Identification of Malicious URLs: A Purely Lexical Approach

Julio Rodrigues[1], Charles de Barros[1], Diego Dias[2(✉)], Marcelo de Paiva Guimarães[3], Elisa Tuler[1], and Leonardo Rocha[1]

[1] Universidade Federal de São João Del Rei (UFSJ), São João Del Rei, Brazil
{charlesbarros,etuler,lcrocha}@ufsj.edu.br
[2] Universidade Federal do Espírito Santo (UFES), Vitória, Brazil
diego.dias@ufes.br
[3] Universidade Federal de São Paulo (UNIFESP), São Paulo, Brazil
marcelode.paiva@unifesp.br

Abstract. Internet users are increasingly exposed to security vulnerabilities stemming from malicious Uniform Resource Locators (URLs), which act as conduits for cyber threats. These threats, often orchestrated by sophisticated cybercriminals, underscore the importance of comprehending the intricate dynamics involved to devise robust defense mechanisms. This scholarly exposition delineates an efficacious approach for discerning diverse categories of malicious URLs leveraging machine learning algorithms. Notably, our methodology obviates the necessity of directly accessing such URLs for extracting pertinent information, relying solely on attributes inherent within the lexical composition of the URLs. The empirical analyses are predicated on meticulously curated datasets from reputable repositories such as Kaggle and PhishTank, culminating in competitive performance vis-à-vis existing literature that predominantly focuses on network-centric or content-based features.

Keywords: Machine Learning · Malware detection · Feature Extraction

1 Introduction

Users' vulnerability to internet security is one of the weakest points in the digital ecosystem. The URLs in our daily lives act as direct gateways to malicious online activities, requiring just a few clicks to carry out a cyber attack. This easy access facilitates the spread of digital threats, significantly increasing the potential for infection. However, the effectiveness of these attacks often depends on additional characteristics of the URL itself and the sophisticated camouflage techniques employed by cybercriminals. Understanding the dynamics behind these tactics is fundamental to developing more effective online protection strategies.

In 2019, the cybersecurity landscape in Brazil confronted formidable challenges with the proliferation of cyber attacks leveraging malicious URLs, as

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
O. Gervasi et al. (Eds.): ICCSA 2024, LNCS 14814, pp. 397–407, 2024.
https://doi.org/10.1007/978-3-031-64608-9_26

underscored by the findings of the Abranet report, which positioned Brazil among the fifteen nations with the highest incidence of victims. However, it is reasonable to posit that these alarming statistics have substantially exacerbated in subsequent years, particularly in light of the protracted duration of the COVID-19 pandemic. This unprecedented global crisis has accelerated the adoption of information technologies across diverse sectors, consequently enlarging the attack surface susceptible to potential cyber threats. The dynamic nature of cybersecurity perpetually undergoes evolution, with both defenders and adversaries engaged in an incessant pursuit of enhancing their respective strategies. While security teams diligently endeavor to devise more robust detection mechanisms, the perpetrators of cyber attacks persistently refine their tactics, particularly in the realm of obfuscating malicious URLs. Such perpetual flux necessitates implementing proactive and adaptive measures to mitigate the ever-looming specter of online security risks effectively.

Currently, two popular and practical strategies facilitate the detection of malicious URLs, the first of which is blacklisting services. These services keep a record of URLs that have been reported and identified as malicious, thereby checking whether a suspicious URL is present in their database. Although blacklisting services are helpful in the medium and long term, they often face limitations in detecting newly created or as yet uncatalogued malicious URLs. This gap in coverage of recent URLs highlights the need to complement this approach with other more dynamic and adaptable detection strategies.

The second approach involves building machine learning models to classify URLs. Although this strategy is more complex compared to blacklisting services, it offers a significant advantage in terms of effectiveness in the short, medium, and long term. A well-built machine learning model can accurately classify many URLs when successfully implemented. This approach takes advantage of sophisticated algorithms to analyze various attributes of URLs, identifying subtle patterns that can indicate whether a URL is malicious or legitimate. Using machine learning models represents a promising evolution in malicious URL detection, providing a more adaptable and efficient solution to the ever-changing challenges of the cybersecurity landscape.

