Phishing URL Detection System

A Report Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Shashwat Khare, 20214142

Sourav Paul, 20214056

Sayon Kar, 20214533

Vaishnav Anish, 20214501

Under the guidance of

Dr. Sarsij Tripathi

To the

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
Motilal Nehru National Institute of Technology Allahabad, Prayagraj,
Uttar Pradesh (India) May 2024

UNDERTAKING

I declare that the work presented in this report titled “Phishing Website Detection System”, submitted to the Computer Science and Engineering Department, Motilal Nehru National Institute of Technology, Allahabad, Prayagraj, Uttar Pradesh (India) for the award of **the Bachelor of Technology** degree in **Computer Science & Engineering**, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

May, 2024

Prayagraj U.P. (India)

Shashwat Khare (20214142)

Sourav Paul (20214056)

Sayon Kar (20214533)

Vaishnav Anish (20214501)

CERTIFICATE

Certified that the work contained in the report titled “Phishing Website Detection System”, by Shashwat Khare (20214142), Sourav Paul (20214056)**,** Sayon Kar (20214533), Vaishnav Anish (20214501), has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. Sarsij Tripathi
Computer Science and Engineering Department.

M.N.N.I.T, Allahabad

Prayagraj, Uttar Pradesh (INDIA)

May 2024

Preface

With the rise of the internet, the web has turned into a principal part of our lives, this has brought a lot of new opportunities for growth as well as a lot of new challenges. One of the most dangerous threats out there is called phishing, which is when someone tries to steal your personal info like your login names, passwords, and even credit card numbers by mimicking a website of a legitimate undertaking.

Phishing usually happens through emails or messages that look like they are from a trusted source, but are actually from a bad actor trying to harm us. These phishing websites can be very convincing and very hard to spot, And as the technology has gotten better so has phishing and it's gotten harder to tell the difference between real and fake websites.

To deal with this growing issue, we need better detection systems that can help us identify these fake websites and stop them before they do too much damage. But with the large amount of phishing websites present in the current internet ecosystem, it's not possible to do it by hand. That's why automated systems like machine learning, natural language processing, and pattern recognition have become super important.

Our project's main objective is to classify these phishing websites and to let the user able to know if the website accessed by the user is phishing or safe using machine learning algorithms to reduce the harm done by these websites.

Acknowledgments

We deeply appreciate all those who contributed to the successful development of our “Phishing Website Detection System” project. Our mentor, Dr. Sarsij Tripathi, provided invaluable support, expert guidance, and constructive feedback that were essential throughout this endeavor. His extensive knowledge and encouragement motivated us to excel and confidently address the challenges of Phishing Website Detection.

We also acknowledge the significant contributions of our project team members and colleagues at Motilal Nehru National Institute of Technology (MNNIT). Their collaboration and expertise were instrumental in refining our methods and achieving our project objectives within the academic context. We extend our gratitude to the academic and technical staff at MNNIT for their provision of resources, facilities, and assistance whenever needed.

The success of this project is a result of the collective efforts and support of all involved, and we genuinely thank each individual who contributed to its achievement.

Contents

Chapter 1

Introduction

1.1 Problem Statement

The aim of this project is to create a system to identify phishing URLs efficiently. Using machine learning algorithms and URL pattern analysis, the system will accurately detect phishing URLs to protect users. The solution will also be integrated into web application and web browser extension to prevent phishing-related risks.

1.2 Motivation

Phishing attacks lead to billions of dollars in financial losses and pervasive identity theft through deception to make users disclose private information. While physically identifying URLs is growing harder to detect as scammers improve their scamming methods, URL detection is achievable and essential to safeguarding the general public and private organizations.

Detection of phishing URLs safeguards against unauthorized entry, defending against data infringement and financial fraud. The project helps eliminate phishing URLs via in-depth analysis and a pattern of URL recognition, addresses actual-world dangers, enhances adherence to cyber protection guidelines, and minimizes reliance on digital services.

1.3 Cyber Crime Attacks

A cyber attack refers to an action designed to target a computer or any element of a computerized information system to change, destroy, or steal data, as well as exploit or harm a network. Cyber attacks have been on the

rise, in sync with the digitization of business that has become more and more popular in recent years.

Phishing - A phishing attack occurs when a malicious actor sends emails that seem to be coming from trusted, legitimate sources in an attempt to grab sensitive information from the target.

Ransomware - In a ransomware attack, the target downloads ransomware, either from a website or from within an email attachment, and holds the user's computer hostage.

Denial of Service (DoS) and Distributed Denial of Service (DDoS) - A denial-of-service (DoS) attack is designed to overwhelm the resources of a system to the point where it is unable to reply to legitimate service requests.

SQL Injection - SQL injection attacks target web applications with poorly secured databases. Attackers insert malicious SQL code into database queries, allowing them to access or manipulate data, or even gain administrative privileges.

Man-in-the-Middle (MitM) Attack - This refers to breaches in cybersecurity that make it possible for an attacker to eavesdrop on the data sent back and forth between two people, networks, or computers.

Credential Stuffing - Credential stuffing involves using lists of leaked usernames and passwords to gain unauthorized access to multiple accounts.

Malware - Malware infects a computer and changes how it functions, destroys data, or spies on the user or network traffic as it passes through.

1.4 Phishing URLs

Phishing URLs are malicious web links designed to deceive users into revealing sensitive information, such as login credentials, financial data, or personal details.

These URLs often try impersonating legitimate websites, using similar domain names or disguised links.

Spoofed Domains: Phishing URLs often use misspelled or similar-looking domain names to impersonate trusted websites.

Obfuscated Links: Phishing URLs may use shortened URLs or complex URL structures to hide their true source destination.

Fraudulent Landing Pages: The websites these URLs lead to may look nearly identical to legitimate ones but are designed to steal information.

1.5 Detection of Phishing URLs

1.5.1 How?

Anti-Phishing Software: Use security tools that automatically scan websites for suspicious links.

Education and Awareness: Teach users to recognize the signs of phishing, such as unexpected emails, urgent requests, or suspicious URLs.

Regular Security Audits: Conduct regular security audits and penetration testing to identify vulnerabilities and improve defenses.

1.5.2 Why?

Prevent Data Theft: Detecting phishing URLs helps prevent unauthorized access to sensitive information, such as usernames, passwords, and credit card numbers.

Avoid Financial Loss: Phishing attacks can lead to identity theft, unauthorized transactions, and significant financial damage.

Protect Personal and Corporate Reputation: Falling victim to phishing can harm personal reputation or damage corporate brands.

Maintain Cybersecurity: Early detection of phishing URLs is essential for maintaining overall cybersecurity, protecting networks, and preventing of malware distribution.

1.6 Machine Learning Algorithms

Machine Learning (ML) algorithms play a pivotal role in detecting phishing URLs, helping to combat the rising threat of cyberattacks that aim to deceive users into providing sensitive information.

