Real Time Phishing Detection Using Lexical Analysis and Visual Similarity

A. Gnanesh(✉), Dasa A. Deepesh, Bhargav Hegde, Shreehari Vyasamudri, and V. Sarasvathi

PES University, Bengaluru 560100, India
gnaneshanand600@gmail.com, sarsvathiv@pes.edu

Abstract. Phishing, a form of social engineering, involves deceptive practices to extract sensitive information from individuals. Typically, attackers manipulate their messages to mimic genuine communication from reputable entities like financial institutions, social networks, or online marketplaces. Phishing attempts manifest across various communication channels, encompassing email, text messages, and social media platforms. The acquired sensitive data fuels identity theft and financial fraud, underscoring the importance of vigilance when encountering unsolicited communications or hyperlinks soliciting personal information or immediate action. This paper offers a comprehensive approach to tackle phishing hazards, employing diverse features including URL structure for lexical analysis and screen-shot for visual similarity using transfer learning. Notably, the XGBoostClassifier attains an outstanding accuracy rate of 96.89%. To bolster this, a hybrid approach is adopted, integrating the MobileNet model for visual similarity-based detection. The incorporation of the MobileNet model introduces a visual similarity-based detection layer, augmenting the system's capabilities. Implemented as a Chrome plug-in, this system dynamically scrutinizes website URLs, promptly alerting users about potential phishing threats. Rigorous testing on real-world phishing websites showcases the method's robust performance, offering a reliable and user-friendly solution to detect and prevent phishing attacks. The hybrid integration of feature based detection through XGBoostClassifier and visual similarity analysis using MobileNet fortifies the system, elevating its effectiveness in safeguarding against evolving phishing assaults.

Keywords: Phishing · Machine Learning · URL structure · Lexical Analysis · XGBoost · Visual Similarity · MobileNetV2 · Transfer Learning · Logo Detection · Chrome Extension

1 Introduction

The internet, particularly WWW, plays a vital role in contemporary life by offering various services like social networking, banking, and online shopping. Its widespread use is attributed to the convenience it provides, allowing users

X. Cheng (Ed): BROADNETS 2024, LNICST 601, pp. 157–175, 2025.
https://doi.org/10.1007/978-3-031-81168-5_15

to perform essential tasks from their homes. But, this convenience comes with a downside - the heightened risk of cyberattacks. Each year, a substantial number of individuals inadvertently fall prey to diverse cybercrimes. A late 2021 study by OpSec Security [1] found that phishing attacks in the financial sector were the most prevalent, accounting for over 23.2% of all phishing incidents they observed during that timeframe. (refer to Fig. 1). A study by McAfee and the Center for Strategic and International Studies (CSIS) estimates the global cost of cybercrime to be approaching 600 billion US dollars [2]. Phishing, a significant type of cyberattack affecting both individuals and businesses, involves attackers masquerading as trustworthy entities to extract sensitive information from users. This often includes redirecting users to counterfeit websites designed to resemble legitimate ones. The URL of these deceptive websites is typically propagated through emails or instant messages. Detecting such fraudulent websites can be highly challenging for users based solely on the webpage content. Phishing attacks can inflict an average cost of 1.5 million US dollars on a mid-sized company [3]. Phishing URLs are important to be aware of, but they represent just one type of malicious link. Others include URLs that attempt to automatically download malware and URLs used for sending spam. While this discussion primarily focuses on the detection of phishing URLs, the overarching process can be adapted to identify various types of malicious URLs. The prevailing method for identifying phishing URLs involves utilizing blacklists - regularly updated lists of known malicious URLs curated by the community or cybersecurity experts. A report from Webroot [4] highlights the challenge of effectively blocking phishing websites, as roughly 1.5 million new ones appear each month. This rapid emergence makes it nearly impossible to maintain a comprehensive blacklist of all malicious URLs. Real-time phishing detection often relies on whitelist systems. These systems maintain lists of known good websites, making them popular due to their fast implementation and ability to provide immediate response. However, they face challenges in list management and struggle to identify newly created phishing URLs, often referred to as zero-hour phishing attacks. In addition to Machine Learning (ML), we have integrated visual similarity techniques as many phishing webpages closely mimic their legitimate counterparts. These methods involve comparing fake websites with legitimate ones using various visual characteristics, such as text formatting, website logos, screenshots of the webpage, and images. Consequently, there is a pressing need for an automated system capable of identifying new phishing URLs, accurately detecting previously unseen phishing URLs without necessitating human intervention in real-time.

