Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

WCA: Integration of Natural Language Processing Methods and Machine Learning Model for Effective Analysis of Web Content to Classify Malicious Webpages

Shaheetha Liaquathali[1,*], Vadivazhagan Kadirvelu[1]

[1] Department of Computer and Information Science, Annamalai University, Tamil Nadu 608002, India

ARTICLE INFO

Article history:
Received 7 December 2023
Received in revised form 12 May 2024
Accepted 23 May 2024
Available online 20 June 2024

Keywords:

Count; Term frequency and inverse document frequency; Machine learning model; Phishing; Malicious webpages

ABSTRACT

Malicious websites have become a pervasive concern in the digital realm, targeting careless users as well as organizations. It may result in substantial financial losses, identity theft, data intrusions, and damage to reputation. In order to create efficient countermeasures, it is crucial to comprehend the effects of interacting with such websites. There are several ways to classify malicious webpages. Web content analysis is one such way for protecting internet users from malicious activities. It entails analysing websites to identify potential hazards, such as phishing attempts, malware distribution, and fraudulent activities. Traditional methods relied on rule-based systems, but recent advances in natural language processing and machine learning have opened up new avenues for increasing the precision and scalability of web content analysis to classify malicious webpages. Most of the exciting research work focuses more on URL alone for risk free processing. This paper introduces novel method for analysing web contents especially textual contents of the webpages for classification. Among various tags in web technology, proposed method focuses on div, paragraph and meta tags. The textual contents of these tags are extracted and vectorized using three different vectorizers in natural language processing and classify the webpages using machine learning models. Seven different machine learning models are used for performance evaluation. The result shows that a combined textual content of three distinct tags with count vectorizer + random forest achieves the higher accuracy of 93.46% with 1000 features.

1. Introduction

Malicious websites offer serious security risks to the digital world. These websites deceive users into giving important information, infecting their devices with malware, or other unlawful acts [1]. Phishing attacks employ malicious websites to deceive visitors into entering sensitive information like login passwords or bank details [2].

* *Corresponding author.*
E-mail address: shahee.aasc@gmail.com

https://doi.org/10.37934/araset.47.1.105122

Malicious websites serve as platforms for the distribution of malicious software, such as viruses, worms, ransomware, and spyware. These dangerous programs may be unintentionally downloaded and executed by users, compromising systems or destroying data [3]. Malicious websites can use browser or plugin vulnerabilities to automatically download and install malware on visitors' devices [4]. Social engineering, such as phony login prompts or urgent alerts, is used by malicious websites to trick users into providing sensitive information or executing hazardous activities [5]. By effectively recognizing and blocking these webpages, users can avoid financial loss, data breaches, and unauthorized access to sensitive information. It prevents infections and malware proliferation, decreasing the impact on individuals, organizations, and the internet ecosystem [6]. Malicious webpage classification can be done using many methods. The blacklist method [7] is a well-known technique to detect dangerous websites. Blacklisting involves keeping a database of harmful websites and comparing visiting webpage's URLs or other identifiers to it. If matched, the page is harmful. Security companies and organizations frequently generate and maintain blacklists. Despite being a simple technique, it has some limitations.

This method is reactive in nature; therefore, it can only block websites that have already been classified as harmful and placed on the blacklist. It may take time for new threats to be detected and added to the blacklist.

i. Keeping a blacklist current takes time and resources. Security teams must regularly check and update the blacklist to add new threats and remove outdated items.
ii. Blacklisting can consume considerable computational and network bandwidth. For every webpage request, matching URLs or other identifiers against a big blacklist database can cause latency and degrade security system performance.

The next method is heuristic analysis [8] which is the process of looking at different aspects of a website by using a set of rules or formulas that have already been set up. These criteria are intended to spot patterns frequently connected to malicious behaviour, such as suspicious URLs, a lot of pop-up advertising, concealed iframes, or obfuscated JavaScript code. Compared to blacklisting, heuristic analysis is a better generalize detection method, although it still has significant drawbacks.

i. Rules may not encompass the full spectrum of benign webpages, resulting in false positives in which benign webpages are incorrectly identified as malicious. False positives can interfere with user access and degrade their overall experience.
ii. Occasionally, heuristic analysis may over generalize features associated with malicious websites. For instance, if a heuristic rule flags a certain JavaScript function as possibly harmful, it may do the same for legitimate websites that employ the function for legitimate purposes.
iii. Heuristic analysis, especially large-scale or real-time analysis, can slow performance.

To overcome these limitations, researchers prefer to implement a machine learning model capable of autonomously classifying websites as malicious or benign [9]. In this method, a huge dataset of webpages that have been classified as malicious or benign is gathered, appropriate features are extracted (such as URL structure, HTML content, and JavaScript behaviour), and a machine learning model is trained to classify webpages based on these features. This paper focuses on processing textual contents of the webpages which include malicious and benign. Train and classify these websites using machine learning models. URL lexical analysis is the most common classification technique for malicious webpages due to its risk-free nature. URL lexical analysis

focuses predominantly on the structure, syntax, and components of a URL to identify patterns or indicators of malicious intent. The most widely used features are domain, subdomain, path, query parameters, special characters, length, etc. URL lexical analysis is often faster than webpage content analysis because it only entails analysing the URL. However, it has limited features and does not consider contextual information about the webpage. In URL lexical analysis, it is crucial to identify significant features that can distinguish between benign and malicious URLs [10]. However, not all extracted features are informative or contribute significantly to classification. It may not fully utilize the dataset's capabilities. Furthermore, redundant features can add computational effort and noise to analysis. On the other side, webpage content analysis is the process of looking at a website's text, pictures, scripts, metadata, and other embedded elements. Web page content analysis can capture the semantic meaning and context of the content, enabling a more in-depth understanding of the webpage's purpose and possible risks. As contrasted to URL lexical analysis, webpage content analysis may necessitate loading and parsing the webpage, which introduces a delay. This research work focuses on textual contents from div, paragraph and anchor tags of the webpages.