The subsequent sections of this paper are delineated as follows: Sect. 2 expounds upon the employed methodology for development; Sect. 3 furnishes an analysis of the acquired results; Sect. 4 deliberates upon the principal conclusions derived from this study and posits a set of considerations for prospective future endeavors.

2 Methodology

In this section, we outline the development process and the crucial decisions in designing an effective method for classifying URLs based on machine learning. It is essential to highlight that the entire development of this work utilized the

Python[1] programming language in conjunction with the open-source scikit-learn[2] machine learning library.

2.1 Feature Extraction

In our endeavor to extract highly predictive information from URLs, we opted for a purely lexical approach, focusing solely on the URL's structure. This decision stemmed from the recognition that extracting features about content or network attributes, as elaborated upon in diverse related studies, typically necessitates direct URL access for data collection. However, our primary objective is to discern the essence of a URL without necessitating direct access, thereby mitigating potential exposure to cybersecurity threats.

The features can be divided into two groups: those related to the structure of the URL, such as the length and specific characters, and those linked to the content of the page, such as the use of pop-ups and cookies. Thus, the characteristics of malicious domains can include the presence of an IP address in the URL, the use of at, double slashes "//", hyphens, HTTPS, the domain's favicon, the use of pop-ups, the length of the URL, the number of links and the number of open ports.

In this investigation, we extracted a set of 21 characteristics from the URLs (Table 1), employing a methodology grounded in simple statistical analysis and examining pertinent literature. Among these features, seven emerged from statistical analysis of the databases, while the remaining 14 were identified through scrutiny of the existing literature. We meticulously extracted all these characteristics, focusing on the lexical attributes encapsulated within the URLs. However, after applying feature selection techniques, we reduced this number to 11 features, as shown.

One of the elementary features extracted encompassed the enumeration of distinct symbols (tokens) manifested within the URLs. Intriguing disparities emerged concerning these quantities, notably observed in the prevalence of the "=" symbol within URLs. Remarkably, URLs categorized under defacement exhibited an average occurrence of approximately 11 times more instances of this symbol than other URL types in the collected database.

In addition to the symbol count, we extracted characteristics related to the measurement of URL segments. In this context, metrics such as the length and number of characters of the URLs and the size of the directory were considered. These characteristics are potential indicators for distinguishing malicious instances from safe ones.

2.2 Feature Selection

Once the feature extraction process was complete, we began analyzing the relevance of each feature. To conduct this analysis, we adopted a strategy of recursive elimination of features with cross-validation. In other words, we utilized a

[1] More information at: https://www.python.org/.
[2] More information at: https://scikit-learn.org/stable/.

Table 1. Features Extraction

Features	Description
protocol	Communication protocol used
exclamation	Number of occurrences of the '!' character in the URL
slash	Number of occurrences of the '/' character in the URL (ignoring the protocol - https://)
dirs_qtd	Number of directories in the URL path
digits_qtd	Number of digits in the URL
equals	Number of occurrences of the '=' character in the URL
question	Number of occurrences of the '?' character in the URL
dots	Number of occurrences of the character '.' in URL
fstdir_length	Length in characters of the first directory in the URL path
url_length	Total length in characters of the URL
question	Number of occurrences of the '-' character in the URL
underscore	Number of occurrences of the character '_' in the URL
ats	Number of occurrences of the '@' character in the URL
commercial	Number of occurrences of the character '&' in the URL
plus	Number of occurrences of the '+' character in the URL
star	Number of occurrences of the '*' character in the URL
hashtag	Number of occurrences of the character '#' in the URL
cipher	Number of occurrences of the character '$' in the URL
percent	Number of occurrences of the character '%' in the URL
url_punc	Number of occurrences of non-alphanumeric characters in the URL
susp_words	Boolean indicating the presence of "suspicious" words in the URL (e.g., free, bonus, payment, extra, lucky, ...)

bottom-up approach, dividing the database into five parts and eliminating features with less relevance in the classification at each step. By the end of this process, we obtained a set of 11 features with greater predictive power.