ML algorithms for phishing detection are typically trained on large datasets containing both phishing and legitimate URLs. The training process involves learning patterns that distinguish phishing URLs from safe ones.

Common ML Algorithms for Phishing Detection –

Logistic Regression

Decision Trees

Random Forests

Gradient Boosting Machines (GBM)

Support Vector Machines (SVM)

Chapter 2

Related Works

Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning

Yang et al. [9] developed a method called Multidimensional Feature Phishing Detection (MFPD) using deep learning. They extracted character sequence features from URLs for classification, incorporating a dynamic category choice method to speed up detection. By combining URL statistics, webpage code, and text characteristics with deep learning results, they verified detection accuracy. Testing on a dataset with millions of URLs yielded a 98.99% accuracy with a low false positive rate of 0.59%, demonstrating strong performance in accuracy, false positives, and speed.

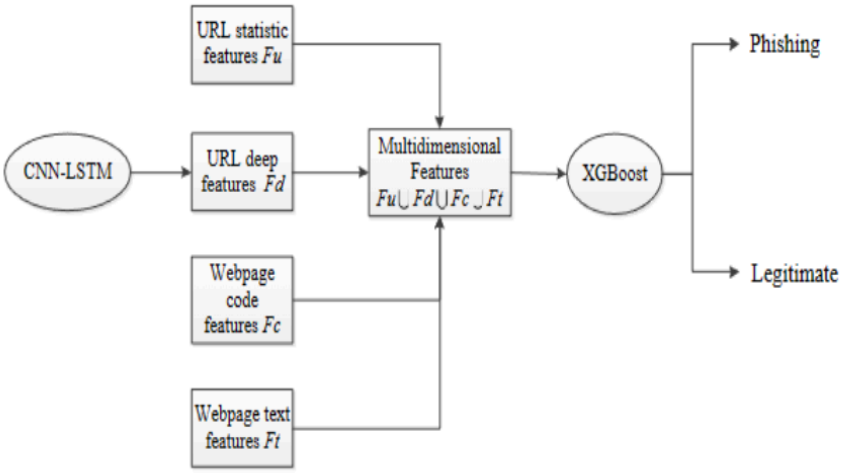
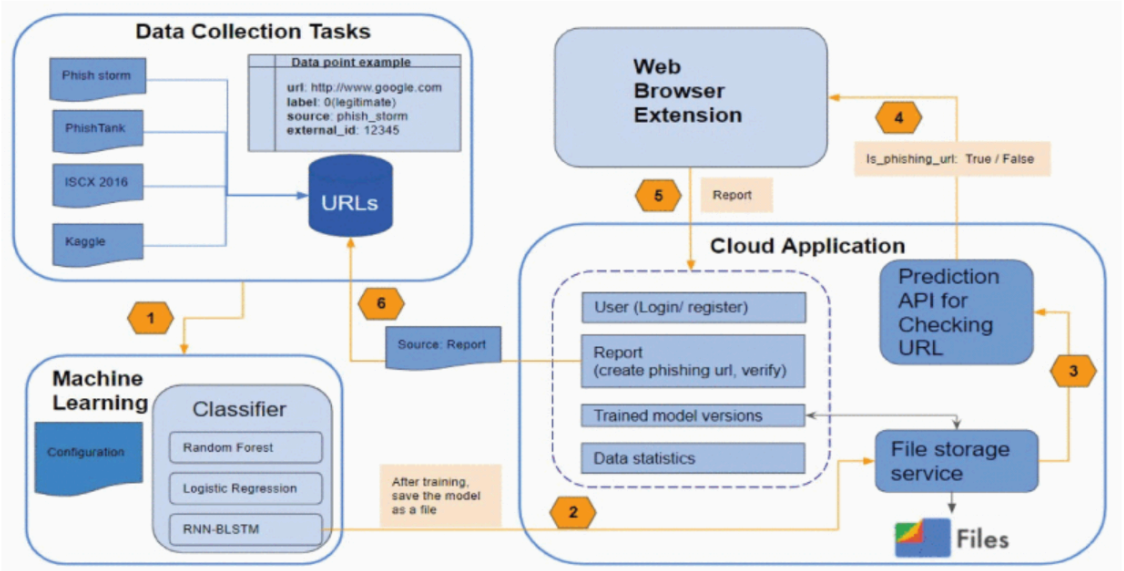


FIGURE 7. Multidimensional feature algorithm.

A Deep Learning-Based Framework for Phishing Website Detection

Tang et al. [10] introduced a deep learning-based framework for phishing website identification, implemented as a Google Chrome extension. This browser plug-in delivers real-time risk assessments and prompts warnings upon phishing detection. Their approach integrates whitelist filtering, blacklist interception, and machine learning (ML) prediction to enhance accuracy, minimize false positives (FP), and accelerate processing time. Utilizing seven datasets sourced from four distinct repositories, they employed Support Vector Machine (SVM), Logistic Regression, Random Forest, and RNN-GRU models. RNN-GRU demonstrated the highest accuracy at 99.18%, establishing it as the most effective model according to their findings.



Architecture of the deep learning-based framework for detecting phishing URLs.

Detecting Phishing Websites Using Machine Learning

Ali et al.[11] introduced a method to enhance phishing website identification accuracy using particle swarm optimization (PSO). By selecting PSO-based website features, they emphasized the importance of each feature in distinguishing phishing from legitimate sites. Applying PSO reduced unnecessary characteristics in training datasets by 7-57%. They employed various learning methods including back-propagation neural network (BPNN), support vector machine (SVM), k-nearest neighbor (KNN), decision tree (C4.5), random forest (RF), and naive Bayes classifier (NB). Compared to standalone models, PSO-based algorithms showed superior accuracy and improvements in true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and false negative rate (FNR) by utilizing a reduced set of website attributes

ML Model	Train Accuracy	Test Accuracy
Decision Tree	0.979	0.937
Random Forest	0.992	0.964
XGBoost	0.996	0.964
MLP	0.825	0.811
KNN	0.931	0.832
Naive Bayes	0.734	0.738
AdaBoost	0.954	0.953
Gradient Boosting	0.993	0.964
SVM	0.600	0.596

Comparison of Results obtained by different models

Chapter 3

Methodology

3.1 Requirement Analysis

3.1.1 Feature Engineering

Feature engineering is essential before training a machine learning (ML) model because it helps transform raw data into a format that models can effectively understand and process. Here's why it's crucial:

Data Preparation: It ensures the data is properly formatted, dealing with missing values, inconsistent data, and various types of data.

Model Performance: By selecting and creating relevant features, we can improve model accuracy and reduce noise and overfitting.

Handling Categorical Data: It converts categorical data into numerical formats that ML models can process, using techniques like one-hot encoding.

Creating New Features: Feature engineering allows for creating new, meaningful features from existing data, often leading to better insights and improved model performance.

