Computer Communications 175 (2021) 47–57

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment

Brij B. Gupta [a,b,*], Krishna Yadav [a], Imran Razzak [c], Konstantinos Psannis [d], Arcangelo Castiglione [e], Xiaojun Chang [f]

[a] National Institute of Technology Kurukshetra, Kurukshetra, 136119, Haryana, India
[b] Department of Computer Science and Information Engineering, Asia University, Taichung, Taiwan
[c] Deakin University, Australia
[d] University of Macedonia, Greece
[e] University of Salerno, Fisciano, Salerno, Italy
[f] Monash University Clayton Campus, Australia

ARTICLE INFO

Keywords:
Accuracy
Blacklist
Features
Machine learning
Phishing
Real-time
Random forest
URLs

ABSTRACT

In recent times, we can see a massive increase in the number of devices that are being connected to the internet. These devices include but are not limited to smartphones, IoT, and cloud networks. In comparison to other possible cyber-attacks, these days, hackers are targeting these devices with phishing attacks since it exploits human vulnerabilities rather than system vulnerabilities. In a phishing attack, an online user is deceived by a seemingly trusted entity to give their personal data, i.e., login credentials or credit card details. When this private information is leaked to the hackers, this information becomes the source of other sophisticated attacks. In recent times many researchers have proposed the machine learning-based approach to solve phishing attacks; however, they have used a large number of features to develop reliable phishing detection techniques. A large number of features requires large processing powers to detect phishing, which makes it very much unsuitable for resource constrained devices. To address this issue, we have developed a phishing detection approach that only needs nine lexical features for effectively detecting phishing attacks. We used ISCXURL-2016 dataset for our experimental purpose, where 11964 instances of legitimate and phishing URLs are used. We have tested our approach against different machine learning classifiers and have obtained the highest accuracy of 99.57% with the Random forest algorithm.

1. Introduction

There has been an exponential growth in the number of organizations, and new technologies are continually being explored for broader applicability. Every day millions of new websites are being developed that have login portals to extract credentials from users. As these websites are in large volume, it is becoming challenging to verify their credibility. According to a report published by Dofo [1], more than 5.16 million domain names were registered in the month of April 2020. As the number of users is increasing, it becomes easy for attackers to lure more users. In most cases, phishing attacks start with fraudulent emails that appear to come from a legitimate source. This email consists of malicious links that are redirected to a fake website when clicked by a user. Afterward, the user ends up giving their confidential information like login id, passwords, and credit card details. Sometimes, attackers perform phishing attacks to disseminate malware in the network.

The word phishing was first introduced in the 1990s via America Online. The hacker back then constructed an algorithm that was used to generate the credit card numbers. They use these generated credit card numbers to register an AOL account. When there was a match between the generated and real credit card numbers, they used to create an account whose motive was to ultimately spam other people in AOL's community. Later on, when online users were informed about this scam, hackers switched their phishing platform from messenger to email as it was easy to create an exploitable email, and it was very difficult to catch hackers through emails. Modern days phishing is not only limited to email and websites, but also with the links that appear in online ads, status updates, tweets, and Facebook posts. Some of these links are fraudulent where credentials are being massively stolen.

Fig. 1 represents a traditional/generic phishing scenario (i.e., mass-email phishing campaigns). Fig. 1 shows that an attacker hosts a

* Corresponding author.
E-mail address: gupta.brij@gmail.com (B.B. Gupta).

https://doi.org/10.1016/j.comcom.2021.04.023
Received 21 November 2020; Received in revised form 9 April 2021; Accepted 22 April 2021
Available online 30 April 2021

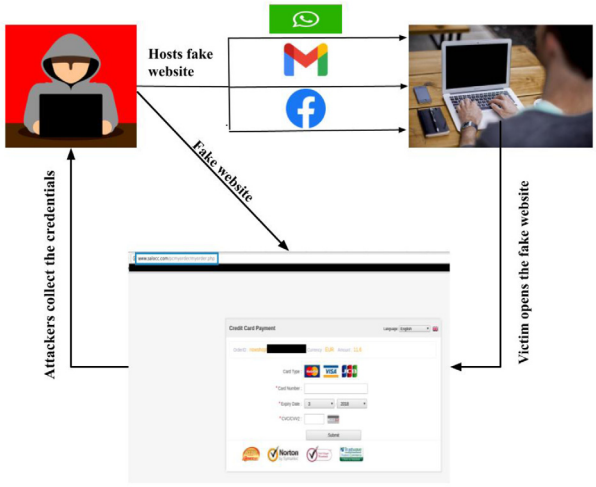

Fig. 1. Traditional Phishing attack.

fake website and sends it to the victim via emails or any other messaging platform. The fake website hosted seems very appealing and authentic to the users. Sometimes hackers also display some lucrative messages inside the hosted fake website, which makes users enter their credentials. The hosted website is controlled by the attackers, and all the entered credentials are redirected to the attackers. Sometimes phishing attacks are used by users to install the malware in the victim's machine. This malware will change the victims machine to the botnet [2,3]. These botnets are used by the attackers to launch DDoS and several other attacks (i.e. Phishing, SQL injection, XSS attacks, authentication/authorization issues, etc.) [4–8].

In 2019, a security threat report of Symantec [9] suggested that one out of 170 URLs in 2018 is malicious and used for performing phishing attacks. As per the report [9], phishing attacks have caused a loss of $1.4 billion in 2018 and $26 billion from June 2016 to July 2019. Check Point security report [10] suggested that 64% of organizations have experienced a phishing attack in 2018. Moreover, IBM also reported [11] that the loss from phishing attacks is not only limited to the revenue generated by the organization but also leads to losing customers, brand, and reputation loss. The report also shows that organizations have lost about 1% of their customers due to a data breach. Phishing attacks were the root cause of these data breaches. Authors at [12] have identified that most of the users are unable to identify the correct websites and select the fake website based on their content and seemingly professional look. In [13], authors believed that people were propelled in sharing their credentials, assuming that email senders already had enough information about them.

Several software have been developed using blacklisting and heuristic approaches to prevent phishing attacks, such as Google Safe Browsing, McAfee SiteAdvisor, Netcraft Anti-phishing Toolbar, Spoof Guard [14]; still, victims are being phished, and credentials are being stolen. With the increase in the novelty of phishing attacks, the blacklisting approach is no more appropriate for phishing detection. Minor changes or mismatch in URLs from the URLs in the blacklist database, i.e., replacing Top-Level Domain, directory path, brand name, Query String substitution, etc., in URLs, can make malicious links undetectable. In [15], authors believed that earlier proposed phishing detection approaches such as heuristic and visual similarity-based techniques produced high false-positives. Therefore, it is important to develop an anti-phishing solution that can detect the new phishing web sites with higher accuracy and without using blacklisted URLs. Many researchers have proposed machine learning-based solutions to prevent phishing attacks [16–18]; however, most of these machine learning approaches suffer from various issues like high false-positive rates, high response time, and intervention of third parties information. In this paper, we have proposed an anti-phishing solution that can detect phishing URLs in a real-time environment without requiring any third party information and also with a very low response time. In our approach, we focused on achieving high accuracy with a limited number of features to detect phishing attacks. Thus, we studied the most important features in the literature and came up with nine lexical-based features to develop a highly accurate phishing detection approach. We evaluated our approach with several machine learning classifiers and obtained the highest accuracy of 99.57%. To provide readers a good insight about our features, we applied different algorithms, such as Spearman correlation, K best, and Random forest, and calculated feature importance scores.

The rest of the paper is organized as follows: The related work is discussed in Section 2, and the proposed approach is discussed in Section 3. Section 4 provides details about the implementation and analysis of results. Finally, Section 5 concludes the paper and discusses some future work.