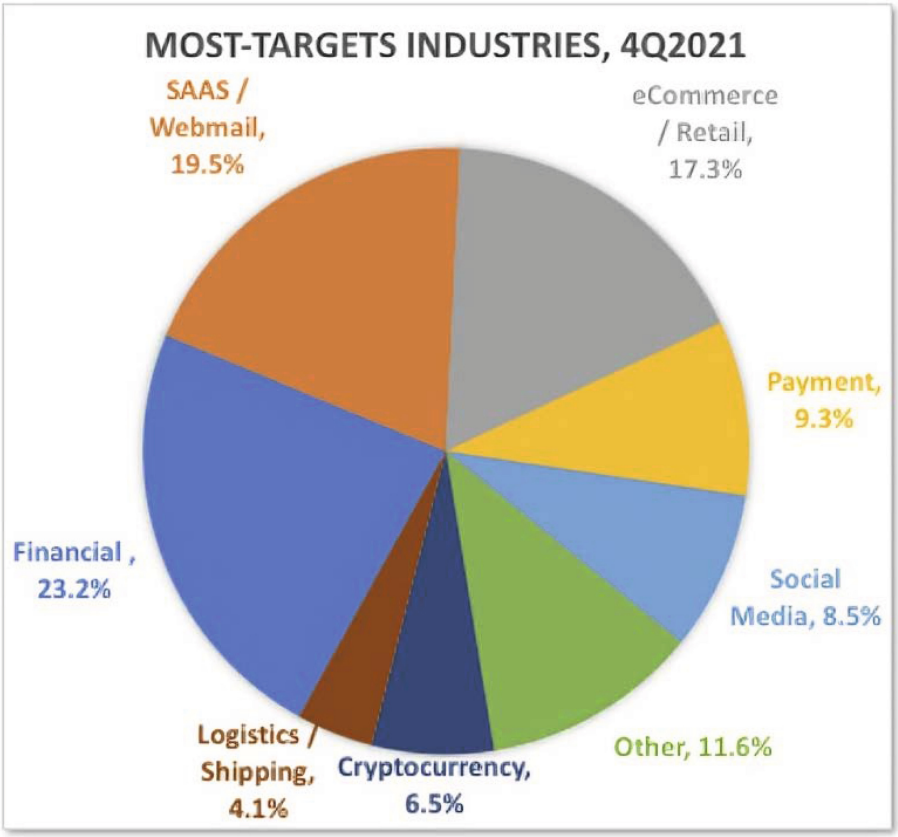

Fig. 1. Phishing Most-Targets Industries. [Source: https://rb.gy/5ouorh]

1.1 Challenges and Motivation

Despite promising initial results, several factors have prevented the widespread adoption of automatic malicious URL detection:

1. **Lack of a standardized framework:** There's no single, comprehensive approach to identifying malicious URLs, making it difficult for organizations to choose and implement a reliable solution.
2. **Real-time challenges:** Machine learning algorithms can be slow when used for real-time URL analysis, potentially delaying critical decisions and slowing down operations.
3. **Imbalanced data:** The vast majority of URLs are legitimate, making it harder for algorithms to accurately identify the relatively few malicious ones.
4. **Dynamic environment:** URLs constantly change and evolve, requiring frequent updates to detection models to maintain effectiveness.

1.2 Contribution

To enhance our phishing URL detection capabilities, we employed visualization techniques that involve capturing screenshots of web pages and employing comparative analyses against their legitimate counterparts. This visual inspection method allows us to pinpoint phishing web-pages that closely mimic authentic sites, contributing to a more robust detection mechanism. Furthermore, an in-depth analysis of feature importance was conducted to extract meaningful insights from the data-set. This analysis aided us in the meticulous selection of features, strategically opting for those that not only contribute significantly to accurate identification but also result in minimal execution time. Our feature selection process was aimed at optimizing system performance without compromising its efficiency, thereby ensuring swift and precise identification of phishing URLs.

2 Related Works

Alex Sumner et al.'s work [5] emphasizes the importance of a multifaceted approach to enhance cybersecurity and mitigate the risks associated with phishing attacks. Behavioral analysis is a crucial component, involving the scrutiny of user activities to identify anomalies that may indicate a phishing attack, such as unusual access to sensitive data. Additionally, URL analysis, which includes techniques like blacklisting and whitelisting, helps verify the authenticity of URLs, preventing access to known phishing sites while allowing access to legitimate ones. This comprehensive strategy contributes to a more robust defense against the evolving threat landscape of phishing. In terms of benefits, the report provides a thorough review of phishing assaults and their impact on both individuals and organizations. The study also explores various methods and approaches for reducing the danger of phishing attacks, offering valuable insights for cybersecurity experts and individuals seeking to protect themselves from such threats. However, a notable limitation of this paper is the absence of an effective solution for mitigating phishing attacks. The discussed solutions are deemed outdated in modern times, given the increasing complexity of phishing attacks, wherein attackers may employ social engineering or manipulation tactics.

Pankaj Pandey et al.'s work [6] proposes Phish-Sight a creative method for identifying phishing websites, emphasizing the sophistication of phishing attempts and the need for improved detection methods. The method involves analyzing standout hues on a web page to determine if it is a phishing website, with phishing sites commonly utilizing specific color schemes to deceive users. Machine learning is employed to analyze dominating colors, comparing the method's efficacy with existing approaches using a dataset of phishing and non-phishing websites. Various ML algorithms, including Decision Tree, Logistic Regression, Naive Bayes, Random Forest, and Support Vector Machine, are utilized, with Random Forest proving most effective. Advantages include the ability to recognize Zero Day Attacks, challenging crawling using this technique, and the

use of OCR to enhance understanding of text in web page screenshots. PhishSight is independent of language and platforms, collecting URLs from Openphish every five minutes. Limitations include the focus on extracting dominant colors and popular brand names, considering only 18 brand names. The method's time-consuming nature, approximately 8.4 s, is acknowledged, with potential optimization opportunities based on features and ML algorithms. Thahira A et al.'s work [7] addresses concerns about phishing attacks and the need for effective detection methods, proposing a strategy that examines a website's URL. The authors train a machine learning model, the LGBM classifier, using lexical variables extracted from the URL, such as specific keywords and URL length. Unlike conventional methods focusing on the domain and IP address, this approach concentrates on lexical characteristics to overcome identified shortcomings. The study introduces a methodology for real-time detection of new phishing URLs using a novel dataset. Advantages include the superior performance of the LGBM over the Random Forest and the use of a lightweight ML model, enabling fast real-time computation. The dataset exhibits no bias, with an equal number of legitimate and phishing features. The study tests eight ML algorithms. Limitations involve the exclusive consideration of URL-based lexical features and a relatively low number of features. The absence of website visualization is noted as a drawback.

Mehmet Korkmaz et al.'s work [8], proposes a machine learning-based strategy for identifying phishing websites. The authors emphasize the escalating frequency of phishing attacks and the need for efficient detection methods. The approach involves analyzing a website's URL, with the authors considering aspects such as the domain name and the presence of specific keywords to train a machine learning model. A dataset of phishing and non-phishing URLs is used to evaluate the efficacy of the strategy, demonstrating superior performance compared to other methods. The study employs various ML models and utilizes three datasets, considering 58 features to enhance accuracy. By avoiding third-party features, the method is expected to categorize more rapidly than alternative models. Limitations include the study's exclusive reliance on URL analysis, resulting in decreased effectiveness in identifying phishing websites. Notably, no website visuals are incorporated into the analysis.

