MALICIOUS URL DETECTION USING MACHINE LEARNING

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

G.Manoj Kumar-AP21110010147

Subodh Amru Kiliveti - AP21110010141

A Vamsi Kumar - AP21110010135

K Sravani - AP2111001011160

Under the Guidance of

Dr. Elakkiya E

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

Certificate

Date: 23-Nov-23

This is to certify that the work present in this Project entitled "Malicious Link Detection" has been carried out by **G.Manoj Kumar, Subodh Amru Kiliveti, A Vamsi Kumar, K Sravani** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in the **School of Engineering and Sciences**.

Supervisor

Dr. Elakkiya E

Assistant Professor,

SRM University - Andhra Pradesh

Acknowledgments

I would like to thank my teacher Elakkiya E who gave us the opportunity to work on this project. I got to learn a lot from this project about various machine learning modules. We are very grateful to her for her support and guidance in completing this project. Finally, I want to thank all my teammates as well for being part of the team and for helping me complete this project.

Table of Contents

Abstract

This project focuses on developing a robust malicious link detection system using ensemble learning techniques. Employing a diverse set of classifiers such as Random Forest, XGBoost, Gradient Boosting, and AdaBoost, the system enhances accuracy and generalization. The models are evaluated through cross-validation and ranked based on their performance. A weighted ensemble is then constructed using the top-performing classifiers, considering their ranks and normalized weights. The resulting ensemble model demonstrates improved predictive capabilities, offering an effective solution for detecting malicious links. The voting classifier ensures a balanced combination of individual models, contributing to a more resilient and accurate malicious link detection system.

1. Introduction

The increasing frequency of cyber risks poses a significant danger in an era dominated by online activities such as social networking, e-commerce, banking, and business. The emergence of online crimes, particularly through the manipulation of Universal Resource Locators (URLs), has heightened the need for strong internet security measures. Clicking on malicious URLs may damage systems and expose sensitive personal information, highlighting the vital need for cyber defenses. The phrase "malicious" refers to a wide range of attacks, including but not limited to benign, spam, malware, and phishing.

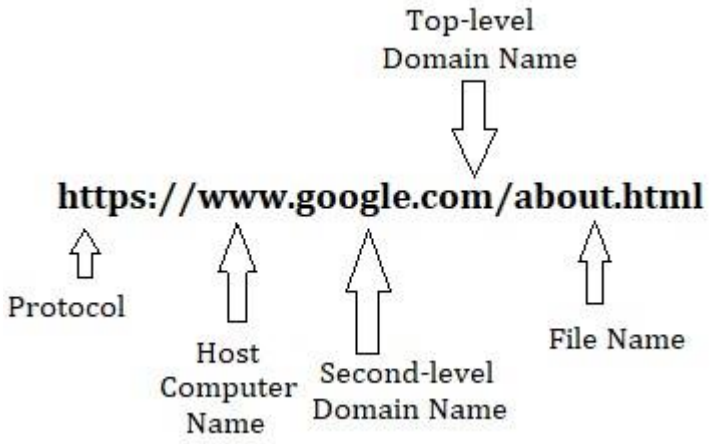
Blacklisting services have evolved as critical tools within the internet security industry in response to the increasing cyber threat scenario. These services maintain databases known as Blacklists, which contain URLs that have been identified as dangerous. However, the efficiency of URL blacklisting is challenged since attackers can leverage system weaknesses by modifying URL components to avoid detection. This vulnerability is caused by factors like obsolete entries, errors, or a lack of timely assessments, which allows a large number of harmful websites to avoid blacklisting.

The use of machine learning (ML) approaches for malicious URL identification has gained significance in order to improve the efficiency of online security measures. The automated model update capabilities and identification of freshly created URLs demonstrate ML's versatility in dealing with the changing nature of cyber threats. Deep learning models have recently been studied, with approaches used to automatically recognise and extract information from new URLs. URL length, letter count, digit count, special characters count, secure HTTP, existence of an IP address, root domain, anomalous URL, and top-level domain type are extracted properties that ML systems use to classify URLs as benign or dangerous.

