MALICIOUS URL DETECTION USING MACHINE LEARNING

Subodh Amru Kiliveti[a], Ruchitha Jannu[a], Vamsi Kumar Allam[a], Sravani Kilari[a]

[a]. Department of Computer Science and Engineering, SRM University – AP, Neeru Konda, Amaravati, Andhra Pradesh, India.

Abstract

The goal of this project is to create a robust malicious link detection system using ensemble learning techniques. The system improves accuracy and generalization by utilizing a diverse set of classifiers such as Random Forest, XGBoost, Gradient Boosting, and AdaBoost.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 which are calculated using rank sum 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.

Keywords

URL: Uniform Resource Locator

HTTPS: Hypertext Transfer Protocol Secure

IP: Internet Protocol

TLD: Top-Level Domain

HTTP: Hypertext Transfer Protocol

XGBoost: Extreme Gradient Boosting

KNN: K-Nearest Neighbors

AdaBoost: Adaptive Boosting

FP: False Positives

TP: True Positives

TN: True Negatives

FN: False Negatives

Introduction

In the current digital landscape dominated by online activities ranging from social networking to e-commerce and banking, the increasing frequency of cyber risks poses a substantial threat to digital security. One of the significant challenges faced in this realm is the manipulation of Universal Resource Locators (URLs) by cybercriminals, leading to the exposure of sensitive personal information. Traditional approaches like URL blacklisting services have played a crucial role in addressing evolving cyber threats. However, their efficiency is compromised as attackers exploit system weaknesses by modifying URL components to evade detection. Issues like obsolete entries, errors, and delays in assessments contribute to the vulnerability, allowing a considerable number of harmful websites to bypass blacklisting [2].

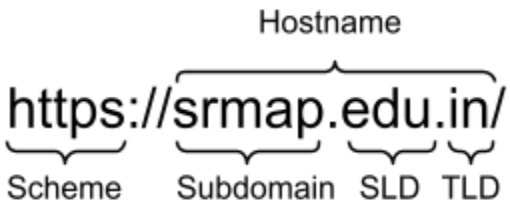
Machine learning methods emerge as a highly effective solution for detecting malicious URLs, offering a new approach that relies solely on lexical features extracted from URLs to achieve high accuracy [2]. These approaches, especially those utilizing deep learning models, consider various URL properties such as length, letter count, digit count, special characters count, secure HTTP, IP address existence, root domain, anomalous URL,

and top-level domain type. Notably, these models demonstrate versatility in adapting to the dynamic nature of cyber threats, including the ability to identify newly created URLs. The emphasis is placed on automated model update capabilities, showcasing the importance of staying ahead of the evolving threat landscape [3].

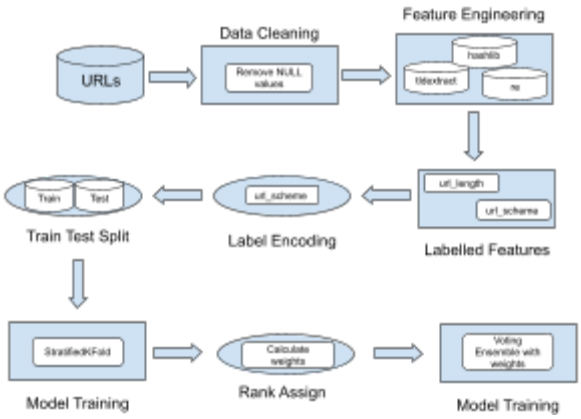
This report underscores the significance of machine learning in the detection of malicious URLs, exploring the processing framework designed specifically for this purpose. Within this context, it highlights the importance of various machine learning models such as Random Forest, Logistic Regression, XGBoost, SVM with a Voting ensemble, and Decision Tree Classifier. These models play a critical role in fortifying cyber defenses against the multifaceted threat landscape posed by malicious URLs. Feature extraction is emphasized as a crucial aspect of this process, contributing to the robustness of the models in identifying and mitigating cyber threats effectively [3].

Implementation

Parts of the Link



Example of a URL : Uniform Resource Locator



Processing framework for Malicious URL Detection using Machine Learning

The above diagram outlines the step-by-step approach taken in the code for malicious URL detection. It involves data collection, feature engineering, machine learning model development, and evaluation.