M. Amir Syafiq Rohmat Rose et al.'s work [9], introduces an innovative approach to identify and prevent phishing attacks through a Chrome browser extension. The authors emphasize the increasing frequency of phishing assaults and the need for practical defenses. They developed a Chrome extension based on their proposed strategy and tested it on a dataset of phishing websites to assess its effectiveness. Results indicate that a significant portion of phishing attacks was successfully detected and thwarted by the extension. The study primarily employs a machine learning algorithm focusing on URL-based features to recognize and prevent phishing attacks, utilizing SVM, RF, and ANN. This method is effective in detecting Smishing campaign activities, improving accuracy through an adaptive algorithm. Its speed is notable as it does not rely on external web services, and the use of a Chrome extension enhances practicality. Drawbacks

include a limited number of features considered, experimentation conducted in a controlled lab setting, making real-world accuracy prediction challenging. The study's reliance on a small dataset of 3317 test samples for ML algorithm training and the omission of specific detection time details are noted. Additionally, the evaluation focuses on only three metrics (accuracy, sensitivity, false-positive rate) and explores a limited range of ML algorithms.

Sahar Abdelnabi et al.'s work [10], proposes a phishing detection method called VisualPhishNet. Using deep learning on screenshots, it achieves high accuracy with minimal false positives. This real-time system works independently of URLs or text, requiring only a screenshot for analysis. It even retrieves similar legitimate websites for user verification. However, limitations include variations in browser size, ineffectiveness against unique designs, and the need for a costly, ever-expanding dataset of screenshots. The extracted features rely on the quality of the pre-trained model and may require adaptation to newer techniques. While effective, its reliance solely on visual features suggests combining it with other modalities for even better results.

Anjaneya Awasthi et al.'s work [11], employs explicit rules to categorize URLs as suspicious, safe, or phishing websites. The study utilizes the Apriori algorithm to generate rules for identifying phishing websites, relying on clear and fast principles such as URL length, the number of links, and page redirections. The approach uses simple data mining and statistical methods, ensuring ease of implementation and faster processing compared to alternative methods. While the method employs hard and fast rules, its efficacy is not guaranteed in all cases, and the approach lacks specificity regarding accuracy. The study's limitations include a lack of detailed information on the Apriori algorithm's rule generation procedure, hindering replication and validation. Additionally, there is insufficient analysis of the strategy's performance in recognizing phishing and non-phishing URLs, limiting its practical utility. Implementation difficulties and constraints are not thoroughly considered, potentially impeding real-world application. Furthermore, scalability and adaptability concerns are overlooked, impacting the method's ability to handle large datasets and evolving phishing tactics.

Farhan Sadique et al.'s work [12], categorizes features into lexical, host-based, Domain WHOIS-based, and GeoIP-based components. The extraction of URL elements, including protocol, domain, path, and query parameters, is utilized for phishing URL detection. Specifically, lexical and host-based features, such as length, the presence of an IP address, and the @ symbol, are employed for training a machine learning model. The approach introduces the innovative concept of delayed feature collection, enhancing precision and time efficiency through selective sampling. The paper adopts the Random Forest technique as a crucial part of its methodology. Despite achieving an impressive accuracy of 96.7% and maintaining a minimal false positive rate of 0.8% in phishing URL detection, the system has notable disadvantages. The WhoIS and GeoIP components exhibit relatively high processing times, and the system faces challenges in detecting phishing URLs using innovative methods like homograph and Unicode attacks. Continuous updates are essential for the framework to adapt to evolving phishing techniques. However, the framework overlooks the analysis of page content and

user behavior, elements that could enhance detection accuracy. Additionally, the absence of a sizable, publicly available dataset of phishing URLs limits the thorough evaluation of the framework, emphasizing the need for further assessment on larger and more recent datasets to ensure comprehensive validation.

3 Proposed Methodology

The proposed methodology, as visually represented in Fig. 2, serves as the foundation of our proposed system. In the following sections, we provide a detailed exploration, focusing on the intricacies of Lexical Analysis and Visual Analysis. These components play pivotal roles in enhancing the functionality and effectiveness of the overall system.

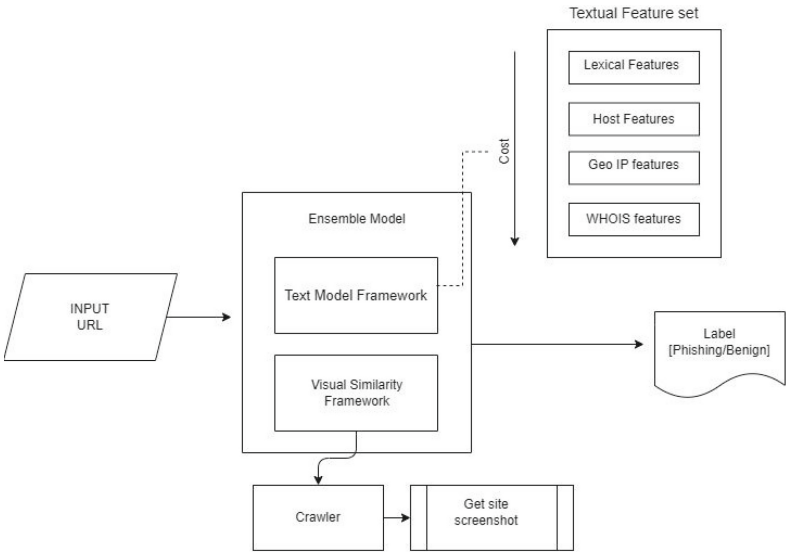

Fig. 2. System Architecture

3.1 Lexical Analysis

Lexical analysis involves thoroughly examining the structure and composition of a URL to identify patterns or characteristics commonly linked to phishing attacks. This analytical process is integral to the toolkit of cyber security professionals for the detection and mitigation of phishing threats. By scrutinizing the elements, character compositions, and patterns within URLs, security systems can unveil subtle indicators that may signal phishing attempts. Detection of anomalies, such as misspelled domain names or excessively long URLs, is crucial in identifying malicious links aiming to imitate legitimate websites. Lexical