Most of the recent research uses features that can be generated from webpage contents or URLs for classification. But in this research, employs natural language processing (NLP) techniques to solve the feature generation problem. NLP methods can convert text into numerical representations that may be used as features in machine learning models. These representations utilize vectorizers, such as bag-of-words, TF-IDF, and hashing, which capture the salient information of the text while preserving the semantic relationships between words [11]. Seven different machine learning algorithms are utilized for experiments and performance assessments. The result shows that combined textual contents of three distinct tags with count vectorizer + random forest achieve the higher accuracy of 93.46%with 1000 features. This paper makes several significant contributions to the field of web content analysis and malicious webpage classification. First, we introduce a novel method that goes beyond traditional URL-centric approaches, focusing on the analysis of textual content within div, paragraph, and meta tags. This departure from the norm enables a more comprehensive understanding of webpage content. Second, the incorporation of three different vectorizers in natural language processing enhances the extraction and processing of textual information, contributing to the precision of our classification model. Third, the utilization of seven distinct machine learning models for performance evaluation demonstrates the robustness and versatility of our approach. Lastly, our results reveal a remarkable accuracy of 93.46% with 1000 features, underscoring the efficacy of our proposed method. These contributions collectively advance the state-of-the-art in web content analysis and provide valuable insights for the development of more effective countermeasures against malicious activities on the internet.

The structure of the paper is as follows. Part I discusses the significance of the research and current methodologies. Additionally, it emphasizes the contribution of the current paper. Part II covers previous study, findings, and remarks. The proposed work is fully described in Part III. The experimental results are shown in Part IV. The paper is concluded in Part V.

2. Related works

Malicious website detection protects users from online threats like malware, phishing, and data theft. The researchers devised a number of techniques for identifying fraudulent websites. The malicious website's features, including HTML and JavaScript, were listed by Wan *et al.,* [12]. Malak *et al.,* [13] evaluated a dataset of 66,506 URLs to detect dangerous websites using ML and DL models. Three different kinds of features such as lexical-based, network-based, and content-based were utilized for detection for malicious webpages. The dataset's most discriminative properties were

extracted using correlation analysis, ANOVA, and chi-square. The results showed that Nave Bayes (NB) was the best model for detecting malicious URLs using the applied data, with an accuracy of 96%.A framework for the detection of malicious web pages is provided by Sirageldin *et al.,* [14]. When utilizing machine learning algorithms to identify fraudulent web pages, two different factors (URL keywords and page content) were considered. The dataset included both good and bad websites. Alexa, malwareurl, phishtank, malwaredomainlist.com, StopBadWare, mwsl.org.cn dataset was used. The gathered features are divided into two categories: training and test. The model had a 97% accuracy rate and no false positives. Machine learning and deep learning were used by Saleem *et al.,* [15] to classify harmful webpages by content. The approach uses only HTML tags, event methods, DOM keywords, and JavaScript functions from online content. The investigations made use of data from the Kaggle dataset. More than 206 features are extracted from the dataset. The top-scoring features are chosen for the experiment using the selectKbest method. The results show that support vector machine (SVM) has 88% accuracy and random forest (RF) 93%.Desai *et al.,* [16] proposes a Chrome Extension to help users spot phishing websites. The experiment made use of the UCI dataset. The dataset has 11055 records with 30 features. Only 22 of the dataset's 30 features were taken into consideration for the experiment. The outcome reveals that the Random Forest algorithm achieves 96% accuracy, SVM achieves 93.5% accuracy, and KNN algorithm achieves 93% accuracy. Saleem *et al.,* [9] suggested a lightweight malicious URL detection algorithm. The authors note that blacklists and reputation-based malicious URL detection methods are incapable of detecting new threats. To address this problem, the authors propose a novel approach that employs a set of lexical features, such as the URL's length, the presence of specific characters, and the number of subdomains, to train a machine learning model to detect malicious URLs. According to the study, k-nearest neighbour (k-NN) algorithms and random forest (RF) algorithms can both detect malicious URLs with 98% and 99% accuracy, respectively. With the help of rank-based, bag-of-words, web page-based, and lexical criteria, Pradeepa *et al.,* [17] devised a method for identifying malicious URLs. The datasets from Phistank and Kaggle were used. The outcome demonstrates that Random Forest offers 99% accuracy. Saleem *et al.,* [18] implements NLP to vectorize URL terms. Machine learning and deep learning models are used for classification. Two datasets (D1 and D2) are utilized for the experiment. Three vectorization methods vectorize URL text such as Count, TF-IDF, Hashing. With the D1 dataset, the Decision Tree (DT) with count vectorizer and Random Forest (RF) with TF-IDF vectorizer both reach 92.4% accuracy. With the D2 dataset, the Decision Tree (DT) with TF-IDF vectorizer achieves a higher accuracy of 99.5%. The Artificial Neural Network (ANN) model has an accuracy of 89.6% with the D1 dataset and 99.2% with the D2 dataset. Table 1 summarizes the recent works in the field.

