL*s and 5000 phishing *URL*s. This dataset includes details on both legal and fraudulent *URL*s. Sources of fraudulent websites include PhishTank, OpenPhish, and legitimate websites like Alexa and Common Crawl [36].

6 Experimental Evaluations and Survey Findings

The findings reported in the phishing literature are important because they will aid in the identification of the algorithms that will be used to detect phishing in *URL*. Detailed information is provided in Table 2 where the best-performing algorithms are reported. Additionally, the metrics are briefly explained in the appendix.

Table 2. Performance evaluation by researchers with metrics: [Acc]uracy, [P]recision, [Rec]all, [F1]-Score. Studies [6,24,37] used other metrics.

Ref	Best Performing Algorithm	P	Rec	Acc	F1
[5]	*DT*			97.40	96.30
[39]	Gradient Tree Boosting (*GTB*)			97.42	
[29]	eXtreme Gradient Boosting (*XGBoost*)	95.78	96.77	96.71	96.27
[11]	*RF*	94.00	94.00	94.05	93.20
[35]	*RF*	97.00		97.98	
[13]	*RF*	97.40		99.29	98.22
[21]	*SVM*			91.28	
[42]	*CNN*	99.57	100.00	99.80	99.78
[8]	*CNN*	99.00	99.20	99.20	99.20
[4]	*CNN*	96.53	95.09	95.78	95.81
[41]	*CNN*	97.33	93.78	95.60	95.52
[45]	*CNN*			98.30	94.95
[7]	*CNN*	92.35	98.09	99.02	95.13
[40]	*LSTM*	99.88	99.82	99.97	99.85
[3]	*GRU*	98.00		97.56	
[19]	*BiGRU*	99.40	99.50	99.50	99.40
[44]	*BiGRU*	99.64	99.43	95.55	99.54
[20]	Transformer			96+	
[17]	LURL			97.40	
[34]	EXPOSE			97+	
[12]	GramBedding	97.59	98.26	98.27	99.73
[2]	Adaptive Neuro-Fuzzy Inference System (*ANFIS*)			98.30	
[14]	Multi-Modal Hierarchical Attention Model (*MMHAM*)	97.84	96.66	97.26	97.24

We now list our observations on automated *URL*-based website phishing detection strategies employing machine learning algorithms.

Feature Selection process has a significant impact on the performance of an automated website phishing detector. The specific features must be chosen before the classification process can begin for both machine learning and deep learning approaches. However, if a deep learning-based approach is used, the feature extraction process can be done automatically because these algorithms are capable of identifying the key characteristics on their own; as a result, deep learning features can also be used if researchers are attempting to come up with new sets of features. For a *URL*-based website phishing attack detector to operate well, a combination of features directly connected to the *URL* is required. For instance, combining Domain Name System (*DNS*), domain, and lexical elements of the *URL* will improve the detector's accuracy. There is one thing to keep in mind, though, and that is to avoid using too many features for classification as this

could lead to bias and over-fitting, both of which would impair the detector's effectiveness.

Algorithms from the fields of machine learning and deep learning used by researchers to combat the problem of phishing. Researchers initially employed heuristic-based approaches to tackle these issues, but as machine learning models advanced, this strategy was swiftly supplanted. The manual feature extraction was a vital component of the machine learning-based method because it influenced how well the algorithms worked. Deep learning-based approaches, however, are currently quite popular because the models can now automatically infer the semantics of the *URL*, eliminating the need for manual extraction. Although the essence of these works has been simplified, the underlying architecture is still a conundrum. As a result of this survey, we can see that developing a *URL*-based detector using deep learning-based algorithms yields better results. Additionally, someone who has little prior domain expertise about what features to choose for categorization purposes may benefit from a deep learning method because this can be done automatically.

Based on the classification accuracy of these algorithms in this domain, it can be suggested that *RF* algorithms in the area of machine learning perform the best with an accuracy of 99.29% with *DT* being another excellent machine learning algorithm that comes in second place with an accuracy of 97.40%. *LSTM* is an algorithm that is the best choice (accuracy 99.96%) and *CNN* is the second-best-performing algorithm with accuracy of 99.79% for the deep learning-based approaches.