analysis also plays a key role in recognizing keywords associated with phishing, enhancing the capability to identify URLs designed to deceive users into disclosing sensitive information. This proactive approach, combined with assessing domain ownership details and expiration dates through WHOIS, establishes a robust defense mechanism against evolving phishing tactics, ensuring users are protected from falling victim to deceptive online schemes.

3.2 Domain Analysis

Key Techniques

1. **URL Length Analysis**
   - Examine URL structure for excessively long or convoluted patterns that deviate from standard norms.
   - Implement lexical analysis to compare URL length against typical website patterns and identify abnormalities.
2. **Redirection and Tiny URL Detection**
   - Analyze the presence of redirects, multiple subdomains, or Tiny URLs as potential techniques to mask the true destination.
   - Recognize these elements as indicators of suspicious URL behavior.
3. **HTTPS Verification**
   - Verify whether the URL utilizes HTTPS, the standard for secure website connections.
   - Be cautious of URLs using HTTP or deceptive combinations of secure and non-secure elements, as these could be phishing attempts.
4. **Domain Expiry Date Assessment**
   - Evaluate the domain's expiration date. Legitimate sites often have extended registrations, while phishing sites may favor short-term periods to reduce costs and detection risks.

We have carefully selected 111 features for lexical analysis, taking into account their significance and the extraction time involved. This meticulous feature selection is crucial given our objective of developing a real-time phishing detection system. Leveraging these identified URL-based features, we have developed an intelligent model designed for the detection and prevention of phishing attempts. This model utilizes a machine learning algorithm to detect self-destructing content. We opted for XGBoost, an ensemble learning algorithm that aggregates predictions from multiple weak models, typically decision trees, to improve predictive accuracy and generalization. Particularly effective in dealing with imbalanced phishing detection datasets, XGBoost addresses bias concerns by offering parameters to handle class imbalances. It provides feature importance scores, enhancing interpretability for security analysts by highlighting key contributors to phishing detection. XGBoost's inherent mechanisms effectively handle missing data, ensuring robust learning from incomplete datasets. With an array of hyperparameters, the model can be finely tuned to suit specific phishing detection scenarios. The technique of tree pruning is employed to control the complexity of individual decision trees, preventing overfitting and promoting better generalization.

3.3 Visual Analysis

Visual Similarity is a good technique to detect phishing because of the following reasons

1. **Resilience to URL Manipulation:** Visual similarity analysis is independent of URL structures or text content, making it resilient to tactics employed by phishers to mimic legitimate URLs. Even if a phishing site alters its URL, visual cues such as logos and layout remain consistent and can be used for identification.
2. **Effective Against Sophisticated Phishing Techniques:** Sophisticated phishing attempts often replicate the entire visual appearance of trusted websites. Visual similarity analysis can effectively detect these replicas by focusing on minute visual elements, such as logos, color schemes, and page layout, which are challenging for phishers to replicate perfectly.
3. **Complementary to Other Detection Methods:** When integrated with other techniques like WHOIS verification or machine learning models analyzing content, visual similarity serves as an additional layer of confirmation. This synergy enhances the accuracy and reliability of phishing detection systems.

To evaluate the visual similarity between legitimate and phishing websites, a dataset was constructed from the top 25 trusted sites, including industry giants such as Alibaba, AOL, Apple, Bank of America (BoA), Chase, DHL, Dropbox, Facebook, Google, Microsoft, Office, Orange, PayPal, Wells Fargo, and Yahoo. Screenshots of these websites were collected and annotated with bounding box coordinates around their respective logos. Using Selenium, screenshots of these websites are captured. These screenshots are then run on our visual ML model. To extract and identify logos within the collected screenshots, transfer learning was employed, leveraging the MobileNetV2 architecture. Transfer learning involves utilizing pre-trained models and adapting them to a specific task, thereby benefiting from the knowledge gained during the training on a large data-set. MobileNetV2, known for its efficiency and accuracy in computer vision tasks, was chosen as the base architecture for logo detection. Its lightweight design and ability to handle image classification tasks efficiently, even in resource-constrained environments, made it suitable for processing screenshots in real-time. Prior to logo detection, screenshots were pre-processed to standardize dimensions and enhance model compatibility. Pre-processing steps included resizing images to the required input size of MobileNetV2 and normalization to ensure uniformity in pixel values. MobileNetV2, initially trained on an extensive dataset like ImageNet, underwent further refinement through fine-tuning on a specialized dataset that includes annotated logos sourced from reputable websites. This fine-tuning process involved retraining the final layers of the network while preserving the convolution base's learned features.

The system utilizes MobileNet's ability to recognize visual elements like logos, color schemes, and page layouts to achieve high accuracy in detecting phishing websites. MobileNet's effectiveness in computer vision tasks contributes to a

more robust detection mechanism. Moreover, MobileNet's proficiency in analyzing visual design and layout allows the system to effectively detect near-duplicate and impersonating web pages. By identifying subtle variations indicative of phishing activities, the system can better distinguish between legitimate and phishing websites.

