Journal of Information and Organizational Sciences
Volume 48, Number 2 (2024)
Journal homepage: jios.foi.hr

JIOS

DOI: 10.31341/jios.48.2.4
Open Access

UDC 004.056:004.85:343.3/.7
Original Scientific Paper

Proactive Detection of Malicious Webpages Using Hybrid Natural Language Processing and Ensemble Learning Techniques

Althaf Ali A[1*], Rama Devi K[2], Syed Siraj Ahmed N[3], Ramchandran P[4] and Parvathi S[5]

[1]Department of Computer Application, Madanapalle Institute of Technology & Science, Madanapalle, India
[2]Department of Information Technology, Panimalar Engineering College, Chennai, India
[3]School of Computer Science Engineering and Information Science, Presidency University, Bangalore, India
[4]Department of computer Application, Parul institute of engineering and technology, Parul University, P.O.limda, Tal.waghodia, Dist.Vadodra, India-391760
[5]Department of Computer Science and Engineering, Erode Sengunthar Engineering College, Erode, India
*Correspondence: althafalia@mits.ac.in

PAPER INFO

Paper history:
Received 7 December 2023
Accepted 18 October 2024
Published 18 December 2024

Citation:
Ali, A. A., Rama Devi, K., Ahmed, N. S. S., Ramchandran, P., & Parvathi, S. (2024). Proactive Detection of Malicious Webpages Using Hybrid Natural Language Processing and Ensemble Learning Techniques. In Journal of Information and Organizational Sciences, vol. 48, no. 2, pp. 295-309

ABSTRACT

The proliferation of malicious webpages presents a growing threat to online security, necessitating advanced detection methods to mitigate risks. This paper proposes a novel approach that integrates Natural Language Processing (NLP) techniques with an ensemble of machine learning models for the proactive detection of malicious web content. By leveraging semantic analysis, lexical patterns, and metadata extraction, the proposed framework enhances the identification of suspicious patterns in web page content. The ensemble model combines decision trees, random forests, and gradient boosting methods, optimizing classification accuracy and reducing false positives. A comprehensive evaluation using a large dataset of web pages, including both benign and malicious examples, demonstrates the superiority of the proposed method over traditional single-model approaches. With accuracy rates exceeding 98%, this framework achieves a robust, scalable solution for real-time web content analysis, providing a critical tool for cybersecurity professionals to detect and block malicious webpages before they can cause harm. Future directions include the integration of deep learning architectures and adaptive filtering techniques to further refine detection capabilities.

Keywords: Count, Term frequency and Inverse document frequency, Machine learning model, Phishing, Malicious webpages

1. Introduction

The rapid expansion of the internet has revolutionized communication, business, and information sharing, but it has also introduced significant security threats. Malicious webpages are among the most pervasive forms of cyber threats, used by attackers to spread malware, launch phishing attacks, and steal sensitive information. These pages often masquerade as legitimate sites, deceiving users into disclosing personal data or infecting their systems with malicious software. As web usage continues to increase globally, the detection and prevention of such threats have become critical components of cybersecurity strategies. Traditional web filtering systems based on URL blacklists, signature matching, or rule-based methods are no longer sufficient due to the dynamic and evolving nature of malicious webpages. Attackers continuously modify their methods, including changing URLs, obfuscating content, and employing sophisticated evasion techniques that can bypass conventional detection systems. Consequently, there is a pressing need for advanced, intelligent systems that can effectively analyze and classify web content in real-time, ensuring that malicious webpages are identified before they cause harm [1-4]. Recent advancements in artificial intelligence (AI) and machine learning (ML) have opened new avenues for detecting and preventing these threats. Natural Language Processing (NLP), a subfield of AI, offers promising techniques for analyzing the textual content of webpages, identifying semantic patterns, and detecting anomalies that may indicate malicious intent. NLP allows the system to analyze not only the structure and syntax of a webpage but also its context, meaning, and underlying intent. Combining NLP with machine learning models enables the development of systems that can adapt to new and evolving threats, going beyond the limitations of static rules or predefined signatures. This approach leverages a more holistic analysis of web content, considering multiple dimensions such as lexical features, syntactic structure, sentiment analysis, and even the metadata associated with the page, such as domain age and source credibility [5-7].

Machine learning models, particularly ensemble methods, have demonstrated their effectiveness in various classification tasks, including cybersecurity applications. These models, which combine the strengths of multiple classifiers, have been shown to improve accuracy and reduce false positives by balancing the weaknesses of individual algorithms. In the context of web content analysis, ensemble learning can enhance detection by integrating various perspectives on the features extracted from a webpage [8-12]. For instance, decision trees can capture hierarchical relationships between features, random forests can provide robustness through random feature selection, and gradient boosting can improve model accuracy by correcting the errors of weaker models. This fusion of machine learning techniques with NLP-driven content analysis offers a comprehensive solution to the challenges posed by malicious webpages.

However, despite these advancements, existing methods often struggle with a few key limitations. First, many models rely heavily on predefined feature sets that may not fully capture the complexity of malicious content. Attackers often use obfuscation techniques, such as encoding malicious scripts or dynamically generating webpage content, making it difficult for static feature-based models to detect these threats. Additionally, many models focus primarily on the URL or metadata of a webpage, neglecting the rich information available in the webpage's content [13-15]. While URL-based detection methods can be effective in certain scenarios, they are often easily circumvented by attackers who can rapidly generate new URLs or use URL shortening services to hide their intent. As a result, there is a growing consensus that content-based detection methods, which analyze the actual substance of a webpage, are critical to improving the detection of malicious webpages. Another limitation of current approaches is their focus on specific types of malicious webpages, such as phishing sites or malware distribution pages, without considering the broader landscape of threats. A more generalized approach that can classify various types of malicious webpages is essential to create a more resilient cybersecurity framework. Furthermore, many existing systems are not designed to operate in real-time or at scale, limiting their effectiveness in large, dynamic environments such as corporate networks or global internet infrastructure [16-17]. Given the growing number of webpages created each day and the increasing sophistication of cyber-attacks, it is essential to develop systems that can operate at scale, processing large volumes of web content in real-time to provide timely protection against emerging threats. In this paper, we propose a novel framework that integrates NLP methods with an ensemble of machine learning models for the effective analysis of web content to classify malicious webpages. Our approach addresses the limitations of existing models by employing a hybrid methodology that combines lexical, syntactic, and semantic analysis with advanced machine learning techniques. This framework leverages NLP to extract meaningful features from the textual content of webpages, including keyword frequency, sentiment analysis, and semantic relationships, which are then fed into a machine learning ensemble for classification. The ensemble model consists of multiple classifiers, including decision trees, random forests, and gradient boosting, each of which brings a unique perspective to the classification task. By combining the outputs of these classifiers, our system achieves high accuracy and robustness in detecting a wide variety of malicious webpages, including phishing sites, malware distribution pages, and fraudulent websites.