2. Related work

In this section, we discuss various significant works and existing approaches for phishing attack detection that have been proposed in the literature. Phishing attack detection approaches can be classified into two categories; user education-based and software-based. Software-based approaches can be further categorized into four categories; the blacklisting method, visual similarity method, machine learning method, and hybrid method. Before getting a detailed insight into the existing works proposed in the literature that detect phishing, it is very important to understand some terminology of an URL. Section 2.1 clearly discusses the URL terms, and the rest of the subsection discusses the existing phishing detection approaches.

2.1. Anatomy of an URL

The Uniform Resource Locators are used to identify a webpage that is represented in Fig. 2. It has seven different parts, i.e., Protocol, Domain name, Path, Parameter, Subdomain, Top-level domain, and Query. A protocol defines how our web browser should communicate with a web server. Some of the very common protocols are HTTPS (Hypertext Transfer Protocol Secure), FTP (File Transfer Protocol), POP (Post Office Protocol), SMTP (Simple Mail Transfer Protocol), and IMAP (Internet Message Access Protocol). Further, a domain name is a unique reference that identifies a web site on the internet. The path refers to a unique location where a file or directory exists in a web server, e.g., /home/address/image.jpeg. A subdomain is a subdivision of the main domain name. For example, mail.stanford.edu and cs.iitb.ac.in, are subdomains of stanford.edu and iitb.ac.in. A domain name always includes the TLD(top-level domain); in the case of stanford.edu, edu is a top-level domain. A query is generally found in dynamic web pages. A query is always followed by a question mark. When a client makes a request for a page on a server, it takes a query string and runs the program. For example, https://example.com/over/path/there?name=jason. In this URL, name=jason is a query.

2.2. User education

At the present time, many of the users that are connected to the internet are unaware of Internet security and cyber-threats. These days hackers take advantage of users' limited knowledge to launch different phishing attacks. At this time, it becomes essential to educate the user about Internet security and practices to be followed in order to sustain information security. However, authors at [19] have mentioned that there is a need to develop a good mechanism to fight against phishing attacks as educating people about the legitimacy of a website is a tedious and time-consuming task.

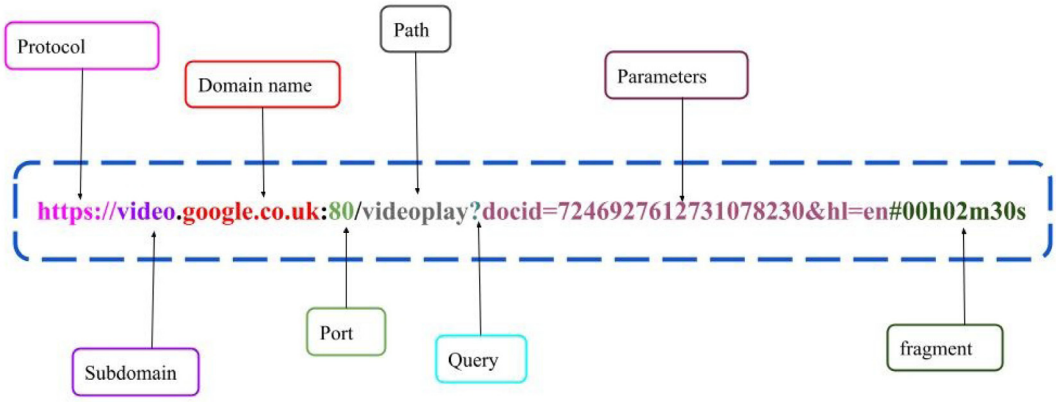

Fig. 2. Anatomy of an URL.

2.3. Blacklisting phishing URLs

A classical technique for detecting phishing URLs is to blacklist phishing or malicious URLs. In this approach, a database is maintained containing a large number of phishing URLs, domains, and IP addresses. When a user visits a new URL, the visited URL is searched in the database. If the visited URL is found, then the URL is classified as malicious, otherwise not. For the proper detection of phishing URLs, the blacklist database needs to be updated frequently. Authors at [20] have suggested that the blacklist database needs to be updated every 12 h of the initial phishing test, and in [21], authors have observed that many attackers make minor modifications to URL such as domain name, file path, Query String which lets phishing URL go undetected by the employed blacklist-based system. Authors at [22] suggested that blacklist features alone do not perform well; however, when it is conjugated with other features such as lexical and host-based, it may produce a great result. Due to the simplicity of the blacklist method, it is still widely used in many systems to detect phishing attacks.

2.4. Visual similarity-based techniques

Most of the users clicked on fake websites just by looking at the appearance of the URL since it looks like an authentic one. People do not pay much attention to the URL and SSL (Secure Socket Layer) certificates of websites. Visual similarity-based phishing technique utilizes features like HTML tags, CSS, image logo to find the similarity. If the similarity between suspicious websites and legitimate websites exceeds a certain threshold, then the suspicious website is categorized as phishing. The information extraction process from images has been clearly described in [23]. On the basis of extracted information, similarity can be calculated. Authors at [24] have proposed a visual similarity-based phishing detection scheme using images and CSS with a target website finder and have obtained an accuracy of 80%. Authors at [25] proposed a WhiteNet approach where a database of whitelisted pages is maintained based on their URLs. The embeddings of the phishing web pages are compared to the whitelist pages and the decision is made based on visual similarity. Phishing webpages were found to be visually similar to the whitelist. The disadvantage of a visual similarity-based scheme is that it cannot detect newly launched phishing websites. Moreover, it also generally suffers from a low accuracy rate. The legitimate Facebook website with the correct URL www.facebook.com and the visually similar benign Facebook website with URL www.sanagustinturismo.co/facebook.com/ is shown in Figs. 3(a) and 3(b), respectively. From the figures, it can be concluded that the user may perceive the phishing websites as legitimate ones by looking only at the visual properties since they do not pay much heed to the URL.

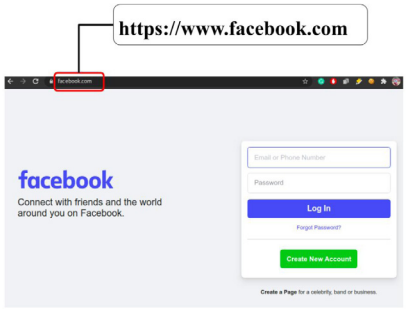

Fig. 3(a). Legitimate Facebook webpage.

Fig. 3(b). Phishing webpage of Facebook.

2.5. Machine learning based techniques

In recent times, many authors have proposed machine learning approaches for phishing detection, which includes a vast database of phishing and legitimate websites. The features related to the URL, page content, DNS, etc., are extracted, and the new dataset is made with the selected feature set. The dataset is then pre-processed and fed into various machine learning algorithms. The results and accuracy depend upon the selected feature sets and the selected machine learning algorithms. Authors at [16] have proposed an approach that is independent of URLs language and third party information. Additionally, it can detect phishing URLs in real-time. They have used 27 NLP based features such as Raw word count, brand name count, etc., with

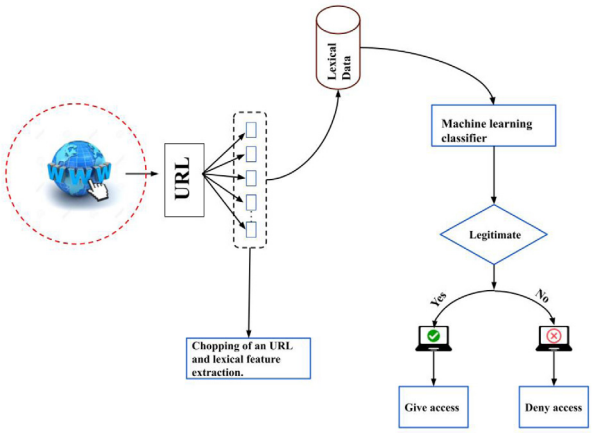

Fig. 4. Proposed Phishing detection architecture.

different machine learning algorithms where accuracy of 97.98% has been obtained with random forest. Authors at [26] have used URL and content-based features and have achieved an accuracy of 96.58%. However, their proposed approach has one disadvantage of high false-positive rates. Authors at [27] have presented a phishing detection approach where they extracted 19 features at the client-side to distinguish malicious links from legitimate ones. They have used 2141 phishing pages and 1918 legitimate web pages to prepare a dataset and have obtained an accuracy of 99.39%. Their proposed approach is available for client-side desktop applications and can be used for real-time phishing detection. In literature, many approaches have also used the natural language processing method for phishing detection. In [28], authors have utilized the NLP method to detect phishing emails and obtained an accuracy of 95%. They have used a blacklisted bag of words with a dataset that contains 5009 and 5000 instances of phishing and legitimate emails.