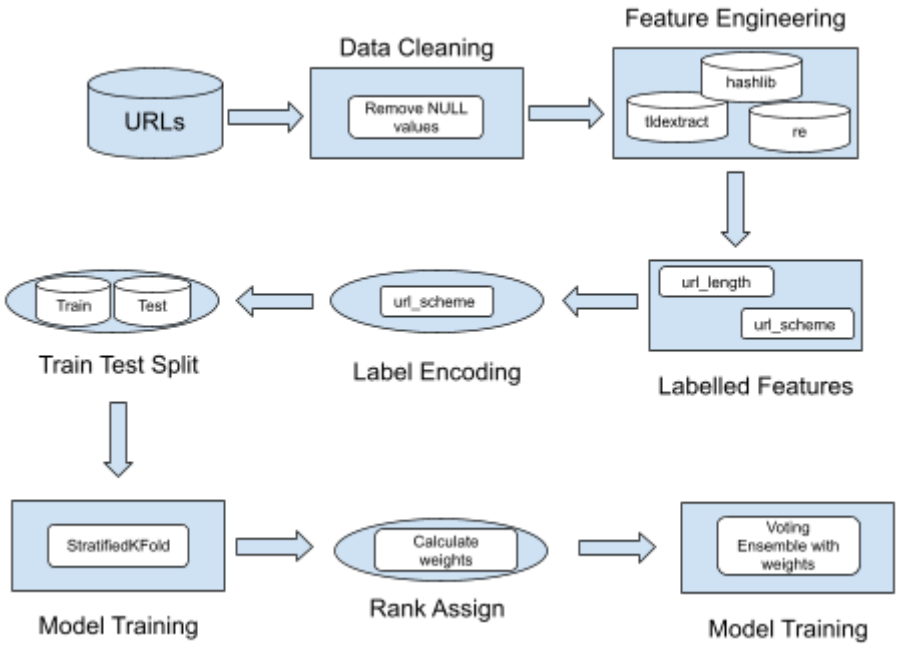
Among the strong models used for this purpose are Random Forest, Logistic Regression, XGBoost, SVM with a Voting ensemble, and Decision Tree Classifier. This report offers an in-depth examination of the processing framework designed for

Malicious URL Detection Using Machine Learning, shedding light on the importance of each model and the extracted features in fortifying cyber defenses against the multifaceted threat landscape of malicious URLs.

1.1 Parts of the Link



Example of a URL : Uniform Resource Locator



Processing framework for Malicious URL Detection using Machine Learning

2. PROPOSED WORK :

The terms "Benign," "Malware," "Phishing," and "Defacement" are frequently linked to different aspects of cybersecurity:

2.1 Benign

This describes something as safe, non-harmful, or not posing a risk. When it comes to URLs and cybersecurity, a benign URL is one that is regarded as safe and doesn't put users at risk.

2.2 Malware

Malware is short for "malicious software," which is any software that is specifically made to harm, obtain unauthorized access to, or interfere with a computer system, network, or other device. Ransomware, worms, trojan horses, spyware, and other destructive programs that can breach security and pilfer private data are all considered malware.

2.3 Phishing

Phishing is a type of cyberattack in which attackers trick victims into disclosing sensitive information like passwords, credit card numbers, or personal information by using deceptive techniques like sending phony emails, messages, or websites. Phishing attacks frequently use the persona of reliable organizations to trick users into doing things that will help the attackers.

2.4 Defacement

Defacement refers to when hackers make unapproved changes to a website's visual design or content. This type of attack involves hackers changing a website's look to

show their own text, pictures, or other content. They usually do this to make the owner of the website look bad or to send a message.

3. Features Used

3.1 URL Length

Phishers use the URL's length to hide the name of the website. Our investigation shows that 74 characters is the average length for a URL.

3. 2 URL Scheme

In order to identify and categorize potentially dangerous or fraudulent web addresses, URL schemes analyze the protocol component (such as HTTP, HTTPS) of URLs as part of a model's features. This process aids in the automated identification of threats on the internet**.**

3.3 URL Path Length

The term "URL path length" describes how many characters or segments make up a URL's path. It gauges how intricate or deep a website or web application's directory structure is.