Table 1
Summary of recent works

No	Proposed Method	Features	Accuracy	Remarks
1	Malak *et al.,* [13]	URL lexical features, Content features, Reputation features, Network features	Nave Bayes: 96%	The dataset is underutilized. Depending on the features chosen, the outcome could vary.
2	Sirageldin *et al.,* [14]	URL keywords and page content	97%	Features are limited. Depending on the features chosen, the outcome could vary.
3	Saleem *et al.,* [15]	HTML tags, event methods, DOM keywords, and JavaScript functions	SVM: 88% RF: 93%	Features are limited. Longer processing time. Unable to be extended.
4	Desai *et al.,* [16]	URL lexical features, Content features, Reputation features, Network features	RF: 96%, SVM: 93.5% KNN: 93%	Features are limited. Depending on the features chosen, the outcome could vary.
5	Saleem *et al.,* [9]	URL lexical features, Content features, Reputation features, Network features	K-NN: 98%, RF: 99%	Features are limited. Depending on the features chosen, the outcome could vary.
6	Pradeepa *et al.,* [17]	Rank-based features, Bag-of-words features, Web page content features, URL lexical features	RF: 99%	Features are limited. Depending on the features chosen, the outcome could vary
7	Saleem *et al.,* [18]	URL vectorized features using NLP method	DT with TF-IDF vectorizer: 99.5%.	URLs alone may not be adequate for better classification.

While the existing body of literature has made commendable strides in the realm of malicious webpage classification, our proposed system aims to bridge a notable research gap. Prior studies have predominantly focused on URL-centric analyses, often overlooking the comprehensive examination of textual content within webpages. The proposed system introduces a novel approach by prioritizing the analysis of div, paragraph, and meta tags, thereby providing a more nuanced understanding of webpage content. This departure from conventional methodologies addresses the need for a more holistic examination of potential threats, ultimately contributing to the advancement of web content analysis in the context of malicious activities. Our work thus complements and extends the existing research landscape by offering a fresh perspective and insights into an underexplored facet of malicious webpage classification.

3. Proposed Systems

The proposed system (Web Content Analysis-WCA) incorporates various machine learning techniques including Logistic Regression (LogR), Gaussian Naive Bayes (GNB), K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), Gradient Boosting (GB), and Extreme Gradient Boosting (XGB). For assessing the model's efficacy, it is essential to ensure accurate data selection and balancing. The experiment utilized well-established datasets such as URL dataset (ISCX-URL2016) [19], UNB [20], and phistank [21]. Imbalanced data poses a common challenge in machine learning, where one class has significantly more samples than the others [21-24]. This imbalance can lead to biased models that prioritize the majority class and perform poorly with the minority class [25,26]. In order to resolve this issue and prevent skewed results, an equal number of benign and malicious URLs have been selected for the experiment. Table 2 provides the summary of the total benign and malicious URLs used.

Table 2
Dataset Summary

No	Type	Count
1	Benign	5530
2	Malicious	5882

In our experiment, focuses solely on the textual content of webpages especially textual contents in <para>, and <meta> tag. Algorithm 1 presents the text extraction process. Most of the textual content on webpages is organized using by using these tags. Paragraphs in HTML are defined by the <p> tag. It is one of the most popular tags for organizing and exhibiting textual content on web pages. HTML's tag is a versatile and widely used tag for dividing a webpage. To organize the content, tags are utilized. The <meta> tag adds metadata to a webpage. Metadata delivers vital information to browsers, search engines, and other web services.

```
    Algorithm #1: Textual content extraction
    function
extract_text_from_webpage(html_content):
    div_texts = []
    meta_texts = []
    para_texts = []

      # Parse the HTML content
      soup = parse_html(html_content)

      # Extract text from <div> tags
    div_tags = find_div_tags(soup)
      for div_tag in div_tags:
    div_text = extract_text(div_tag)
        if div_text is not empty:
    add_to_list(div_texts, div_text)

      # Extract text from <meta> tags
    meta_tags = find_meta_tags(soup)
      for meta_tag in meta_tags:
    meta_text = extract_content(meta_tag)
        if meta_text is not empty:
    add_to_list(meta_texts, meta_text)

      # Extract text from <p> tags
    para_tags = find_para_tags(soup)
      for para_tag in para_tags:
    para_text = extract_text(para_tag)
        if para_text is not empty:
    add_to_list(para_texts, para_text)

      # Return the extracted text from all the
tags
      return div_texts, meta_texts, para_texts
```

i. parse_html (html_content) represents the function to parse the HTML content and create a parse tree.
ii. find_div_tags(soup), find_meta_tags(soup), and find_para_tags(soup) represent functions to locate , <meta>, and <p> tags respectively in the parse tree.
iii. extract_text (tag) represents the function to extract the text content from a given tag.
iv. extract_content(tag) represents the function to extract the content attribute value from a given tag.
v. add_to_list(list, text) represents the function to add non-empty text to a given list.

Textual data must first be transformed into numbers because machine learning algorithms often operate on numerical data. This method, referred to as vectorization, involves converting text data into a numerical format that machine learning algorithms can understand. Algorithm 2 presents the text vectorization process. Vectorization methods like Count, Term Frequency-Inverse Document Frequency (TF-IDF), and Hashing are used to encode textual data as numerical vectors [22]. These methods record the semantic and syntactic connections between various words and sentences.

```
    Algorithm #2. Text Vectorization
    function vectorize_text(texts,
max_features):
        vectorizer = initialize_vectorizer()

        # Set the maximum number of features
    set_max_features(vectorizer,
max_features)

        # Fit the vectorizer on the text data
    fitted_vectorizer =
fit_vectorizer(vectorizer, texts)

        # Transform the text data into feature
vectors
    feature_vectors =
transform_text(fitted_vectorizer, texts)

        return feature_vectors
```

i. initialize_vectorizer() represents the function to initialize the text vectorizer. The choice of vectorizer (e.g., Bag-of-Words, TF-IDF, and Word2Vec) and the configuration of parameters would depend on the specific vectorization technique you want to use.
ii. set_max_features(vectorizer,max_features) represents the function to set the maximum number of features for the vectorizer. This ensures that the resulting feature vectors have the desired dimensionality.
iii. fit_vectorizer(vectorizer, texts) represents the function to fit the vectorizer on the given texts. This step calculates any necessary statistics or parameters from the text data required for vectorization.

iv. transform_text (fitted_vectorizer, texts) represents the function to transform the text data into feature vectors using the fitted vectorizer. This step converts the textual data into numerical representations according to the chosen vectorization technique.

v. feature_vectors is the resulting feature matrix containing the vectorized representation of the input texts.