2.6. Hybrid features based technique

The hybrid approach uses the combination of several features such as URL-based, Content-based, and Domain-based features for phishing detection. In [29], authors have suggested that the combination of several features may boost up the accuracy rate of phishing URLs detection. They have discussed a machine learning-based hybrid approach and image checking approaches for the detection of phishing websites [29]. They have used hyperlink based features, third-party based features, and URL obfuscation features and have obtained an accuracy of 99.55%.

3. Proposed approach

3.1. Design objectives

The main objective of this research is to develop a machine learning-based phishing detection system that can help users to check the legitimacy and maliciousness of an URL within a minimum amount of time. We aim to develop a mechanism that can extract the feature vectors from URL whenever a user visits that URL. Afterward, the feature vectors are pre-processed and fed into several machine learning algorithms to validate the legitimacy of an URL. The other objectives of our proposed approach are described below.

- *Reliable security*: We aim to build the phishing detection approach that produces the minimum number of false positives.
- *Real-time security*: A reliable phishing detection mechanism must block the malicious URL without revealing the credentials to the hackers.
- *Low response time*: We aim at building an anti-phishing approach that can detect phishing URLs as early as possible. The low response time will give hackers comparatively less time to steal the credentials.
- *Detection of new phishing websites*: We aim at building an anti-phishing system that outperforms the current blacklisting techniques for phishing detection and can detect new phishing websites in the coming future.
- *Scalability: We aim at developing a phishing detection approach that can be embedded in a device with constrained resources, i.e., IoT, to the devices powered up by multiple CPUs and GPUs.*

3.2. Architecture of proposed approach

The architecture of our proposed system is shown in Fig. 4. In Fig. 4, we can see that whenever a user visits a new URL, the URL is chopped into different segments. We developed a feature extraction mechanism that obtains different lexical based information from URL. Our developed feature extraction mechanism is briefly discussed in Section 3.3. Our feature extraction mechanism consists of multiple scripts written in python. The lexical information obtained is then mapped into respective features, and the instance of an URL is prepared that contains all the information necessary to classify the URL as legitimate or phishing. The data from an instance of an URL is then preprocessed, and the preprocessed data is fetched into machine learning algorithms to check the legitimacy of an URL. If the website is found legitimate, then it is given access to the clients to use, otherwise, the website is blocked.

3.3. Features extraction algorithm

In this section, we clearly discuss our lexical features and the algorithms we have used to extract lexical data from the URLs. The most contributing lexical features were developed by analyzing several available lexical features proposed by different researchers [30,31]. We applied several algorithms to calculate feature importance and came up with the optimal number of features. In our features, we have expanded the number of top-level domains and have included new domains that could present in phishing URLs. Moreover, we have also expanded the number of delimiters in a delimiters list. Expanding the list has increased the feature importance that can be seen in Fig. 8 and ultimately accuracy. The lexical data extracted were then used to make a dataset to train machine learning classifiers.

1. ***No.of token in a domain***: Tokenizing is the process of chopping the URL into several pieces. A token refers to a segment that has been broken down into sequence. In this feature, we break down URLs into the tokens and take their count. Generally, it is seen that phishing URLs are longer and contain a greater number of a token count. For e.g., URL: http://clubeamigosdopedrosegundo.com.br/last/
   Tokenized URL=[‘http’, ‘clubeamigosdopedrosegundo’, ‘com’, ‘br’, ‘last’]. Algorithm 1 gives an idea about the way of tokenizing URLs and taking their count. In algorithm 1, tokenize function helps in creating a token which is stored in a list of tokenized_list . In an URL u, there might be multiple tokens i.e $T = \{t_1, t_2, t_3, \ldots, t_n\}$ where T is a list of tokenized_list. The count gives the total number of tokens present in the T and stores them in D. In algorithm 1, m_1 indicates the feature vector, i.e., No.of token, and index indicates the position where data is stored.

Algorithm 1: Count the number of tokens in the URL

```
Input: URL
Output: Number of token count in a URL
Procedure token_count(URL)
        Initialize count ←0
        Initialize index←0
        tokenized_list=tokenize(URL)
        for all each token in tokenized_list do
                count←count+1
Store D[m1].index=count
```

Algorithm 2: Find the number of top level domains in an URL

```
Input: URL
Output: Number of top level domain present in URL
Procedure find_domain(URL)
        Initialize top_level_domain P={p1, p2, ..pn}
        Initialize count ←0
        Initialize index ←0
        d=find_top_level_domain(URL)
        for all each domain present in d do
            if domain is present in P then
                count←count+1
Store D[m2].index=count
```

2. ***No. of top-level domain:*** It is observed that attackers use multiple top-level domains within a domain name. This feature takes the count of the number of top-level domains present in a URL. Algorithm 2 gives an idea to extract this feature from URLs. In algorithm 2, we have maintained a top_level_domain list which contains 920 top-level domains, i.e., $P = \{p_1, p_2, tp_3 \ldots p_y\}$. In one URL, there may be multiple top-level domains present. Let $d = \{d_1, d_2, d_3, \ldots, d_n\}$ be the individual top-level domain present in a URL u, such that $d_k \in P$ where d_k is any instance of top-level domain present in d. The function *find_top_level_domain* in algorithm 2 searches through the URL to find the number of top-level domains. The domain present is stored in list D. All the domains present in the d are iterated and matched against the top level domain in list P, and the count is stored in the dataset D accordingly.
3. ***Length of an URL:*** Generally, the phishing URLs are longer in length. Attackers embed multiple domains' names, longer paths to perform malicious activity that makes an URL longer. Algorithm 3 counts each character present inside the length of an URL and gives the total length.

Algorithm 3: Find the length of an URL

```
Input: URL
Output: Length of URL
Procedure length(URL)
        Initialize Length←0
        Initialize index←0
        for all each character in URL do
                Length→Length
Store D[m9].index=Length
```

Algorithm 4: Find the number of digits in a query

```
Input: URL
Output: Number of digits in a query
Procedure digit_in_query(URL)
        Initialize Y→0
        Initialize index→0
        Q=query(URL)
        If Q is not empty then
            for all each query in a Q do
                xk= digits_in_query(query)
                Y=Y+xk
            Store D[m8].index=Y
        else
            Store D[m8].index=-1
```

4. ***No of digits in a Query*:** To calculate the digit in a query, we first tokenize the text followed by a question mark. In algorithm 4, let us say for URL u, there might be more than one query i.e., $Q = \{q_1, q_2, q_3, \ldots, q_n\}$. Let $X = \{x_1, x_2, x_3, \ldots, x_n\}$ be the total number of digits present in each query in Q. The function *digits_in_query iterates through all of the queries present in Q and counts the number of digits present.* For an URL u, let us say the total number of digits present is Y, which is the sum of the number of digits present in each individual query, i.e., x_k, then Y is represented by Eq. (1).

$$Y = \sum_{k=1}^{n} x \text{ where } n \text{ is any arbitrary number} \tag{1}$$

5. ***No. of dots in URL:*** It is used to check the number of dots present in a URL. Generally, legitimate websites do not have many dots in a URL. On the other hand, phishing websites have comparatively more numbers of dots. Dots are usually used to misguide the legitimacy of the website. For example, www.facebook.com can be represented as www.face.book.com. In such a case, many users do not pay much attention to the URL and become a victim to the attackers.
6. **No. of delimiter in a domain:** Delimiter is the bag of the word containing character like *(.), (;), ({) (|), (+)*. We have created a list of delimiters containing 26 characters initialized by V in algorithm 6. There may be multiple domains present in one phishing URl. In algorithm 6, the function *find_available_domain* first finds the number of domains present inside the URL. The function *delimiter_count* checks the available characters inside the domain against the bag of word V and gives the count. The count is increased accordingly and stored in D. The presence of a delimiter, for example, a dashed or a + symbol in URL www.face-book.com, www.face+book.com can make a phishing URL look like a legitimate one. The analysis result in Fig. 8 indicates that there are relatively more delimiters in a domain of phishing URLs than legitimate URLs.