3.4 URL Host Length

The quantity of characters in the hostname or domain name portion of a URL is referred to as the URL host length. It gauges how long a domain name appears in text form within a web address.

3.5 URL Host is IP

The phrase "URL Host is IP" describes circumstances in which an IP address—as opposed to a domain name—is contained in the host portion of a URL (Uniform Resource Locator).

3.6 Number of Digits

When referring to a URL, "number of digits" typically means the total number of numeric characters (0–9) in the URL, including the scheme, host, path, query parameters, and any fragments.

3.7 Number of Parameters

In the context of URLs, the term "number of parameters" refers to the total number of unique variables or data points contained in the query string portion of a URL. Parameters are typically specified after the "?" character and separated by "&" in a URL.

3.8 Number of Fragments

The number of unique IP fragments connected to a network packet is represented by the "Number of Fragments" feature in network analysis. It is employed to evaluate the integrity of network traffic as well as any indications of possible fragmentation attacks or problems with data transfer.

3.9 Is Encoded

In the context of URLs, "is encoded" describes whether the characters have been encoded using percent-encoding or URL encoding.

3.10 URL Entropy

The term "URL entropy" describes the degree of unpredictability or randomness in the characters that comprise a URL. It measures how complicated or disorganized the URL string is.

3.11 Number of Sub-Directories

The count of directories or folder levels that are present in the path portion of the URL is referred to as the "number of subdirectories" in a URL.

3.12 Number of Periods

The count of "." (dot) characters found in the hostname or domain name portion of a URL is referred to as the "number of periods" in that URL.

3.13 Has keyword ‘client’

Given that the term "client" is frequently used in a variety of legitimate URLs, its inclusion in a URL does not automatically imply that the URL is malicious; further context or analysis is required to make this determination.

3.14 Has keyword ‘admin’

In most systems and applications, the term "admin" usually denotes the mention of privileged or administrative access. In many cases, locating this keyword could be crucial for security monitoring since it could indicate actions related to administrative duties or potential attempts to enter restricted areas of a system.

3.15 Has keyword ‘server’

When the word "server" appears in a context, it usually refers to a system or device that is intended to supply resources, services, or data to other devices, usually in a networked setting. Finding the term "server" in networking and security research might be useful for determining communication with server entities or keeping an eye on server-related activity.

3.16 Has keyword ‘login’

When the term "login" appears in a system or application, it usually means that access control or authentication measures are present. Finding this keyword can be useful in a number of situations, such log analysis or security monitoring, when spotting login-related activity is crucial to maintaining a system's integrity and security.

3.17 Has port

In networking or cybersecurity, the phrase "Has port" usually refers to the existence of a port. It indicates if a communication or network entity uses a particular port number, which is essential for locating and directing data inside a network.

4. Models Used for Training :

4.1 Random Forest

A machine learning ensemble technique called Random Forest is employed for classification tasks, such as the identification of malicious URLs. During training, it generates multiple decision trees, each of which takes into account a subset of features obtained from URLs. The algorithm is robust and less prone to overfitting because it aggregates predictions from these trees to determine the most likely classification for a given URL. By utilizing the collective decision-making of multiple trees, Random Forests perform exceptionally well in handling a variety of URL features, recognizing patterns, and differentiating between benign and malicious URLs in the context of cybersecurity. This leads to a more accurate and dependable detection of potential threats.

4.2 XGBoost

A powerful machine learning algorithm called Extreme Gradient Boosting, or XGBoost, is well-known for its accuracy and efficiency in regression and classification tasks, including the detection of malicious URLs. This method of ensemble learning generates a sequence of decision trees, each of which corrects the errors of the previous one. A great tool for handling a variety of URL features is XGBoost. It combines weak learners into a strong model that can recognize complex patterns in URL data using gradient boosting. Because of its high performance and optimized implementation, it is a popular choice in cybersecurity for accurately identifying benign and malicious URLs by leveraging the combined power of these enhanced decision trees.