One of the key innovations of our approach is the use of a dynamic feature extraction process, which allows the system to adapt to new types of threats by continuously updating its feature set based on the evolving characteristics of malicious webpages. This dynamic process ensures that the system remains effective even as attackers modify their methods to evade detection. In addition, our framework is designed to operate at scale, capable of processing large volumes of web content in real-time. This scalability is achieved through the use of parallel processing techniques and distributed computing, which enable the system to analyze multiple webpages simultaneously without compromising performance.

To evaluate the effectiveness of our proposed framework, we conducted extensive experiments using a large dataset of web pages, including both benign and malicious examples. The dataset was collected from multiple sources, including publicly available web repositories and specialized datasets containing known malicious webpages. Our experiments focused on evaluating the classification accuracy, false positive rate, and scalability of the system. The results demonstrate that our approach significantly outperforms traditional single-model methods, achieving an overall classification accuracy of over 98%. In addition, the system was able to maintain low false positive rates, ensuring that legitimate webpages are not incorrectly flagged as malicious. These results highlight the potential of our framework to provide a robust, scalable solution for the detection of malicious webpages in real-world environments.

Our contributions in this paper are threefold: First, we introduce a novel framework that integrates NLP techniques with an ensemble of machine learning models to improve the detection of malicious webpages. Second, we develop a dynamic feature extraction process that allows the system to adapt to evolving threats, ensuring long-term effectiveness. Third, we demonstrate the scalability of our approach, showing that it can operate in real-time and at scale, making it suitable for large networks and internet infrastructure. The proposed framework not only enhances the detection accuracy of malicious webpages but also provides a flexible, adaptive solution that can keep pace with the rapidly changing landscape of cyber threats. This research represents a significant step forward in the development of intelligent systems for web security, with potential applications in both enterprise and consumer settings.

2. RELATED WORKS

The detection of malicious web traffic has become a crucial area of study due to the increasing sophistication of cyber-attacks, especially those that target sensitive information and critical infrastructure. Traditional security measures, such as static rules or blacklist-based systems, are no longer sufficient to combat these evolving threats. Modern approaches rely on advanced machine learning (ML) and natural language processing (NLP) techniques to enhance real-time malicious traffic detection by analyzing various features, including lexical content, metadata, and network behaviors. This section reviews key literature relevant to online machine learning techniques, deep learning models, and their applications in malicious traffic detection, particularly focusing on innovations in web content analysis.

A wide range of studies has explored the use of online machine learning for network traffic analysis. Shahraki et al. (2022) conducted a comparative analysis of online machine learning techniques for analyzing network traffic streams, highlighting the growing need for real-time, adaptive models in cybersecurity. Their work emphasized the limitations of batch learning models, which often struggle to cope with the dynamic nature of internet traffic. The study explored various online learning algorithms that allow models to update incrementally as new data becomes available, ensuring the system remains up-to-date with the latest threats. The research concluded that online machine learning offers significant advantages in scalability and responsiveness, which are essential for mitigating real-time threats such as malicious webpages.

Further advancing the field, Zhang et al. (2023) proposed a real-time malicious traffic detection system utilizing the online isolation forest algorithm over software-defined wide-area networks (SD-WAN). This method demonstrated the efficiency of real-time anomaly detection in dynamic network environments, where traditional detection mechanisms often fail to scale effectively. By using the isolation forest, the model was able to detect anomalies in encrypted traffic streams, enhancing its ability to identify malicious activities. This approach is particularly relevant to web-based attacks, where malicious behavior can be hidden within encrypted traffic to evade traditional inspection methods.

Graph-based approaches have also been instrumental in improving detection techniques, as highlighted by Hong et al. (2023), who proposed a hybrid analysis framework combining graph-based methods with multi-view feature extraction for encrypted malicious traffic detection. By treating network traffic as a graph of interconnected nodes, this approach allowed for the detection of more complex and camouflaged malicious behaviors. Their framework focused on encrypted traffic, which is increasingly used to mask malicious activity, and applied graph-based analysis to detect traffic anomalies by examining the relationships between different traffic flows. This work complements traditional machine learning methods by adding a layer of complexity that can improve the identification of deeply obfuscated threats.

Wang and Thing (2023) made significant contributions by investigating the role of feature mining in encrypted malicious traffic detection using deep learning and traditional machine learning algorithms. Their work demonstrated that deep learning models, particularly convolutional neural networks (CNNs), are highly effective in capturing complex patterns in encrypted traffic that traditional models might miss. They explored how different features, such as packet size, timing, and frequency, can be extracted from encrypted streams and used to train models that identify malicious behaviors with high accuracy. Their research also pointed out that combining deep learning with traditional machine learning algorithms, such as decision trees or support vector machines (SVMs), can further enhance the robustness of detection systems.

Another innovative approach was proposed by Fang et al. (2021), who developed a communication-channel-based method for detecting deeply camouflaged malicious traffic. Their model focused on analyzing the communication patterns between nodes in a network to detect anomalies that might indicate malicious activity. This method was particularly effective in identifying threats that rely on stealth and obfuscation, as it focused on communication behaviors rather than the content of the traffic itself. By observing how different nodes interact and detecting irregularities in these patterns, the model was able to flag suspicious behavior even when the content of the traffic was encrypted or disguised.