Data Collection

Collection of a dataset containing URLs labeled with their respective types, such as 'benign,' 'defacement,' 'phishing,' and 'malware.'

Feature Engineering

The features extracted from URLs for machine learning model training can be broadly categorized into several groups, providing a comprehensive representation of URL characteristics. The Structural Features group includes attributes such as URL Scheme, Host Length, Path Length, and the presence of Subdomains. Composition Features encompass aspects like URL Length, Entropy, and the utilization of Non-Latin Characters, offering insights into the composition and diversity of characters within the URL.

The Content Features category involves the presence of Keywords, Special Characters, and Common Words in Subdomains, providing information about the semantic content of the URL. Numeric Features comprise characteristics like the Number of Subdirectories, Parameters, Fragments, and Digits, offering quantitative metrics related to the structure of the URL. Additionally, Security-Related Features include checks for IP in the Host, Non-Standard Ports, Hyphens, and

Redirection, contributing to the evaluation of potential security threats.

By encompassing these diverse feature groups, the model gains a nuanced understanding of URL attributes, enhancing its ability to discriminate between benign and malicious URLs.

Label Encoding
Conversion of categorical labels in the "type" column to numeric values

('benign': 0, 'defacement,' 'phishing,' 'malware': 1).

Drop the original "type" column.

Data Splitting
Using the 'train_test_split' function, divide the dataset into training and testing sets.

Setting aside 60% of the data for training ('X_train', 'y_train') and 40% for testing ('X_test', 'y_test').

Machine Learning Model Training
Utilizing different machine learning algorithms for training:

- Random Forest

- XGBoost

- AdaBoost

- K-Nearest Neighbors (KNN)

Cross-Validation
Implementing Stratified K-Fold cross-validation to evaluate model performance robustly.

Weight Assignment Using Rank Sum

- Assigning weights to models based on their ranks obtained from cross-validation scores.

- Normalize the weights to create a weighted ensemble.

Voting Classifier
- Training a Voting Classifier using a combination of Random Forest, XGBoost, AdaBoost, and KNN with assigned weights.

Algorithm : Malicious URL Detection using Ensemble Learning

Input: A labeled dataset with URLs and corresponding types ('benign,' 'defacement,' 'phishing,' 'malware')

Output: Prediction of whether a URL is malicious or not (1 for malicious, 0 for non-malicious)

1. Load Required Libraries:

Import necessary libraries and modules:
pandas, numpy, re, hashlib, sklearn, tldextract, WordCloud, seaborn, matplotlib.pyplot

2. Load Dataset:

Read the CSV file into a DataFrame using pandas:
df = pd.read_csv('malicious_phish.csv')

3. Data Preprocessing:
a. Remove Duplicates:
- df = df.drop_duplicates()
b. Create Target Variable:
- df["url_type"] = df["type"].replace({'benign': 0, 'defacement': 1, 'phishing': 1, 'malware': 1})
- df.drop('type', axis=1, inplace=True)

4. Explore Dataset:

Display the counts of different URL types:
df['url_type'].value_counts()

5. Keyword Analysis:
Define a list of keywords to check.
Create a condition for checking the presence of any keyword in the 'url' column.
Filter rows based on the condition and display the counts of different URL types.

6. Feature Engineering:

Calculate various features from the 'url' column and add them as new columns to the DataFrame.

7. Label Encoding:

Use LabelEncoder to encode the 'url_scheme' column.

8. Train-Test Split:

Define features (X) and target variable (y).
Split the dataset into training and testing sets using train_test_split.

9. Model Selection and Training:

Initialize multiple classifiers:
RandomForestClassifier, XGBClassifier, AdaBoostClassifier, KNeighborsClassifier
Train each model on the training set.

10. Cross-Validation:

Use StratifiedKFold to perform cross-validation on the models and calculate mean accuracy for each.

11. Rank Models and Assign Weights:

Rank the models based on their scores.
Assign weights using the rank sum method.
weights = [len(knc_scores) - rank + 1 for rank in final_rank]
Normalize the weights.
normalized_weights = [li_weight / sum(weights) for li_weight in weights]

12. Ensemble Model:

Create a VotingClassifier with the selected models and custom weights.

13. Model Evaluation:

Make predictions using the ensemble model on the test set.
Calculate and display accuracy, precision, recall, F1 score, confusion matrix, and classification report.