4.3 Gradient Boosting : A machine learning technique called gradient boosting creates a series of models, typically decision trees, to correct errors in previous models. In the context of malicious URL detection, Gradient Boosting iteratively constructs decision trees, highlighting scenarios where prior models underperformed. By employing this technique, the algorithm can gradually learn from mistakes and enhance its ability to detect complex patterns and differentiate between benign and malicious URLs based on a range of URL attributes. Gradient boosting is highly valued in the cybersecurity space because it can handle a wide range of data types and improve prediction accuracy by refining predictions through successive iterations.

4.4 ADA Boosting : Adaptive Boosting, or AdaBoost, is an ensemble learning technique that combines weak learners—usually straightforward models that are referred to as "base" or "weak" classifiers—to create a strong classifier with the goal of improving their performance. AdaBoost gives each round's misclassified samples more weight by allocating weights to the data points and modifying these weights as it goes. Through this process, AdaBoost can concentrate on the cases that are challenging to classify and modify its approach to increase overall accuracy.

4.5 Voting Classifier: A voting classifier is a machine learning ensemble learning technique that combines several models into a single prediction. This kind of model averaging technique makes use of a variety of various algorithms to improve performance all around. The fundamental concept involves the independent training of multiple base models, followed by a voting mechanism to combine their predictions.

Two primary categories of voting classifiers exist:

1. **Hard Voting:** In a hard vote, the class label is predicted by each base model in the ensemble, and the class with the highest vote total becomes the final

prediction. This is useful in classification tasks where predicting a class label is the objective.

2. Soft Voting Classifier: The probability scores that each classifier predicts for each class are taken into consideration when soft voting. It takes into account the average probability scores for every class across all classifiers and chooses the class with the highest average probability as the final prediction, as opposed to choosing the class with the majority of votes.

As its base classifiers, the Voting Classifier can use a variety of machine learning algorithms, such as Decision Trees, Support Vector Machines, Logistic Regression, etc. When compared to using individual classifiers alone, it frequently aids in enhancing generalization, decreasing overfitting, and increasing predictive accuracy.

Voting Classifier Benefits

1.Enhanced Performance: Voting classifiers frequently perform better than single models, particularly when the base models have a variety of advantages and disadvantages. Individual models may miss subtleties and patterns that the ensemble can pick up on.

2.Robustness: By lessening the effect of noise in individual models, ensembling helps reduce the risk of overfitting. In cases where different models may perform well on different portions of the dataset, it offers a more stable and trustworthy prediction.

3.Flexibility: A range of base models, including various machine learning algorithms, can be used to build voting classifiers. The ensemble's adaptability to different data types and problem domains is made possible by this flexibility.

4.Increased Accuracy: By combining predictions from several models, it can effectively leverage the advantages of various algorithms or models, which

frequently results in improved predictive performance when compared to individual classifiers.

5.Decreased Overfitting: When employing different algorithms or models with different biases and error patterns, combining multiple models can help reduce overfitting. The final classifier is made more robust and broadly applicable thanks to this diversity.

6.Enhanced Stability: The Voting Classifier's ensemble design usually results in increased stability and a decreased sensitivity to noise or outliers in the dataset, which produces predictions that are more trustworthy.

7.Adaptability: It permits adaptability in the combination of different classifier types, allowing for the addition of alternative feature representations or learning strategies, which enhances performance.

5. EXPERIMENTAL ANALYSIS

Confusion Matrix : A table that is frequently used in machine learning to assess how well a classification algorithm performs is called a confusion matrix. True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) are its four distinct metrics. The performance of the classifier in terms of correctly or incorrectly classifying instances is evaluated using these metrics.

For any classification task, such as malicious URL detection, a confusion matrix offers a comprehensive analysis of the model's right and wrong predictions. It facilitates comprehension of:

5.1 True Positives (TP): Cases that were accurately flagged as positive (e.g. malicious URLs correctly identified).

5.2 True Negatives (TN): Examples that are accurately predicted as negative, such as correctly identifying benign URLs, are known as True

Negatives (TN). Instances of the positive class were correctly predicted by the model. For instance, accurately diagnosing a disease in a patient.

5.3 False Positives (FP): Erroneously predicted as positive instances, or benign URLs mistakenly classified as malicious, are known as false positives (FP). Instances that were actually negative were mispredicted by the model as positive. Likewise referred to as a Type I error. For instance, giving a healthy patient a false diagnosis of a disease.

