WhatAPhish: Detecting Phishing Website

Himanshi Mathur, Vanshika Goel, Vibhu Agrawal
Department of Computer Science
Indraprastha Institute of Information Technology, Delhi
{himanshi18037, vanshika18202, vibhu18116}@iiitd.ac.in

Abstract

Website Phishing costs internet users billions of dollars per year. Phishers steal personal information and financial account details such as usernames and passwords, leaving users vulnerable in the online space. Traditionally, the ad-hoc methods have been used to detect phishing attacks based on content, URL of the webpage, etc. It is difficult to detect phishing websites because of the use of URL obfuscation to shorten the URL, link redirections and manipulating link in such a way that it looks trustable. We present a machine learning based approach combining several features to attain the best results. Among various classification models, Random Forest Classifier and Support Vector Machine were able to classify most accurately with least number of false positives. After analysing, we used SVM which achieved an accuracy of 96.29% on test data and 97.95% on training data with a set of 25 features spanning over broad categories of URL parsing, pages' source code and URLs embedded therein and domain based statistics. We then created a web-platform WhatAPhish to deploy the obtained results. The code and the trained models are available at https://github.com/himanshi18037/WhatAPhish.

1. Introduction

The COVID-19 pandemic has boosted the use of technology in every sector, resulting in shifting of major activities like organising official meetings, attending classes, shopping, payments, etc. from physical to online space. This gives rise to more opportunities for phishers to carry out attacks impacting the victim financially, psychologically and professionally. Phishing refers to luring techniques used by identity thieves to fish for personal information in a pond of unsuspecting internet users. Phishers steal details like usernames, passwords, account numbers, credit card details, etc.

In 2013, just 450 thousand phishing attacks led to financial losses of more than 5.9 billion dollars [2]. The number of users in online space is increasing exponentially day by day thus making phishing an even more serious concern. As per CheckPoint Research Security Report 2018, 77% of IT professionals feel their security teams are unprepared for today's cybersecurity challenge. The same report indicates that 64% of organizations have experienced a phishing attack in the past year. These problems necessitate the need for moving beyond traditional approaches to a more sophisticated machine learning based approach.

Detecting phishing URLs is not so easy especially because of the dynamic web structure where the content on most websites is loaded dynamically, personalised for the user. Also, most phishing URLs are shortened using services like bit.ly, tinyurl.com, ow.ly, etc. which hides the target phishing URL. So, we aimed to identify features based on URL, static structure of the webpage, and the URL reputation to detect a potential phishing attack. Additionally, we built an end user solution for the web users to provide a safe and reliable online space.

We explore some of the recent works done in the domain, followed by describing the dataset used, methodology followed and WhatAPhish architecture. We conclude by stating the directions for future work and results obtained.

2. Literature Review

There are primarily three modes of phishing detection: [1]

1. **Content-Based Approach:** Analyses text-based content of a page using copyright, null footer links, zero links of the body HTML, links with maximum frequency domains. Using only pure TF-IDF algorithm, 97% of phishing websites can be detected with 6% false positives.
2. **URL Based Approach:** Uses page rank and combines it with other metrics derived from URL based on apriori knowledge. This method can detect upto 97% phishing websites.

3. **Machine Learning Approach:** Uses different machine learning models trained over features like if URL contains @, if it has double slash redirecting, pagerank of the URL, number of external links embedded on the webpage, etc. This approach could get upto 92% true positive rate and 0.4% false positive rate.

Mohammad et al [3] proposed a dataset based on 30 features, with each feature being categorical and classified as legitimate, suspicious or phishing. Each datapoint is then classified as legitimate or phishing. The dataset has 11055 datapoints with 6157 legitimate URLs and 4898 phishing URLs. The dataset was accompanied by a set of rules to categorise features for any new URL.

M E Pratiwi et al used neural network perceptron on data provided by UCI Machine Learning [4] and were able to achieve an accuracy of 83.38% by using 18 features [5]. A. Alswailem et al used random features extracted from URL and page content out of 36 available features and achieved a minimum accuracy of 53% and maximum 98% by Random Forest Classifier. [1].