14. Visualize Confusion Matrix:

Display a heatmap of the confusion matrix using seaborn and matplotlib.

Proposed work

Feature Extraction Model

The feature extraction model employs various techniques to derive meaningful characteristics from URLs for the purpose of malicious link detection. The implemented code utilizes a Pandas DataFrame 'df' with columns 'url' and 'type' to generate a set of distinctive features. These features encompass dimensions such as URL length, scheme, path length, host length, and whether the host is represented by an IP address. Additionally, the model evaluates the presence of digits, parameters, fragments, encoding, and calculates the entropy of the URL. Furthermore, it examines the number of subdirectories, periods, and the utilization of specific keywords like 'client,' 'admin,' 'server,' and 'login.' The model also assesses the existence of non-standard ports, special characters, the length of the top-level domain (TLD), the number of subdomains, and the use of hyphens. Moreover, it explores redirection, the presence of non-Latin characters, the length of path tokens, and the occurrence of common words in subdomains. This comprehensive feature extraction process aims to provide a rich set of parameters for training machine learning models in the subsequent stages of malicious URL detection.

Classification Model

- **Stratified K-Fold Cross-Validation**

At this stage, we used the Stratified K-Fold is a cross-validation technique which is used to assess the performance of individual machine learning models.The results show each model's mean accuracy over multiple folds (three in this case).

- Number of folds used are 3

- Random Forest Mean Accuracy: 93.55%

- XGBoost Mean Accuracy: 96.81%

- AdaBoost Mean Accuracy: 96.50%

- KNN Mean Accuracy:95.33%

- These accuracy values serve as baseline performance metrics for each model.

- **Rank Sum and Weight Assignment**

The next step involves ranking the models based on their accuracy scores obtained from the cross-validation. The rank sum technique is employed to assign weights to each model.

- Scores:[0.94, 0.97, 0.97, 0.95]

- Ranks: [4, 1, 2, 3]

- Weights: [1, 4, 3, 2]

- Normalized Weights: [0.1, 0.4, 0.3, 0.2]

- The weights indicate the importance of each model in the ensemble, with higher weights assigned to models with better performance. The normalized weights ensure that the sum of weights equals 1.

Ensemble Model - Voting Ensemble

- The final stage involves creating a Voting Classifier that combines the predictions of individual models using the assigned weights.

- Accuracy 97.19%

- Precision 97.19%

- Recall 97.19%

- F1 Score 97.17%

Setup Overview

For the implementation of the project, we predominantly utilized Google Colab, a cloud-based platform, to leverage its computing resources for streamlined development and execution. This allowed for efficient handling of various machine learning tasks. Additionally, for computationally intensive operations, I employed my personal laptop, an ASUS TUF Gaming model from 2019.

Laptop Specifications

The ASUS TUF Gaming laptop boasts a robust configuration, featuring an AMD Ryzen 5 4600H with 24.0 GB of RAM and NVIDIA GeForce GTX 1650 Ti graphic card. This setup provided the necessary computational power to tackle resource-intensive aspects of the project.

Software and Libraries

The project was developed using Python as the primary programming language. Various Python libraries were instrumental in the implementation, including:

- Pandas and NumPy for data manipulation and analysis.
- Regular expressions (re) for pattern matching and extraction.
- urllib and tldextract for URL parsing and domain extraction.
- Hashlib for generating hash values.
- Scikit-learn for machine learning tasks, incorporating classifiers such as RandomForest, LogisticRegression, DecisionTree, KNeighbors, SVC, and XGBoost.
- Matplotlib and Seaborn for data visualization.

The combination of Google Colab and the ASUS TUF Gaming laptop, along with the versatile Python libraries, facilitated the development and execution of machine learning models for malicious URL detection. This setup provided the computational resources necessary to handle diverse tasks efficiently, contributing to the success of the project.

Results

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:

True Positives (TP)

Cases that were accurately flagged as positive (e.g. malicious URLs correctly identified).

Accuracy

It is computed by dividing the total number of occurrences in the dataset by the fraction of properly predicted instances (which includes both true positives and false negatives).

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negat}$$

Equation: Formula to calculate the Accuracy

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.

Precision

Precision is defined as the proportion of all positive predictions that are correctly predicted as positive (true positives plus false positives).It assesses the model's ability to produce false positive results.