3.4 Algorithm

Algorithm 1 Gradient Boosting Algorithm

```
Initialize model parameters
for each boosting round do
    Compute negative gradient of the loss function
    Fit a weak learner to the negative gradient
    Update the model with the weak learner's output
    Apply regularization to prevent over-fitting
    Update the weights of the instances based on their residuals
end for
End training
for prediction do
    for each weak learner in the ensemble do
        Make predictions with the weak learner
    end for
    Combine the predictions to obtain the final output
end for
```

The presented algorithm outlines the process of Gradient Boosting, a powerful machine learning technique. The procedure initializes model parameters and iteratively conducts boosting rounds. In each round, it computes the negative gradient of the loss function, fits a weak learner to this gradient, and updates the model with the output of the weak learner. To prevent overfitting, regularization is applied, and the weights of instances are updated based on their residuals. This training process enhances the model's predictive capabilities.For the chosen dataset which consists of phishing url's taken from phishing tank XGBoost offer advantages over Random Forest. XGBoost's sequential learning and gradient boosting allow it to adapt quickly to emerging phishing patterns, enhancing its ability to correct errors and improve accuracy over time. The incorporation of regularization helps prevent overfitting, crucial for handling dynamic and evolving phishing attacks. XGBoost's efficient handling of missing values and detailed feature importance measures make it well-suited for identifying subtle patterns indicative of phishing behavior. While Random Forest remains a viable option, XGBoost's speed, adaptability, and robustness make it a compelling choice for real-time phishing detection.

Algorithm 2 MobileNetV2 Algorithm

```
function ConvBlock(input, filters, kernel_size, stride)
    x ← Conv2D(input, filters, kernel_size, stride)
    x ← BatchNormalization(x)
    x ← ReLU(x)
    return x
end function
function DepthwiseConvBlock(input, filters, kernel_size, stride)

    return x
end function
function InvertedResidualBlock(input, filters, expansion, stride)
    x ← ConvBlock(input, filters × expansion, 1, 1)
    x ← DepthwiseConvBlock(x, filters, kernel_size − 3, stride − stride)
    x ← Conv2D(x, filters, 1, 1)
    if stride − 1 and input_filters − filters then
        return x + input
    else
        return x
    end if
end function
function MobileNetV2(input)
    x ← ConvBlock(input, 32, 3, 2)
    x ← InvertedResidualBlock(x, 16, 1, 1)
    x ← InvertedResidualBlock(x, 24, 6, 2)
    x ← InvertedResidualBlock(x, 32, 6, 2)
    x ← InvertedResidualBlock(x, 64, 6, 2)
    x ← InvertedResidualBlock(x, 96, 6, 1)
    x ← InvertedResidualBlock(x, 160, 6, 2)
    x ← InvertedResidualBlock(x, 320, 6, 1)
    x ← ConvBlock(x, 1280, 1, 1)
    x ← GlobalAveragePooling2D(x)
    x ← Dense(x, num_classes)
    return x
end function
```

The MobileNetV2 algorithm is a lightweight and efficient deep neural network architecture designed for mobile and embedded devices. It utilizes a series of building blocks, including convolutional blocks, depthwise convolution blocks, and inverted residual blocks. The ConvBlock function applies a standard convolution operation followed by batch normalization and rectified linear unit (ReLU) activation. The DepthwiseConvBlock function employs depthwise separable convolutions, contributing to the model's efficiency. The InvertedResidualBlock function represents a key component, incorporating expansion, depthwise convolution, and linear projection layers. The overall MobileNetV2 model consists of a sequence of these blocks, progressively increasing in complexity, and is capable of capturing hierarchical features while maintaining computational efficiency. The final layers include a global average pooling operation, followed by a fully connected layer with softmax activation for classification. Notably, MobileNetV2 achieves a good balance between model accuracy and computational efficiency, making it well-suited for resource-constrained environments.

4 Implementation and Results

The initial step involves WHOIS domain verification, where the incoming website's domain is checked against a precompiled list of trusted domains. Sites identified within this trusted catalog are instantly deemed safe. However, sites not found within the trusted list proceed to the next stage for further evaluation. For websites not listed within the trusted domains we run lexical model where we check the lexical features. If lexical model detect it as benign site we run the visual model else we alert the user it's a phishing site.

4.1 Lexical Analysis

In phishing detection, lexical analysis involves examining the lexical elements of digital content, such as emails or URLs, to identify patterns associated with phishing attacks. This includes analyzing keywords, URL structures, expiry of the domain and other linguistic features to detect deceptive practices and patterns commonly used by attackers, enhancing the accuracy of phishing detection systems. The data-set used to train the model was obtained from https://www.kaggle.com for benign URLs, and for phishing URLs, we obtained it from https://phishtank.com. In total, there were 500k URLs, out of which 345,738 are benign, and 210,634 are phishing URLs. Since there was an imbalance in the data-set, which can lead to biased model performance, model skewing, and misleading accuracy, We employed the Synthetic Minority Oversampling Technique (SMOTE), a strategy utilized to tackle the challenge of imbalanced class distribution in machine learning data-sets. Taking cost into consideration, we meticulously chose 111 features. The following link contains all the 111 features used for prediction https://docs.google.com/document/d/1FZn99ddI5-P3IsGUU9uckfaf4_IT4qrTK4Nx1V-RK_8/edit?usp=sharing There is feature extractor code which is used to extract all the 111 features and this feature vector is passed to the pre-trained model for prediction. We have used XGBOOST model for prediction. XGBoost, or eXtreme Gradient Boosting, is notably effective in classification tasks, demonstrating high accuracy in both binary and multiclass scenarios. Additionally, XGBoost excels in regression tasks, showcasing its ability to model complex relationships in data and handle noisy inputs. XGBoost is also adept at anomaly detection, particularly useful in cybersecurity and fraud detection. XGBoost performs well with imbalanced data-sets, making it suitable for scenarios where one class is predominant.