We continue this process until we find the optimum number of features, meaning the total number of features eliminated has little or no impact on the model's performance. In Fig. 1, we observe the result of feature selection using the XGBoost algorithm [2] as a base. An apparent discrepancy exists between the predictive power of the communication protocol used by the URLs and the rest of the attributes. This behavior stems from the well-defined patterns found in the database, where all defacement URLs use the HTTP protocol, while phishing and malware URLs predominantly utilize the HTTPS protocol. The 11 features selected by the XGBoost algorithm can be described as follows:

– **protocol**: The protocol, also known as the scheme, is the first part of a URL and determines how different files on a web server are displayed and transferred to the user. The protocol can be HTTP or HTTPS.

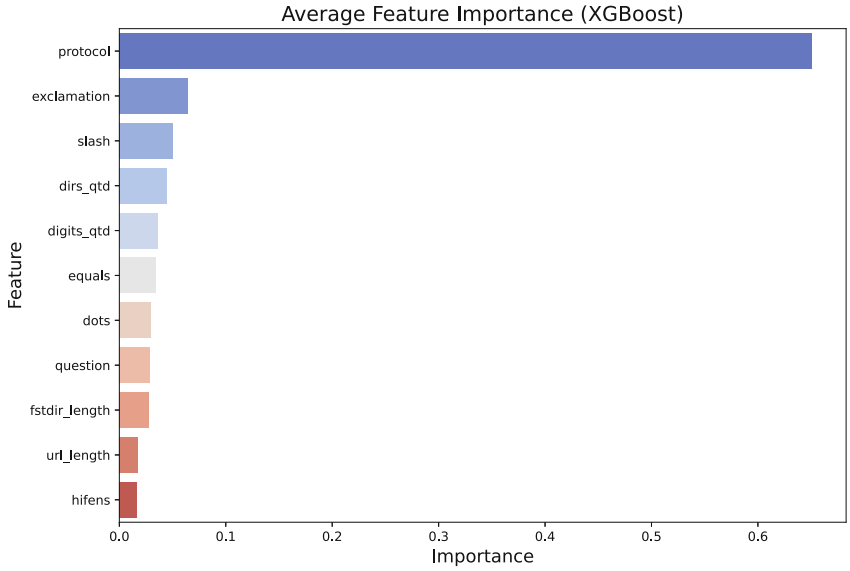

Fig. 1. The predictive power of the 11 features

- **exclamation**: The exclamation mark (!) is not commonly used in standard URLs. However, in some web applications, it can be used as part of a technique known as "hashbang" (#!) to enable client-side navigation.
- **slash**: The slash (/) separates different parts of a URL, such as the domain, path, and parameters1.
- **dirs_qtd**: This characteristic probably refers to the number of directories in a URL, which are the parts of the URL separated by slashes (/).
- **digits_qtd**: This characteristic probably refers to the number of digits in a URL.
- **equals**: The equals sign (=) is used in URLs to separate names and parameter values in query strings1.
- **dots**: These (.) are used in URLs to separate different parts of the domain1.
- **questions**: The question character (?) is used in URLs to start the query string, which contains additional parameters for the request1.
- **fstdir_lenght**: This characteristic probably refers to the length of the first directory in a URL.
- **url_lenght**: This characteristic refers to the total length of the URL.
- **hyphens**: These (-) are often used in URLs to separate words, especially in SEO-friendly URLs.

2.3 Database and Balancing

To build the supervised machine learning model, we used URLs from two databases:

– **Kaggle**[3]: Database made up of 651,191 URLs collected from various sources, distributed among four classes;
– **PhishTank**[4]: Database composed exclusively of phishing URLs. 105,905 instances were collected.

We observed a significant class imbalance after applying the join process and removing duplicate instances from the databases (Fig. 2). That necessitated the adoption of class balancing techniques before training the classifiers. The class distribution was Benign (57%), Defacement (27%), Phising (13%) and Malware (3%).