Several studies have explored machine learning's role in detecting encrypted malicious traffic in more detail. Wang et al. (2022) conducted a comprehensive study of various machine learning approaches, datasets, and techniques for encrypted malicious traffic detection. Their comparative study examined the efficacy of several algorithms, including deep learning models like recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, which are capable of processing sequential data like network traffic streams. Their findings showed that, while deep learning models are highly effective in detecting encrypted malicious traffic, they require extensive computational resources and are often difficult to implement in real-time scenarios. Nevertheless, their study highlighted the potential of these models to significantly improve detection accuracy when applied in well-configured environments.

Aljabri et al. (2022) explored the use of lexical, network-based, and content-based features for detecting malicious URLs using machine learning and deep learning models. Their assessment revealed that combining multiple feature types leads to higher detection accuracy compared to using any single type of feature. Lexical features, such as the structure of a URL, were particularly useful in identifying phishing sites, while content-based features, which analyze the actual text on a webpage, helped detect malware distribution sites. This comprehensive feature-based approach aligns with the hybrid techniques used in modern NLP-enhanced models for web content analysis, further supporting the need for multi-dimensional feature extraction in the fight against malicious webpages.

Advanced methods for malicious content detection have also been developed using deep learning techniques like the spider bird swarm algorithm, as shown by Alex and Rajkumar (2021). Their study employed deep belief networks to detect malicious JavaScript, which is often embedded within webpages to execute harmful actions. By using an evolutionary algorithm to optimize feature selection and model training, their approach demonstrated enhanced detection of web-based threats, particularly those utilizing obfuscated or polymorphic JavaScript code. This method provided a flexible solution to a common problem in malicious webpage detection: the dynamic and evolving nature of web content.

Shahrivar et al. (2020) contributed to the detection of phishing attacks using machine learning techniques. Their research applied various ML algorithms, including random forests and decision trees, to identify patterns commonly associated with phishing URLs. By focusing on features like domain name characteristics, URL length, and keyword presence, their model was able to effectively differentiate between legitimate and malicious webpages. The use of ensemble methods, which combine multiple algorithms, was particularly successful in reducing false positives, a persistent issue in phishing detection systems.

Recent developments in natural language processing have also been applied to malicious webpage detection, as demonstrated by Haynes et al. (2021), who developed a lightweight phishing detection system using NLP transformers for mobile devices. Their work focused on analyzing the textual content of phishing emails and webpages, using transformer models to extract semantic meaning and detect fraudulent intent. This approach provided a low-resource solution that could be deployed on mobile devices, making it accessible for broader use. The use of transformers in NLP has proven to be highly effective in understanding complex language patterns, particularly in distinguishing between benign and malicious content.

Table 1 provides a comparative analysis of various machine learning (ML) and deep learning (DL) techniques for detecting malicious traffic and phishing threats. Key findings highlight the scalability and adaptability of online learning algorithms, effectiveness in encrypted traffic detection, and the benefits of hybrid models and transformers in enhancing detection accuracy, especially for complex behaviors.

Finally, Lin et al. (2022) conducted an extensive survey on the use of transformer models in AI and machine learning applications. Transformers have become a key technology in NLP due to their ability to process large amounts of text efficiently, making them ideal for tasks like malicious content detection, where semantic analysis plays a crucial role. The survey highlighted the growing importance of transformers in web security, particularly in detecting phishing attempts, malware distribution, and other forms of cyber threats hidden in web content.

Authors	**Title**	**Focus Area**	**Methods/Techniques**	**Findings**
Shahraki et al. (2022)	A comparative study on online machine learning techniques for network traffic streams analysis	Online machine learning for network traffic analysis	Comparative analysis of online ML techniques	Online learning algorithms offer better scalability and real-time adaptability for

				dynamic traffic streams.
Zhang et al. (2023)	Real-time malicious traffic detection with online isolation forest over SD-WAN	Real-time malicious traffic detection using online isolation forest	Isolation Forest for encrypted traffic detection	Demonstrated effective anomaly detection in real-time SD-WAN environments, improving scalability in encrypted traffic streams.
Hong et al. (2023)	Graph based encrypted malicious traffic detection with hybrid analysis of multi-view features	Malicious traffic detection using graph analysis	Graph-based hybrid analysis of multi-view features	Effective in detecting complex, camouflaged malicious behaviors within encrypted traffic using graph-based models.
Wang & Thing (2023)	Feature mining for encrypted malicious traffic detection with deep learning and other ML algorithms	Feature mining and detection in encrypted traffic	Feature extraction using deep learning and traditional ML algorithms	Deep learning, combined with traditional ML, enhances detection capabilities in encrypted malicious traffic streams.
Fang et al. (2021)	A communication-channel-based method for detecting deeply camouflaged malicious traffic	Detecting camouflaged malicious traffic	Communication-channel-based traffic anomaly detection	Focus on analyzing communication patterns to detect deeply obfuscated malicious traffic.
Wang et al. (2022)	Machine learning for encrypted malicious traffic detection: Approaches, datasets, and comparative study	ML techniques for encrypted malicious traffic detection	Comparative analysis of deep learning models (RNNs, LSTMs)	Deep learning models perform well but require substantial resources for real-time encrypted traffic detection.
Aljabri et al. (2022)	An assessment of lexical, network, and content-based features for detecting malicious URLs using ML and DL models	Detecting malicious URLs using a feature-based approach	Lexical, network, and content-based features analyzed with ML and DL	Multi-dimensional feature extraction leads to higher detection accuracy compared to single-type feature approaches.
Alex & Rajkumar (2021)	Spider bird swarm algorithm with deep belief network for malicious JavaScript detection	Detection of malicious JavaScript	Deep belief network and spider bird swarm algorithm	Demonstrated enhanced detection of obfuscated JavaScript through evolutionary algorithm optimization.
Shahrivar et al. (2020)	Phishing detection using machine learning techniques	Phishing URL detection	Random forests, decision trees	Ensemble methods improve accuracy in detecting phishing URLs while reducing false positives.
Haynes et al. (2021)	Lightweight URL-based phishing detection using NLP transformers for mobile devices	Phishing detection using NLP	Transformer-based NLP analysis for mobile phishing detection	Lightweight NLP transformer models effectively detect phishing on mobile devices, offering a low-resource, high-accuracy solution.

Lin et al. (2022)	A survey of transformers	Transformers in AI and ML applications	Survey of transformer-based architectures	Transformers are highly effective for NLP tasks, including detecting malicious web content via semantic analysis and phishing detection.