Feature Selection and Dimensionality Reduction: It helps us identify and keep the most important features, reducing complexity and enhancing generalization.

Feature Scaling: Ensures that all features are on a similar scale, preventing certain features from disproportionately influencing the model.

Improving Interpretability: It can focus on features that are easier to understand, making it simpler to explain model predictions.

1) get_domain_length(url):

This function takes a single string argument url representing a URL. It imports the urlparse function from the urllib.parse module to parse the URL into its components. It extracts the domain name from the parsed URL using the netloc attribute, which contains the domain and port information. The function returns the calculated length of the domain name.

Feature Engineering:

- The domain_length feature can be used for various data analysis and machine learning tasks:
  - **Feature Selection:** If we suspect that domain length might be related to a certain phenomenon or target variable, we can use the domain_length feature as a potential predictor in our model. For example, extremely long domain names might indicate spam or phishing attempts.
  - **Data Cleaning and Filtering:** We might want to filter out URLs with excessively long or short domain names, which could be erroneous or irrelevant to our analysis.
  - **Data Exploration:** The domain_length feature can help us understand the distribution of domain lengths in our dataset and identify potential patterns or outliers.

Phishing websites sometimes use unusually long or short domain names to bypass detection filters.

This feature can capture such deviations from typical domain name lengths and be used as a numerical predictor in a machine learning model.

2) special_char_ratio_in_url(url):

It calculates the ratio of special characters(excluding letters, digits, and common punctuation symbols like periods, commas, and hyphens) within a URL and stores the result in a new column of our DataFrame.

Feature Engineering:

- Phishing URLs sometimes employ excessive special characters to obfuscate their content, bypass detection filters, or mimic legitimate website elements.
- A high special character ratio, especially when combined with other suspicious features like high digit ratio or presence of obfuscation, can be a potential red flag.

The special character ratio provides a normalized value representing the relative proportion of special characters within a URL, making it less sensitive to variations in URL length.

3) is_domain_ip(url):

This function effectively identifies whether the domain part of a URL is an IP address and stores the result in a new column of our DataFrame. If the conversion is successful (IP address), the function returns 1. If the conversion fails (not an IP address), the function returns 0.

Feature Engineering:

- The is_domain_ip feature can be used for various data analysis and machine learning tasks:
  - **Feature Selection:** If we suspect that URLs with IP addresses as domains might be related to a certain phenomenon or target variable, we can use the is_domain_ip feature as a potential predictor in our model. For example, URLs directly using IP addresses might indicate specific network configurations or internal resources.
  - **Data Cleaning and Filtering:** We can filter out URLs with IP addresses as domains, depending on your analysis goals.
  - **Data Exploration:** The is_domain_ip feature can help us understand the distribution of IP-based domains in our dataset and identify potential patterns or anomalies.

4) tld_length(tld):

It effectively extracts the top-level domain (TLD) from URLs, calculates its length, and stores both the TLD and its length in our DataFrame. This

function takes a string argument tld (top-level domain) and it checks if the tld is not empty (None). If it's not empty, it returns the length of the tld string and if it's empty, it returns -1.

Feature Engineering:

- The 'tld' and 'tld_length' features can be used for various data analysis and machine learning tasks:
  - **Feature Selection:** If we suspect that URLs with specific TLDs or TLD lengths might be related to a certain phenomenon or target variable, we can use these features as potential predictors in our model. For example, a high TLD length might indicate a private or internal domain.
  - **Data Cleaning and Filtering:** We might want to filter out URLs with specific TLDs or TLD lengths, depending on our analysis goals.
  - **Data Exploration:** The 'tld' and 'tld_length' features help us to understand the distribution of TLDs and their lengths in our dataset, potentially revealing patterns or anomalies.

5) url_character_prob(url):

It effectively calculates the average probability of each character appearing in a URL and stores it in a new column of our DataFrame.This feature provides additional information about the character composition of URLs, potentially aiding in further analysis and insights. It Returns the calculated average character probability.

Feature Engineering:

- The 'URLCharProb' feature can be used for various data analysis and machine learning tasks:
  - **Feature Selection:** If we suspect that URLs with a specific average character probability might be related to a certain phenomenon or target variable (e.g., spam or phishing attempts), we can use this feature as a potential predictor in our model. URLs with high average character probability might indicate unusual character distributions.

- **Data Cleaning and Filtering:** We might want to filter out URLs with extremely high or low average character probabilities, as they might be suspicious or generated by bots.
- **Data Exploration:** The 'URLCharProb' feature helps us understand the distribution of average character probabilities in our dataset, potentially revealing patterns or anomalies.

6) number_of_subdomains(url):

The function takes a single string argument url representing a URL.It uses urlparse(url) to parse the URL into its components. It extracts the domain name from the netloc attribute of the parsed URL.It basically counts the number of occurrences of the dot character ('.') in the extracted domain name.Each dot separates a subdomain from the next, so the count of dots indicates the number of subdomains. It returns the calculated number of subdomains as an integer.

If the domain name is empty (no dots), the function returns 0.

Feature Engineering:

- The 'NoOfSubDomain' feature can be used for various data analysis and machine learning tasks:
  - **Feature Selection:** If we suspect that URLs with a specific number of subdomains might be related to a certain phenomenon or target variable (e.g., website complexity, organization structure), we can use this feature as a potential predictor in our model.
  - **Data Cleaning and Filtering:** We might want to filter out URLs with unusually high or low numbers of subdomains, as they might be suspicious or generated by bots.
  - **Data Exploration:** The 'NoOfSubDomain' feature can help us understand the distribution of subdomain counts in our dataset, potentially revealing patterns or anomalies.

- Phishing websites often employ multiple subdomains to appear more legitimate.

- A higher number of subdomains compared to typical websites could be a potential indicator of phishing.

In machine learning, this feature can be used as a numerical predictor within a model to help distinguish phishing URLs from legitimate ones.

7) obfuscation_ratio(url):

It calculates the ratio of characters within a URL that belong to common obfuscation patterns and stores the result in a new column of our DataFrame. It returns the calculated obfuscation_ratio, which is calculated by dividing the total number of obfuscated characters (num_obfuscated_chars) by the total number of characters in the URL (total_chars).

Feature Engineering:

- The 'ObfuscationRatio' feature can be used for various data analysis and machine learning tasks:
  - **Feature Selection:** If we suspect that the proportion of obfuscated characters might be related to a certain phenomenon or target variable (e.g., the likelihood of phishing attempts), we can use the 'ObfuscationRatio' feature as a potential predictor in your model.
  - **Data Cleaning and Filtering:** We might want to filter out URLs with a high obfuscation ratio, as they are more likely to be malicious or suspicious.
  - **Data Exploration:** The 'ObfuscationRatio' feature helps us understand the distribution of obfuscation ratios within our dataset, potentially revealing patterns related to the prevalence and severity of obfuscation techniques.