3. Dataset

3.1. Dataset Description

We used the dataset provided by UCI Machine Learning repository [4] collated by Mohammad et al. The 30 features can further be subdivided into three categories:

3.1.1 URL and derived features

These features are included in the address of the website. Phishers generally adopt following methods to attack:

1. Long URL: Suspicious or phishing domains are hidden under long URL
2. Providing IP instead of URL: The IP address are not well recognised by general public and hence phishing URLs can easily be spoofed.
3. Using shortened URLs: Shortened URLs always redirect and hence seem unsuspicious to naked eyes.
4. "@" symbol in URL: Web browsers ignore anything preceeding "@" symbol, hence the phishing part can follow the "@" symbol.
5. URLs with "//": "//" is used to redirect the URL, hence can lead to landing up on a phishing site post redirections.
6. URL with "-": Legitimate websites rarely use "-". However, phishing websites use "-" in URLs to mimic the names of legitimate websites.
7. Number of subdomain: Legitimate websites generally use no or only one sub-domain, however, phishers generally redirect via multiple sub-domains.
8. Use of HTTPs security: Websites on HTTPS are generally secure and have a valid certificate issued by a trusted authority. However, phishing websites generally operate over unprotected HTTP layer or do not have a valid HTTPS certificate.
9. Period for which Domain has been registered: Legitimate websites usually operate over several years. Most phishing websites operate for a short period of time and do not have domain registered for more than one year.
10. Favicon: Website favicons are used to relate identity to URL, and generally load from the same domain. If it is being loaded from an external website then, it can be an attempt to spoof the identity of URL and a possible attack.
11. Ports: All websites running over HTTP use port 80 and running over HTTPS use port 443. The other ports should remain closed for security reasons.
    However, when we inspected some popular websites like www.google.com, www.linkedin.com, www.yahoo.com, etc. we found most of them have their FTP (21), SSH (22) and other non-standard ports open. Hence we decided to drop this feature.
12. Use of "https" in domain part: Phishers can use "https" in domain part to trick users into believing that the URL passes through secure "HTTPS" protocol.

3.1.2 Page's souce code based features

A common trick employed by phishers is to make the interface textually and graphically very similar to a legitimate webpage. However, these features are difficult to tap especially when the content is loaded dynamically. However, there are a number of distinguishing features between legitimate and phishing websites based on code structure of the webpage:

Based on URLs embedded in webpage

The URLs being accessed/accessible by the webpage generally carry a good amount of information about their nature. If the links belong to the website itself, it increases the credibility of the website. Few features identified on the basis of the embedded URLs are:

1. Embedded objects' URLs: Legitimate pages share their domains with the objects embedded in them. However, phishing websites usually load embedded objects from external resources to resemble them.

2. URL of Anchor tag: Anchor tag in HTML is used for hyperlinking. A legitimate website will never have a void source in anchor tag. However, phishers can use it to discard useless information for them and to redirect personal information to alternate sources.

3. Tags: Legitimate pages have the domain name for the page and domain name of URLs in it's <Meta>, <Script> and <Link> tags as same. However, they usually differe for suspicious websites.

4. Server Form Handler (SFH): Legitimate websites always take action on the content submitted via a form. However, if the form handler is void, or of a different domain than the actual website, then the chances of being phishing are higher.

5. Submitting information to a mail: Legitimate websites generally process the information submitted either on frontend itself or on some backend. However, a phisher might redirect the information to his personal mail.

6. Abnormal URL: The host name is usually included in the URL of all the objects on webpage. However, since this feature was already well explored by other factors above, we decided to drop it.

HTML and Javascript based features

HTML and Javascript can be used to hide malicious code in a seemingly well-behaved website. Some of the features recognised are:

1. Number of times website redirects: Legitimate websites usually redirect only once, whereas phishing redirects usually more than 4 times.

2. Status bar customisation: Phishers generally use Javascript to modify the URL of the webpage as seen in the address bar and it is mostly different from the actual URL.

3. Right-click disabled: Right click can be used to view the source code of a webpage. However, to eliminate this risk of being caught, phishers generally disable right click.

4. Pop-Up windows: If legitimate websites, use pop-up window, it is mostly for an alert purpose. However, phishing websites usually collect information through pop-up windows.
   However, it seemed infeasible to detect pop-up windows without actually visiting the webpage. Hence, we decided to drop this feature.

5. IFrame redirection: Phishers can overlap a webpage with an invisible frame and redirect to a different website/server using it.

3.1.3 Domain based features

The domains of legitimate websites are generally established for long duration and have good statistical properties. However, phishing websites usually live for shorter time periods and do not earn good indicators.

1. Age of the domain: Legitimate domains have a minimum age of 6 months. However, the phishing websites are shot-lived.

2. DNS: Legitimate sites are mostly recognised by publicly available WHOIS database, and have a non-empty DNS record. Phishing websites are mostly not recognised by WHOIS database.

