Detecting Phishing Websites with Random Forest

Shinelle Hutchinson(✉), Zhaohe Zhang(✉), and Qingzhong Liu(✉)

Sam Houston State University, Huntsville, TX, USA
{sdh053, zxz003, liu}@shsu.edu

Abstract. Phishing has been a widespread issue for many years, claiming countless victims, some of which have not even realized that they fell prey. The sole purpose of phishing is to obtain sensitive information from its victims. There have yet to be a consensus on the best way to detect phishing. In this paper, we analyze web-based phishing detection by using Random Forest. Some important URL features are identified and our study shows that the detection performance with feature selection is improved.

Keywords: Phishing · Random forest · Classification · Website Detection

1 Introduction

Phishing is a cyberattack rooted in scare tactics, with the sole purpose of eliciting personally identifiable information (PII) from its victims. An attacker disseminates a fraudulent version of a legitimate website, usually via email, telephone or text messages [1], in the hope that the victim would believe the claims made in the email. A successful phishing attack can result in an attacker obtaining credit card details and login information.

With the ever-increasing number of Internet users, comes even more data that needs to be protected. This data includes login information and credit card information, both of which are priceless to cyber criminals. These cybercriminals would try anything to gain this information. One such way that has been overutilized is via phishing websites.

Phishing can occur in three forms: web-based phishing where a website is duplicated to resemble a trusted website and tricks users into submitting sensitive information [2]. Email-based phishing, where an attacker sends email to countless users claiming some account issue, in hope some of them fall for it. Email phishing usually involves web-based phishing as well [2]. Malware-based phishing where malicious code is injected into a legitimate website and when the user visits that site, the malicious soft-ware is installed on the user's system [2].

Our paper focuses on web-based phishing detection and aims to identify the most relevant subset of features that can accurately identify phishing URLs, using the Random Forests classifier.

S. Hutchinson and Z. Zhang—equal contribution.

2 Related Work

Recent research in phishing detection saw the use of sophisticated approaches that all prove worthwhile. We've studied four such approaches in an attempt to better the current detection rate of phishing URLs.

One method proposed for the detection of phishing websites was the use of Public Key Certificates. In an article by Dong et al. [2], they highlighted a method of using the structure of the X.509 certificates to confirm phishing websites. Their method used machine-learning to build a phishing detection system. This system contained a Certificate Downloader, Feature Extractor, Classification Executor and a Decision Maker. A total of 42 features were selected based on the structure and content of the X.509 certificate. Decisions were made based on the output of six classification algorithms including Naïve Bayes Tree and Random Forests. This model was an improved alternative to traditional detection approaches, like blacklisting, as it could be run by end users in real time. It also did not require frequent updates from a central server, which eliminated any vulnerabilities users may have faced if they were using blacklists. However, this model was susceptible to sophisticated learning attacks such as evasion and poisoning.

Rao and Ali [3], developed a desktop application called PhishShield, which was able to detect phishing URLs based on heuristics and whitelists. Decisions are made based on Footer links with NULL values, Zero links in the body of the HTML, Copyright Content, Title Content and Website Identity. PhishShield prides itself on being able to detect phishing sites that employ Image Based Phishing Attacks. Rao and Ali also claim that a heuristic approach resulted in fewer false positives and false negatives. Unfortunately, PhishShield fails when its parse, JSoup, fails and also when all the filters are bypassed.

Rao and Pais [4] proposed a phishing detection approach that used automation of human behavior. Their model would submit fake credentials to the suspected phishing website in an attempt to determine if the site was indeed phishing. This approach was able to detect phishing websites that used captcha verification on its login page and those that contained embedded HTML text. One advantage over the other approaches discussed so far is that this method does not require any data to be trained. However, this approach can only detect phishing websites that use login pages.

Kumar et al. [5] proposed a datamining approach to extract URL links from emails. The proposed algorithm had two phases to detect the phishing website: phase one extracted all URLs from an email, the application (DC scanner) was used to verify the domain authority and other information that might be hidden under the HTML code; phase two checked that the web page sent the contents of a form to a web server. The application verified that the URL links in the web page was the same as in the domain. If the URLs pass all the checks in phases one and two, the email/link was deemed as non-phishing website. Unfortunately, this design was unable to detect malware-based phishing attacks.

Our approach would aim to improve on a machine learning take on phishing detection in hopes of achieving higher accuracy than previously attainable.

3 Methodology

The aim of phishing websites is to trick users into submitting their private information like login credentials or credit card information. Some of these websites are easily recognizable as illegitimate but others require deeper analysis. One method of conducting a deeper analysis of websites is to look at their URLs.