A high obfuscation ratio can be a significant red flag for phishing attempts.

This feature can be used as a numerical predictor in a machine learning model.

8) letter_ratio_in_url(url):

It effectively calculates the letter ratio in a URL and it is calculated as If total_chars is greater than 0, then the letter_ratio is equal to the the number of letters (num_letters) divided by the total number of characters (total_chars).

Feature Engineering:

- Phishing URLs often contain excessive text or word manipulation to appear legitimate. This can lead to a higher proportion of letters compared to genuine URLs.
- A high letter ratio might indicate an attempt to mimic legitimate website names or keywords through excessive text usage.
- It provides a normalized value representing the relative proportion of letters within a URL, making it less sensitive to variations in URL length.

9) fd_length(url):

It calculates the length of the first directory level within the path component of a URL and stores the result in a new column of our DataFrame.It splits the path by forward slashes (/) using the split('/') method. It attempts to access the second element (index 1) of the split list, which should represent the first directory level after the leading slash.If successful, the function returns the length of the first directory level string.

Feature Engineering:

- Phishing URLs sometimes employ unusually long or short first directory levels to obfuscate their structure or appear more legitimate.

A significant deviation from typical directory level lengths could be a potential red flag, especially in combination with other suspicious features.

10) tld_legimate_prob(url):

It calculates a score representing the legitimacy of a URL's Top-Level Domain (TLD) and stores the result in a new column of our DataFrame.If the extracted TLD is present in the legitimate_tlds list, the function returns 1.0, indicating a high probability of the TLD being legitimate. If the TLD is not

found in the list, the function returns 0.0, suggesting a lower probability of legitimacy.

Feature Engineering:

- Phishing URLs often employ uncommon or newly registered TLDs to bypass detection filters and appear more legitimate.
- A TLD legitimacy score can provide valuable information about the typicality of the TLD used in the URL.

11) having_ip_address(url):

The function employs regular expressions to search for IP addresses in various formats including IPv4 and IPv6.

Feature Engineering:

- Phishing URLs often use IP addresses instead of domain names to avoid blacklists or to appear more legitimate. Detecting IP addresses in URL helps us to identify anomalous patterns that may indicate phishing attempts.
- The presence of an IP address in a URL may not always indicate malicious intent, but when combined with other features and contextual information, it can contribute to a more comprehensive analysis of the URL's legitimacy.

12) abnormal_url(url):

It checks if a URL contains its own hostname within the path and stores the result in a new column of our DataFrame.It uses the re.search(hostname, url) function to search for the extracted hostname as a substring within the entire URL. If the regular expression search finds a match (i.e., the hostname is present within the URL path), the function returns 1, indicating a potential abnormality. If no match is found, the function returns 0.

Feature Engineering:

- Legitimate URLs typically have a clear separation between the hostname and the path.

- Phishing URLs sometimes include the hostname within the path to obfuscate their true destination or bypass security measures.
- The presence of the hostname within the path might indicate an attempt to:
  - Mimic a legitimate website by incorporating its domain name into the path.
  - Disguise the actual destination of the URL.
  - Bypass certain security filters that might block URLs based solely on the domain name.

13) no_of_dir(url):

It calculates the number of directory levels (indicated by forward slashes) within the path component of a URL and stores the result in a new column of our DataFrame. It returns the total number of directory levels found in the URL.

Feature Engineering:

- Phishing URLs sometimes employ excessively long or complex paths with multiple directory levels to obfuscate their true destination or bypass certain security filters.
- An unusually high number of directory levels compared to typical website URLs could be a potential red flag, especially in combination with other suspicious features.

14) no_of_embed(url):

It calculates the number of occurrences of the substring "//" within the path component of a URL and stores the result in a new column of our DataFrame. It uses the urldir.count('//') method to count the number of occurrences of the substring "//" within the extracted path. It returns the total number of occurrences of "//" found in the URL path.

Feature Engineering:

- Phishing URLs sometimes employ embedded domains within their paths to obfuscate their true destination or bypass security measures.

- The presence of "//" often indicates a subdomain within the path, and an unusually high number of occurrences could be a potential red flag.

15) suspicious_words(url):

It checks for the presence of certain suspicious words within a URL and stores a binary indicator (1 or 0) in a new column of our DataFrame.It uses the re.search function with a regular expression pattern to search for the presence of specific words within the URL string.

Feature Engineering:

- Phishing URLs often contain words that try to manipulate users into clicking or entering sensitive information.
- The presence of these suspicious words can be a red flag and a valuable feature in machine learning models for phishing detection.
- The chosen list of suspicious words might not be exhaustive and may not capture all phishing tactics.
- Legitimate websites might also use some of these words in their URLs for various reasons.

16) google_index(url):

It utilizes the googlesearch library to search for the given URL using Google Search.

It retrieves the top 5 results returned by Google Search. If the URL appears in any of the top 5 search results, the function returns 1, indicating that the URL is likely indexed by Google.

Otherwise, it returns 0.

Feature Engineering:

- Phishing URLs often target specific websites or services.
- Legitimate websites are typically indexed by Google and appear in search results.
- A URL that is not indexed by Google might raise suspicion as it could be a recently created phishing site.

- Using this feature in conjunction with other relevant features enhances the effectiveness of our machine learning model for phishing detection.

3.2 Machine Learning Model

How Random Forest Classifier Works?

Ensemble of Decision Trees: A random forest consists of a collection ("forest") of individual decision trees. Each tree is constructed using a subset of the training data and different subsets of features. This process, called bootstrapping or bagging introduces randomness and helps us to diversify the ensemble.

Training: To build the forest, the algorithm creates multiple decision trees by selecting random samples from the dataset and choosing a subset of features at each split. This randomness helps ensure that each tree is different, reducing the risk of overfitting.

Classification: In the classification phase, each tree in the forest predicts an outcome, and the final prediction is typically determined by a majority vote among the trees. This aggregation helps improve accuracy and robustness.

Benefits: Random forests offer several advantages:

Robustness: By using multiple trees, the classifier is less prone to errors caused by outliers or overfitting.

Versatility: It can handle both numerical and categorical data.

Feature Importance: The algorithm can rank features based on their importance, aiding in feature selection and interpretation.

Why Random Forest is Important for Phishing URL Detection?

Handling Complexity: Phishing URL detection involves complex patterns and features like URL structure, domain name, presence of special

characters, length, etc. Random forests can effectively manage this complexity due to their ensemble nature and flexibility with various types of data.

Resilience to Noise: Phishing URLs can be diverse and sometimes mimic legitimate patterns to evade detection. The randomization in random forests helps make the classifier resilient to noise and variation in the data.

Accuracy: Since random forests use a majority voting system across multiple decision trees, the resulting classifier tends to have high accuracy. This accuracy is crucial in phishing URL detection, where a wrong prediction can lead to security risks.