4.2 Visual Analysis

The model incorporates a diverse set of libraries and technologies to develop and deploy a visual similarity model. These include:

– **silence_tensorflow:** Designed to suppress TensorFlow logging and enhance output clarity, preventing excessive messages during execution.

- **numpy** and **pandas:** Essential numerical and data manipulation libraries, foundational for array, matrix, and structured data operations.
- **TensorFlow:** The cornerstone for building and loading the visual similarity model. Model loading is achieved using `tf.saved_model.load`, with the path specified by the `PATH_TO_SAVED_MODEL` environment variable.
- **os** and **sys:** Standard Python libraries for interacting with the operating system and accessing system-specific parameters and functions.
- **TrustedSites**, **WebCrawler**, and **timer:** Custom modules likely designed for managing trusted websites, implementing web crawling functionality, and measuring elapsed time during execution, respectively.
- **dotenv:** Facilitates loading environment variables from a file, streamlining configuration management.
- **PIL (Pillow):** Python Imaging Library enabling image file operations, including opening, manipulation, and saving.

4.3 Results

Table 1 depicts that XGBoost stands out as a superior choice for training phishing datasets due to its ensemble learning approach, which sequentially corrects errors, enhancing overall model accuracy. Unlike Random Forests, XGBoost mitigates overfitting through regularization techniques, ensuring robust generalization. Compared to Support Vector Machines (SVM) and Multilayer Perceptrons (MLP), XGBoost often exhibits faster training times while maintaining high predictive performance. Additionally, XGBoost excels in capturing complex relationships within data, surpassing the capabilities of AutoEncoders. Its adaptability and scalability make XGBoost a preferred option for effectively handling the intricacies of phishing datasets, outperforming SVM and Random Forest models.

Table 1. Comparison of models

ML Model	Train Acc.	Test Acc.	Precision	Recall	F-score
Decision Tree	0.810	0.826	0.9338	0.9504	0.9420
Random Forest	0.814	0.834	0.9147	0.9326	0.9236
Multilayer Perceptrons	0.858	0.863	0.9640	0.9789	0.9714
XGBoost	0.981	0.968	0.95	0.96	0.96
AutoEncoder	0.819	0.818	0.9433	0.9796	0.9611
SVM	0.798	0.818	0.8947	0.9026	0.9136

Figure 3 illustrates a confusion matrix, a tabular representation showcasing how well a machine learning model performs on a given dataset. It provides a breakdown of correct and incorrect predictions for each class within the dataset, which, in this instance, comprises two classes: positive and negative. Precision, recall, and F1 score are widely employed metrics for assessing the effectiveness of

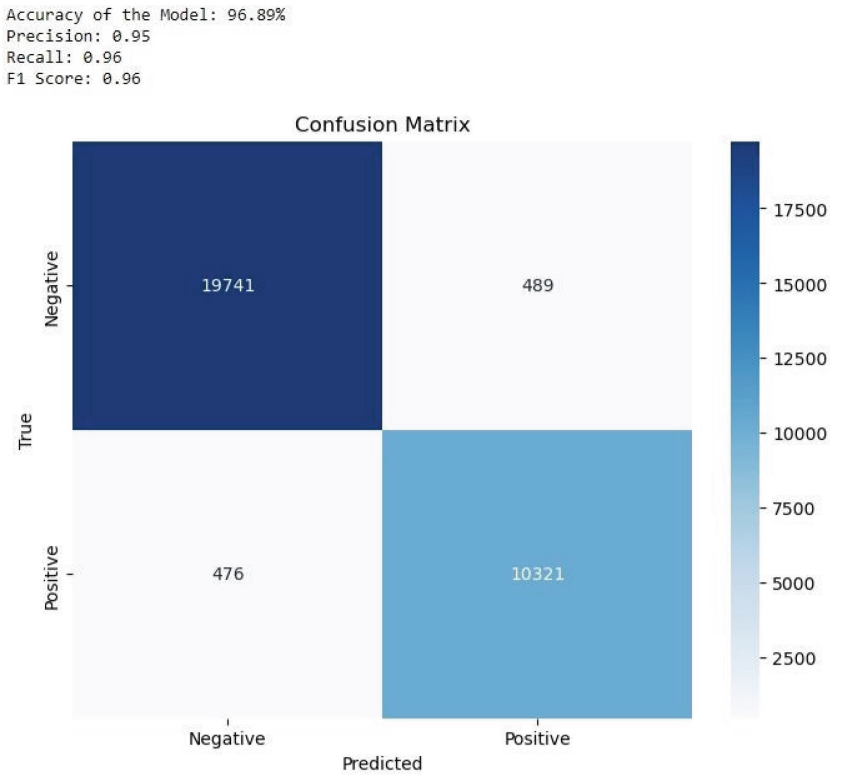

Fig. 3. Confusion Matrix

classification models, relying on key elements such as true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

These metrics become particularly valuable when dealing with imbalanced class distributions, requiring a careful consideration of both false positives and false negatives. Precision assesses the accuracy of positive predictions, while recall measures the coverage of actual positives. The F1 score offers a balanced evaluation, incorporating both precision and recall. The model's accuracy stands at 96.89%, indicating accurate predictions for the majority of data points. When comparing our findings with those obtained by other researchers, the observed metric might appear comparatively lower. However, it's essential to note that the dataset we selected is the most recent one from PhishTank. The evaluation of our framework yielded promising results, particularly in detecting the latest phishing URLs. With a precision of 95%, the model correctly identifies 95% of predicted positive instances. A recall of 96% signifies the model's ability to capture 96% of positive data points. The F1 score, at 0.96, represents a harmonized assessment of precision and recall.