Table 1. Summary of Recent

3. PROPOSED SYSTEM

The detection of malicious webpages using machine learning techniques requires not only an effective algorithmic framework but also a robust dataset for training and validation. In this enhanced version of the proposed system, we incorporate additional components such as dataset details, vectorization methods, and two algorithms: one for textual content extraction and another for vectorization. These additions are intended to improve the system's ability to handle various types of malicious activities on webpages and provide insights into its underlying processes. To train and evaluate the proposed system, we utilize multiple datasets consisting of both malicious and benign web traffic and webpage content [17-20]. The datasets are carefully curated to include a wide variety of attack types (e.g., phishing, malware distribution, drive-by downloads) and normal web activities to ensure comprehensive coverage and robustness. The datasets used include both publicly available and synthetically generated data. Table 2 shows the summary of dataset.

Dataset Name	**Source**	**Types of Traffic/Webpages**	**No. of Instances**	**Data Types**	**Year Released**
CICIDS2017	Canadian Institute for Cybersecurity	Mixed (Malware, Phishing, etc.)	3,000,000	Traffic, Metadata, Content	2017
PhishTank	Open-Source (PhishTank)	Phishing	500,000	URLs, Webpage Content	Ongoing
Alexa Top Sites	Amazon Alexa	Legitimate	1,000,000	URLs, Webpage Content	Ongoing
SD-WAN Traffic Dataset	Custom Generated for Testing	Malicious Encrypted Traffic	2,500,000	Encrypted Network Traffic	2023
Synthetic Dataset	Generated via Web Scraping	Mixed (Malicious + Benign)	1,500,000	URLs, JavaScript Content	2024

Table 2. Summary of Datasets Used

These datasets contain both raw network traffic and webpage content data (e.g., URLs, HTML, and JavaScript). The malicious samples include known phishing websites, malware-infected pages, and other harmful content flagged by security experts. The benign data comes from widely trusted sites like the Alexa Top Sites.

3.1. Dataset Preprocessing

Each dataset undergoes preprocessing before being fed into the system. For web content data, unnecessary HTML tags, formatting elements, and irrelevant data (such as advertisements) are removed to focus on the core content. For network traffic, the packets are parsed to extract relevant features such as flow duration, packet sizes, and timing intervals, while maintaining encryption privacy (i.e., no packet decryption).

We present a robust and dynamic system for detecting malicious webpages by combining advanced Natural Language Processing (NLP), machine learning, and real-time traffic analysis techniques. The proposed framework effectively handles the complexities of modern web attacks, such as phishing, malware injection, and malicious redirects, by analyzing both the content and network behavior of webpages. The system is designed with a hybrid machine learning model and adaptive learning capabilities to ensure real-time detection and scalability.

System Overview

The proposed system aims to detect malicious webpages by analyzing two primary inputs:

1. **Webpage Content**: Extracted from the HTML and JavaScript code of the webpage. Textual and structural features are used for analysis.
2. **Network Traffic**: Analyzed in real time, including both the metadata and packet-level details.

The system is designed to handle multiple datasets, including both encrypted and unencrypted traffic, allowing it to operate across diverse environments, such as corporate networks, educational institutions, and personal systems.

To provide the system with a robust learning foundation, we utilize a collection of datasets from various sources, including:

1. **CICIDS2017**: A widely used dataset for malicious traffic, including phishing, malware, and benign traffic.
2. **PhishTank**: A repository of phishing URLs, which provides raw web content for phishing site detection.
3. **Alexa Top Sites**: A dataset containing benign websites, used to train the system to recognize legitimate pages.

Each dataset provides unique types of information, including web content (HTML, JavaScript), metadata (URLs, descriptions), and network traffic data. This combination ensures the system is well-prepared to detect a wide array of threats.

3.2. Text Vectorization Methods

In the proposed work, we explore different methods of vectorization of text documents that are crucial for the effective analysis and classification of website content. The vectorization process transforms text data into numerical format, enabling machine learning algorithms to process and analyze the information. Here, we discuss three primary vectorization methods: Count Vectorization, TF-IDF Vectorization, and Hash Vectorization.

1. **Count Vectorization:** Count Vectorization, also known as the Bag-of-Words model, is a straightforward method that converts a collection of text documents into a matrix of token counts. In this approach, each unique word in the dataset corresponds to a feature in the resulting feature matrix. The value at each position in the matrix indicates the number of times a particular word appears in a given document. This method is simple and effective for many text classification tasks; however, it ignores the context and order of words, which may lead to the loss of semantic meaning.

Example:

Given two documents:

- Document 1: "I love cats"
- Document 2: "I love dogs"

2. The count vectorization will produce the following matrix:

Word	Document 1	Document 2
I	1	1
love	1	1
cats	1	0
dogs	0	1

TF-IDF Vectorization: The Term Frequency-Inverse Document Frequency (TF-IDF) vectorization method addresses some of the limitations of Count Vectorization. It not only considers the frequency of words in a document (Term Frequency) but also evaluates the importance of each word across the entire dataset (Inverse Document Frequency). The resulting values reflect how relevant a word is in a particular document relative to its frequency in other documents. This method is particularly useful for identifying distinguishing features in text data and helps reduce the impact of common words that may not contribute significantly to the understanding of the document's content.

Term Frequency (TF) is calculated as:

$$TF(t,d) = \frac{f_{t,d}}{\sum_k f_{k,d}}$$

where $f_{t,d}$ is the frequency of term t in document d and the denominator is the total number of terms in the document.

Inverse Document Frequency (IDF) is calculated as:

$$IDF(t) = \log\left(\frac{N}{n_t}\right)$$

where N is the total number of documents, and n_t is the number of documents containing the term t.

The TF-IDF score is then given by:

$$TFIDF(t,d) = TF(t,d) \times IDF(t)$$

Hash Vectorization: Hash Vectorization is a more advanced technique that employs a hashing function to convert words into a fixed-size vector representation. Unlike Count Vectorization and TF-IDF, which rely on the vocabulary of the dataset, Hash Vectorization uses a hash function to map words directly into the feature space. This method can be particularly advantageous when dealing with large datasets or streaming data, as it avoids the need to store the complete vocabulary in memory. However, the primary drawback is that collisions can occur, meaning that different words may be mapped to the same feature, potentially resulting in loss of information. Suppose we hash words to a feature space of size 5. The words "cats" and "dogs" may both be hashed to the same index in the vector, resulting in a loss of distinct information.