Feature Importance and Interpretability: Phishing detection systems often need to explain why a URL is considered suspicious. Random forests can provide insights into which features contribute most to the prediction, aiding in interpretability and understanding of phishing tactics.

Scalability: Given the vast number of URLs in use on the internet, scalability is important. Random forests can efficiently handle large datasets, making them suitable for real-time phishing detection systems.

3.3 Database Setup

We have a database where we store URLs along with their associated results. The process of adding a new URL and obtaining its result involves several steps:

Checking the Database: When we receive a new URL, we first check our database to see if it already exists. If it does, we retrieve the existing result and display it to the user.

Processing with the Model: If the URL isn't in the database, we use our model to analyze it and produce a result. This model could be any kind of computational or machine learning process that interprets the URL to provide the desired information.

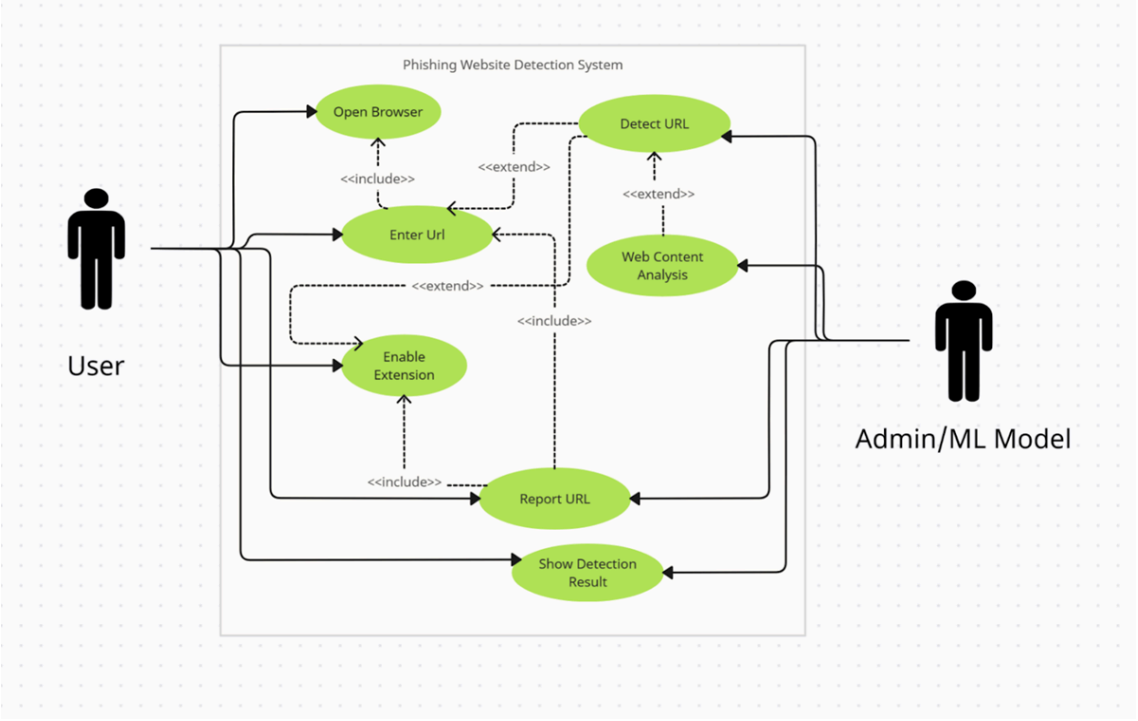
Updating the Database: After the model generates a result for the new URL, we add both the URL and its result to the database. This step ensures that

future queries for the same URL will return the existing result without needing to run the model again.

Retraining the Model: Over time, as we add more URLs and results to the database, the model might require retraining. This step involves updating the model with the latest data from the database to ensure it remains accurate and relevant.

By following this approach, we aim to improve efficiency by reducing redundant processing, and also ensure that our model stays current with the latest information from our growing database.

3.4 Use Case Diagram



Chapter 4

Experimental Setup

4.1 Protocol Stack

4.1.1 Data Collection

This protocol outlines the process of gathering datasets containing URLs labeled as either phishing (malicious) or benign (safe).

Data Sources: We have gathered datasets containing URLs labeled as phishing or benign. These datasets have taken from open-source repositories, security organizations, or data collection websites.

Open-Source Repository: Open-source repositories like GitHub often provide datasets contributed by researchers, security professionals, or enthusiasts. The PhishDataset repository likely contains URLs labeled as phishing or benign, along with additional information or metadata.
https://github.com/ESDAUNG/PhishDataset?tab=readme-ov-file

Security Organizations: Security organizations like PhishTank continuously monitor and track phishing activities on the internet. They maintain databases of reported phishing URLs along with relevant details such as the type of phishing (e.g., email, website), timestamps, and user reports.
https://phishtank.org/

Data Collection Websites: Kaggle and the UCI Machine Learning Repository serve as centralized platforms for hosting diverse datasets across different domains, including cybersecurity. These platforms offer datasets contributed by researchers, organizations, and individuals worldwide.
https://www.kaggle.com/
https://archive.ics.uci.edu/

Data Extraction Protocol: This protocol focuses on extracting URLs from different sources, such as email archives, web scraping, or public datasets.

4.1.2 Sources of Data Extraction:

Email Archives: Email archives contain a wealth of information, including URLs embedded within email bodies or attachments. Phishing emails often contain malicious links disguised as legitimate ones. By extracting URLs from email archives, we can identify and analyze potential phishing attempts, gather insights into attackers' tactics, and develop strategies to mitigate risks.

Web Scraping: Web scraping basically involves extracting data from websites. In the context of phishing, we can scrape URLs from various sources such as phishing websites, forums, social media platforms, or online marketplaces where phishing links may be shared or advertised. Web scraping tools can automate the process of extracting URLs from web pages, allowing us to collect a large volume of data efficiently.

Public Datasets: Public datasets available on platforms like Kaggle, GitHub, or academic repositories already contain URLs relevant to phishing. These datasets include labeled URLs indicating whether they are benign or malicious. By extracting URLs from such datasets, we can build upon existing work, compare different datasets, or combine multiple datasets to create comprehensive datasets for analysis.

4.1.3 Data Preprocessing

Data preprocessing refers to the preparatory steps taken to clean, transform, and organize raw data into a format suitable for analysis or modeling.

Feature Engineering: It basically means extracting meaningful features from URLs. Feature engineering involves selecting or creating relevant attributes (features) from raw data that best represent the underlying patterns or characteristics of the dataset. In the context of phishing URLs, we extracted various components such as:

1. **Domain-related features**: Extract domain, subdomain, TLD, and domain length.
2. **Path-related features**: Analyze path structure and length.
3. **Character-based features**: Count occurrences of specific characters or sequences.
4. **Semantic features**: Analyze URL semantics, including keywords indicating phishing behavior.