In order to detect phishing websites with high precision and recall, we carefully select specific features that represent each URL. These features are used in training and testing inputs. We divide this input into four different arrays: training input, training output, testing input and testing output. We then use the training input array to train our Random Forest classifier. Once the classifier is trained to accept the specific features and make the appropriate decision, we used the testing input to test the classifier. This process [9] is summarized in Fig. 1. These predictions are then evaluated based on accuracy, precision, recall and F-score.

We've recorded these four scores for five different subsets of features and compared them. Our goal was to find the subset of features that give the highest precision and recall rates.

3.1 Data

We collected our training and test data from the UCI phishing dataset [6] that is publicly available. This dataset contained 2456 unique URL instances and a total of 11,055 URLs of which 6,157 are phishing and 4,898 are legitimate sites. Each row represented a URL and each URL was previously parsed and represented according to 30 features which could determine whether or not the URL is used for phishing or just identify a specific feature as suspicious for a particular URL. The features considered include whether or not an IP Address is used instead of a URL, the length of the URL, the presence of link tags to the same domain as the webpage and whether or not the webpage uses IFrames. At the end of each row, there is a result which identifies the true nature of the URL, 1 if it is a phishing site and −1 if it a legitimate site.

3.2 Feature Selection

In any classification scheme, there are features which seem more prominent than others in achieving a correct classification. These features, combined with other salient features or even less salient features, can perform outstandingly. The difficulty arises when we must determine what are the most relevant features from a set and what combination of features give us near perfect classification accuracies. From the 30 features, we identified five subsets. These were grouped as shown below.

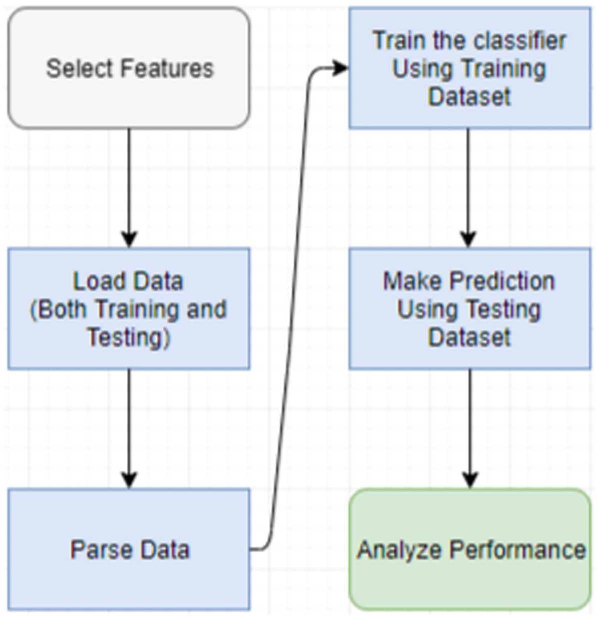
Select Features
Load Data (Both Training and Testing)
Parse Data
Train the classifier Using Training Dataset
Make Prediction Using Testing Dataset
Analyze Performance

Fig. 1. Process flowchart for each selected feature set.

Set A: Web-presence related features. We chose these four features to determine if it is practical to determine the nature of a URL simply by looking at its presence on the internet, and not by any structural features of the URL itself. These features included: Domain age, Website traffic, Page Rank, Google Index.

Set B: Features with only two (2) possible outcomes {−1,1}. This makes the data more binary and eliminates the possibility of uncertain URLs. These features included: having_IP_Address, Shortining_Service, having_At_Symbol, double_slash_redirecting, Domain_registeration_length, Favicon, port, Prefix_Suffix, HTTPS_token, Request_URL, Submitting_to_email, Abnormal_URL, Redirect, on_mouseover, RightClick, popUpWidnow, Iframe, age_of_domain, DNSRecord, Page_Rank, Google_Index, Statistical_report.

Set C: Features with three (3) possible outcomes {−1,0,1}. These features allow for uncertainty in their output and includes: URL_Length, having_Sub_Domain, SSLfinal_State, URL_of_Anchor, Links_in_tags, SFH, web_traffic, Links_pointing_to_page.

Set D: Combination of Sets B and C. This set contains all 30 features.