The proposed method (Web Content Analysis-WCA) contains three steps. The first step entails extracting the required text from the three distinct tags of the web pages. Text is pre-processed during the second step. Preprocessing includes all the steps that are needed for text vectorization. Cleaning text eliminates special characters and only takes into account text and numerals. After cleaning the text, it will be changed to lowercase, stop words will be removed, and it will be lemmatized to reduce the number of features while vectorizing. Figure 1 depicts the entire process.

Several NLP approaches, such as bag of words, TFIDF, and hash vectorizer, are used to convert text into a real-valued vector. In step three, seven different machine learning algorithms are used to analyse the data that has been vectorized.

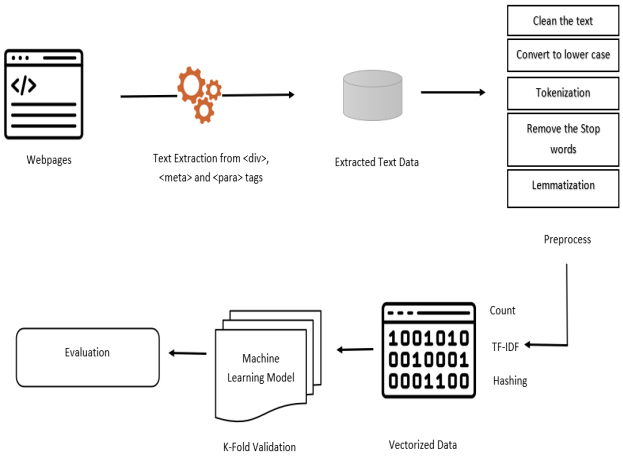

Fig. 1. Proposed System (WCA)

Several NLP approaches, such as bag of words, TFIDF, and hash vectorizer, are used to convert text into a real-valued vector. In step three, seven different machine learning algorithms are used to analyse the data that has been vectorized.

The experimental setup consists of Windows 10, an I5 processor running at 3.2 GHz, and 8 GB of RAM. Python is used for programming in Jupyter Notebook with the sklearn package. The performance metrics that are considered are accuracy, precision, recall, and f1-score. The number of features that vectorizers output ranges from 500 to 2000. The contents of each tag are separately evaluated. Table 3 to Table 6 show the results of count vectorizer experiments.

Table 3
Performance of Count Vectorizer with tag

Vectorizer	Machine Learning Model	No. of Features	Accuracy	Precision	Recall	F1-Score
Count	LOGR	500	81.63	78.47	91.38	83.93
	KNN		79.74	78.60	86.16	81.54
	GNB		74.26	69.86	97.01	80.41
	DT		84.52	81.01	93.16	86.38
	RF		87.06	84.38	92.77	88.23
	GB		81.50	77.71	92.64	84.08
	XGB		85.67	82.03	93.95	87.32
	LOGR	1000	82.41	79.24	91.53	84.54
	KNN		77.80	77.39	84.68	79.98
	GNB		73.57	69.07	98.06	80.22
	DT		84.17	80.70	93.05	86.13
	RF		87.16	84.68	92.64	88.31
	GB		82.16	78.15	93.33	84.66
	XGB		85.62	82.11	93.74	87.26
	LOGR	1500	85.30	82.35	92.50	86.84
	KNN		80.56	79.50	86.93	82.36
	GNB		75.12	70.39	97.60	81.03
	DT		81.10	78.63	91.29	83.79
	RF		85.75	83.92	91.40	87.12
	GB		80.29	76.35	92.89	83.35
	XGB		86.15	82.61	93.91	87.67
	LOGR	2000	85.83	83.02	92.57	87.25
	KNN		78.88	77.47	87.54	81.45
	GNB		76.14	71.30	97.26	81.55
	DT		84.25	82.02	90.63	85.74
	RF		87.47	86.01	91.38	88.36
	GB		80.40	76.60	92.69	83.40
	XGB		85.47	81.65	94.37	87.26

Table 4
Performance of Count Vectorizer with <meta> tag

Vectorizer	Machine Learning Model	No. of Features	Accuracy	Precision	Recall	F1-Score
Count	LOGR	500	84.35	82.65	90.24	85.83
	KNN		78.07	77.04	85.31	80.26
	GNB		83.47	82.14	88.73	84.94
	DT		85.70	83.06	92.26	87.14
	RF		87.41	85.45	92.20	88.47
	GB		79.50	75.34	94.46	83.16
	XGB		84.17	81.52	91.97	85.96
	LOGR	1000	86.86	85.20	91.57	87.95
	KNN		76.63	74.51	88.66	80.00
	GNB		80.41	93.32	66.86	77.42
	DT		87.39	85.27	92.31	88.45
	RF		88.86	87.58	92.04	89.60
	GB		77.36	75.24	90.80	80.52
	XGB		84.73	81.79	92.55	86.44
	LOGR	1500	87.69	85.88	92.45	88.74
	KNN		76.99	72.96	93.33	81.26
	GNB		72.83	93.84	50.25	63.81
	DT		85.71	83.17	92.45	87.31
	RF		89.24	87.98	92.31	89.96
	GB		79.16	75.19	94.34	82.94
	XGB		85.19	82.47	92.60	86.83
	LOGR	2000	87.76	85.71	92.84	88.85
	KNN		76.65	73.67	90.60	80.36
	GNB		72.49	94.35	49.37	63.07
	DT		87.42	85.21	92.45	88.50
	RF		89.44	88.19	92.45	90.13
	GB		79.20	75.18	94.51	83.01
	XGB		85.48	82.83	92.59	87.03