Data Cleaning: Data cleaning is essential to ensure the quality and consistency of the dataset. This process typically involves removing duplicates, normalizing data, and handling missing values which are essential steps to ensure that the dataset is consistent, reliable, and suitable for analysis. Clean data reduces noise and improves the effectiveness of our machine learning algorithms.

Labeling: Accurate labeling is critical for supervised learning tasks, where models learn from labeled examples. Whether done manually or automatically, labeling ensures that each URL is correctly classified as phishing or legitimate, forming the basis for our training and evaluating machine learning models.

4.1.4 Machine Learning Model Development

Machine Learning Model Development focuses on creating and evaluating a machine learning model for detecting phishing websites.

Feature Selection: Feature selection involves choosing the most relevant attributes from the dataset to train the machine learning model effectively.

Opting for the most appropriate features to train the model -

'url_length','domain_length', 'is_domain_ip', 'tld_length',
'char_continuation_rate', 'URLCharProb', 'NoOfSubDomain',
'HasObfuscation', 'NoOfObfuscatedChar', 'ObfuscationRatio',
'NoOfLettersInURL', 'LetterRatioInURL', 'NoOfDigitsInURL',
'DigitRatioInURL', 'NoOfAmpersandInURL',
'NoOfOtherSpecialCharsInURL', 'SpecialCharRatioInURL', 'IsHTTPS',

'TLDLegitimateProp', 'use_of_ip','abnormal_url', 'count%', 'count?', 'count-', 'count=', 'count-www','count@', 'count_dir', 'count_embed_domain', 'count-https', 'count.','count-http', 'short_url', 'hostname_length', 'sus_url', 'count-digits','count-letters', 'google_index', 'fd_length'

4.1.4.1 Model Selection

Random Forest Classifier :- We have chosen Random Forest Classifier as the machine learning algorithm for this task as we wanted our model to handle large datasets and have Built-In Cross-Validation with resistance to overfitting. Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees.

4.1.4.2 Model Training

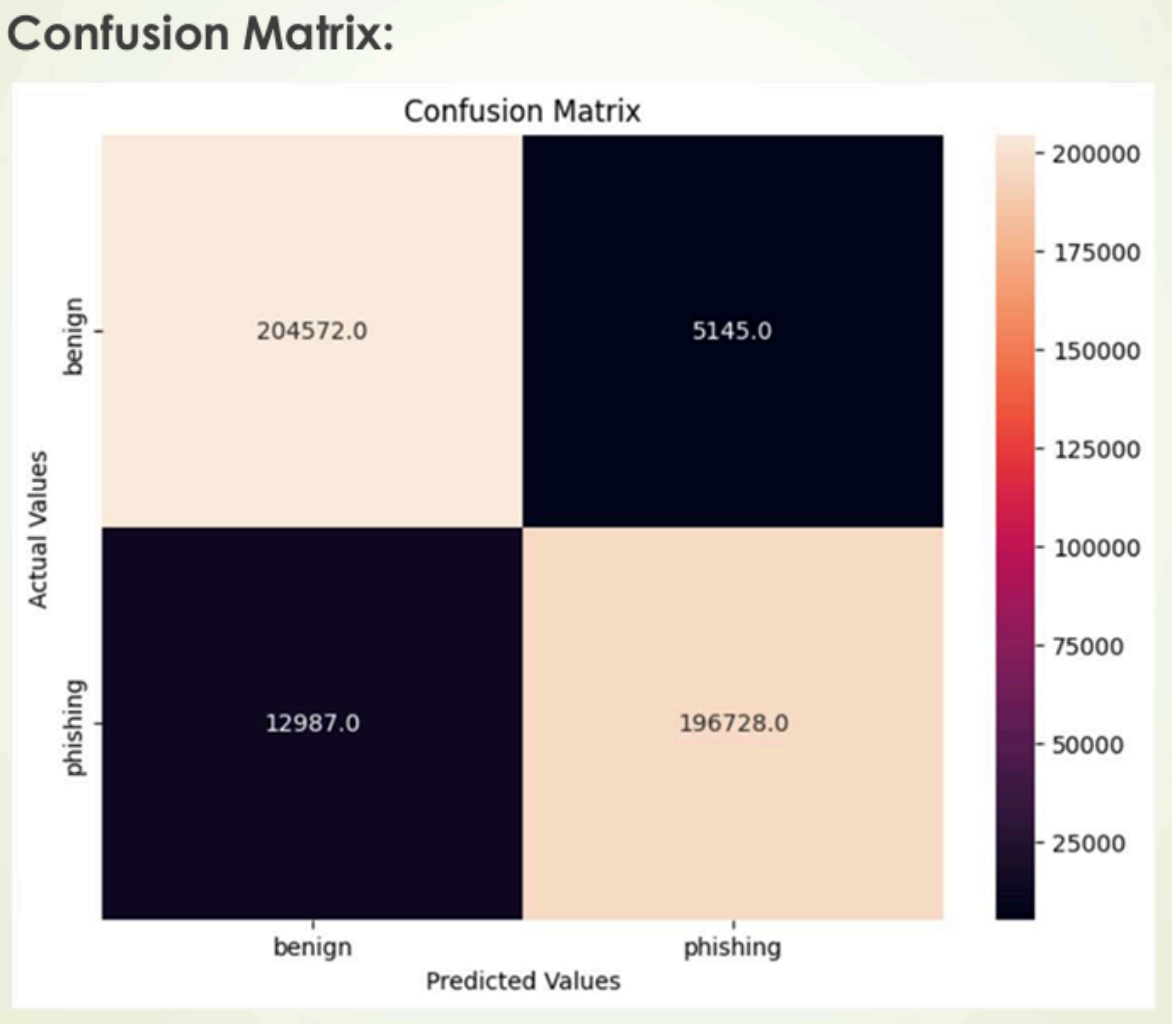
Splitting the dataset to training and testing data: The dataset is split into training and testing sets using the train_test_split function from scikit-learn. This step is crucial to evaluate the model's performance on unseen data. The parameter test_size=0.2 specifies that 20% of the data will be reserved for testing, while the remaining 80% will be used for training. By using the parameter stratify=Y, it ensures that the distribution of classes remains consistent across both the training and testing sets.

Model training (on training data): The Random Forest Classifier model is initialized with parameters such as the number of estimators (decision trees) and the maximum number of features to consider for each split. Afterwards, the model undergoes training on the provided training data using the fit method.

Prediction (on testing data): Subsequently, upon completion of training, the model applies the predict method to forecast the labels for the testing data.

4.1.4.3 Model Evaluation

We have computed the accuracy of the model to assess its execution on both the training and testing datasets. The precision score speaks to the extent of accurately classified occurrences out of the add up to number of occurrences. In this case, the accuracy on the training information is roughly **99.16%**, whereas the accuracy on the testing information is roughly **95.68%**.