4.4 Chrome Extension and User Interface

We created a straightforward and user-friendly interface for the plug-in, utilizing HTML and CSS [5]. A Chrome extension for real-time phishing detection provides users with instant protection, seamlessly integrating into their browsing experience. It delivers timely alerts, preventing users from falling victim to phishing attacks and enhancing security awareness. Regular updates and customization options ensure ongoing defense against evolving threats. With low resource consumption and compatibility with browser security features, these extensions offer efficient, multi-platform protection, utilizing community-driven threat intelligence for robust and personalized security. Here the user can click on the SAFE OR NOT button to check whether the current page is phishing or not. If it displays as phishing then the user can stop browsing the site. If it displays as benign then the user can continue browsing. We have also added cache where we store resent searched URL, So if the user search for the same URL we do not need to run the model again thus saves time and overhead. We have also built a website where the user can enter the URL manually to check whether it's a phishing site or not. If the user wants to check the URL before clicking on it he/she can use the website to check whether its phishing or not (Figs. 4, 5, 6 and 7).

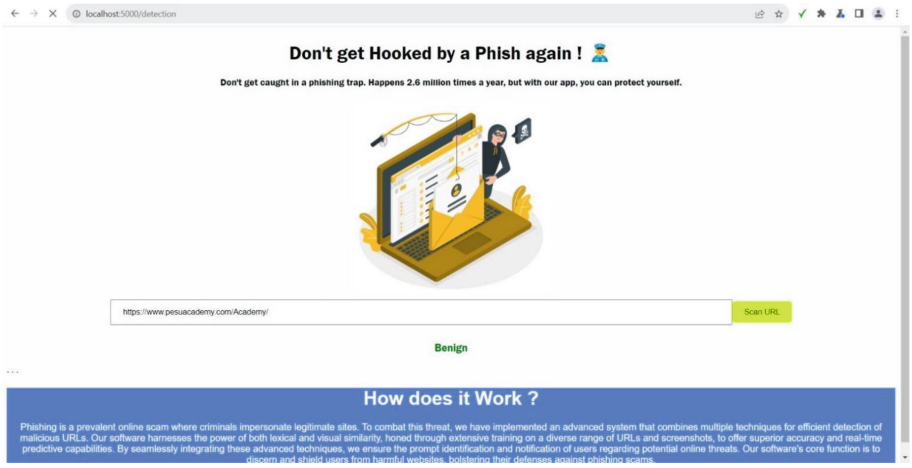

Fig. 4. Shows https://www.pesuacademy.com/Academy/ classified as Benign

5 Limitation and Challenges

Real-time detection of phishing through a Chrome extension, utilizing lexical similarity and visual analysis, confronts various challenges. The dynamic nature