Table 5
Performance of Count Vectorizer with <para> tag

Vectorizer	Machine Learning Model	No. of Features	Accuracy	Precision	Recall	F1-Score
Count	LOGR	500	79.35	75.91	91.97	82.60
	KNN		77.29	75.90	86.42	80.10
	GNB		71.65	67.43	97.81	79.00
	DT		83.47	80.54	91.72	85.49
	RF		86.90	85.68	90.94	88.04
	GB		80.55	77.79	90.97	83.25
	XGB		84.53	81.59	92.62	86.43
	LOGR	1000	81.31	78.31	91.46	83.89
	KNN		78.18	76.01	88.08	81.04
	GNB		72.12	67.81	97.57	79.21
	DT		84.00	81.04	92.33	86.02
	RF		87.81	86.88	91.46	88.88
	GB		80.71	77.92	91.09	83.39
	XGB		85.24	82.65	92.57	86.99
	LOGR	1500	84.31	81.30	92.54	86.22
	KNN		78.51	75.53	90.68	81.78
	GNB		73.66	69.03	97.26	80.03
	DT		83.54	80.66	91.96	85.61
	RF		88.24	87.21	91.58	89.18
	GB		81.71	78.93	91.02	84.02
	XGB		86.17	83.56	92.93	87.72
	LOGR	2000	85.07	82.12	92.91	86.87
	KNN		78.31	75.36	90.89	81.66
	GNB		74.37	69.63	97.25	80.46
	DT		85.48	82.80	92.14	87.03
	RF		87.29	85.68	91.86	88.47
	GB		82.06	79.03	91.65	84.39
	XGB		86.82	84.33	93.23	88.28

Table 6
Performance of Count Vectorizer with combined tags (<meta><para>)

Vectorizer	Machine Learning Model	No. of Features	Accuracy	Precision	Recall	F1-Score
Count	LOGR	500	85.02	82.34	92.76	86.84
	KNN		82.54	78.65	94.05	85.20
	GNB		75.06	70.22	97.14	80.81
	DT		89.11	87.58	92.40	89.81
	RF		92.67	93.97	91.92	92.86
	GB		84.84	82.80	91.35	86.46
	XGB		91.03	89.50	94.24	91.67
	LOGR	1000	88.58	86.28	93.61	89.58
	KNN		82.94	78.89	94.61	85.57
	GNB		76.90	71.80	97.06	81.93
	DT		89.58	87.99	92.89	90.28
	RF		93.46	95.17	92.11	93.56
	GB		85.69	83.57	92.16	87.26
	XGB		91.73	90.44	94.39	92.26
	LOGR	1500	89.39	87.48	93.47	90.22
	KNN		81.83	77.89	93.56	84.57
	GNB		80.18	75.21	96.87	84.10
	DT		88.27	87.14	91.82	89.24
	RF		93.00	94.84	91.55	93.11
	GB		85.99	83.48	92.91	87.58
	XGB		91.94	90.45	94.80	92.48
	LOGR	2000	90.02	88.35	93.54	90.74
	KNN		81.42	77.37	94.03	84.38
	GNB		79.13	73.88	97.31	83.43
	DT		88.24	87.09	91.77	89.20
	RF		92.96	94.96	91.29	93.04
	GB		86.07	83.68	92.74	87.63
	XGB		92.06	90.69	94.75	92.58

Table 7 shows the summary of performance of count vectorizer. The result shows that combined textual contents of three different tags with random forest (RF) gives better result of 93.46% accuracy with 1000 features. Timing parameters play a crucial role in the performance and efficiency of the WCA system. Given the substantial computational requirements associated with tasks such as text extraction, preprocessing, and the application of machine learning algorithms, a clear understanding of timing parameters becomes essential. This includes examining the time required for tasks like HTML parsing, tag-specific text extraction, vectorization, and model training. Furthermore, as the proposed system aims to process web content in real-time, timing considerations become pivotal for ensuring its practical applicability. Understanding the temporal aspects of the system's operations is essential for evaluating its responsiveness, especially in scenarios where timely analysis of web content is crucial, such as in cybersecurity applications.

Table 7
Summary of Count Vectorizer

Tag	Machine Learning Model	No. of Features	Accuracy	Precision	Recall	F1-Score
Div	RF	2000	87.47	86.01	91.38	88.36
Meta	RF	2000	89.44	88.19	92.45	90.13
Para	RF	1500	88.24	87.21	91.58	89.18
Combined	RF	1000	93.46	95.17	92.11	93.56

Table 8 to Table 11 shows the results of TF-IDF vectorizer experiments.