Confusion Matrix for Our Model

4.1.5 Model Deployment

Model Deployment involves making the trained machine learning model accessible for real-time phishing detection.

Model Export: After training the model, we have exported the model to a serialized file format suitable for deployment. In this example, the trained Random Forest Classifier model is serialized using the Python pickle module and saved as a file named 'trained_model.sav'.

```
import pickle
filename = 'trained_model.sav'
pickle.dump(rf, open(filename, 'wb'))
```

API Development: We have created an API to expose the trained model, allowing it to be accessed and used for real-time phishing detection.We have utilized FastAPI, a Python web framework, for API development and we have used ngrok for tunneling the local server to a public URL, and we have also employed uvicorn as the ASGI server.

Integration with Other Systems: We have integrated the api exposing model with various systems for seamless utilization:

- **MERN web application:** We have integrated the API with a web application developed using the MERN stack, enabling users to perform phishing detection through the web interface.
- **Google Chrome extension:** We have integrated the API into a Google Chrome extension, allowing users to conveniently check the legitimacy of URLs directly from their browser.

4.1.6 Monitoring and Maintenance

Monitoring and Maintenance involves ensuring the ongoing effectiveness and reliability of the phishing detection system. By proactively monitoring performance, addressing issues, adapting to emerging threats, and continuously improving based on feedback, the system can effectively safeguard users against phishing attacks.

Model Retraining: Regular updates of the database with new URLs and their predicted results by the model are conducted by us. This updated dataset is then used to retrain the model periodically. By doing so, the model stays current with emerging trends in phishing attacks, ensuring its effectiveness in detecting new threats.

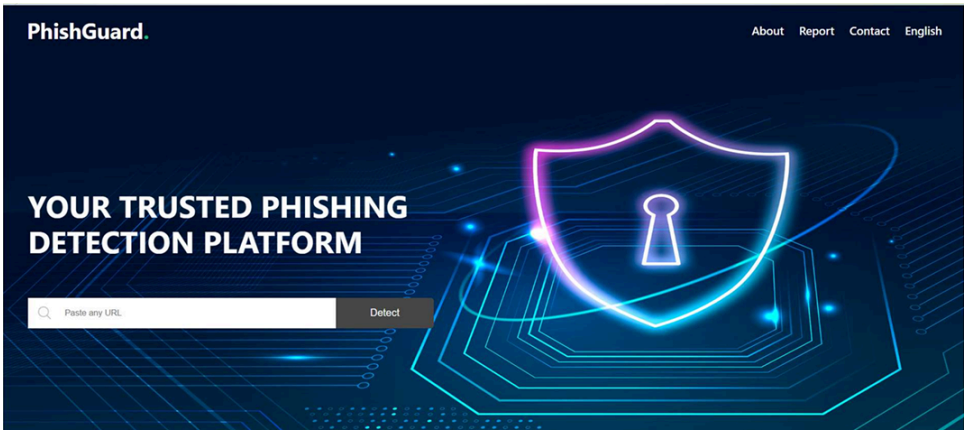
Feedback Loops: We have integrated a reporting feature into the website to gather data on false positives and false negatives. Users can report instances where the model incorrectly identifies a legitimate URL as phishing (false positive) or fails to detect a phishing URL (false negative). This feedback loop allows for continuous improvement of the model by adjusting its parameters or incorporating new training data based on the reported issues.

4.1.7 User Interface

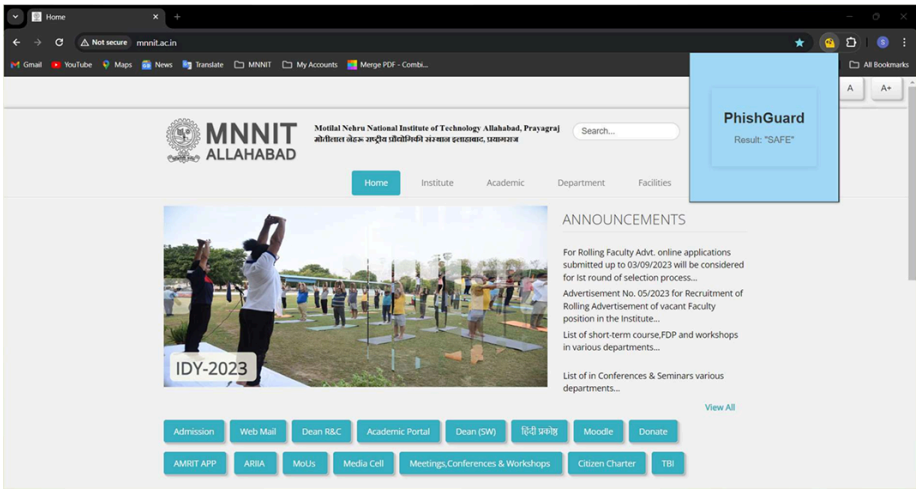
User Interface focuses on providing intuitive interfaces for users to interact with the phishing detection system.

Web-based Dashboard: We have developed a user-friendly dashboard to visualize the results of the phishing detection process. Users can access this dashboard through a web browser. It provides an interactive interface where users can view flagged URLs identified as potential phishing threats, as well as URLs classified as safe.

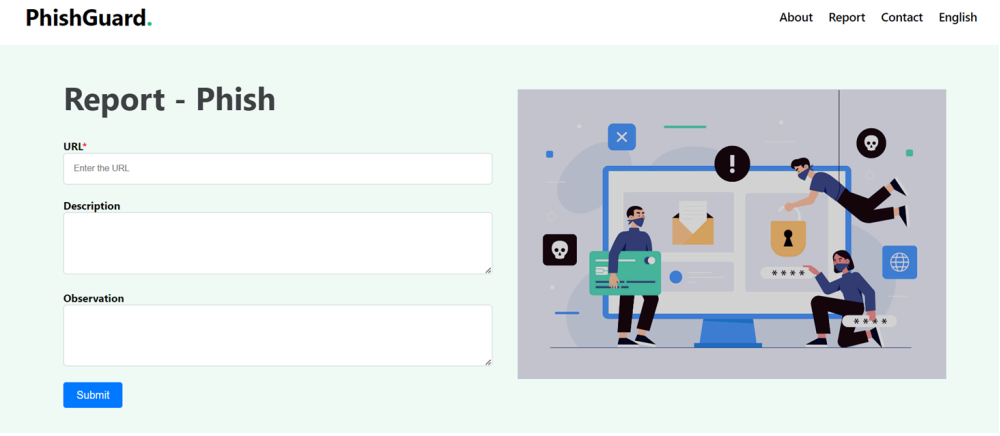
Extension: We have created an on-click extension for web browsers to enable users to quickly check the legitimacy of URLs directly from the browser's search bar. When a user enters a URL in the search bar and clicks on the extension icon, the extension sends the URL to the phishing detection system for analysis. The system then provides feedback on whether the URL is classified as phishing or safe, which is displayed to the user. This extension enhances user convenience and promotes safe browsing practices by providing real-time feedback on website safety.