Algorithm 5: Find the length of the longest path in URL Input: URL

```
Input: URL
Output:Length of longest path in URL.
Procedure longest_path(URL)
        Initialize index→0
        G=find_paths(URL)
        If G is not empty then
            for all each path in G do
                fk=find_length(path)
            H=max(F)
            Store D[m7].index=H
        else
            Store D[m7].index = -1
```

Algorithm 6: Find the number of delimiter in path and domain

```
Input: URL
Output:Number of delimiter in path and domain
Procedure delimiter_count(input)
        Initialize V='@', '/','='.....
       Initialize count←0
        for all each character in input do
          if character is in V then
                    count=count+1
       return count

Procedure number of delimiter(URL)
       Initialize C1←0
       Initialize C2←0
        Initialize path_index←0
       Initialize domain_index←0
        G=find_available_path(URL)
       R=find_available_domain(URL)
          if G is not empty then
       for all each path in G do
          count=delimiter_count(path)
           C1=C1 + count
       Store D[m6].path_index=C1
           else
              Store D[m6].path_index=-1
          if R is not empty then
             for all each domain in R do
                count=delimeter_count(domain)
                C2=C2 + count
             Store D[m5].domain_index=C2
          else
             Store D[m5].domain_index=-1
```

7. ***No. of delimiter in the path:*** We analyzed the URLs and found that the delimiter in the path of the phishing URL is about two times more than in the legitimate URL. This analysis can be found in Fig. 8. The phishing URL path contains multiple files like .png, .js followed by a delimiter. These files can be harmful in performing malicious activities. In algorithm 6, the function *find_available_path* finds the number of paths present in an URL. The function *delimiter_count* then gives the number of delimiters present inside the path.
8. ***Length of longest token in the path:*** Generally, hackers try to hide some malicious information in the long token of a URL path. Our analysis regarding the longest path in Fig. 8 indicates that phishing URLs have a relatively 3X longer path than the legitimate URLs. In algorithm 6, the function *find_paths* finds the multiple paths present inside the URL in a list *G*. The length of each path in list *G* is then calculated by function *find_length*, and the length of the longest path is calculated, which is stored in *H*. In some URLs, if the path is not present, then -1 is stored in a dataset *D*.
9. ***Domain length:*** Generally, it is found that hackers use long domains to make a URL look legitimate. A legitimate URL generally contains short domains. Fig. 8 gives this comparative analysis between the domain length present between legitimate and phishing URLs. In this feature, we calculate the domain length of each URL and store them in a dataset.

4. Experimental results and performance evaluation

4.1. Experimentation algorithms

We have evaluated our phishing detection approach with the help of four different machine learning algorithms, i.e., Random forest, k-Nearest-Neighbors, Logistic regression, and Support vector machine. In this section, we give a brief introduction to the classifiers used and the algorithms that have been used to calculate feature importance.

Table 1
Distribution of data in dataset.

S. No	Feature name	Min value	Max value
1.	No. of token in domain	0	17
2.	No. of top level domain	0	5
3.	Length of an URL	26	318
4.	No. of dots in URL	1	26
5.	No. of delimiter in domain	1	26
6.	No. of delimiter in path	0	39
7.	Length of longest token in path	0	248
8.	No of digit in a Query	−1	103
9.	Domain length	0	15

The logistic regression is commonly used for classification [32]. It calculates the probability of an instance belonging to a particular class. It computes a weighted sum of the input features plus the bias term. It then produces a result in the form of 0 or 1. In a support vector machine, the algorithms work by finding the best hyperplane that separates the two classes [33]. In our proposed approach, we have used the non-linear SVM classifier. K-Nearest-Neighbors (KNN) works by finding the distance between the new data point and existing data points [34]. In KNN, distance computing algorithms such as Euclidean distance, Manhattan distance, and Minkowski distance are used to cluster similar data points into one group. A random forest is an ensemble approach that uses decision trees as their building block [35]. It first constructs the number of decision trees, and each decision tree outputs its own decision of classification. Now the voting is done on the decision generated by each decision tree. Voting is a process of giving weightage to the decision made by several decision trees.

Feature selection is a very important part when we are building a phishing detection approach. A good set of features can boost accuracy, whereas the redundant features may sometimes increase the noise in a dataset, which ultimately decreases the accuracy of an algorithm. So, it is very important to include the features that can contribute the most to the accuracy. We have used three algorithms such as feature correlation [36], K best [37], and random forest [38], to find the feature importance.

Feature correlation is a way to understand the degree of dependency between multiple features and categorical attributes. There are two types of correlation algorithms: Spearman and Pearson. In our approach, we have used the Spearman correlation algorithm. The range of Spearman correlation lies between 0 and 1. The correlation value near 1 means that the features are highly contributing, whereas a value near 0 means features are barely contributing.

4.2. Experimental details

Experiments have been performed on a machine having a core i5 processor with a clock speed of 3.4 GHz, 8 GB RAM, and a 2 GB graphics card. Python has been used for the complete implementation of the proposed approach. In recent times, Python has been very popular for machine learning problems due to the availability of its large amount of libraries, such as Scikit-learn, NumPy, and the flexibility of language itself. We have used the following Python libraries to implement our anti-phishing approach.

- ***Scikit-learn:*** It is used to implement different machine learning algorithms such as SVM, Random forest, Logistic regression, and KNN.
- ***Scipy*****:** It is used in the statistical analysis of data, such as finding a correlation between the data.
- ***Pandas*****:** It is used to perform data analysis and for data manipulation.
- ***Matplotlib*****:** It is used to find the data distribution and to plot the results in pictorial forms.

Table 2
Results from different classifiers.

Algorithms	Precision (%)	Confusion matrix	Recall (%)	F1 score (%)	Accuracy (%)
Random forest	99.7	[1937 11] [6 2039]	99.46	99.58	99.57
k-Nearest-Neighbor	98.67	[1937 11] [27 2018]	99.45	99.06	99.04
Support vector machine	96.87	[1918 30] [64 1981]	98.50	97.68	97.64
Logistic regression	94.96	[1874 74] [103 1942]	96.3	95.625	95.56

	Phishing	Legitimate
Phishing	True Positive	False Positive
Legitimate	False Negative	True Negative

Fig. 5. Confusion matrix.

- ***Time***: It is used to capture the response time of our proposed approach.

4.3. Dataset and its pre-processing

To evaluate our proposed approach, we have used a dataset "ISCXURL-2016" from the data repository of the University of Canada Brunswick. It contains spam records, phishing links, malware, defacements, and benign URLs [39]. We have extracted 10 000 benign and 9964 phishing URLs from the repository. To prevent data bias, we have used benign and phishing data in nearly equal proportion.