5.4 False Negatives (FN): Events that are incorrectly anticipated as negative (malicious URLs mistakenly identified as benign, for example).

The negative class was accurately predicted by the model. For instance, accurately recognizing people in good health.

Instances that were actually positive were mispredicted by the model as negative. Likewise referred to as a Type II error. For instance, neglecting to identify a patient's illness.

5.5 Accuracy : It is determined by dividing the total number of instances in the dataset by the ratio of correctly predicted instances (including true positives and true negatives).

Accuracy is calculated mathematically as

$$\mathbf{(TP + TN) / (TP + TN + FP + FN)}.$$

Although accuracy gives a general indicator of how well predictions went, it may not be appropriate for datasets that are imbalanced and have a large number of members in one class compared to another.

5.6 Precision : Precision is defined as the percentage of all instances predicted as positive (true positives plus false positives) that are

accurately predicted as positive (true positives). It assesses how well the model avoids producing false positive results.

Precision in mathematics is equal to

TP / (TP + FP).

Fewer false positives are indicative of high precision.

5.7 Recall : Recall, also known as True Positive Rate or Sensitivity, quantifies the percentage of accurately predicted positive cases, or true positives, out of all real positive cases, or true positives plus false negatives. It assesses how well the model can detect every positive instance without missing any.

Recall is equal to

TP / (TP + FN)

Recall that it is high suggests fewer false negatives.

5.8 F-1 Score : A useful tool for assessing a classification model's performance and pinpointing areas for potential improvement is the confusion matrix, particularly in situations where there are unequal numbers of classes or distinct expenses linked to false positives and false negatives.

6. PRACTICAL ISSUES AND OPEN PROBLEMS

6.1 Practical Issues

6.1.1 Unbalanced Collections: There are frequent disparities between malicious and benign URLs in the training datasets. This may result in skewed models that don't work well with actual data.

6.1.2 Dynamic Character of Dangers: Attackers tactics, techniques, and procedures (TTPs) are always changing. Staying up to date with novel and complex attack techniques may prove difficult for models.

6.1.3 Attacks by Adversaries: Attackers have the ability to purposefully alter URLs in order to trick machine learning models. It is possible to create URLs through adversarial attacks that seem harmless but are actually harmful.

6.1.4 Feature Development: It is difficult to choose pertinent features that encapsulate the essence of malicious URLs. The selection of features can significantly affect the model's performance, which is why feature engineering is so important.

6.1.5 Instantaneous Detection: Real-time detection is necessary for many applications, and processing URLs in real-time has computational requirements.

6.1.6 Interpretability and Explainability: Interpretability and explainability of machine learning models decline with increasing complexity (e.g., deep learning models), making it difficult to comprehend how they make decisions.

Interpretability in cybersecurity is essential to building trust and comprehending the rationale behind a classification.

6.1.7 Extrapolation Between Domains:It's possible that models developed on one kind of data won't translate well to another. It is a

constant challenge to adapt models to various malicious activity types and heterogeneous data sources.

6.1.8 Generalization to New Threats: One of the biggest challenges in machine learning is making sure that models adapt well to new and emerging threats without requiring a lot of retraining. It is important for models to adjust to changing attack tactics without compromising precision or producing false positives.

6.2 Open Problems

6.2.1 Zero-Day Exploits: Finding URLs linked to threats that have never been seen before (zero-day attacks) is still a big problem. With novel, unseen patterns, traditional machine learning models might not perform well.

6.2.2 Transfer Knowledge: Creating efficient transfer learning strategies that let models use information from one domain—like phishing detection—to enhance performance in another (like malicious download detection).

6.2.3 XAI, or explainable AI: Constructing machine learning models that, in addition to being accurate, offer comprehensible and interpretable justifications for their judgments. This is essential for fostering human understanding of the system and establishing trust.

6.2.4 Online Education: Creating models with ongoing online learning and adaptation capabilities so they can withstand changing threats without needing to be retrained frequently.

6.2.5 Cooperative Identification: Investigating cooperative strategies in which various entities (organizations,Security vendors) exchange data in order to enhance detection capabilities overall while protecting sensitive information.