3. Website Traffic: Legitimate domains have a lot of visitors, hence are ranked among the top 100,000 in Alexa database.However, if a website is not even recognised by Alexa, then it is deemed to be phishing.

4. Pagerank: Legitimate domains generally have a page rank between 0.2 and 1. Higher page rank implies that it is an important domain.

5. Google Index: Legitimate websites are usually indexed by Google. Phishing websites generally, being short-lived do not make it to the index.
   However, we did not find any publicly available API to use for determining if a website has Google index and thus we dropped this feature.

6. Numbers of links pointing to a page: Legitimate websites generally have a lot of external links pointing to them.
   But, no reliable source was found from which this information can be extracted. So, we dropped this feature.

7. Statistical Report based: A few publicly available databases like PhishTank are maintained and periodically updated to identify phishing websites. If a website is found in this database labelled as phishing, the probability of being actually phishing is very high.

3.2. Preprocessing

3.2.1 Feature Selection

We analysed each feature and studied how they can be extracted for a new URL. We studied the relevance of the features in the present context and decided if the feature was still relevant. We dropped 5 features out of 30, owing to feature drift. These were Port Number, Abnormal URL, Pop-up Window, Google Index and Number of Links Pointing to a Page. We did not reduce features any further as binary classification models usually tend to overfit the training data.

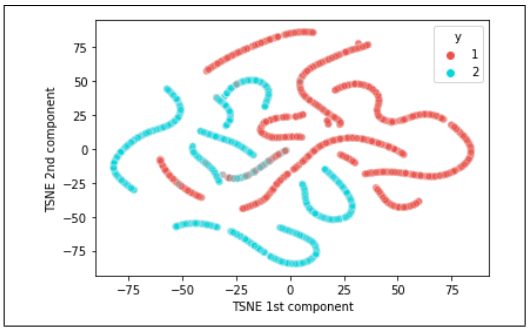

Figure 1. t-SNE plot of the training data. 1 represents legitimate URLs and 2 shows phishing URLs

3.2.2 Visualising the Dataset

To see separability of the phishing class from the legitimate class URLs, we plotted the t-SNE curve as shown in Figure 1. The curve implied that though the classes are separable, they are not clustered together, and either transformation of the features or non-linear model is required to obtain good results.

3.2.3 Encoding the dataset

The dataset used a {-1, 0, 1} for {'phishing', 'suspicious', 'legitimate'}. However, most models require the values to be positive. Hence we relabelled the three as {2, 0, 1} respectively. Also, a few models interpret the labels as numbers. Hence, we one-hot encoded the feature labels as well.

3.2.4 Splitting the dataset

We splitted the available data into training and testing data using 80:20 split. Post that, since we had only 7075 data points in the training data, we trained it using 5 fold cross validation. Hence, we achieved a train:val:test split of 64:16:20.

4. Methodology

4.1. Model Details

After splitting the dataset, we shortlisted the different models for the classification of URLs. We tried to find the best hyperparameters using GridSearchCV. The different models deployed and parameters tuned beyond the default parameters are:

1. *Logistic Regression:* It is a statistical model that uses a logistic function to classify the data points. We fixed the maximum number of iterations to 2500.

2. *Categorical Naive Bayes:* It is a probabilistic model that assumes the features to be independent of each other.

3. *Decision Tree:* It creates a classification model that learns by creating decision boundaries. The maximum depth used is 17.

4. *Random Forest Classifier:* It is an ensemble classification model which takes the average of results from multiple decision trees and optimally predicts. The number of estimators taken is 200.

5. *K-Nearest Neighbours:* KNN calculates the nearest k neighbours for each data point and returns the majority label among them. The hyperparameters used are n_neighbours as 3 and the metric for distance evaluation is 'euclidean' distance.

6. *Support Vector Machine:* SVM classifies the given labelled training data by creating an optimal hyperplane for classification. The hyperparameters used are kernel as rbf, gamma value as 0.1 and C = 10

7. *XGBoost:* XgBoost is based on decision trees that use a gradient boosting framework for classification. The hyperparameters are: silent = False, scale_pos_weight = 1, learning_rate = 0.01, colsample_bytree = 0.4, subsample = 0.8, objective = 'binary:logistic', n_estimators = 1000, reg_alpha = 0.3, max_depth = 4, gamma = 10

4.2. Evaluation Metrics

For testing the results obtained, we used 3 parameters: Accuracy, Recall and False Positive Rate (FPR).