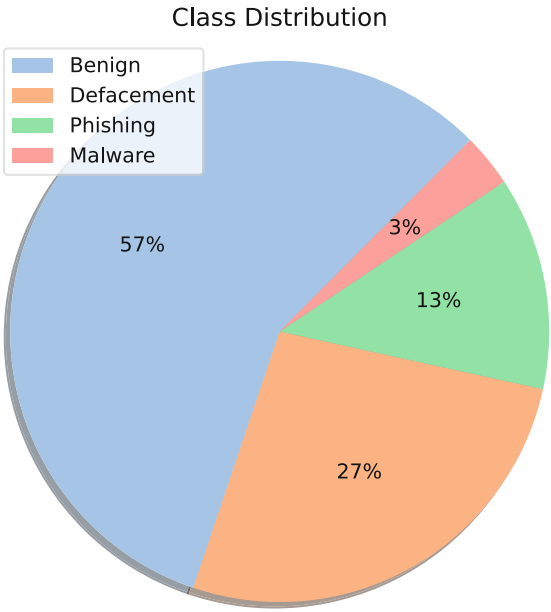

Fig. 2. Proportion of classes in the database

We used the F1-Score metric in our study. The F1 Score is a metric that combines precision (how many of the model's positive predictions are correct) and recall (how many of the actual positive instances were correctly identified by the model) into a single value. It is beneficial when we have an unbalanced data set, i.e., when one class is much more frequent than the others. In the context of classification metrics such as the F1 Score, the terms malware, benign, phishing, and defacement are usually referred to as classes or labels in a data set. Malware, a term derived from malicious software, is an umbrella term for

[3] Available at: https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset.
[4] Available at: https://phishtank.org/phish_archive.php.

any software designed to infiltrate a device without the user's consent and cause damage, system interruptions, or data theft.

There are several distinct categories of malware, each with its characteristics and infection methods. The most common examples of malware are adware, spyware, ransomware, and viruses. Benign usually refers to instances that are considered normal or safe. For example, in a malware detection dataset, samples free of malware would be labeled benign. Phishing refers to instances identified as phishing attempts (an online fraud technique where attackers try to trick users into revealing personal information, such as passwords or credit card details). Defacement refers to instances employing attacks where the visual content of a website is altered by the attackers, usually for vandalism or advertising purposes. They can be defined as follows:

- Adware is an abbreviation for "advertising-supported software", a form of malware that displays unwanted, sometimes malicious, advertising on a computer or mobile device's screen;
- Spyware is a category of malware that operates covertly on a device, monitoring user activity and stealing sensitive information such as financial data, account details, and login credentials;
- Ransomware, a particularly pernicious form of malware, is a serious threat that encrypts the user's data and demands a ransom for its recovery; and
- Virus is malicious code that spreads between computers and networks, often causing significant damage.

At this stage, we opted for a hybrid approach to applying balancing techniques to the database. While the most straightforward path would involve the indiscriminate application of random oversampling, duplicating instances of the minority classes, this approach might not add helpful information to the training and classification processes. It would also significantly increase the data volume, which would considerably increase the computational cost of training the models.

However, after conducting initial tests with the hold-out method, we observed that the F1 Score metric values were similar for the benign, phishing, and defacement URL classes. This preliminary result suggested the potential application of random undersampling to majority class instances (benign). Subsequently, we implemented this technique, reducing the number of safe URLs in the database from 428,081 to 200,000 instances. This approach provided a safety margin to prevent the loss of information if large numbers of non-redundant URLs were removed.

At this stage, we equalized the number of safe and phishing URL instances through random undersampling and data scraping from PhishTank's database. However, addressing the imbalance between defacement and malware URLs posed a challenge. Despite efforts in data scraping, similar to the approach taken for phishing URLs, we encountered difficulty locating databases of sufficient size to compensate for the shortage of such instances.

However, we chose to apply a balancing technique based on synthetic minority oversampling, known as SMOTE [1]. In other words, several synthetic instances

of the defacement and malware classes were generated based on the existing URLs of these types in the database.

At the end of this process, we had a fully balanced database of 800,000 URLs, 200,000 instances of each class, and 11 characteristics. With this database prepared, we could select the machine learning algorithms, formulate the models, and then evaluate them.

2.4 Classifier Selection

We chose a set of three classifiers to build the supervised machine learning models. The first is XGBoost, classified as an ensemble method, which uses a set of several classifiers with the boosting technique, calibrating and reinforcing the classifiers at each iteration. The remaining two algorithms correspond to methods considered classic in the literature: logistic regression and KNN [3].