The effective preprocessing of the dataset and the selection of the right classifier highly affect the result and accuracy of the machine learning-based approach. Data preprocessing is a data mining technique, which cleans the raw data and delivers more relevant data. Raw data contains numerous numbers of missing values, outliers, and unnecessary features. Without preprocessing, biases and deviation may be found in a dataset that may lead to inaccurate assumptions and results. Preprocessing of data involves many steps, such as data cleaning, integration, reduction, and discretization [40]. In phishing, where we are focusing on minimizing the false positive rate, data pre-processing becomes a crucial step. Our dataset contains large numbers of infinite and NaN (Not a number) values that are replaced by the mean value using the NumPy library in Python. Once the data gets cleaned, we apply feature scaling to the dataset. Feature scaling is a process of placing data in the same range such that one variable is not dominated by others [40]. The distribution of our data in our dataset is presented in Table 1.

If we see the distribution of data, we observe that there is a high variation in data. For such data, scaling is necessary to scale the data in an acceptable range, i.e., between 0 and 1 or centered around the mean of data [40]. Scaling on data is important so that the variance of the features lies in the same range. If the magnitude of the variance of one feature is more than the other, then that particular feature might dominate other features in the dataset. For the feature scaling, we have used a standard scaler. Standard scaling includes the subtracting mean value from the individual data value and then dividing by the standard deviation so that the resulting distribution has a unit variance [41]. The categorical variables 'legitimate' and 'phishing' have been changed to a numerical value using a one-hot encoding algorithm [42]. One hot encoding creates one binary attribute per category, either 0 or 1. Once the dataset gets standardized, it is then split into the training set and testing set. The training set data contains 15 971 instances, whereas the testing set contains 3993 instances.

Table 3
Performance measure used in our approach.

Performance measure	Formula	Description
TPR	TP/(TP+FN)	It is the total number of phishing URLs that are classified as phishing out of total URLs.
TNR	TN/(TN+FP)	It is the total number of legitimate URLs that are classified as legitimate out of total URLs.
FPR	FP/(FP+TN)	It is the total number of legitimate URLs classified as phishing out of total URLs.
FNR	FN/(FN+TP)	It is the total number of phishing URLs classified as legitimate out of total URLs.
Precision	TP/(TP+FP)	It is the total number of URLs detected as phishing out of total phishing URLs
Recall	TP/(TP+FN)	Total number of legitimate URLs classified as legitimate and phishing URLs classified as phishing.
F1-score	2 ∗ (precision ∗ recall)/ (precision + recall)	The harmonic mean of precision and recall.
Accuracy	(TP+TN)/ (TP+TN+FN+FP)	Total number of overall correctly classified instances.

Table 4
Result of true and false positive classification.

Algorithms	True positive rate (%)	False positive rate (%)	True negative rate(%)	False negative rate(%)
Random forest	99.70	0.53	99.46	0.30
k-Nearest-Neighbor	98.62	0.54	99.45	1.37
Support vector machine	96.77	1.49	98.50	3.22
Logistic regression	94.79	3.67	96.32	5.20

4.4. Performance evaluation and results

The results obtained are evaluated on the basis of the confusion matrix and other metrics that can be found in Table 3. A confusion matrix is a summary of correctly and incorrectly predicted instances of a class in a classification problem that is given in Fig. 5. We have used 15 971 instances as training data, whereas 3993 instances as testing data. Table 2 presents all the results generated from different classifiers. Here, we can see that the highest accuracy was offered by Random forest, which is 99.7%. We can see that only six phishing URLs are wrongly classified as legitimate instances out of total phishing instances from the confusion matrix in Table 2. The random forest produces the false positive with a rate of 0.53% that renders our proposed approach very reliable and comprehensible. However, a recall score of 99.46% suggests that we are able to classify 99.46% of the total instances correctly. In the cases where we are classifying legitimate and phishing URLs, precision matters the most than the recall score. Among all our classifiers, logistic regression performs relatively bad by producing a false positive rate of 3.67%.

For performance evaluation, we have to analyze true positive rate (TPR), true negative rate (TNR), false-positive rate (FPR), false-negative rate (FNR), and accuracy. These metrics and their calculation process are clearly described in Table 3. Table 4 presents the results of these metrics with different classifiers. We further have also calculated the

Table 5
AUC scores of different classifiers.

Classifiers	AUC score
Random forest	0.9967
Support vector machine (SVM)	0.9906
k-Nearest neighbor classifier	0.9881
Logistic regression	0.9558

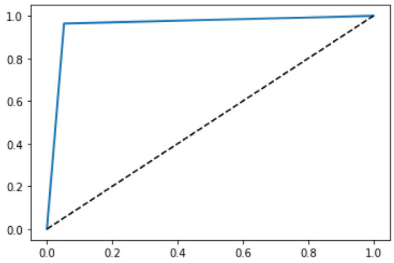

Fig. 6(a). (a) ROC curve of logistic regression.

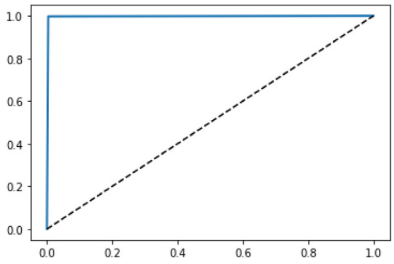

Fig. 6(b). ROC curve of random forest.

precision score, recall score, and f1 score presented in Table 2. It is common in machine learning that whenever the recall value increases, the precision decreases. This is called a precision–recall trade-off. To gain insight into how precision is varying with recall, we have also built the precision–recall curve. We have used one more metric, i.e., receiver operating characteristic (ROC) curve. ROC curve plots the true positive rate against false positive rate (FPR), which helps to calculate the accuracy of an algorithm.

Fig. 6 presents the ROC curve of different classifiers. If any algorithm covers the maximum area under the ROC curve, then it is called the best classifier. It can be seen in Fig. 6 that the maximum area in the ROC curve is covered by random forest classifier and minimum by logistic regression. A perfect classifier has the area under the curve (AUC) score equal to 1, whereas a random classifier has a score equal to 0.5. Table 5 shows the AUC score of different classifiers.

In Fig. 7, recall is plotted along the X-axis, and precision is plotted along the Y-axis. The green dotted line depicts how precision is decreasing when there is an increase in recall score. Fig. 7 shows that, initially, every classifier has a precision of 100%; however, it started decreasing after a certain threshold value. The most inclined slope of the green dotted line is logistic regression, i.e., Fig. 7(a) shows that precision is decreasing significantly with the increase in recall value, whereas the least inclined slope of random forest, i.e., Fig. 7(b), shows that there is almost no decrease in precision as recall value increases.

Hyperparameter tuning is a process of choosing the right parameters for a classifier — for example, in a random forest classifier, the accuracy depends on the choice of the right amount of trees in the forest, the total depth of the tree, and maximum leaf nodes. Selecting the right parameter will boost the accuracy of a classifier. We have performed hyperparameter tuning for all the classifiers; however, only SVM gets tuned well. Default parameters were found to be the best parameters in the rest of the classifiers. Fig. 8 shows the result of a false positive rate, precision, recall, F1 score before and after hyperparameter tuning

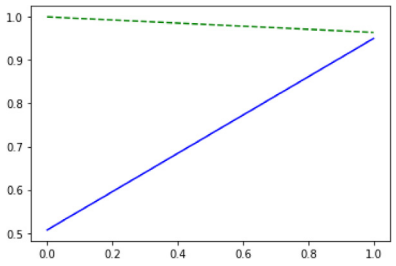

Fig. 7(a). Precision–recall trade off in logistic regression.