Set E: The most important features (those rated 0.01 and up) according to Ran-dom Forest's feature_importances_ attribute. These include: having_IP_Address, URL_Length, Prefix_Suffix, having_Sub_Domain, SSLfinal_State, Do-main_registeration_length, HTTPS_token, Request_URL,URL_of_Anchor, Links_in_tags, SFH, age_of_domain, DNSRecord, web_traffic, Page_Rank, Google_Index, Links_pointing_to_page.

3.3 Experiment Design

Random Forest is a supervised classification algorithm that makes use of several classification trees [7]. A classification is made by passing each input vector down each tree, randomly. Each tree gives a classification, or vote, and the forest chooses the classification with the most instances, or votes [8]. We decided to use this algorithm because it is unexcelled in accuracy among its counterparts [8], it runs efficiently on large datasets and it can handle missing values [7].

3.4 Evaluation Methods

To fully evaluate the effectiveness of a classification model, you must include its precision and recall scores. In order to evaluate the performance of our classifications, we've calculated the Accuracy, Precision, Recall and F1 Scores for each set tested. Each metric is calculated based on True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) scores.

Precision measures the number of instances that have been correctly classified and is a measure of the classifier's exactness. It is the number of positive predictions divided by the total number of positive instances predicted. For us, precision answers the question, "Of all the URLs labeled as phishing, how many are actually phishing?" The formula to calculate precision is given by Eq. 1 below.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Recall measures the number of positive instances that the classifier correctly identified from the set of all positive instances. In other words, recall measures the number of instances that were missed [1]. Recall is a measure of the classifier's completeness. For us, recall answers the question, "Of all the URLs that are truly phishing, how many did we identify as phishing?" The formula to calculate recall is given by Eq. 2 below.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Simply looking at a classifier's precision and recall scores would not constitute a substantial evaluation. As such, we include the F1 score, which is the weighted average of precision and recall scores. The formula to calculate F1 Score is given by Eq. 3 below.

$$F1Score = \frac{2TP}{2TP + FP + FN} \tag{3}$$

Accuracy, simply put, is the number of correct classifications made out of all instances in the test data. The formula to calculate accuracy is given by Eq. 4 below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

The Receiver Operating Characteristic (ROC) curve is ideal for representing binary classifications like this one. The curve is plotted with the False Positive Rate (FPR) on the x-axis and the True Positive Rate (TPR) on the y-axis. The Area Under the ROC Curve (AUC) shows how well the classifier is able to discriminate between phishing and legitimate URLs. The formulas to calculate FPR and TPR are given by Eqs. 5 and 6 below.

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

$$TPR = \frac{TP}{TP + FN} \tag{6}$$

3.5 Results

In this section we provide details on the performance of our classification model. As previously mentioned, thirty (30) initial features were broken down into five subsets and investigated. The overall performance of these subsets is reported in Table 1.

Table 1. Classifier performance

	Precision	Recall	F1 Score	Accuracy	FPR	TPR	AUC
Set A	0.4283	0.5119	0.4664	48.8%	0.488	0.4669	0.4908
Set B	0.7148	0.783	0.7474	76.9%	0.2169	0.7575	0.7703
Set C	0.9213	0.9154	0.9183	92.9%	0.0845	0.9393	0.9273
Set D	0.9559	0.9414	0.9486	95.5%	0.0585	0.9663	0.9538
Set E	0.9711	0.9479	0.9593	96.5%	0.052	0.9781	0.963

Sets C, D and E were able to achieve great precision and recall rates, above 91.5%. Set E outperformed the other sets, achieving 97.1% precision and 94.8% recall. This set produced results that surpass some previous research [1] and closely follow others [3]. High precision and recall rates are directly related to high quality feature selection. Set A underperformed with 48.8% accuracy and 51.2% precision, while Set B produced average results, obtaining 76.9% accuracy and 71.5% precision. The increasing performance of each set can be easily seen in Fig. 2.

Each set contained varying amounts of features. The relation between feature count and performance is shown in Fig. 3. We can observe one profound characteristic: the number of features is not as important as the importance of the features within the set. Set B contained 21 features to obtain 76.9% accuracy while Sets C and E used 8 and 16 features respectively and achieved over 92% accuracy.