The main reason for selecting these algorithms is to observe and compare the performance of a more recent and sophisticated classifier, such as XGBoost, with older and more classic algorithms, such as KNN. At this point, important questions arise, such as the relationship between classifier performance and the computational cost associated with training. Another hypothesis to explore is the influence of the Knowledge Discovery in Databases (KDD) process on the final performance of the classifiers, i.e., whether there is a significant difference between the classifiers when applied to a well-prepared and refined database.

2.5 Model Evaluation

To evaluate the results, we employed the 10-fold cross-validation technique. Following the completion of the entire battery of tests, we derived the mean and standard deviation of the Macro F1 metric to assess the performance of each algorithm.

Finally, we fine-tuned the hyperparameters for each algorithm through a tripartite experiment using 5-fold. Subsequently, we conducted a two-tailed t-test for each pair of algorithms to determine whether the models exhibited statistically significant differences or similarities at a specified confidence level.

3 Results

After fine-tuning the hyperparameters, the results obtained with each 10-fold cross-validation algorithm are summarized in Table 2. Initially, we observe good and similar performance between the XGBoost and KNN algorithms, which will be discussed later with the statistical test. The low standard deviation is also a good indication of the model's performance. This characteristic could signify good uniformity in the base and a low presence of *outliers*.

Table 2. Macro F1 Scores achieved by the algorithms

Algorithm	Mean	Standard Deviation
Logistic Regression	73.39%	0.39%
XGBoost	94,76%	0,06%
KNN	92,29%	0,08%

To ascertain the superior performing algorithm, a two-tailed t-test was conducted with a significance level set at $\alpha = 0,05$, indicating a 5% probability of erroneously inferring a difference between the algorithms when such disparity does not exist. Following the execution of this statistical test across the algorithm pairs, it was discerned that all algorithms exhibited statistically significant differences. Consequently, XGBoost emerged as the optimal classifier, demonstrating the highest score in the Macro F1 metric.

4 Conclusion and Future Work

In this study, the initial objectives were successfully achieved, with the extraction of characteristics capable of defining the nature of each URL and obtaining satisfactory results. In addition, the metrics evaluated, analyzing only the lexical scope of the URL, are promising, showing comparable or slightly lower performance than related works that use characteristics related to the network and the content of URLs, which require direct access to them.

In conclusion, the methodology employed throughout the development of this work surpasses many of the cited related studies in terms of robustness and reliability. This superiority stems from the consistent application of cross-validation techniques for model evaluation, feature selection, and hyperparameter refinement, alongside the confident utilization of statistical tests to ascertain the optimal performing models for URL classification.

Future research endeavors could entail a more thorough investigation into the distinguishing characteristics of URLs, aiming to elucidate more precise demarcations between various malicious URLs. As depicted in Fig. 3, it becomes evident that additional pertinent features are requisite to effectively differentiate instances belonging to the malware class from other URL categories. The disparity in false negatives and positives surpasses any other URL classification, underscoring the critical need for refinement. Notably, false negatives in this classification possess significant potential for detrimental consequences, as they erroneously portray malware-infected URLs as safe, posing substantial security risks.