Table 8
Performance of TF-IDF vectorizer with tag

Vectorizer	Machine Learning Model	No. of Features	Accuracy	Precision	Recall	F1-Score
TF-IDF	LOGR	500	83.28	80.93	90.19	84.92
	KNN		75.07	74.51	79.39	75.75
	GNB		79.03	74.32	95.51	83.02
	DT		84.39	80.89	92.96	86.24
	RF		87.19	84.60	92.79	88.34
	GB		82.21	78.61	92.48	84.56
	XGB		85.77	82.33	93.73	87.40
	LOGR	1000	83.45	81.12	90.24	85.06
	KNN		74.54	74.18	78.90	75.58
	GNB		75.46	70.60	97.84	81.29
	DT		84.15	80.71	92.77	86.03
	RF		87.28	84.78	92.77	88.42
	GB		82.36	78.61	92.88	84.73
	XGB		85.89	82.29	94.10	87.53
	LOGR	1500	84.65	82.90	90.07	85.84
	KNN		75.90	74.25	86.43	79.04
	GNB		79.59	74.41	96.82	83.62
	DT		81.41	78.51	91.99	84.07
	RF		85.47	83.59	91.17	86.85
	GB		81.43	77.75	92.82	84.16
	XGB		86.95	83.65	94.01	88.31
	LOGR	2000	85.09	83.59	89.97	86.18
	KNN		79.20	76.61	89.15	81.79
	GNB		80.68	76.03	95.44	84.14
	DT		84.59	82.55	90.51	85.97
	RF		87.39	85.90	91.36	88.29
	GB		81.07	77.47	92.30	83.80
	XGB		87.01	83.73	94.03	88.35

Table 9
Performance of TF-IDF vectorizer with <meta> tag

Vectorizer	Machine Learning Model	No. of Features	Accuracy	Precision	Recall	F1-Score
TF-IDF	LOGR	500	85.16	83.98	89.92	86.38
	KNN		76.95	74.12	89.13	80.38
	GNB		83.57	83.58	86.58	84.63
	DT		86.22	83.57	92.62	87.60
	RF		88.61	86.74	92.88	89.52
	GB		80.44	76.10	95.03	83.90
	XGB		84.83	82.01	92.40	86.49
	LOGR	1000	87.08	86.29	90.38	87.93
	KNN		73.32	69.80	92.57	78.84
	GNB		80.87	91.21	70.14	78.73
	DT		87.75	85.06	93.49	88.89
	RF		89.82	88.29	93.13	90.52
	GB		76.16	75.85	87.43	78.69
	XGB		85.73	83.00	92.76	87.24
	LOGR	1500	87.78	87.23	90.49	88.52
	KNN		70.91	67.84	92.18	77.31
	GNB		75.87	93.46	56.81	68.84
	DT		88.31	85.80	93.52	89.34
	RF		90.17	88.64	93.45	90.86
	GB		76.05	75.85	87.04	78.50
	XGB		85.27	82.36	92.86	86.90

LOGR	2000	88.51	88.02	90.94	89.18
KNN		67.90	65.07	87.25	73.70
GNB		75.07	93.90	54.96	67.58
DT		88.26	85.88	93.11	89.22
RF		90.22	88.86	93.18	90.86
GB		76.21	75.85	87.35	78.64
XGB		85.44	82.43	93.08	87.06

Table 10
Performance of TF-IDF vectorizer with <para> tag

Vectorizer	Machine Learning Model	No. of Features	Accuracy	Precision	Recall	F1-Score
TF-IDF	LOGR	500	80.98	79.58	87.96	83.02
	KNN		76.99	74.21	89.03	80.33
	GNB		79.61	75.76	93.59	83.12
	DT		84.27	81.78	91.53	86.08
	RF		87.23	86.44	90.72	88.31
	GB		79.70	77.01	90.04	82.45
	XGB		85.40	83.21	91.87	87.02
	LOGR	1000	82.58	80.97	88.95	84.37
	KNN		71.48	68.66	83.12	74.38
	GNB		79.78	75.25	95.12	83.48
	DT		84.66	81.90	92.26	86.48
	RF		87.41	86.34	91.21	88.50
	GB		80.26	77.09	91.26	83.05
	XGB		85.45	83.09	92.21	87.11
	LOGR	1500	84.82	83.18	90.22	86.25
	KNN		73.21	69.96	91.57	78.53
	GNB		82.31	78.62	93.30	84.90
	DT		85.86	83.54	92.04	87.33
	RF		88.26	87.49	91.23	89.15
	GB		80.94	77.72	91.75	83.60
	XGB		86.61	84.19	92.82	88.03
	LOGR	2000	85.60	83.80	91.12	87.01
	KNN		72.69	69.00	93.54	78.66
	GNB		83.79	80.43	92.93	85.90
	DT		85.29	82.82	91.74	86.83
	RF		88.38	87.46	91.57	89.29
	GB		81.15	77.82	91.80	83.77
	XGB		86.61	84.09	92.98	88.06

Table 11
Performance of TF-IDF vectorizer with combined tags (<meta><para>)

Vectorizer	Machine Learning Model	No. of Features	Accuracy	Precision	Recall	F1-Score
TF-IDF	LOGR	500	87.98	87.43	90.55	88.69
	KNN		79.17	74.71	94.54	82.96
	GNB		83.87	79.60	94.92	86.18
	DT		87.05	85.54	91.18	88.05
	RF		92.74	94.25	91.74	92.90
	GB		85.00	83.31	90.94	86.52
	XGB		90.36	89.35	93.74	91.26
	LOGR	1000	89.39	88.88	91.50	89.99
	KNN		78.18	73.63	94.73	82.27
	GNB		84.24	79.93	94.78	86.39
	DT		87.86	86.58	91.96	88.96
	RF		93.19	94.74	92.04	93.32
	GB		86.52	84.29	92.77	87.98
	XGB		91.91	90.93	94.07	92.38
	LOGR	1500	89.95	89.49	91.86	90.51
	KNN		82.09	79.21	90.97	84.22
	GNB		87.95	85.23	93.84	89.11
	DT		85.66	84.28	91.04	87.17
	RF		93.12	95.09	91.48	93.20
	GB		86.27	84.16	92.30	87.69
	XGB		91.79	90.73	94.10	92.29
	LOGR	2000	90.07	89.64	91.86	90.59
	KNN		79.82	76.83	90.38	82.56
	GNB		86.45	82.04	96.31	88.31
	DT		88.19	87.32	91.35	89.11
	RF		93.22	95.14	91.65	93.31
	GB		86.36	84.31	92.37	87.81
	XGB		92.28	91.28	94.41	92.73

Table 12 shows the summary of performance of TF-IDF vectorizer. The result shows that combined textual contents of three different tags with random forest (RF) gives better result of 93.22% accuracy with 2000 features.