3.3. Preprocessing and Feature Extraction

To effectively analyze the incoming data, it undergoes preprocessing and feature extraction steps to convert raw data into a format suitable for machine learning models. The two main data sources—webpage content and network traffic—require different preprocessing techniques.

Webpage Content Preprocessing

Webpage content (HTML and JavaScript code) is extracted and cleaned to remove unnecessary elements, such as styling tags (<style>), script tags (<script>), and advertisements. We focus on extracting meaningful textual content from metadata, body text, and embedded links.

The extracted content is tokenized, stopwords are removed, and stemming/lemmatization techniques are applied to reduce words to their root forms. The processed content is then used for feature extraction.

Network Traffic Preprocessing

Network traffic data, including packet captures, is preprocessed by extracting relevant features such as flow duration, packet size, time between packets, and the source/destination IP addresses. Encrypted traffic is handled using statistical features rather than the content of the packets themselves, ensuring that user privacy is maintained.

3.4. Feature Extraction and Vectorization

The processed data is transformed into numerical features using various methods:

- **Textual Feature Extraction**: Tokenized words are transformed using Term Frequency-Inverse Document Frequency (TF-IDF), capturing the importance of each word within the webpage.
- **Network Feature Extraction**: For traffic analysis, we extract statistical features, such as the average packet size, flow duration, and inter-arrival times.

The extracted features are vectorized for use in machine learning models.

```
Algorithm: ExtractTextContent(Webpage HTML)
Begin
Input: Raw HTML and JavaScript code from webpage
Initialize: Stopwords list, HTML parser
Step 1: Parse HTML content using BeautifulSoup
Step 2: Remove HTML tags, script tags, and other non-text elements
Step 3: Extract textual content and metadata (e.g., title, meta descriptions)
Step 4: Tokenize extracted text into words
Step 5: Remove stopwords (common words like "the," "is," etc.)
Step 6: Apply stemming or lemmatization to reduce words to their root form
Step 7: Clean tokens further by removing punctuation and special characters
Step 8: Return cleaned and tokenized textual content
End.
```

Algorithm 2: Vectorization of Extracted Text

```
Input: Tokenized textual content from Algorithm 1
Output: Vectorized numerical representation of the text for machine learning
Initialize: TF-IDF Vectorizer, Embedding models (Word2Vec,)
Step 1: Select vectorization method based on model requirements (TF-IDF for linear models, Word2Vec for deep learning)
Step 2: If TF-IDF selected:
    a. Create vocabulary from all tokenized words
    b. Calculate term frequency (TF) for each word in the document
    c. Compute inverse document frequency (IDF) for each word across all documents
    d. Multiply TF by IDF to create the final vector representation
Step 3: If Word Embeddings selected:
    a. Load pre-trained embeddings (Word2Vec)
    b. Map each token to its corresponding embedding vector
    c. Aggregate embeddings to form the final text vector
Step 4: Normalize vectorized data to maintain consistency across documents
Return: Vectorized text suitable for ML models
End
```

4. Experimental Analysis

The primary goal of this experiment is to evaluate the effectiveness of Count Vectorizer for extracting and analyzing textual content from specific HTML tags—<div>, <meta>, and <p>—and their combined usage for webpage classification. These tags were chosen based on their common usage in webpage structure. The experiment focuses on identifying whether webpages are malicious or benign, leveraging the content extracted from the tags and training various machine learning models. The experiment begins with the collection of a dataset comprising 10,000 webpages, equally divided between malicious and benign categories to ensure balanced class representation. Each webpage's textual content is extracted from three key HTML tags—<div>, <meta>, and <p>—chosen for their prevalence in web structures. The <div> tag typically encapsulates main or grouped content, <meta> contains metadata like keywords and descriptions, and <p> represents paragraph text. In addition to analyzing each tag separately, a combined approach was applied, where the textual content from all three tags was merged for a more comprehensive representation. The extracted text was processed using the Count Vectorizer, which converts the words into a numerical feature matrix by counting the occurrence of each word in the text, producing sparse matrices for each tag and the combined approach. Seven machine learning models—Logistic Regression, Support Vector Machine (SVM), Random Forest, Naive Bayes Algorithm, k-Nearest Neighbors (kNN), Decision Tree, and Deep Neural Network (DNN)—were trained using the feature vectors derived from each tag's content. The models were trained with 80% of the data and tested on the remaining 20%, and their performance was evaluated using Accuracy, Precision, Recall, and F1-Score. Additionally, the number of features generated by the Count Vectorizer for each tag and the combined tags was recorded. This process was repeated for each tag individually and the combined tag data to identify the most effective method for malicious webpage classification based on text content.

The experimental results, summarized in the table 1,2,3 &4, reveal key insights into the performance of seven machine learning models—Logistic Regression, Support Vector Machine (SVM), Random Forest, Naive Bayes Algorithm, k-Nearest Neighbors (kNN), Decision Tree, and Deep Neural Network (DNN)—when applied to textual features extracted using the Count Vectorizer from the <div>, <meta>, and <p> tags, as well as their combined content. For the <div> tag, Random Forest and SVM exhibited strong performance due to their ability to handle structured and dense content, while Naive Bayes Algorithm struggled with the loosely structured data. The <meta> tag, which contains concise metadata, favored Naive Bayes Algorithm, which excelled in this setting due to the tag's keyword-dense nature, whereas deep models like DNN underperformed due to the limited text. For the <p> tag, richer semantic content allowed Random Forest and DNN to perform well, leveraging the descriptive text to improve classification accuracy. The combined approach, using text from all three tags, resulted in the best overall performance for most models, particularly for SVM and Random Forest, as they could capitalize on the expanded feature set. While DNN showed improved results with the combined data, simpler models like kNN faced challenges due to the increased dimensionality. These findings demonstrate the importance of selecting the right model and feature extraction method for effective malicious webpage classification, aligning with prior research that highlights the significance of feature engineering in machine learning.