1. *Accuracy:* It is the ratio of number of correct predictions to the total number of input samples. Since the objective requires most of the URLs to be classified correctly, hence high accuracy is one of the metrics.

2. *Recall:* It is the ratio of number of true positives to the total number of predicted positives. As we want the websites predicted as positive, to be legitimate only, high recall is desired.

3. *False Positive Rate (FPR):* It is the ratio of number of samples incorrectly identified as positive to to total number of actually negative samples. The requirement is to minimise the number of phishing websites identified as legitimate as it can lead to heavy losses for the person visiting the website. Thus low FPR is one of the metrics used.

5. Results and Analysis

The models trained and their performances on the validation data are present in Table 1. Since we used 5-fold method, we averaged the results obtained across each fold. We have evaluated the accuracy, recall and FPR.

Classification Model	Accuracy	Recall	FPR
Logistic Regression	89.92	92.17	12.91
Categorical Naive Bayes	92.53	94.41	9.83
Decision Tree	95.29	96.2	5.84
Random Forest	96.25	96.95	4.61
K-Nearest Neighbours	95.17	95.97	5.82
Support Vector Machine	96.11	96.78	4.73
XGboost	93.79	94.69	7.34

Table 1. Classification Models Results (in percentage)

We aim to increase accuracy, increase recall and decrease false positive rate so that most of the points are classified correctly and the number of phishing websites labelled as legitimate is reduced.

Random Forest Classifier and Support Vector Machine perform similar on the given dataset and have higher accuracy and lower FPR compared to other models. Apart from Logistic Regression and Categorical Naive Bayes, the performance of other models is also comparable.

Logistic Regression assumes linearly separable classes. However, as can be seen from the t-SNE plot, classes are separable, but not in a linear fashion. Hence logistic classifier fails to perform well. Naive Bayes assumes independence of features which may not be entirely true. In this case features like pagerank and statistical report are closely associated, similarly, Request URL and URL of Anchor also seem to be tightly coupled. This coupling occurs because most phishing websites tend to follow similar patterns in their static features. Thus Naive Bayes could not capture the patterns well.

Decision tree is able to classify non-linearly separable patterns, and hence performs well on the given data after pruning, but as Random Forest is an ensemble model using Decision Trees which combines the output of various Decision Trees in a random fashion, it provides an improvement over decision trees as reflected in the results. K-nearest neighbours works on similarity of features. A lot of the phishing websites have some common features such as less than six months of activity, less than one year of domain registration length, etc. Hence KNN can classify well based on these features. However, with 'euclidean' distance, non-convex patterns may not be classified well, thus impacting performance slightly.

Support Vector Machine works well for linearly separable data. The data is not linearly separable directly, but after applying 'rbf' kernel, the data becomes separable and SVM is able to learn well from the data. XGBoost is an ensemble model based on decision tree which uses gradient boosting to reduce errors. Since it uses techniques similar to gradient descent, it is not very well suited for categorical data and even after one-hot encoding does not perform well.

Dataset	Accuracy	Recall	FPR
Training	97.95	97.86	1.93
Testing	96.29	96.42	3.87

Table 2. Value of chosen metrics for SVM on entire training and testing data (in percentage)

We did not use Neural Networks due to limited amount of data available. Neural Networks being so powerful have a high tendency to overfit very easily. Thus, with 11,055 data points including training, validation and testing data, we refrained from using neural networks like ANN and CNN.

So, we decided to use SVM as the final model. It was performing well with high accuracy, high recall and low FPR. The problem at hand is binary classification and SVM is a comparatively simpler model with tolerance to outliers. Additionally it offers explainability for the involved features.

The metrics obtained for SVM over the complete training and testing data are indicated in Table 2.

6. WhatAPhish: Design and Implementation

WhatAPhish is a web-based platform that accepts an URL as input and predicts its status as 'Phishing' or 'Legitimate'. The web platform has been developed using flask.

An input URL is expected to be complete, i.e., with 'http' or 'https' and any following characters required to land up on the destined page. We used various tools to extract different features on the fly. We used curl and BeautifulSoup to scrape the webpage and its headers, tldextract to obtain domain name from a given URL, string parsing to identfy features extractable from the URL's text itself, WHOIS database to verfiy authenticity of the link. Apart from that we found pagerank of the URL from Open Page Rank, page popularity using Alexa database and the perceived reputation of the website using PhishTank API.