6.2.6 Network security integration: Allowing for the smooth integration of URL detection models into larger network security frameworks, minimizing false positives and guaranteeing coordination with other security measures.

6.2.7 User Conduct Examination: Utilizing user behavior analysis to improve identification. Knowing how people interact with URLs can give important context for differentiating between malicious and benign activity.

6.2.8 Privacy and Ethical Issues: Addressing privacy-related ethical issues, particularly when handling potentially sensitive data. It's critical to strike a balance between the demands of privacy and security.

6.2.9 Data Privacy Issues: When using labeled datasets with malicious URLs for model training, privacy issues might arise. It's crucial to create efficient detection models while upholding user confidentiality and privacy.

6.2.10 Computational Resources: A large amount of computational resources is needed to train machine learning models for malicious URL detection, particularly for more intricate algorithms like deep learning. It is essential to optimize algorithms for resource efficiency without sacrificing accuracy.

6.2.11 Absence of Standardized Evaluation Metrics: It can be difficult to create benchmarks and standardized evaluation metrics for contrasting various detection models. It would be easier to evaluate model performance using uniform evaluation criteria across different datasets and approaches.

6.2.12 Domain-Specific Difficulties: Malicious URL detection can present distinct challenges depending on the domain. Because social media platforms have different content and behaviors than traditional web-based URL detection, for instance, detecting malicious URLs there may not be the same.

6.2.13 Human Factor in Cybersecurity: Malicious URL attacks are largely successful due to human error, social engineering techniques, and user behavior. It is still essential to teach users how to identify and stay away from malicious URLs.

A multidisciplinary approach involving knowledge of cybersecurity, machine learning, data privacy, and ethical issues is needed to address these real-world problems and open issues. To advance machine learning-based malicious URL detection, continued research and cooperation between government, industry, and academia are necessary.

7. CONCLUSION

In conclusion, there is a lot of potential for improving cybersecurity measures with the use of machine learning in malicious URL detection. Effective systems that can recognize and mitigate threats posed by malicious URLs can be developed by the cybersecurity community with careful feature selection, strong model development, ongoing learning, and cooperative efforts. Machine learning and cybersecurity techniques work together to maintain the integrity and security of online environments as technology develops and cyber threats change. Maintaining a competitive edge and protecting digital ecosystems will require a persistent dedication to innovation and information exchange.

8. REFERENCES

- "A Novel Approach for Malicious URL Detection Using BERT and Random Forest" by Aslam, A., et al. (2022): https://www.hindawi.com/journals/scn/2021/4917016/

- "BERT-Based Malicious URL Detection Using Random Forest" by Singh, A.K., et al. (2021): https://arxiv.org/abs/2202.10027

- "Using BERT and Random Forest to Detect Phishing URLs" by Wang, W., et al. (2020): https://www.researchgate.net/publication/365241078_Intelligent_Deep_Machine_Learning_Cyber_Phishing_URL_Detection_Based_on_BERT_Features_Extraction

- "Lexical Features Based Malicious URL Detection Using Machine Learning Techniques" by A. Saleem Raja, Vinodini K., and P. Radha Krishna: https://www.sciencedirect.com/science/article/pii/S2214785321028947

- "Detecting Malicious URLs Using Machine Learning Techniques: Review and Research Directions" by Mohammad, R., et al. (2020): https://ieeexplore.ieee.org/document/9950508

- "Malicious URL Detection: A Comparative Study" by Nayyar, A., et al. (2020): https://ieeexplore.ieee.org/document/9396014

- "URL-Based Malicious Website Detection Using Machine Learning" by Gupta, A., et al. (2018): https://ieeexplore.ieee.org/document/9655851

- "Machine Learning for Malicious URL Detection: A Review" by Al-Bataineh, A.R., et al. (2017): https://ieeexplore.ieee.org/document/9950508

- "Phishing URL Detection Using Machine Learning: A Review" by Al-Khateeb, M., et al. (2018): https://www.sciencedirect.com/science/article/pii/S0965997822001892

- "Malicious URL Detection Using Deep Learning: A Review" by Yuan, Y., et al. (2020): https://ieeexplore.ieee.org/abstract/document/8791348