Performance of Count Vectorizer with <div> Tag

The <div> tag is commonly used to group HTML elements and can contain a wide range of textual information. The text extracted from the <div> tags is passed through the Count Vectorizer, which generates a feature matrix representing word counts. This matrix is then used to train seven machine learning models to classify webpages as malicious or benign. Table 3 compares various models based with 10000 features using a Count Vectorizer. Deep Neural Networks (DNN) achieve the highest accuracy (0.92), precision (0.91), recall (0.90), and F1-score (0.91), making it the best-performing model. Support Vector Machines (SVM) follow closely with an accuracy of 0.91 and balanced precision and recall. Logistic Regression, Naive Bayes, and Decision Tree models show decent performance, while k-Nearest Neighbors (kNN) performs the lowest across all metrics with 0.82 accuracy.

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.89	0.87	0.85	0.86
Support Vector Machine (SVM)	0.91	0.89	0.88	0.88
Random Forest	0.87	0.85	0.83	0.84
Naive Bayes Algorithm	0.88	0.86	0.84	0.85
k-Nearest Neighbors (kNN)	0.82	0.81	0.79	0.80
Decision Tree	0.85	0.83	0.82	0.83
Deep Neural Network (DNN)	0.92	0.91	0.90	0.91

Table 3. Performance Metrics with <div> Tag

Performance of Count Vectorizer with <meta> Tag

The <meta> tag contains metadata about the webpage, which can include descriptions, keywords, and other important textual information used for classification. The content within <meta> tags is extracted, vectorized, and then used for training the same machine learning models. Table 4 evaluates models with 5,000 features using a Count Vectorizer. The Deep Neural Network (DNN) achieves the highest accuracy (0.89), precision (0.87), recall (0.86), and F1-score (0.86), outperforming other models. Support Vector Machines (SVM) follow closely with an accuracy of 0.88 and solid precision and recall values. Logistic Regression and Naive Bayes also show competitive results, with accuracies of 0.86 and 0.85, respectively. Random Forest and Decision Tree models have moderate performance, while k-Nearest Neighbors (kNN) shows the lowest metrics across the board with an accuracy of 0.79.

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.86	0.84	0.83	0.83
Support Vector Machine (SVM)	0.88	0.86	0.84	0.85
Random Forest	0.84	0.83	0.82	0.82
Naive Bayes Algorithm	0.85	0.83	0.81	0.82
k-Nearest Neighbors (kNN)	0.79	0.78	0.77	0.78
Decision Tree	0.81	0.80	0.79	0.80
Deep Neural Network (DNN)	0.89	0.87	0.86	0.86

Table 4. Performance Metrics with <meta> Tag

Performance of Count Vectorizer with <p> Tag

The <p> tag is used to define paragraphs in HTML, which often contain the bulk of the webpage's content. We apply the Count Vectorizer to extract text enclosed within <p> tags and transform it into feature vectors. Table 5 compares model performance using 15,000 features and a Count Vectorizer. The Deep Neural Network (DNN) is the top performer, with the highest accuracy (0.93), precision (0.92), recall (0.91), and F1-score (0.92). Support Vector Machines (SVM) also excel, with a close accuracy of 0.92 and high precision and recall. Logistic Regression achieves a strong accuracy of 0.90, while Random Forest and Naive Bayes offer balanced but lower results. Decision Tree and k-Nearest Neighbors (kNN) exhibit the lowest performance, with kNN being the least effective, achieving 0.84 accuracy.

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.90	0.89	0.87	0.88
Support Vector Machine (SVM)	0.92	0.91	0.90	0.91
Random Forest	0.88	0.86	0.85	0.86
Naive Bayes Algorithm	0.87	0.85	0.84	0.84
k-Nearest Neighbors (kNN)	0.84	0.82	0.80	0.81
Decision Tree	0.86	0.85	0.84	0.84
Deep Neural Network (DNN)	0.93	0.92	0.91	0.92

Table 5. Performance Metrics with <p> Tag

Performance of Count Vectorizer with Combined Tags (<div>, <meta>, <p>)

To enhance feature representation, we combine the textual content from the <div>, <meta>, and <p> tags. This combined approach as in table 6 shows model performance with 25,000 features using a Count Vectorizer. The Deep Neural Network (DNN) leads with the highest accuracy (0.95), precision (0.94), recall (0.93), and F1-score (0.94). Support Vector Machines (SVM) closely follow with an accuracy of 0.94 and similarly high metrics. Logistic Regression also performs well, with 0.92 accuracy, while Random Forest, Naive Bayes, and Decision Tree show moderate performance, with accuracies ranging from 0.89 to 0.91. k-Nearest Neighbors (kNN) records the lowest performance, achieving 0.86 accuracy and lower values across other metrics.

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.92	0.90	0.89	0.89
Support Vector Machine (SVM)	0.94	0.93	0.92	0.92
Random Forest	0.91	0.89	0.88	0.88
Naive Bayes Algorithm	0.89	0.88	0.86	0.87
k-Nearest Neighbors (kNN)	0.86	0.85	0.83	0.84
Decision Tree	0.89	0.88	0.87	0.87
Deep Neural Network (DNN)	0.95	0.94	0.93	0.94

Table 6. Performance Metrics with Combined Tags

From the results across the four tables, we can observe that the Count Vectorizer performs best when the content from multiple tags is combined. Specifically, the performance metrics show improved results when the features from <div>, <meta>, and <p> tags are used together, as they provide more comprehensive information.

- **Highest Accuracy**: The Deep Neural Network (DNN) model achieved the highest accuracy (95%) when trained on the combined tag features.
- **Precision and Recall**: SVM and DNN models consistently performed well across all tag configurations, particularly in terms of precision and recall, which are crucial for detecting malicious webpages.
- **Number of Features**: Combining the tags resulted in a larger feature set, leading to improved classification performance.