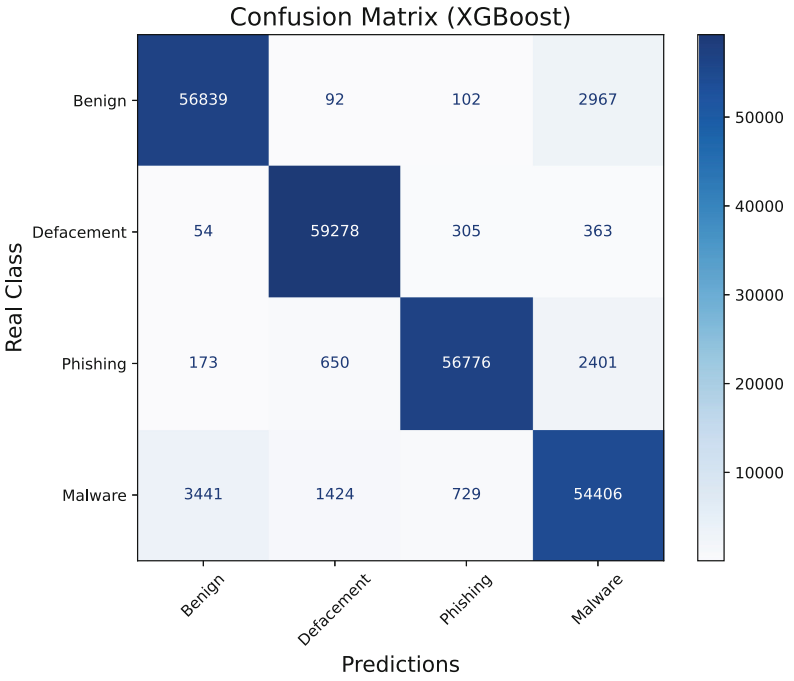

Fig. 3. False negative analysis

Another avenue for exploration could involve investigating the levels of correlation among different characteristics. Figure 4 illustrates a notable correlation between features. For instance, we can observe a conspicuous example between attributes corresponding to the count of directories and the count of "/" symbols within URLs. Given their direct proportional relationship, researchers could consider strategies for feature consolidation or similar approaches.

Lastly, contemplating the reduction of database volume could prove beneficial. That entails scrutinizing and eliminating redundant URLs from the database, specifically those that contribute minimal informative value to the classification task. Implementing the instance selection technique holds promise for substantially diminishing data volume while preserving model efficacy and curtailing the computational overhead associated with training.

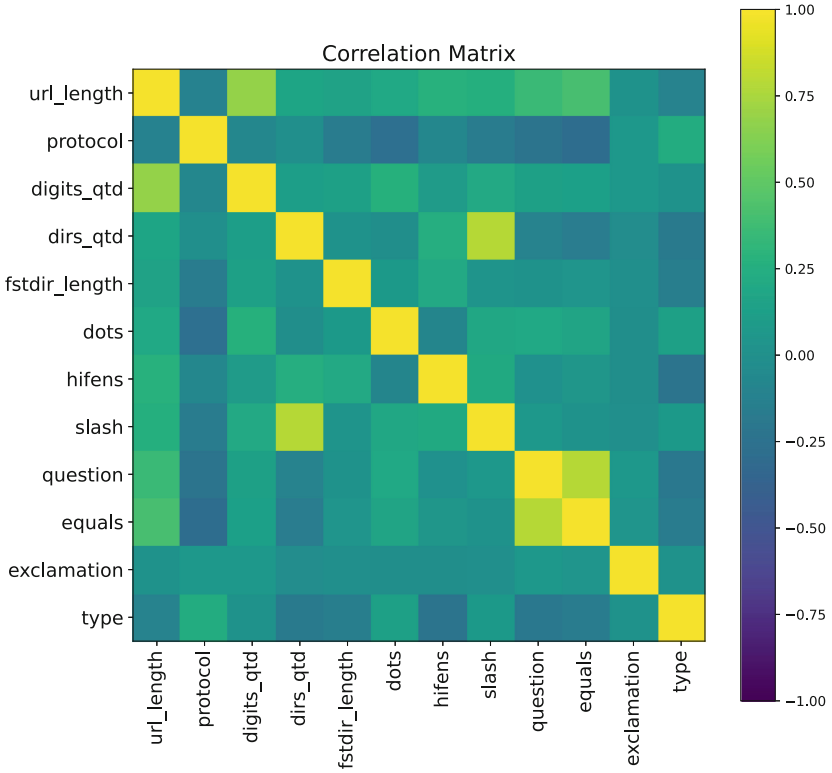

Fig. 4. Levels of correlation between characteristics

Acknowledgements. This study received partial financial support from AWS, CNPq, CAPES, FINEP, and Fapemig.

References

1. Bowyer, K.W., Chawla, N.V., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. CoRR **abs/1106.1813** (2011). http://arxiv.org/abs/1106.1813
2. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. KDD '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939785, https://doi.org/10.1145/2939672.2939785
3. Fix, E., Hodges, J.: Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties. USAF School of Aviation Medicine (1951). https://books.google.com.br/books?id=4XwytAEACAAJ