Datasets utilized were not from a single source, and each researcher used a separate dataset to develop their system. As a result, the lack of a shared dataset can be a concern because one dataset may contain certain phishing site data while the other one does not. Furthermore, because phishing *URL* databases are not open-source, many academics do not use them. This is advantageous because attackers may acquire publicly accessible datasets and use them to extract key attributes and tailor their assaults accordingly. The drawback of that is that it might be laborious and time-consuming for a researcher to create a dataset.

7 Conclusions

We discussed *URL*-based phishing detection approaches, focusing on the features, algorithms, and datasets used by researchers. We observed that lexical analyzers are effective tools for detecting *URL*-based phishing since they can detect phishing on the fly (real-time detection), and they can also correctly identify newly constructed malicious websites. However, more effort needs to be put into making the detector more robust because attackers are always coming up with new ways to use phishing attacks to evade the defenses. One approach to do this is to use adversarial phishing samples to train the model, and these samples can be produced using an Generative Adversarial Network (*GAN*).

Google Sites is increasingly used to create websites, and fraudsters use it to build phishing websites and conduct phishing attacks. The problem, in this

case, is that because sites created with Google Sites disclose less information in the *URL*, the approaches covered in this survey may not be adequate to thwart phishing attempts made using Google Sites. For such websites, a combination of *URL*-based and content-based features need to be used to make the detection techniques effective.

Acknowledgements. This work was supported in part by funding from NSF under Award Numbers CNS 1715458, DMS 2123761, CNS 1822118, NIST, ARL, Statnett, AMI, NewPush, and Cyber Risk Research.

Appendix

The metrics used to assess the performance of the algorithms are described below. We use N to represent the number of legitimate and phishing websites, with P denoting phishing and L denoting legitimate.

Precision is the proportion of phishing attacks ($N_{P\to P}$) classified correctly as phishing attacks to the total number of attacks detected ($N_{L\to P} + N_{P\to P}$). $Precision = \frac{N_{P\to P}}{N_{L\to P}+N_{P\to P}}$

Recall is the proportion of phishing attacks ($N_{P\to P}$) classified correctly to total phishing attacks ($N_{P\to P} + N_{P\to L}$). $Recall = \frac{N_{P\to P}}{N_{P\to P}+N_{P\to L}}$

Accuracy is the proportion of phishing and legitimate sites that have been correctly classified ($N_{L\to L} + N_{P\to P}$) to the total number of sites $Accuracy = \frac{N_{L\to L}+N_{P\to P}}{TotalSites}$

F1-Score is a widely used evaluation metric that combines the model's recall and precision into a single score for binary classification models. $F1 - score = \frac{2*(Precision*Recall)}{Precision+Recall}$

Table 3. Dataset sources and the size of the data used for experiments in the literature

Ref	Dataset				
	Dataset source		Dataset size		Total Samples
	Legitimate	Phishing	Legitimate	Phishing	
[19]	Common Crawl	PhishTank	800k	759k	1,500k
[37]	DMOZ	PhishTank	55k	55k	100k
[11]	DMOZ	PhishTank	100k	15k	115k
[24]	Alexa	PhishTank	110k	32k	142k
[21]	Yahoo directory, DMOZ	PhishTank	2k	32k	34k
[2]	Google Search Operator	PhishTank, MillerSmiles	6k	6.8k	12.8k
[35]	Yandex.XML	PhishTank	36k	37k	73k
[13]	Kaggle [23]	PhishTank	40k	60k	100k
[6]	DMOZ	PhishTank, MillerSmiles	54k	52.8k	106.8k
[34]	DMOZ, Alexa, Phish-storm	PhishTank, OpenPhish, Phish-storm	96k	96k	192k
[14]	DMOZ	PhishTank	4k	4k	8k
[17]	Alexa	PhishTank	7k	6k	13k
[42]	Common Crawl	PhishTank	10.6k	10.6k	21.2k
[40]	Alexa, DOMZ	PhishTank, OpenPhish, MalwareURL, MalwareDomain, MalwareDomainList	79k	62k	141k
[7]	Alexa, Yandex, Common Crawl	PhishTank, OpenPhish, MalwareDomain	278k	278k	556k
[39]	Google Search Operator, Yahoo, Alexa, Common Crawl	PhishTank, MillerSmiles, OpenPhish	10.4k	11.9k	22.3k
[45]	Alexa	PhishTank	343k	70k	413k
[41]	Alexa	PhishTank	245k	245kk	490k
[44]	Common Crawl	PhishTank	800k	759kk	1559k
[4]	Common Crawl	PhishTank	1140k	1167kk	2307k
[3]	Common Crawl	PhishTank	2220k	2353k	4573k
[5]	Alexa	PhishTank	85k	60k	145k
[8]	Alexa	PhishTank	10k	9.7k	19.7k
[29]	Kaggle (Source not mentioned)	Kaggle (Source not mentioned)	-	-	11k
[12]	Custom Crawler developed	PhishTank, OpenPhish	400k	400k	800k
[20]	Alexa, Common Crawl	PhishTank, OpenPhish	25.96k	25.96k	51.9k