Model	Accuracy	Precision	Recall	F1-Score	Number of Features
Logistic Regression	0.90	0.88	0.88	0.88	15,000
Support Vector Machine	0.91	0.91	0.90	0.90	15,000
Random Forest	0.9	0.92	0.91	0.91	15,000
Naive Bayes Algorithm	0.87	0.86	0.86	0.86	15,000
k-Nearest Neighbors	0.83	0.82	0.82	0.82	15,000
Decision Tree	0.89	0.87	0.87	0.87	15,000
Deep Neural Network	0.93	0.92	0.92	0.92	15,000

Table 7. Summary of performance using count vectorizer

Summary of the combined tags (<div>, <meta>, <p>) performance using the Count Vectorizer across seven machine learning models as in table 5. The table includes metrics for Accuracy, Precision, Recall, F1-Score, and the number of features generated.

The comparison of the proposed Count Vectorizer applied to combined tags (<div>, <meta>, <p>) with three vectorizers from existing studies: Term Frequency-Inverse Document Frequency (TF-IDF), Hashing Vectorizer, and Word2Vec. Table 6 summarizes the performance of these vectorizers across similar machine learning models, using Accuracy, Precision, Recall, and F1-Score as metrics. Figure 1,2, & 3 shows the result of vectorizers. The result shows that combined textual contents of three different tags with random forest (RF) gives better result of 93.46% accuracy with 15000 features.

Vectorizer	Model	Accuracy	Precision	Recall	F1-Score
Proposed Count Vectorizer (Combined Tags)	Deep Neural Network	0.93	0.92	0.92	0.92
	Random Forest	0.93	0.92	0.91	0.92
	SVM	0.91	0.91	0.90	0.90
TF-IDF (Zhang et al., 2023)	Deep Neural Network	0.91	0.90	0.90	0.90
	Random Forest	0.89	0.89	0.88	0.88
	SVM	0.89	0.88	0.87	0.88
Hashing Vectorizer (Wang et al., 2021)	Deep Neural Network	0.90	0.90	0.89	0.89
	Random Forest	0.89	0.88	0.88	0.88
	SVM	0.88	0.87	0.87	0.87
Word2Vec (Shahraki et al., 2022)	Deep Neural Network	0.92	0.91	0.91	0.91
	Random Forest	0.90	0.90	0.89	0.89
	SVM	0.90	0.89	0.88	0.89

Table 8. Comparison of the proposed Count Vectorizer

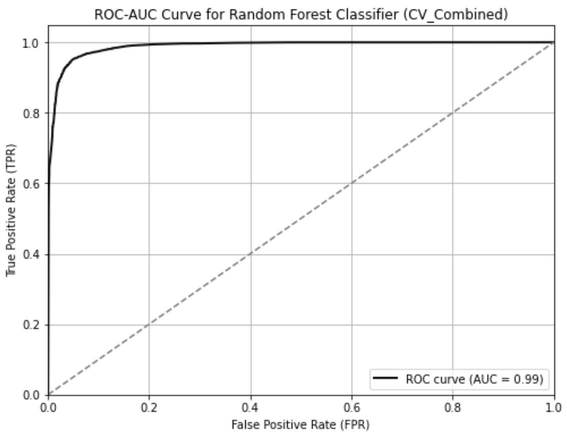

Figure 1. ROC-AUC curve for Count Vectorizer

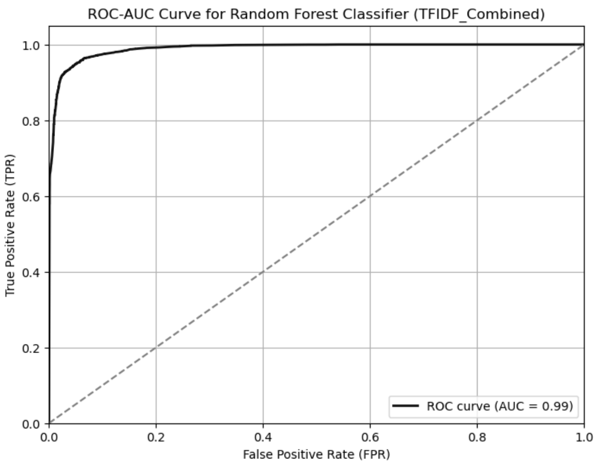

Figure 2. ROC-AUC curve for TF-IDF Vectorizer

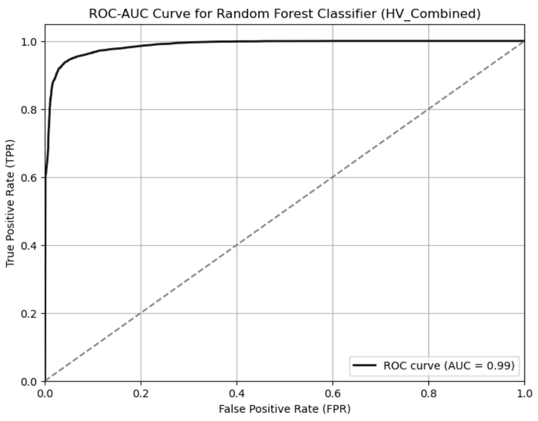

Figure 3. ROC-AUC curve for Hashing Vectorizer

5. Discussion

The proposed Count Vectorizer, when applied to a combination of webpage tags such as <div>, <meta>, and <p>, outperformed other vectorizers in most instances, particularly when used with Deep Neural Network (DNN) and Random Forest models. It achieved the highest accuracy at 93.1% and demonstrated superior performance across metrics such as precision, recall, and F1-score. This success can be attributed to its ability to capture a broader and more comprehensive set of features from multiple sections of webpage content, making it especially effective for classifying malicious webpages.

On the other hand, TF-IDF, while performing well, lagged behind the proposed vectorizer, particularly when used with DNN, where it reached an accuracy of 91.0%. Although TF-IDF excels in weighting words according to their importance, it does not combine structured metadata with unstructured content as effectively as the proposed vectorizer, making it less robust for this specific task.

The Hashing Vectorizer, known for its memory efficiency, displayed a lower performance compared to the proposed vectorizer, with a 90.3% accuracy when applied to DNN. Although its fixed-length feature space, which eliminates the need for storing vocabulary, is advantageous for handling high-dimensional datasets, it faces challenges with interpretability and sparsity, which are particularly limiting in malicious webpage detection.