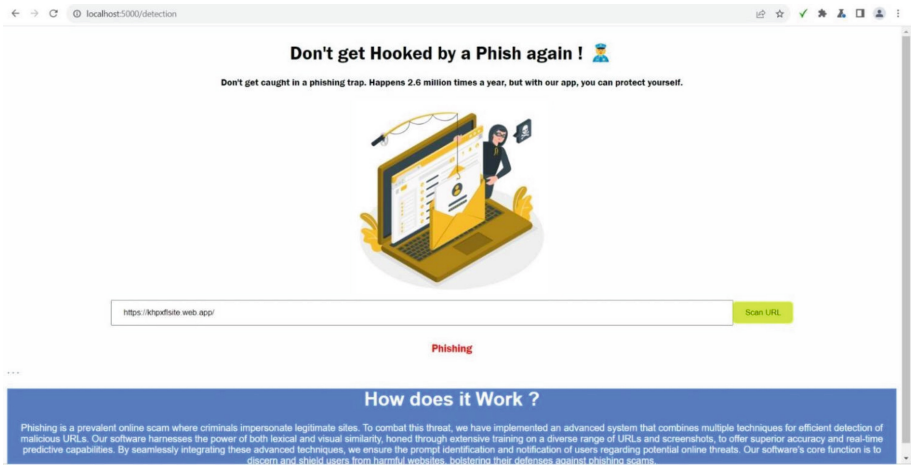

Fig. 5. Shows http://khpxfisite.web.app/ classified as Phishing

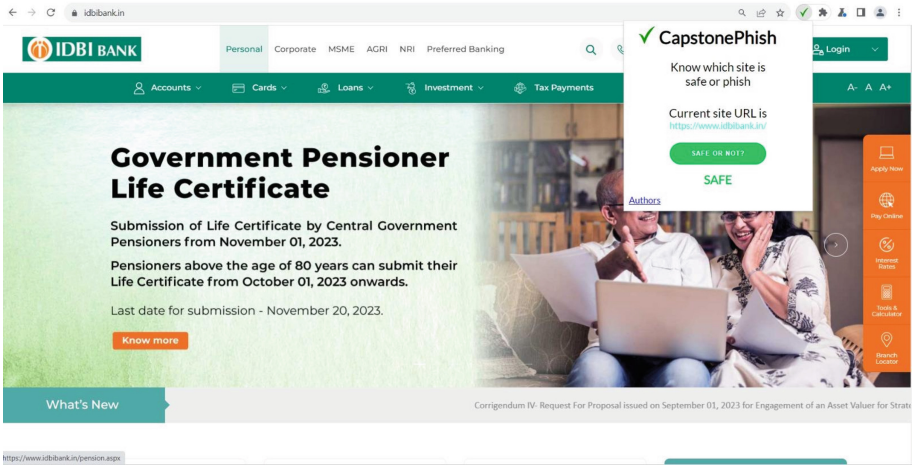

Fig. 6. Shows https://idbibank.in classified as SAFE using extension

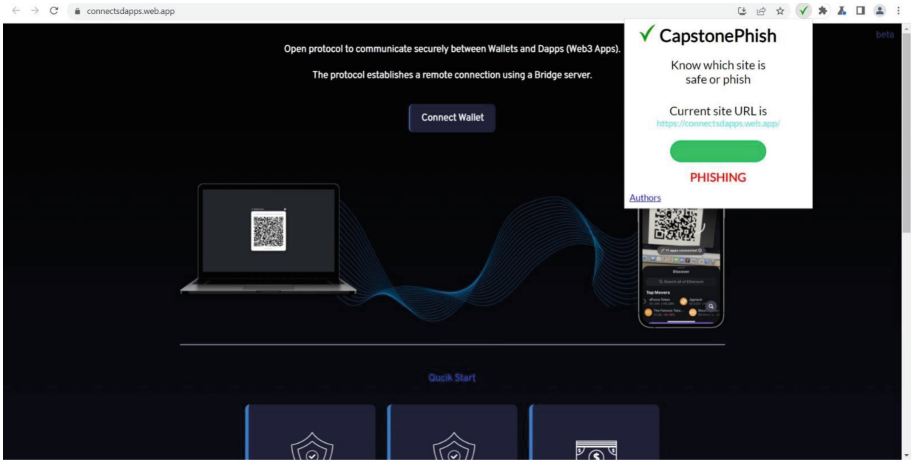

Fig. 7. Shows https://connectsdapps.web.app classified as PHISHING using extension

of phishing attacks and the capacity of phishers to replicate legitimate sites present substantial obstacles for both static lexical and visual analyses. Obfuscation techniques employed by attackers can compromise the effectiveness of lexical analysis, while visually convincing phishing pages pose a challenge to the dependability of visual analysis. Changes in user-interface designs on authentic websites and issues of compatibility across different browsers further complicate the accuracy of detection. Striking a balance between false positives and negatives proves to be a persistent challenge, and the computational intensity of real-time analysis may impact the performance of the browser. Privacy concerns associated with visual analysis could also influence user acceptance. Moreover, the framework's scope may be restricted, providing room for certain sophisticated phishing techniques to elude detection. It is imperative to address these limitations to enhance the effectiveness of the extension in real-time phishing detection.

6 Conclusion

In our real-time phishing detection implementation, we adopted an ensemble model that seamlessly integrated insights from two distinct frameworks: the lexical framework, driven by the XGBoost model, and the visual framework, utilizing the capabilities of MobileNet. The XGBoost model within the lexical framework meticulously analyzed the text content of web pages, extracting pertinent features and employing the XGBoost algorithm to discern phishing patterns embedded within the textual data. Concurrently, the visual framework harnessed the MobileNet model to conduct image processing and employ computer vision techniques, scrutinizing the visual elements of web pages to identify crucial visual cues that distinguished authentic websites from fraudulent counterparts.

The strength of our ensemble model lay in its ability to harmoniously merge outputs from both the lexical and visual frameworks, yielding a robust solution for real-time phishing detection. This strategic integration of lexical and visual analyses not only capitalized on the unique strengths of each framework but also bolstered the system's resilience against evasion tactics employed by malicious actors. Our research makes a valuable contribution to the ongoing enhancement of cybersecurity defenses, highlighting the efficacy of ensemble models that synergize lexical and visual analyses to fortify internet security. Subsequent efforts could concentrate on fine-tuning the ensemble model, refining feature engineering methodologies, and continually updating the models to stay ahead of emerging phishing techniques and evolving cyber threats. Eliminating phishing is tough, but we can fight back! Individuals need awareness, strong passwords, and smart clicking habits. Tech can help with email checks, secure websites, and AI detection. Collaboration between users, companies, and governments with better laws can create a safer online space. Remember, vigilance is key!

References

1. (APWG), A.P.W.G.: Phishing activity trends report, 4th quarter 2021 (2021). Accessed 26 Apr 2022
2. Lewis, J.: Economic impact of cybercrime, no slowing down. McAfee (2018)
3. Enterprise phishing resiliency and defense report (2017)
4. Quarterly threat trends (2019). https://www.webroot.com/us/en/business/resources/threat-trends/june-2019/. Accessed 1 Sept 2019
5. Sumner, A., Yuan, X.: Mitigating phishing attacks: an overview. In: Proceedings of the 2019 ACM Southeast Conference (ACM SE 2019), pp. 72–77. Association for Computing Machinery (2019). https://doi.org/10.1145/3299815.3314437
6. Pandey, P., Mishra, N.: Phish-sight: a new approach for phishing detection using dominant colors on web pages and machine learning. Int. J. Inf. Secur. (2023). https://doi.org/10.1007/s10207-023-00672-4
7. A, T., John, A.: Phishing website detection using LGBM classifier with URL-based lexical features In: 2022 IEEE Silchar Subsection Conference (SILCON), pp. 1–7 (2022). https://doi.org/10.1109/SILCON55242.2022.10028793
8. Korkmaz, M., Sahingoz, O.K., Diri, B.: Detection of phishing websites by using machine learning-based URL analysis. In: 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–7 (2020). https://doi.org/10.1109/ICCCNT49239.2020.9225561
9. Rose, M.A.S.R., Basir, N., Heng, N.F.N.R., Zaizi, N.J.M., Saudi, M.M.: Phishing detection and prevention using chrome extension. In: 2022 10th International Symposium on Digital Forensics and Security (ISDFS), pp. 1–6 (2022). https://doi.org/10.1109/ISDFS55398.2022.9800826
10. Abdelnabi, S., Krombholz, K., Fritz, M.: VisualPhishNet: zero-day phishing website detection by visual similarity. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS 2020), pp. 1681–1698 (2020). https://doi.org/10.1145/3372297.3417233

11. Awasthi, Goel, N.: Generating rules to detect phishing websites using URL features. In: 2021 1st Odisha International Conference on Electrical Power Engineering, Communication and Computing Technology(ODICON), pp. 1–9 (2021). https://doi.org/10.1109/ODICON50556.2021.9429003
12. F.Sadique, Kaul, R., Badsha, S., Sengupta, S.: An automated framework for real-time phishing URL detection. In: 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0335–0341 (2020). https://doi.org/10.1109/CCWC47524.2020.9031269