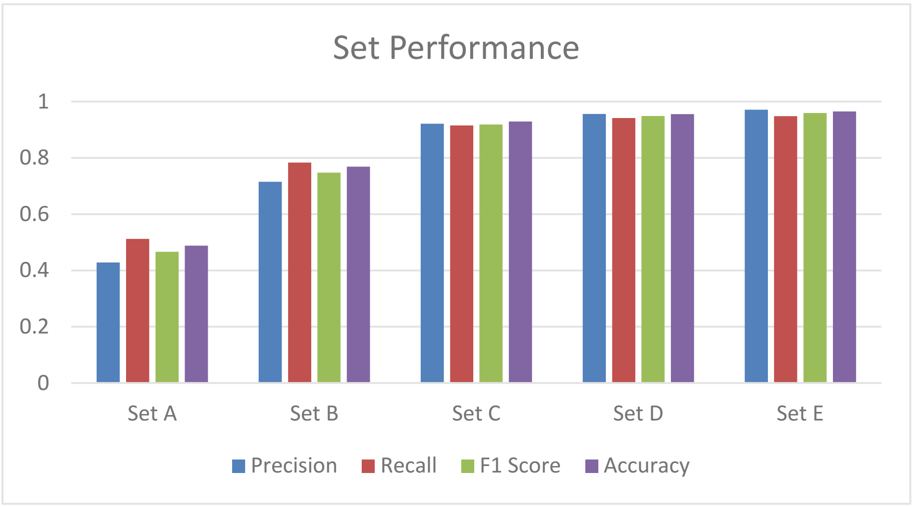

Fig. 2. Overall performance of each set.

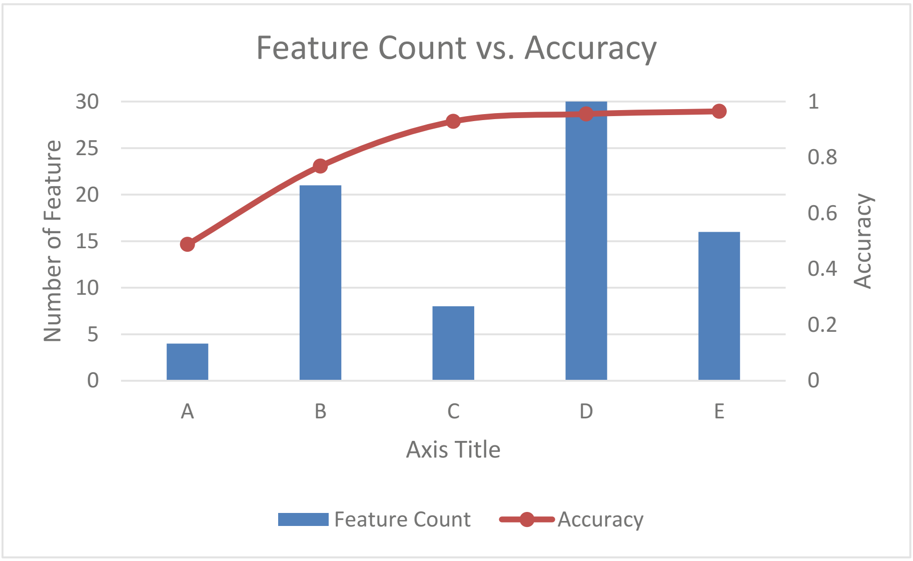

Fig. 3. Accuracy of using different numbers of features.

The ROC Curve is useful in showing the relationship between False Positive Rate and True Positive Rate. The ideal case occurs when the curve has the shortest distance to the upper left corner of the graph. The AUC is an important indicator of classifier performance. We see the Random Forest classifier achieves the best AUC for Set E, with 0.963. This is depicted in Fig. 4.

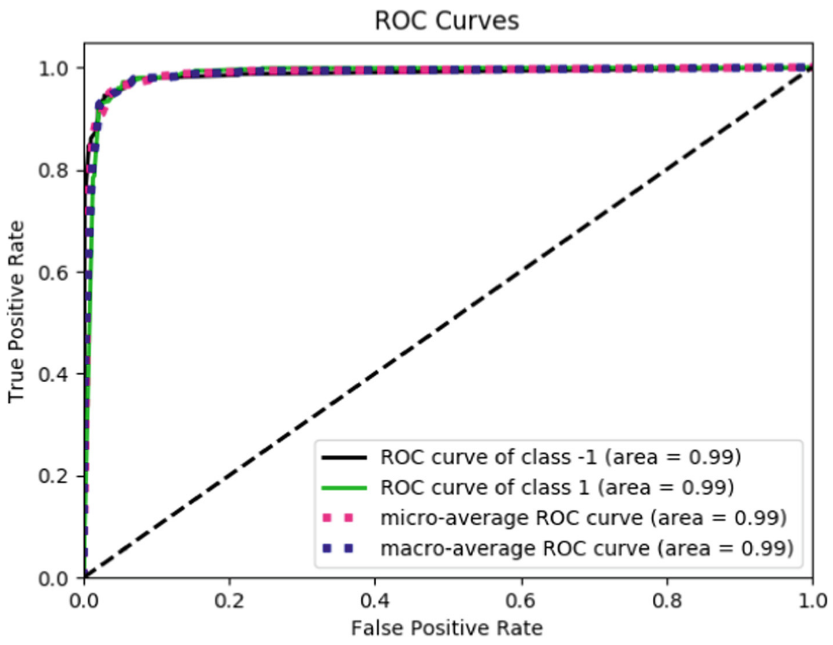

Fig. 4. ROC Curve and AUC for Random Forest classifier with Set E.