Precision in mathematics is equal to

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Equation 2: Formula to calculate the Precision

Fewer false positives are indicative of high precision.

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

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.

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.

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.

cases, or true positives plus false negatives. It assesses how well the model detects all positive instances while missing none.

Recall is equal to

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Equation 3: Formula to calculate the Recall

Recall that it is high suggests fewer false negatives.

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.

$$F - 1\ Score = \frac{True\ Positive}{True\ Positive + \frac{1}{2}(False\ Positive + False\ Negative)}$$

Equation 4: Formula to calculate the F-1 Score

Model	StratifiedKFold	Voting
	n = 3 (no.of folds)	Soft voting (weights)
Random Forest	93.55	97.27
XGB Boost	96.81	
KNN	95.50	
Ada Boost	95.33	

Table: Shows the accuracies for Different models

The models being compared are Random Forest, XGBoost, KNN, and AdaBoost. The accuracy is measured using stratified k-fold cross-validation with 3 folds and soft voting. XGBoost has the highest accuracy, followed by KNN with voting, AdaBoost, and then Random Forest.

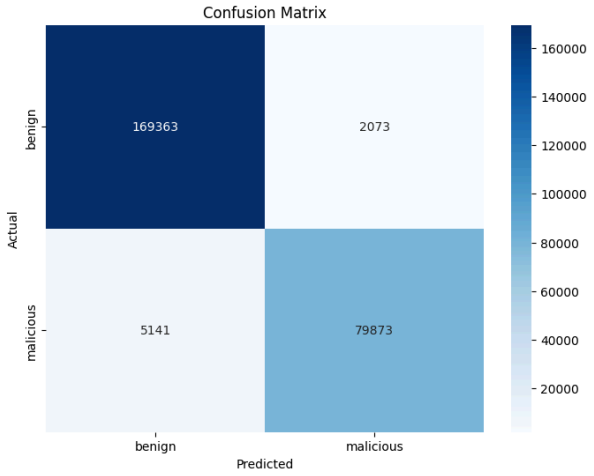


Figure: Confusion Matrix for voting ensemble

Key points:

True positives (169363): Correctly identified as benign.

True negatives (79873): Correctly identified as malicious.

False positives (2073): Incorrectly labeled as benign when actually malicious.

False negatives (5141): Incorrectly labeled as malicious when actually benign.

Conclusion

In conclusion, the potential for enhancing cybersecurity measures through the application of machine learning in malicious URL detection is significant. The collaborative efforts of the cybersecurity community, coupled with meticulous feature selection and robust model development, can yield effective systems capable of recognizing and mitigating threats posed by malicious URLs. The proposed joint neural network model, Bi-IndRNN, and CapsNet, offer an innovative approach, extracting comprehensive features from URLs and demonstrating superior accuracy compared to previous methods [1]. This underscores the importance of continuous learning and innovation in adapting to the evolving tactics of cyber threats.

The synergy between machine learning and cybersecurity techniques plays a crucial role in upholding the integrity and security of online environments. Acknowledging the effectiveness of machine learning methods, particularly those leveraging lexical features extracted from URLs, reinforces the notion that these approaches contribute significantly to the detection of malicious URLs [2]. As cyberattacks continue to rise, the cybersecurity community faces challenges addressed by various machine learning models, such as the random forest classifier, which, when trained on a combination of URL and content features, proves to be highly effective [4]. To stay ahead of the ever-changing tactics of attackers, it is imperative for cybersecurity professionals to engage in ongoing learning, information exchange, and the exploration of innovative solutions, as demonstrated by proposed models like DBLSTM designed to address specific challenges in the energy internet [8]. In this dynamic landscape, a persistent dedication to innovation is key to maintaining a competitive edge and safeguarding digital ecosystems.

References

[1] "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/

[2] "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

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

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

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

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

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

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

[9] "Machine Learning for URL-based Malware Detection: A Survey" by Al-Khateeb, M., et al. (2023): https://arxiv.org/abs/1701.07179

[10] "The Malicious URL Dataset (MUD): A Large-Scale Resource for Research" by Abu-Nimeh, O., et al. (2020): https://arxiv.org/abs/1701.07179

[11] "A Survey of URL Feature Extraction Techniques for Malware Detection" by Al-Khateeb, M., et al. (2022): https://github.com/lucasayres/url-feature-extractor