References

1. Abdelhamid, N.: UCI Machine Learning Repository (2016). https://archive.ics.uci.edu/ml/datasets/Website+Phishing
2. Adebowale, M.A., Lwin, K.T., Sanchez, E., Hossain, M.A.: Intelligent web-phishing detection and protection scheme using integrated features of images, frames and text. Expert Syst. Appl. **115**, 300–313 (2019)
3. Al-Ahmadi, S., Alotaibi, A., Alsaleh, O.: PDGAN: phishing detection with generative adversarial networks. IEEE Access **10**, 42459–42468 (2022)
4. Al-Alyan, A., Al-Ahmadi, S.: Robust URL phishing detection based on deep learning. KSII Trans. Internet Inf. Syst. (TIIS) **14**(7), 2752–2768 (2020)
5. Al-Haija, Q.A., Al Badawi, A.: URL-based phishing websites detection via machine learning. In: 2021 International Conference on Data Analytics for Business and Industry (ICDABI), pp. 644–649. IEEE (2021)
6. AlEroud, A., Karabatis, G.: Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks. In: Proceedings of the Sixth International Workshop on Security and Privacy Analytics, pp. 53–60 (2020)

7. Aljofey, A., Jiang, Q., Qu, Q., Huang, M., Niyigena, J.P.: An effective phishing detection model based on character level convolutional neural network from URL. Electronics **9**(9), 1514 (2020)
8. Alshingiti, Z., Alaqel, R., Al-Muhtadi, J., Haq, Q.E.U., Saleem, K., Faheem, M.H.: A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN. Electronics **12**(1), 232 (2023)
9. APWG: phishing activity trends report (2021). https://apwg.org/trendsreports/. Accessed 14 Nov 2021
10. ARossi: Alexa crawls. https://archive.org/details/alexacrawls?tab=about
11. Aung, E.S., Yamana, H.: URL-based phishing detection using the entropy of non-alphanumeric characters. In: Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services, pp. 385–392 (2019)
12. Bozkir, A.S., Dalgic, F.C., Aydos, M.: GramBeddings: a new neural network for URL based identification of phishing web pages through n-gram embeddings. Comput. Secur. **124**, 102964 (2023)
13. Butnaru, A., Mylonas, A., Pitropakis, N.: Towards lightweight URL-based phishing detection. Future Internet **13**(6), 154 (2021)
14. Chai, Y., Zhou, Y., Li, W., Jiang, Y.: An explainable multi-modal hierarchical attention model for developing phishing threat intelligence. IEEE Trans. Dependable Secure Comput. **19**(2), 790–803 (2021)
15. Common crawl. https://commoncrawl.org/
16. Curlie. https://curlie.org/
17. Dutta, A.K.: Detecting phishing websites using machine learning technique. PLoS ONE **16**(10), e0258361 (2021)
18. Ebubekirbbr: Pdd/input at master · ebubekirbbr/pdd (2019). https://github.com/ebubekirbbr/pdd/tree/master/input
19. Feng, T., Yue, C.: Visualizing and interpreting RNN models in URL-based phishing detection. In: Proceedings of the 25th ACM Symposium on Access Control Models and Technologies, pp. 13–24 (2020)
20. Haynes, K., Shirazi, H., Ray, I.: Lightweight URL-based phishing detection using natural language processing transformers for mobile devices. Procedia Comput. Sci. **191**, 127–134 (2021)
21. Jain, A.K., Gupta, B.B.: PHISH-SAFE: URL features-based phishing detection system using machine learning. In: Bokhari, M.U., Agrawal, N., Saini, D. (eds.) Cyber Security. AISC, vol. 729, pp. 467–474. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-8536-9_44
22. KnowBe4: History of phishing. https://www.phishing.org/history-of-phishing. Accessed 24 June 2022
23. Kumar, S.: Malicious and benign URLs (2019). https://www.kaggle.com/datasets/siddharthkumar25/malicious-and-benign-urls
24. Lee, J., Ye, P., Liu, R., Divakaran, D.M., Chan, M.C.: Building robust phishing detection system: an empirical analysis. In: NDSS MADWeb (2020)
25. Malware domain list. https://www.malwaredomainlist.com/. Accessed 03 Apr 2023
26. MalwareURL: Fighting malware and cyber criminality. http://www.malwareurl.com/. Accessed 03 Apr 2023
27. Marchal, S.: Phishstorm - phishing/legitimate URL dataset (2014). https://research.aalto.fi/fi/datasets/phishstorm-phishing-legitimate-url-dataset
28. MillerSmiles.co.uk: Phishing scams and spoof emails at millersmiles.co.uk. http://www.millersmiles.co.uk/