From the experiments, combined textual contents of three distinct tags with count vectorizer + random forest achieves the higher accuracy of 93.46%with 1000 features as shown in Table 12 and Figure 2. Binary classification models are evaluated using ROC-AUC curves. It offers a comprehensive visual representation of the discriminatory capability of a model in distinguishing positive from negative instances at different probability thresholds. Figure 3, 4 and 5 depicts the performance of each vectorizer with combined tags.

Table 12
Summary of the Combined Tags (<meta><para>)

Vectorizer	Machine Learning Model	No. of Features	Accuracy	Precision	Recall	F1-Score
Count	RF	1000	93.46	95.17	92.11	93.56
TF-IDF	RF	2000	93.22	95.14	91.65	93.31
Hashing	RF	1000	92.22	93.71	91.16	92.36

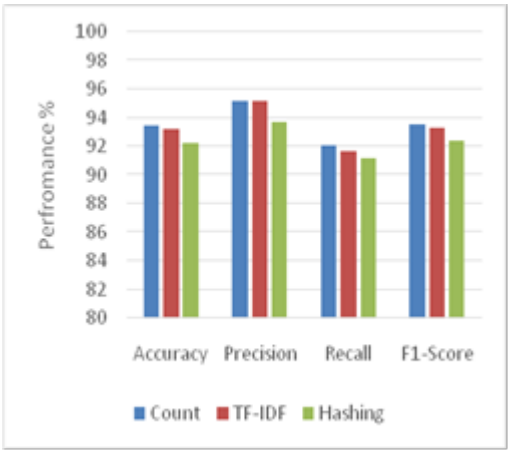

Fig. 2. Performance Comparison

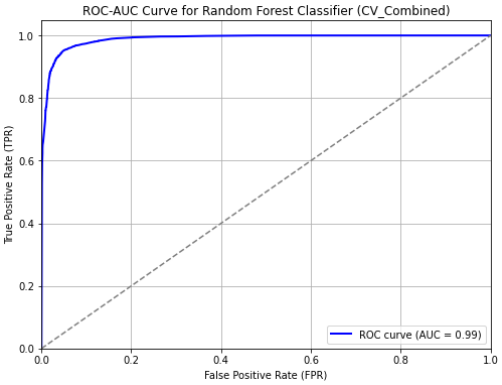

Fig. 3. Proposed System (WCA)

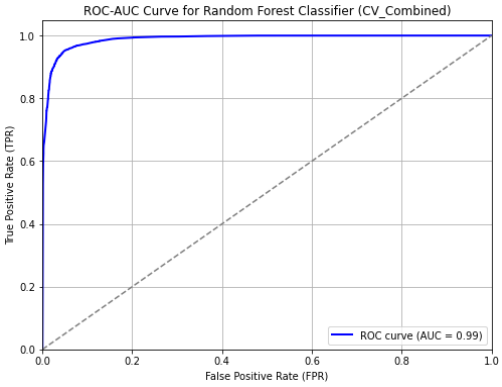

Fig. 4. ROC-AUC curve for Count Vectorizer

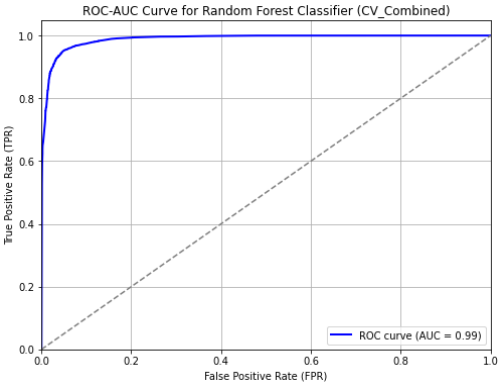

Fig. 5. ROC-AUC curve for Hashing

4. Conclusions

The detection of malicious web pages is the process of identifying and flagging web pages that contain detrimental or malicious content. This is essential for preserving cyber security and protecting users against potential attacks. In this paper, utilizes the textual contents of the webpages for malicious webpage classification. Textual information is extracted from the three distinct tags of the webpages (, <meta> and <para>).After that, NLP vectorizers convert unprocessed text data into numerical feature vectors, which machine learning algorithms can use to process and analyse the text effectively. Text in every tag is tested separately. Finally, a test is conducted on the combined textual contents. The result of the experiment reveals that combined textual contents of three distinct tags with count vectorizer + random forest achieves the higher accuracy of 93.46%with 1000 features. It is important to note that the speed of textual content extraction represents a limitation in this study, suggesting a potential avenue for future enhancements to optimize overall performance.

Acknowledgement
This research was not funded by any grant.