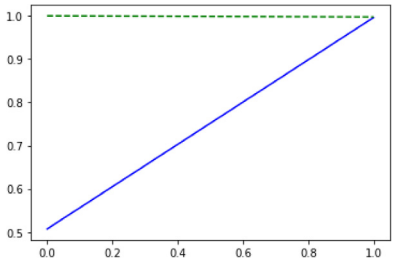

Fig. 7(b). Precision–recall trade off in random forest.

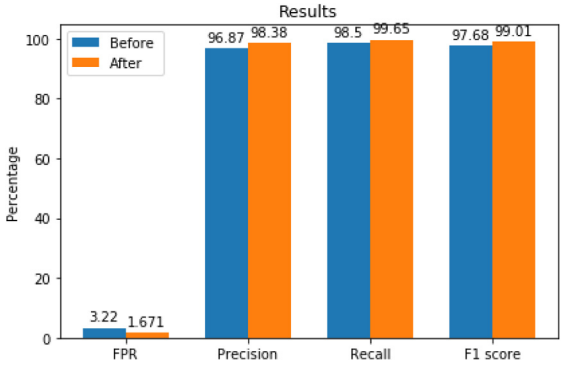

Fig. 8. Result of SVM classifier before and after hyperparameter tuning.

of the SVM classifier. From Fig. 8, we can see that there is an increase in precision from 96.68% to 98.38% after hyperparameter tuning.

Our proposed features are very accurate and give the highest accuracy. However, we apply three different algorithms to measure the importance of features. We have used Spearman correlation, K best, and Random forest to calculate the importance score of each feature. Fig. 9 represents the feature importance scores. The score is varied between 0 and 1. The most important features have a score near 1, whereas less important features have scores nearer to zero. Fig. 9 shows that every algorithm yields different importance to the same features. However, it can be concluded that delimiter in a path, delimiter in a domain, number of dots in a URL, length of URL are some most important features among all.

Response time is the time between the URL fed, and the result predicted. Several processes are carried out during response time, such as feature extraction, preparation of dataset, loading the module, and predicting the result as phishing or legitimate. The minimal amount of response time is very important to prevent phishing attacks in real-time. In our analysis, we obtained response time by testing our anti-phishing solution with some random URLs. The obtained response time from different classifiers is shown in Table 6. The table suggests that random forests have a maximum response time of 51.5 ms, whereas the least response time of 12.2 ms is obtained from SVM. On average, our response time is 22.5 ms, which is very fast to detect phishing attacks. This

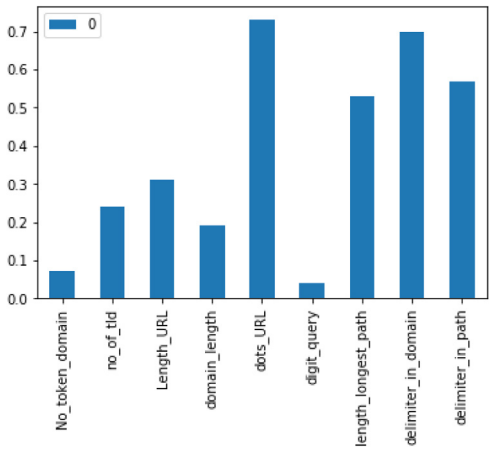

Fig. 9(a). Spearman score.

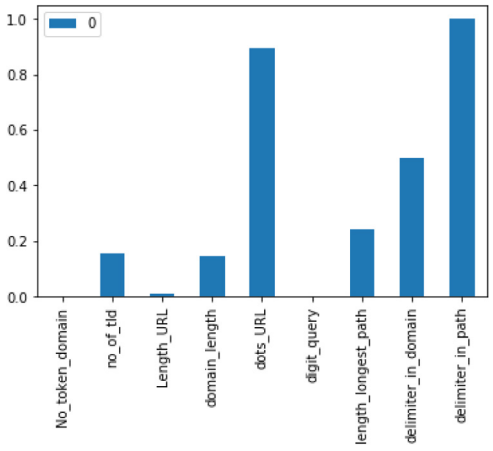

Fig. 9(b). K best score.

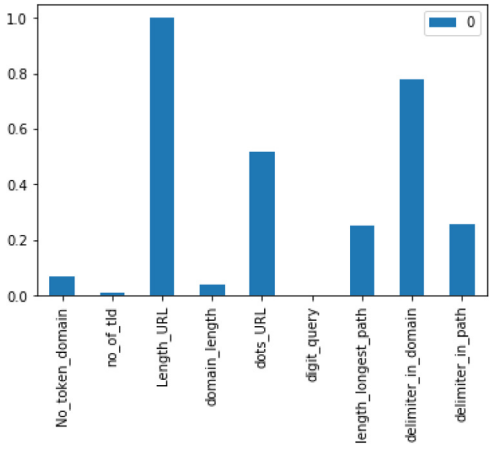

Fig. 9(c). Random forest score.

runtime has a nearly zero percent chance of giving time to the hackers for stealing the credentials from the users. However, this runtime may vary on other systems since runtime is dependent on the configuration of the system.

The data distribution of lexical features varies from legitimate URLs to phishing URLs. Fig. 10(a) shows the distribution of data in legitimate URLs, and Fig. 10(b) shows the distribution of data in phishing URLs. These figures show that the length of legitimate URLs in our dataset

Table 6
Response time of different classifiers.

Classifier	Response time (ms)
Random forest	51.5
Support vector machine (SVM)	12.2
k-Nearest neighbor classifier	12.3
Logistic regression	14
Average	22.5

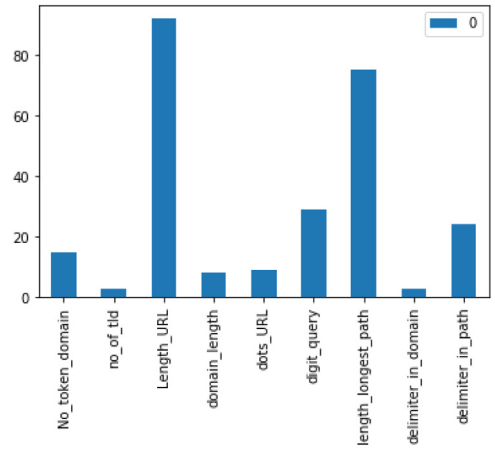

Fig. 10(a). Distribution of legitimate data.

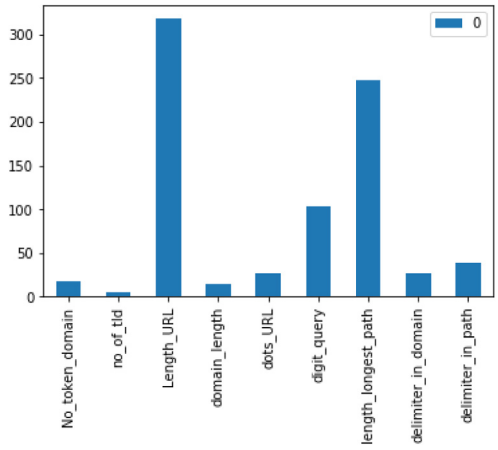

Fig. 10(b). Distribution of Phishing data.

is up to 90 characters, whereas for phishing URLs, it is up to more than 300 characters. Similarly, the length of the path in a legitimate URL is up to 80 characters, whereas in phishing URLs, it is up to 250 characters. Hackers generally try to hide the malicious information inside the long path. In general, we can see that the phishing URLs have a higher distribution of lexical data than legitimate URLs.

We have compared our anti-phishing approach with other anti-phishing approaches based on False positive rate(FPR), accuracy, third-party independency(TPI), and whole page independency(WPI). In [43–45] authors have used third party features. These third-party dependent approaches require high response time, which is not reliable in terms of detecting phishing URLs. However, our proposed approach does not rely on the third party and delivers a very fast response time of 51.5 ms. The approach proposed by the authors [46] requires downloading the source code of the website along with the URL. The source code gives the information regarding the hyperlinks and CSS used that can be used as features to classify between legitimate and phishing websites. Some hackers try to embed the malicious code inside the source code of a

Table 7
Result of different approaches available in a literature.