4 Discussion

In this section we further analyze the performance of Random Forest classifier with our five Sets and identify some limitations of our model.

Intuitively, increasing the number of features tend to yield better results on the performance. However, when we focused on selecting the features that were relatively important, we reduced the feature count by over half compared to the total number of features and were able to achieve better accuracy. This result indicates that adding more features does not necessarily result in better accuracy and may introduce more complexity that could downgrade the efficiency and performance of the classifier.

Our feature importance ranking was done using Random Forest's feature_importances_ attribute. This resulted in 16 features with a rating 0.01 or greater. These 16 features held the most weight and so was expected to perform extremely well. As evident by Set E's performance, using only the most important features are enough to increase the accuracy, precision and recall.

It was hoped that having binary input would make classification simpler for Random Forest. However, Set B's precision and recall scores, although good, were not desirable. In contrast, Set C obtained surprisingly better results although only containing features that could take on 3 possible outcomes. This means that some features had a value of zero, meaning that feature was suspicious, and not definitely phishing or legitimate.

Set A was chosen based on the URL's online presence rather than its structure. The performance of this set simply reiterates that we cannot determine, without a doubt, whether or not a website is phishing based solely on how long the site has been online, how many people visit the site and whether or not a user can visit the site from doing a Google search. These features are not substantial on their own.

We were restricted in the number and type of features under consideration. Had we been able to alter these parameters, we could have included current phishing URLs as reported by PhishTank. Another limitation of our method is that some of our subsets were built based on our judgement of feature importance, including Sets A, B and C.

When we divide the dataset into training and testing sets, one technique we could apply is k-fold cross validation. By partition our dataset into k-parts same sized sets, then running separate testing and training evaluations on those sets, we are confident that k-fold cross validation could significantly improve our assessment of the trained classifier.

5 Conclusion

We were able to improve the accuracy, precision and recall of the Random Forest classifier by using select features from all 30 features. This highlights the significance of first determining which features are most useful in determining whether a URL is phishing. Our findings give several possibilities for improvement. As a further step, we would include more features, find more important features, and combine all important features in hopes of achieving near perfect accuracy, precision and recall. We would also design our own URL parser, so as to be able to include more current phishing URLs. In order to improve the method of selecting important features for testing, we would employ forward feature selection and backward feature elimination methods.

References

1. Phishing—What Is Phishing?. Phishing.org (2018). http://www.phishing.org/what-is-phishing.)
2. Dong, Z., Kapadia, A., Blythe, J., Camp, L.J.: Beyond the lock icon: real-time detection of phishing websites using public key certificates. In: 2015 APWG Symposium on Electronic Crime Research (eCrime). IEEE (2015)
3. Rao, R., Ali, S.: PhishShield: a desktop application to detect phishing webpages through heuristic approach. Procedia Comput. Sci. **54**, 147–156 (2015)
4. Rao, R.S., Pais, A.R.: Detecting phishing websites using automation of human behavior. In: Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security - CPSS 17 (2017). https://doi.org/10.1145/3055186.3055188
5. Kumar, B., Kumar, P., Mundra, A., Kabra, S.: DC scanner: detecting phishing attack. In: 2015 Third International Conference on Image Information Processing (ICIIP) (2015). https://doi.org/10.1109/iciip.2015.7414779
6. Mohammad, R., McCluskey, L., Thabtah, F.: UCI machine learning repository: phishing websites data set. Archive.ics.uci.edu (2015). https://archive.ics.uci.edu/ml/datasets/phishing+websites

7. Gu, S., Wu, Q.: How Random Forest Algorithm Works in Machine Learning. Medium (2017). https://medium.com/@Synced/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674
8. Breiman, L., Cutler, A.: Random forests - classification description. Stat.berkeley.edu (2004). https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
9. Papernot, N.: npapernot/phishing-detection. GitHub (2016). https://github.com/npapernot/phishing-detection