



A Machine-Learning Based Unbiased Phishing Detection Approach ^a

Hossein Shirazi¹, Landon Zweigle¹ and Indrakshi Ray¹

¹*Department of Computer Science, Colorado State University, Fort Collins, CO, USA*
{shirazi, lzweigle, iray}@colostate.edu

Keywords: Phishing, Social Engineering, Fingerprint

Abstract: Phishing websites mimic a legitimate website to capture sensitive information of users. Machine learning is often used to detect phishing websites. In current machine-learning based approaches, the phishing and the genuine sites are classified into two groups based on some features. We feel that this is an inadequate modeling of the problem as the characteristics of different phishing websites may vary widely. Moreover, the current approaches are biased towards groups of over-represented samples. Most importantly, as new features are exploited, the training set must be updated to detect new phishing sites. There is a time lag between the evolution of new phishing sites and retraining of the model, which can be exploited by attackers. We provide an alternative approach that aims to solve the above-mentioned problems. Instead of finding commonalities among non-related genuine websites, we find similarity of a suspicious website to a legitimate target and use machine learning to decide whether the suspicious site is impersonating the target. We define the fingerprint of a legitimate website by using visual and textual characteristics against which a sample is compared to ascertain whether it is fake. We implemented our approach on 14 legitimate websites and tested against 1446 unique samples. Our model reported an accuracy of at least 98% and it is not biased towards any website. This is in contrast to the current machine learning models that may be biased towards groups of over-represented samples and lead to more false-negative errors for less popular websites.

1 INTRODUCTION


Phishing websites attempt to obtain sensitive personal information including usernames, passwords, social security and credit card numbers by mimicking legitimate websites. Machine learning is often used to detect phishing websites. Current approaches (Abdelhamid et al., 2014), (Mohammad et al., 2012), (Shirazi et al., 2018), (Tan, 2018), and (Shirazi et al., 2019) perform an in-depth analysis to find characteristics that are common across phishing websites but help distinguish them from genuine ones. These characteristics form the basis of features that are used by machine learning algorithms. This approach appears counter-intuitive as adversaries use different techniques to make a phishing website similar to a genuine target website, not other phishing instances. The choice of features and their representation often


depends on the skill of the model designer and also on the types of attacks that can be detected by the algorithm. Adversaries are always looking for alternate attack vectors to bypass current learning models and make existing features obsolete. As new attacks emerge, the current models must be upgraded.

(Zou and Schiebinger, 2018) discusses bias in machine learning algorithms and shows how some samples are over-represented, and others are under-represented. (Shirazi et al., 2018) refer to imbalanced dataset concerning the feature values in the context of phishing. Our observation here is if the phishing samples in a training dataset are biased towards the most targeted websites, the detection rate of phishing instances in the more popular sites would be higher than groups with fewer numbers.

We provide a more intuitive approach, which consists of finding similarity between a legitimate website that is targeted and all of the phishing websites that mimic the legitimate website. We define features which we can compare a phishing sample to a target website. This approach produces a novel solution that overcomes the problems associated with bypassing classifiers and dealing with biased datasets. It also

^aThis work is supported in part by funds from NSF Awards CNS 1650573, CNS 1822118 and funding from CableLabs, Furuno Electric Company, SecureNok, AFRL, NIST, and by funds from the State of Colorado sponsored Center for Cyber Security.

^b <https://orcid.org/0000-0002-2721-0628>

^c <https://orcid.org/0000-0002-7166-5267>

does not need to look for new attack vectors.

We propose the idea of fingerprinting legitimate website using its visual and textual characteristics. We also suggest using screenshots of websites instead of relying on the HTML code of websites; this makes bypassing the learning model extremely hard for the attacker. The fingerprint will be compared with the given samples to detect phishing instances.

Each machine learning vector will represent the similarity of a phishing website to a specific target, not similarity to other phishing instances. In this case, the machine learning algorithm will not answer the critical question of phishing detection as *"if the given website is phishing or not"* but it will answer *"if a given website is attacking a specific target or not"*. The learning algorithm in this model improves learning scores based on the similarity of a phishing instance to the target website and do not depend on other samples.

Thus, the model will not skew towards targets with more individuals, as each target has its own learning model and dataset. Accordingly, each site is being judged independently, and it guarantees there is not any bias toward groups of sites with a high volume of samples, and oversampling and undersampling cannot affect the learning scores.

The model is looking for the visual and textual similarity between a given suspicious website and a targeted genuine website. Thus, new attack vectors will not change the learning model, so there is no need to update the model over time unless the target website has been changed.