Approach	FPR (%)	Accuracy (%)	TPI	WPI
Sahingoz et al. [16]	2.95	97.98	Yes	Yes
Jain & Gupta et al. [17]	1.25	99.09	Yes	No
Abdelhamid et al. [43]	n/a	95	No	No
Moghimi et al. [18]	1.35	98.65	Yes	Yes
Chiew et al. [44]	13	93.4	No	No
Xiang et al. [45]	1.4	95.8	No	No
Alsarier et al. [48]	0.0018	98	No	No
Zamir et al. [49]	n/a	97.2	No	Yes
Azeez et al. [50]	n/a	99.1	Yes	No
Jain & Gupta et al. [51]	1.52%	98.4%	Yes	No
Our proposed approach	0.53	99.57	Yes	Yes

website [47]. The extraction of numerous features from the source code increases the feature extraction time. In our proposed approach, the use of limited features reduces this feature extraction time.

Authors at [48] stated that the single classifier is not sufficient to detect phishing attacks accurately, so they integrated different AI meta learners classifiers. Their proposed approach resulted in an accuracy of 98%. Although their proposed approach is very reliable, our anti-phishing approach outperformed them in terms of accuracy. Moreover, to detect URLs, they need third party information, which increases the response time, whereas our approach does not require any such information. Authors at [49] have ensemble the machine learning classifier, i.e., K Nearest Neighbor, Random forest, and bagging, and have obtained the accuracy of 97.2%; however, the FPR and accuracy produced by our methods outperforms their approach and their approach also suffers from whole page independency. Authors at [50] proposed a very novel solution that checks the consistency of URLs with their content. The consistency is a measure of the compatibility between the brand name associated with the URLs and its content. They achieved a very accuracy of 99.1%; however, their proposed approach needs to extract information from the URLs as well as from the content of the web page. This increases the response time of detecting phishing URLs and is not suitable when implemented with resource-constrained devices. Authors at [51] have developed a hyperlinks information-based approach to detect phishing URLs. Their approach obtains information from source code and constructs a DOM tree to extract hyperlink based features. Their proposed approach produces an accuracy of 98.4%, FPR of 1.52%, and FNR of 1.61%. Our proposed approach outperforms their approach in terms of accuracy, FNR, and FPR.

In Table 7, we can observe that our proposed approach has achieved the highest accuracy among all the existing approaches. A good anti-phishing approach must not classify a phishing URL as legitimate. Our proposed approach has only 0.53% of FPR, which makes our anti-phishing approach best among all the approaches. Comparative analysis has been shown in Table 7.

4.5. Novelty and discussions

These days phishing attacks have been massively targeted to resource-constrained devices like IoT [52]. Phishing attacks are made in IoT devices for multiple purposes. Some attackers try to steal credentials, whereas some attackers try to install the malware in these devices. To protect these devices, we need an anti-phishing approach that can detect phishing attacks without utilizing much resources and bandwidth. Since our approach does not need any third-party information, it will consume less bandwidth, and the use of limited features will decrease the overall resources utilization while performing feature extraction from URLs. In addition to resources constrained devices, our anti-phishing approach can also be integrated with any devices that are more prone to phishing attacks. The other several reasons that make our anti-phishing approach very reliable are described below.

1. ***Third-party independency*:** In some of the previous work [43–45] defined in the literature, phishing URLs are detected by searching in the database of blacklisted URLs. Some of the approaches also need third-party information, such as whois records, DNS records. These approaches are time-consuming, whereas our proposed solution overcomes all these limitations. Our approach detects the phishing URL only by looking at the lexical properties of the URL. Our anti-phishing approach detects the phishing attack even without an Internet connection within milliseconds.
2. ***Real-time detection*:** Our proposed approach requires low computational power, low resources and can detect the phishing URL within a millisecond. Therefore, it can be easily deployed in a cloud and IoT environment. Moreover, our proposed model can also be integrated with different applications that clients use regularly. For example, android applications, desktop applications, and chrome extensions to detect phishing URLs in a real-time environment.
3. ***Low response time***: Any anti-phishing approach must be able to detect phishing URLs quickly. Sometimes hackers are so quick that they capture the credentials within a few seconds. Our proposed approach detects the phishing URL within a millisecond. A millisecond of response time in phishing detection is very less enough to give an opportunity to hackers to steal any credentials.
4. ***Detection of new websites*:** Many of the existing approaches still use a blacklist method to detect phishing URLs. Many URLs go undetected since hackers use several techniques to create a website with lexical properties that deviate from the properties defined in the blacklisted database. Our proposed approach overcomes this problem since it uses a machine learning approach and can detect any novel URLs even if it is lexically different from phishing URLs.
5. ***Use of limited features*:** Employing a large number of features might sometimes create noise in the data and ultimately decreases the accuracy of classifiers. Moreover, usage of a large number of features requires high processing power with the increased response time. Our proposed approach uses limited features and solves the issue of high processing power and response time.

5. Conclusion and future work

In this paper, we started with an overview of phishing attacks and several existing approaches to detect this attack. We further described the limitation of all the existing approaches and formulated our novel approach in phishing detection. Our approach used only nine features based on the lexical properties of URLs and produced an accuracy of 99.57%. To clearly extract the features from URLs, we also described the feature extraction algorithms in brief. Later on, we provided the distribution of lexical data in both legitimate and phishing URLs to make readers easier to understand the difference between phishing and legitimate websites. We provided the results by evaluating our approach with different metrics and brought the detailed comparison between our approach and other phishing detection approaches that have been proposed in the literature. We clearly described the relevancy of our proposed approach in integrating with resource-constrained devices. As our future work, we will try to evaluate our approach with some sophisticated deep learning algorithms.

CRediT authorship contribution statement

Brij B. Gupta: Problem formulation, Formal analysis, Supervision. **Krishna Yadav:** Conceptualization, Nonstationary analysis, Writing - original draft. **Imran Razzak:** Problem formulation, Formal analysis, Supervision. **Konstantinos Psannis:** Problem formulation, Formal analysis, Supervision. **Arcangelo Castiglione:** Problem formulation, Formal analysis, Supervision. **Xiaojun Chang:** Problem formulation, Formal analysis, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research work is being supported by sponsored project grant YFRF, under the project Visvesvaraya Ph.D. Scheme of Ministry of Electronics & Information Technology, Government of India and being implemented by Digital India Corporation.