After extracting these features, we re-encoded them using the trained OneHotEncoder and predict its legitimacy using the trained SVM. These results are served back to the user on the web platform. The screens associated with the platform are shown in Figure 2, Figure 3 and Figure 4.

Figure 2. WhatAPhish: User-Interface

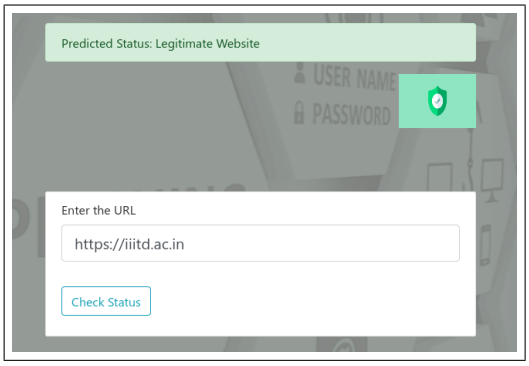

Figure 3. WhatAPhish: A Legitimate URL Detection

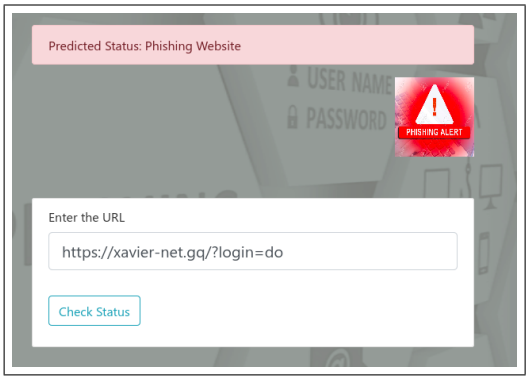

Figure 4. WhatAPhish: A Phishing URL Detection

7. Future Work

Browser Extension: The platform can be converted into a browser extension. Phishing websites usually inflict losses to the users by acting as clickbaits. So, a browser extension can help prevent accidental land ups on these websites, by checking every URL which the browser tries to open, before actually allowing the user to land up on the page.

Caching Results in Database: Currently, for all queries, API hits and web scrapping is done everytime a URL is provided as input. However, if some of these features and results can be cached in a database, the queries can be performed much more quickly and efficiently. So, if a seen URL is provided as input again, then the whole process of feature extraction and classification can be skipped, and results can be served by looking up the database. But, as the websites are prone to updates, the results can be cached only for a few days, say a fortnight, after which the entry becomes invalid and the whole process has to be followed for it again.

Parallel Feature Extraction: As of now, features for an input URL are extracted sequentially. However, if the computation power is available, then most of features can be extracted in parallel. The scrapping using BeautifulSoup and curl, WHOIS, Alexa and other database lookups and string parsing can be done parallely with work-stealing, thus bringing down the query time and increasing efficiency in the real world scenario.

8. Conclusion

In this project, we built WhatAPhish - a mechanism to detect phishing websites. Our methodology uses not just traditional URL based or content based rules but rather employs the machine learning technique to identify not so obvious patterns and relations in the data. We have used features from various domain spanning from URL to HTML tags of the webpage, from embedded URLs to favicon, and databases like WHOIS, Alexa, Pagerank, etc. to check the traffic and status of the website. We were able to obtain an accuracy of more than 96%, recall greater than 96% with a False Positive Rate of less than 5%, thus classifying most websites correctly and proving the effectiveness of the machine learning based technique to attack the problem of phishing websites. We provided the output as a user-friendly web platform whch can futher be extended to a browser extension to provide safe and healthy online space to the users.

References

[1] A. Alswailem, B. Alabdullah, N. Alrumayh, and A. Alsedrani. Detecting phishing websites using machine learning. In *2019 2nd International Conference on Computer Applications Information Security (ICCAIS)*, pages 1–6, 2019.

[2] Ankit Jain and B B Gupta. Phishing detection: Analysis of visual similarity based approaches. *Security and Communication Networks*, 2017:1–20, 01 2017.

[3] R. M. Mohammad, F. Thabtah, and L. McCluskey. An assessment of features related to phishing websites using an automated technique. In *2012 International Conference for Internet Technology and Secured Transactions*, pages 492–497, 2012.

[4] R. M. Mohammad, F. Thabtah, and L. McCluskey. UCI machine learning repository, 2012.

[5] M E Pratiwi, T A Lorosae, and F W Wibowo. Phishing site detection analysis using artificial neural network. *Journal of Physics: Conference Series*, 1140:012048, dec 2018.