29. Mithra Raj, M., Arul Jothi, J.A.: Website phishing detection using machine learning classification algorithms. In: Florez, H., Gomez, H. (eds.) ICAI 2022. CCIS, vol. 1643, pp. 219–233. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19647-8_16
30. Mohammad, R.M.A.: UCI Machine Learning Repository (2015). https://archive.ics.uci.edu/ml/datasets/phishing+websites
31. OpenPhish: Phishing intelligence. https://openphish.com/
32. PhishTank: Join the fight against phishing. https://phishtank.com/
33. RiskAnalytics: Not all threat intel is created equal. https://riskanalytics.com//. Accessed 03 Apr 2023
34. Sabir, B., Babar, M.A., Gaire, R.: An evasion attack against ml-based phishing URL detectors. arXiv preprint arXiv:2005.08454 (2020)
35. Sahingoz, O.K., Buber, E., Demir, O., Diri, B.: Machine learning based phishing detection from URLs. Expert Syst. Appl. **117**, 345–357 (2019)
36. Tan, C.L.: Phishing dataset for machine learning: feature evaluation (2018). https://data.mendeley.com/datasets/h3cgnj8hft/1
37. Tupsamudre, H., Singh, A.K., Lodha, S.: Everything is in the name – a URL based approach for phishing detection. In: Dolev, S., Hendler, D., Lodha, S., Yung, M. (eds.) CSCML 2019. LNCS, vol. 11527, pp. 231–248. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20951-3_21
38. UNB. https://www.unb.ca/cic/datasets/url-2016.html
39. Vaitkevicius, P., Marcinkevicius, V.: Comparison of classification algorithms for detection of phishing websites. Informatica **31**(1), 143–160 (2020)
40. Vinayakumar, R., Soman, K., Poornachandran, P.: Evaluating deep learning approaches to characterize and classify malicious URL's. J. Intell. Fuzzy Syst. **34**(3), 1333–1343 (2018)
41. Wang, W., Zhang, F., Luo, X., Zhang, S.: PDRCNN: precise phishing detection with recurrent convolutional neural networks. Secur. Commun. Netw. **2019**, 1–15 (2019)
42. Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., Woźniak, M.: Accurate and fast URL phishing detector: a convolutional neural network approach. Comput. Netw. **178**, 107275 (2020)
43. Yandex. https://yandex.com/dev/
44. Yuan, L., Zeng, Z., Lu, Y., Ou, X., Feng, T.: A character-level BiGRU-attention for phishing classification. In: Zhou, J., Luo, X., Shen, Q., Xu, Z. (eds.) ICICS 2019. LNCS, vol. 11999, pp. 746–762. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-41579-2_43
45. Zheng, F., Yan, Q., Leung, V.C., Yu, F.R., Ming, Z.: HDP-CNN: highway deep pyramid convolution neural network combining word-level and character-level representations for phishing website detection. Comput. Secur. **114**, 102584 (2022)