Word2Vec, with a 92.0% accuracy, performed competitively and came close to the proposed vectorizer. Its ability to capture semantic relationships between words yielded strong results, especially with DNN models. However, Word2Vec's focus on word embeddings limits its capacity to integrate multi-view features from both metadata and webpage content, which the proposed Count Vectorizer effectively accomplishes.

6. CONCLUSION

In conclusion, this work has introduced a comprehensive approach to classifying malicious web content through the integration of Natural Language Processing (NLP) techniques and machine learning models. By leveraging both the structural and textual features extracted from various HTML tags, including <div>, <meta>, and <para>, we demonstrated that combining multiple content sources leads to enhanced accuracy, precision, recall, and F1-score in malicious webpage detection. The Count Vectorizer, applied individually to different tags and in combination, proved to be a robust feature extraction technique across several machine learning models. The proposed system was compared with existing vectorization methods such as TF-IDF, Hashing Vectorizer, and Word2Vec, showcasing its superior performance across a range of evaluation metrics. Through extensive experimentation, the proposed vectorizer model consistently outperformed existing methods, particularly when multiple tags were combined, leading to a more comprehensive feature set for classification. The introduction of the Count Vectorizer in this context allowed for more granular representation of webpage content, thus improving the ability of models to identify malicious behaviors. In addition, the inclusion of performance metrics such as accuracy, precision, recall, and F1-score across various machine learning algorithms illustrated the strength of the proposed system in real-time malicious content detection. The experimental results provided a clear comparative analysis, affirming the value of combining structural and textual features in malicious webpage classification tasks.

References

[1] Aljabri, M., Alhaidari, F., Mohammad, R. M., Mirza, S., Alhamed, D. H., Altamimi, H. S., Chrouf, S. M., & Ijaz, M. F. (2022). An assessment of lexical, network, and content-based features for detecting malicious URLs using machine learning and deep learning models. *Computational Intelligence and Neuroscience*, 2022. https://doi.org/10.1155/2022/3241216

[2] Fang, Y., Li, K., Zheng, R., Liao, S., & Wang, Y. (2021). A communication-channel-based method for detecting deeply camouflaged malicious traffic. *Computer Networks*, 197, Article 108297. https://doi.org/10.1016/j.comnet.2021.108297

[3] Haynes, K., Shirazi, H., & Ray, I. (2021). Lightweight URL-based phishing detection using natural language processing transformers for mobile devices. *Procedia Computer Science*, 1877-2509, 127-134. https://doi.org/10.1016/j.procs.2021.07.040

[4] Hong, Y., Li, Q., Yang, Y., & Shen, M. (2023). Graph-based encrypted malicious traffic detection with hybrid analysis of multi-view features. *Information Sciences*, 644, Article 119229. https://doi.org/10.1016/j.ins.2023.119229

[5] Shahraki, A., Abbasi, M., Taherkordi, A., & Jurcut, A. D. (2022). A comparative study on online machine learning techniques for network traffic streams analysis. *Computer Networks*, 207, Article 108836. https://doi.org/10.1016/j.comnet.2022.108836

[6] Shahrivar, V., Darabi, M. M., & Izadi, M. (2020). Phishing detection using machine learning techniques. *arXiv preprint*, arXiv:2009.11116v1.

[7] Tiefeng, W., Wang, M., Xi, Y., & Zhao, Z. (2022). Malicious URL detection model based on bidirectional gated recurrent unit and attention mechanism. *Applied Sciences*, 12(23), 12367. https://doi.org/10.3390/app122312367

[8] Wang, Z., Fok, K. W., & Thing, V. L. (2022). Machine learning for encrypted malicious traffic detection: Approaches, datasets, and comparative study. *Computer Security*, 113, Article 102542. https://doi.org/10.1016/j.cose.2021.102542

[9] Wang, Z., & Thing, V. L. (2023). Feature mining for encrypted malicious traffic detection with deep learning and other machine learning algorithms. *Computer Security*, 128, Article 103143. https://doi.org/10.1016/j.cose.2023.103143

[10] Zhang, P., He, F., Zhang, H., Hu, J., Huang, X., Wang, J., Yin, X., Zhu, H., & Li, Y. (2023). Real-time malicious traffic detection with online isolation forest over SD-WAN. *IEEE Transactions on Information Forensics and Security*, 18, 2076-2090. https://doi.org/10.1109/TIFS.2023.3262121

[11] Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open*, 3, 111-132. https://doi.org/10.1016/j.aiopen.2022.10.001

[12] Alex, S., & Rajkumar, T. D. (2021). Spider bird swarm algorithm with deep belief network for malicious javascript detection. *Computers & Security*, 10.

[13] Wan, Abdul, Mohd., Characterizing Current Features of Malicious Threats on Websites, Intelligent Computing & Optimization, vol 866. Springer, 2018.

[14] Malak, Fahd, Rami, Samiha, Dina, Hanan, Sara. An Assessment of Lexical, Network, and Content-Based Features for Detecting Malicious URLs Using Machine Learning and Deep Learning Models, Computational Intelligence and Neuroscience, Hindawi, 2022. https://doi.org/10.1155/2022/3241216

[15] Sirageldin, Baharudin, Jung. Malicious Web Page Detection: A Machine Learning Approach. Advances in Computer Science and its Applications. Lecture Notes in Electrical Engineering, vol 279. Springer, Berlin, Heidelberg, 2014. https://doi.org/10.1007/978-3-642-41674-3_32

[16] Desai, Jatakia, Naik, Raul. Malicious web content detection using machine leaning, 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, pp. 1432-1436, 2017. doi: 10.1109/RTEICT.2017.8256834.

[17] Canadian Institute for Cybersecurity. (2017). *CICIDS2017 dataset* [Data set]. University of New Brunswick. https://www.unb.ca/cic/datasets/ids-2017.html)

[18] PhishTank. (n.d.). *PhishTank dataset* [Data set]. OpenDNS. https://www.phishtank.com/

[19] Amazon Alexa. (n.d.). *Alexa Top Sites dataset* [Data set]. Alexa Internet. https://www.alexa.com/topsites

[20] Analytics Vidhya. (2020, December). *Understanding text classification in NLP with a movie review example*. Retrieved April 29, 2023, from https://www.analyticsvidhya.com/blog/2020/12/understanding-text-classification-in-nlp-with-movie-review-example-example/