References

[1] Eshete, Birhanu, Adolfo Villafiorita, and Komminist Weldemariam. "Malicious website detection: Effectiveness and efficiency issues." In *2011 First SysSec Workshop*, pp. 123-126. IEEE, 2011. https://doi.org/10.1109/SysSec.2011.9
[2] National Cyber Security Centre. "Phishing". https://www.ncsc.gov.uk/guidance/phishing
[3] University of Delaware. "How is Malware Distributed." https://sites.udel.edu/infosecnews/2015/05/18/how-is-malware-distributed
[4] Kaspersky. "What is a Drive by Download." https://www.kaspersky.com/resource-center/definitions/drive-by-download
[5] Kaspersky. "What is Social Engineering". https://usa.kaspersky.com/resource-center/definitions/what-is-social-engineering
[6] Madhubala, R., N. Rajesh, L. Shaheetha, and N. Arulkumar. "Survey on malicious URL detection techniques." In *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 778-781. IEEE, 2022. https://doi.org/10.1109/ICOEI53556.2022.9777221
[7] Shaheetha, L., and K. Vadivazhagan. "Detection of Malicious Domains in the Cyberspace using Machine Learning & Deep Learning: A Survey." In *2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)*, pp. 1540-1543. IEEE, 2022.
[8] Raja, A. Saleem, G. Pradeepa, and N. Arulkumar. "Mudhr: Malicious URL detection using heuristic rules based approach." In *AIP Conference Proceedings*, vol. 2393, no. 1. AIP Publishing, 2022. https://doi.org/10.1063/5.0074077
[9] Raja, A. Saleem, R. Vinodini, and A. Kavitha. "Lexical features based malicious URL detection using machine learning techniques." *Materials Today: Proceedings* 47 (2021): 163-166. https://doi.org/10.1016/j.matpr.2021.04.041
[10] Abdul Samad, Saleem Raja, Sundarvadivazhagan Balasubaramanian, Amna Salim Al-Kaabi, Bhisham Sharma, Subrata Chowdhury, Abolfazl Mehbodniya, Julian L. Webber, and Ali Bostani. "Analysis of the performance impact of fine-tuned machine learning model for phishing URL detection." *Electronics* 12, no. 7 (2023): 1642. https://doi.org/10.3390/electronics12071642
[11] Balasubaramanian, Sundaravadivazhagan, Pradeepa Ganesan, and Justin Rajasekaran. "Weighted ensemble classifier for malicious link detection using natural language processing." *International Journal of Pervasive Computing and Communications* (2023).
[12] Manan, Wan Nurulsafawati Wan, Abdul Ghani Ali Ahmed, and Mohd Nizam Mohmad Kahar. "Characterizing current features of malicious threats on websites." In *Intelligent Computing & Optimization 1*, pp. 210-218. Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-00979-3_21
[13] Aljabri, Malak, Fahd Alhaidari, Rami Mustafa A. Mohammad, Samiha Mirza, Dina H. Alhamed, Hanan S. Altamimi, and Sara Mhd Chrouf. "An assessment of lexical, network, and content-based features for detecting malicious urls using machine learning and deep learning models." *Computational Intelligence and Neuroscience* 2022 (2022). https://doi.org/10.1155/2022/3241216
[14] Sirageldin, Abubakr, Baharum B. Baharudin, and Low Tang Jung. "Malicious web page detection: A machine learning approach." In *Advances in Computer Science and its Applications: CSA 2013*, pp. 217-224. Springer Berlin Heidelberg, 2014. https://doi.org/10.1007/978-3-642-41674-3_32
[15] Raja, A. Saleem, B. Sundarvadivazhagan, R. Vijayarangan, and S. Veeramani. "Malicious webpage classification based on web content features using machine learning and deep learning." In *2022 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*, pp. 314-319. IEEE, 2022.
[16] Desai, Anand, Janvi Jatakia, Rohit Naik, and Nataasha Raul. "Malicious web content detection using machine leaning." In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pp. 1432-1436. IEEE, 2017. https://doi.org/10.1109/RTEICT.2017.8256834
[17] Pradeepa, Ganesan, and Radhakrishnan Devi. "Lightweight approach for malicious domain detection using machine learning." *Scientific and Technical Journal of Information Technologies, Mechanics and Optics.* 22, no. 2 (2022): 262-268. https://doi.org/10.17586/2226-1494-2022-22-2-262-268
[18] AS, Saleem Raja, G. Pradeepa, S. Mahalakshmi, and M. S. Jayakumar. "Natural language based malicious domain detection using machine learning and deep learning." *Scientific and Technical Journal of Information Technologies, Mechanics and Optics.* 23, no. 2 (2023): 304-312. https://doi.org/10.17586/2226-1494-2023-23-2-304-312
[19] Kaggle. "Malicious URLs dataset." https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset?resource=download
[20] University of New Brunswick. "URL dataset (ISCX-URL2016)." (2016). https://www.unb.ca/cic/datasets/url-2016.html
[21] Phishtank. "User Agent String." https://www.phishtank.com/developer_info.php

[22] Analytics Vidhya. "Understanding Text Classification in NLP with Movie Review Example." https://www.analyticsvidhya.com/blog/2020/12/understanding-text-classification-in-nlp-with-movie-review-example-example

[23] Jha, Ashish Kumar, Raja Muthalagu, and Pranav M. Pawar. "Intelligent phishing website detection using machine learning." *Multimedia Tools and Applications* 82, no. 19 (2023): 29431-29456. https://doi.org/10.1007/s11042-023-14731-4

[24] Prasad, Arvind, and Shalini Chandra. "PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning." *Computers & Security* 136 (2024): 103545. https://doi.org/10.1016/j.cose.2023.103545

[25] Aziz, Mohd Zafran Abdul, and Koji Okamura. "A security trending review on software define network (SDN)." *Journal of Advanced Research in Computing and Applications* 6, no. 1 (2016): 1-16.

[26] Nabil, Mohammed, Mohamed Helmy Megahed, and Mohamed Hassan Abdel Azeem. "Design and simulation of new one time pad (OTP) stream cipher encryption algorithm." *Journal of Advanced Research in Computing and Applications* 10, no. 1 (2018): 16-23.