Hosted Website - https://phish-guard.vercel.app/



Google Chrome Extension



Report URLs - https://phish-guard.vercel.app/

Chapter 5

Conclusion

5.1 Limitations

Here are some common limitations of phishing URL detection systems:

False Positives: Phishing URL detection systems might flag legitimate URLs as malicious. This can disrupt normal business operations, affect user experience, and erode trust in the system.

False Negatives: Despite advanced algorithms, these systems may fail to detect some phishing URLs, allowing threats to slip through. Phishers often find ways to evade detection, such as using URL shortening services or domain spoofing techniques.

Evolving Tactics: Phishing tactics constantly evolve. Attackers adapt their methods to avoid detection, making it challenging for static detection rules or models to keep up.

Limited Context: URL detection systems often focus on the URL and its features. This approach might miss phishing attempts that rely heavily on context, such as email content, sender reputation, or other social engineering tactics.

Resource Intensive: Advanced machine learning and deep learning models require substantial computational resources, especially when analyzing large datasets or real-time traffic. This can lead to higher operational costs.

Data Privacy and Compliance: Analyzing URLs and related data can raise privacy concerns, particularly when it involves user data or sensitive information. Compliance with regulations like GDPR can be a challenge.

Reliance on Feature Engineering: Effective detection often relies on feature engineering to extract meaningful patterns from URLs. Incorrect or incomplete feature engineering can lead to reduced model performance.

Maintenance and Updates: Keeping detection systems updated with the latest threat intelligence and adapting to new phishing techniques requires ongoing maintenance and expert knowledge. This can be resource-intensive and may require specialized skills.

Limited Detection Scope: Some phishing URLs may use sophisticated techniques like multi-stage attacks or hybrid approaches (combining social engineering and technical exploits) that are harder to detect with a URL-based system alone.

Integration Challenges: Integrating phishing URL detection systems with existing security infrastructure, such as email filters, firewalls, or endpoint security, can be complex and might require additional customization.

User Education: Even with robust detection systems, phishing attacks often rely on human interaction. Educating users about phishing risks and best practices is crucial, as systems can't replace vigilance and awareness.

5.2 Future Scope

Enhancing Feature Engineering

Content-based features: Content-based features are derived from the data itself, in this case, from the webpage or URL. Examples of such features include:

Keyword Analysis: Identifying significant words or phrases in the URL or webpage content that might be indicative of phishing or legitimacy.

Domain Analysis: Analyzing the domain structure, including subdomains, length, and common phishing patterns.

Lexical Features: Looking at the character composition of the URL, such as the use of special characters, numbers, and their frequency.

HTML Structure: Investigating the structure of the webpage, including tags, scripts, and embedded elements.

Behavioral features: Behavioral features focus on how users or systems interact with the URL or domain. These could include:

Traffic Patterns: Observing the amount of traffic a URL or domain receives over time.

User Interaction Metrics: Studying how users interact with the webpage, such as click-through rates or time spent on the site.

Referral Data: Examining where the traffic to a domain is coming from, including suspicious redirects.

Experiment with feature selection techniques to identify the most important features and reduce noise.

Advanced Machine Learning Techniques

Considering more complex machine learning algorithms or ensemble techniques to improve performance.

Deep learning approaches like CNNs (Convolutional Neural Networks) or RNNs (Recurrent Neural Networks) to capture complex patterns in URLs.

Ensemble learning, such as Random Forests or Gradient Boosting, to combine the strengths of multiple models.

Data Augmentation and Expansion

Collect More Data: Add more phishing and legitimate URLs from public datasets, internal logs, or external sources. This helps reduce overfitting and increases model generalization.

External Datasets: Collaborate with security experts or use external datasets to add diversity and ensure the dataset reflects a wide range of phishing patterns.

Synthetic Data Generation: Create new URLs by modifying existing ones, like changing domains, paths, or parameters.

Random Modifications: Randomly alter URLs, such as adding noise, shuffling components, or inserting extra segments, to create new training examples.

Data Transformation: Apply transformations to existing data, such as encoding or reversing URL components, to add variety to the training set.

Access to Resources: Looking into comprehensive databases, enriching the training dataset with more diverse examples.

Benefits

Prevents Overfitting: More diverse training data reduces overfitting, enhancing the model's ability to generalize.

Enhances Robustness: A larger dataset with varied examples makes the model more robust against changing phishing tactics.

Chapter 6

References

[1] M. M. Gandomi, "Domain length,"
[Online]. Available:
https://www.nature.com/articles/s41598-022-10841-5
[Accessed: May 2022].

[2] J. H. Kim, "Subdomain count,"
[Online]. Available:
https://www.researchgate.net/publication/355896081_Phishing_Detection_Methods_A_Review
[Accessed: 11 November 2021].

[3] E. F. Williams, "Number of ampersands,"
[Online]. Available:
https://www.researchgate.net/publication/358142755_Phishing_Attacks_Detection_--_A_Machine_Learning-Based_Approach
[Accessed: January 2022].

[4] N. O. Perez,
"Presence of "https"," [Online]. Available:
https://www.researchgate.net/publication/355896081_Phishing_Detection_Methods_A_Review
[Accessed: November 2021].

[5] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang,
"An Empirical Analysis of Phishing Blacklists,"
[Proc. 6th Conf. Email Anti-Spam (CEAS'09), Jul. 2009, pp. 59-78].

[6] A. K. Jain and B. B. Gupta,
"A novel approach to protect against phishing attacks at client side using auto-updated white-list,"

[Eurasip J. Inf. Secur., vol. 2016, no. 1, May. 2016].

[7] M. Zouina and B. Outtaj,
"A novel lightweight URL phishing detection system using SVM and similarity index,"
[Human-Centric Comput. Inf. Sci., vol. 7, no. 1, p. 17, Jun. 2017].

[8] Wisdom ML, (2023),
"Malicious URL Detection Using Machine Learning in Python,"
[Online]. Available:
https://wisdomml.in/malicious-url-detection-using-machine-learning-in-python/

[9] Yang, Peng & Zhao, Guangzhen & Zeng, Peng, (2019),
"Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning," [Online]. Available:
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8610190
[IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2892066].

[10] L. Tang and Q. H. Mahmoud,
"A Deep Learning-Based Framework for Phishing Website Detection,"
[Online]. Available:
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9661323
[IEEE Access, vol. 10, pp. 1509-1521, 2022,
doi:10.1109/ACCESS.2021.3137636].

[11] S. Alrefaai, G. Özdemir and A. Mohamed,
"Detecting Phishing Websites Using Machine Learning,"
[Online]. Available:
https://openaccess.iku.edu.tr/items/0f21ddac-ebf3-423c-988b-f15a555ab2ec
[2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1-6, doi: 10.1109/HORA55278.2022.9799917].