References

[1] Domain registered report available at: https://dofo.com/blog/domain-industry-report-april-2020/ Last accessed on May 11, 2020.
[2] Amrita Dahiya, Brij B. Gupta, A reputation score policy and Bayesian game theory based incentivized mechanism for DDoS attacks mitigation and cyber defense, Future Gener. Comput. Syst. 117 (2021) 193–204.
[3] A. Al-Nawasrah, A.A. Almomani, S. Atawneh, M. Alauthman, A survey of fast flux botnet detection with fast flux cloud computing, Int. J. Cloud Appl. Comput. (IJCAC) 10 (3) (2020) 17–53.
[4] C. Esposito, M. Ficco, et al., Blockchain-based authentication and authorization for smart city applications, Inf. Process. Manage. 58 (2) (2021) 102468.
[5] S. Kaushik, C. Gandhi, Ensure hierarchal identity based data security in cloud environment, Int. J. Cloud Appl. Comput. (IJCAC) 9 (4) (2019) 21–36.
[6] Q. Zheng, X. Wang, M.K. Khan, W. Zhang, et al., A lightweight authenticated encryption scheme based on chaotic scml for railway cloud service, IEEE Access 6 (2017) 711–722.
[7] O.O. Olakanmi, A. Dada, An efficient privacy-preserving approach for secure verifiable outsourced computing on untrusted platforms, Int. J. Cloud Appl. Comput. (IJCAC) 9 (2) (2019) 79–98.
[8] C.L. Stergiou, K.E. Psannis, et al., IoT-based big data secure management in the fog over a 6G wireless network, IEEE Internet Things J. (2020).
[9] The Security threat report of Symantec is available at https://docs.broadcom.com/doc/istr-24-2019-en Last accessed on May 11, 2020.
[10] Check Point security report available at: https://www.phishingbox.com/assets/files/images/Check-Point-Research-Information-Security-Report-2018.pdf. Last accessed on May 11, 2020.
[11] The phishing loss report produced by IBM is available at: https://www.ibm.com/security/data-breach. Last accessed on May 11, 2020.
[12] R. Dhamija, J.D. Tygar, M. Hearst, Why phishing works, in: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI2006), 2006, 581-59.
[13] J.S. Downs, M.B. Holbrook, L. Cranor, Decision strategies and susceptibility to phishing, in: Proceedings of the Second Symposium on Usable Privacy and Security (SOUPS 2006), 2006, pp. 79–90.
[14] Information about existing anti-phishing software is available at https://en.wikipedia.org/wiki/Anti-phishing_software Last accessed on May 15, 2020.
[15] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, An empirical analysis of phishing blacklists, 2009, https://kilthub.cmu.edu/articles/An_Empirical_Analysis_of_Phishing_Blacklists/6469805.
[16] O.K. Sahingoz, E. Buber, O. Demir, B. Diri, Machine learning base phishing detection from URLs, Expert Syst. Appl. 117 (2019) 345–357.
[17] A.K. Jain, B.B. Gupta, Towards detection of phishing websites on client-side using machine learning based approach, in: Telecommunication Systems, Springer, 2018.
[18] M. Moghimi, A.Y. Varjani, New rule-based phishing detection method, Expert Syst. Appl. 53 (2016) 231–242.
[19] S. Afroz, R. Greenstadt, Phishzoo: Detecting phishing websites by looking at them, in: Fifth IEEE International Conference on Semantic Computing, 2011.
[20] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, An empirical analysis of phishing blacklists, 2009, https://kilthub.cmu.edu/articles/An_Empirical_Analysis_of_Phishing_Blacklists/6469805.
[21] P. Prakash, M. Kumar, R.R. Kompella, Phishnet: predictive blacklisting to detect phishing attacks, in: Mini-Conference at IEEE INFOCOM, 2010.
[22] J. Ma, L.K. Saul, S. Savage, G.M. Voelker, Beyond blacklists: learning to detect malicious web sites from suspicious URLs, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009.
[23] H. Wang, Z. Li, Y. Li, B.B. Gupta, C. Choi, Visual saliency guided complex image retrieval, Pattern Recognit. Lett. 130 (2020) 64–72.
[24] S. Haruta, H. Asahina, I. Sasase, Visual similarity-based phishing detection using image and CSS with target website finder, in: IEEE Global Communications Conference, 2017.
[25] S. Abdelnabi, K. Krombhoiz, M. Fritz, WhiteNet: Phishing website detection by visual whitelists, in: Cryptography and Security, 2019, (arXiv).
[26] V. Patil, P. Thakkar, C. Shah, T. Bhat, Detection and prevention of phishing websites using machine learning approach, in: Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018.
[27] A.K. Jain, B.B. Gupta, Towards detection of phishing websites on client-side using machine learning based approach, in: Telecommunication Systems, Springer, 2018.
[28] T. Peng, I. Harris, Y. Sawa, Detecting phishing attacks using natural language processing and machine learning, in: IEEE 12th International Conference on Semantic Computing(ICSC), 2018.
[29] R.S. Rao, A.R. Pais, Detection of phishing websites using an efficient feature-based machine learning framework, in: Neural Computing and Applications, Vol. 31, Springer, 2019.
[30] Kholoud Althobaiti, Ghaidaa Rummani, Kami Vaniea, A review of human-and computer-facing url phishing features, in: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS & PW), IEEE, 2019.
[31] Doyen Sahoo, Chenghao Liu, Steven C.H. Hoi, Malicious URL detection using machine learning: A survey, 2017, arXiv preprint arXiv:1701.07179.
[32] M. Maalouf, Logistic regression in data analysis: an overview, Int. J. Data Anal. Tech. Strateg. (2011).
[33] C. Crisci, B. Ghattas, G. Perera, A review of supervised machine learning algorithms and their applications to ecological data, Ecol. Model. 240 (2012) 113–122.
[34] C. Crisci, B. Ghattas, G. Perera, A review of supervised machine learning algorithms and their applications to ecological data, Ecol. Model. 240 (2012) 113–122.
[35] G. Biau, E. Scornet, A random forest guided tour, 2016, pp. 197–227.
[36] C. Croux, C. Dehon, Influence functions of the Spearman and Kendall correlation measures, in: Statical Method and Applications, 2010, pp. 497–515.
[37] D.S. Modha, W.S. Spangler, Feature weighting in k-means clustering, in: Machine Learning, Springer, 2003, pp. 217–237.
[38] B.H. Menze, B.M. Kelm, R. masuch, U. Himmerreich, A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data, in: BMC Bioinformatics, Springer, 2009, 213.
[39] Legitimate and phishing URLs we have used is available here: https://www.unb.ca/cic/datasets/url-2016.html. Last accessed on May 10, 2020.
[40] F. Kamiran, T. Calders, Data preprocessing techniques for classification without discrimination, Knowl. Inf. Syst. (2012) 1–33.
[41] Standard scaling theory available at : https://en.wikipedia.org/wiki/Feature_scaling. Last accessed on July 20, 2020.
[42] I.U. Haq, I. Gondal, P. Vamplew, S. Brown, Categorical features transformation with compact one-hot encoder for fraud detection in distributed environment, in: Australasian Conference on Data Mining, 2018, pp. 69–80.
[43] N. Abdelhamid, A. Ayesh, F. Thabtah, Phishing detection based associative classification data mining, Expert Syst. Appl. 41 (13) (2014) 5948–5959.
[44] K.L. Chiew, E.H. Chang, W.K. Tiong, Utilisation of website logo for phishing detection, Comput. Secur. 54 (2015) 16–26.
[45] G. Xiang, J. Hong, C.P. Rose, L. Cranor, Cantina+ a feature-rich machine learning framework for detection of phishing web sites, ACM Trans. Inf. Syst. Secur. (2011) 21.
[46] A.K. Jain, B.B. Gupta, Towards detection of phishing websites on client-side using machine learning based approach, in: Telecommunication Systems, Springer, 2018.
[47] S. Gupta, B.B. Gupta, PHP-sensor: a prototype method to discover workflow violation and XSS vulnerabilities in PHP web applications, in: Proceedings of the 12th ACM International Conference on Computing Frontiers, 2015, pp. 1–8.
[48] Yazan Ahmad Alsariera, et al., Ai meta-learners and extra-trees algorithm for the detection of phishing websites, IEEE Access 8 (2020) 142532-142542.
[49] Ammara Zamir, et al., Phishing Web Site Detection using Diverse Machine Learning Algorithms, The Electronic Library, 2020.
[50] Nureni Ayofe Azeez, et al., Identifying phishing attacks in communication networks using URL consistency features, Int. J. Electron. Secur. Digit. Forensics 12 (2) (2020) 200–213.
[51] Ankit Kumar Jain, Brij B. Gupta, A machine learning based approach for phishing detection using hyperlinks information, J. Ambient Intell. Humaniz. Comput. 10 (5) (2019) 2015–2028.
[52] A. Tewari, B.B. Gupta, Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework, Future Gener. Comput. (2020).