Our key contributions are stated below.

- Our approach overcomes the following limitations of the current approaches that use machine learning for phishing detection: (i) It makes current models vulnerable against new attack vectors; thus there is a need to update the learning models as new forms of attacks emerge. (ii) There is an imbalance of the phishing datasets used in training samples: they are biased towards frequently targeted websites in a phishing attack. (iii) The strengths of the model depend on the expertise of the designer. Our proposed approach overcomes these limitations.
- We define a new fingerprinting approach based on the visual and the textual traits of legitimate websites.
- Our algorithm is able to detect whether a given suspicious website attacks a specific target.
- We generated our target-based phishing dataset and evaluated it with five different classifiers. We also assessed the effectiveness of each fingerprint

section separately and show which combinations gives us the best results.

The rest of this paper is organized as follows. Section 2 describes the phishing detection algorithms available in the literature. Section 3 discusses our fingerprinting model. Section 4 explains the details of our implementation including how to extract features for fingerprint. Section 5 explains the results of our tests to prove our initial intuition and demonstrate the effectiveness of the model. Section 6 concludes the paper and discusses future work.

2 RELATED WORK

(Cui et al., 2017) monitored more than 19000 phishing attacks for ten months and found over 90% of attacks were a replication or variation of other attacks in the database. In a subsequent work (Cui et al., 2018) a more in-depth analysis was performed. (Shirazi et al., 2019) proposed an approach to simulate these attack replications by generating adversarial samples through direct feature manipulation. New phishing samples were generated by this approach which were able to bypass detection mechanisms. In this work, they investigated the robustness of the machine learning in the context of phishing detection in the phase of adversarial sampling attacks.

(Jain and Gupta, 2018) described a machine learning-based approach that extracts the features from the client-side only. However, their method of dataset creation is biased, as discussed in (Shirazi et al., 2018). (Shirazi et al., 2018) observed two concerns with existing machine learning approaches: a large number of training features were used and there was bias in the datasets used. They defined a subset of features and created the machine learning dataset that can be extracted entirely on the client-side without using any third-party services. The study focused on the features derived from the domain name usage in phishing and legitimate websites and reported an accuracy of 97 – 98% on the chosen datasets.

(Ho et al., 2019) created a large-scale dataset of emails from 92 enterprise organizations and created a detection algorithm to discover spear-phishing emails. The model found hundreds of real spear-phishing emails with a very low false-alarm rate: four per every one million. (Gutierrez et al., 2018) observed that current machine learning-based detection algorithms are vulnerable against structural or semantic change in the message. They implemented machine learning on a large corpus of phishing and legitimate emails and employed under-sampling boost algorithms to handle the class imbalance problem of

phishing datasets. But they did not study the problem of the imbalanced datasets in phishing datasets.

(Van Der Heijden and Allodi, 2019) developed an automated and fully quantitative method based on machine learning and econometrics to measure cognitive vulnerability triggers in a phishing email to predict the degree of success of an attack. Instead of selecting the best features from a machine learning point of view, this study is based on the human cognitive method. The study shows how adversaries convince end-users to give up their sensitive information. These detected metrics can improve learning algorithms and help response teams to prioritize their effort in case of a real attack.

(Marchal et al., 2012) focused on detecting phishing domains and created a proactive mechanism instead of reactive approaches like blacklisting. The second-level domain of the URL and a Markov chain have been used to detect suspicious domain names. They leveraged natural language modeling to create a blacklist based on phishing related vocabulary.

3 PROPOSED SOLUTION

This section elaborates on our model which uses a new way of modeling the phishing problem that makes it more effective and free from bias.

3.1 Problem Modeling

Phishing campaigns usually work based on a three-part scheme. The first part is using email or some form of communication to lure users and redirect them to a phishing page. The fake page closely mimics a trustworthy site. Finally, the user enters their information, which is captured by the adversary. Phishing websites must target at least one genuine site, which we define *target website*, and should be similar in terms of visual and textual traits to the target website to earn the end-users trust.

In our proposed approach, instead of defining features that group the phishing websites together, we relate a suspicious phishing website to its target and define features based on the similarity of a given suspicious website and its target. Figures 1a and 1b hypothetically explain this issue in details. Figure 1a is a scatter-plot of phishing and legitimate samples with two features: *feature 1* in the y-axis and *feature 2* in the x-axis in existing approaches. The green dots represent the position of phishing instances, and blue dots represent legitimate websites. This image clearly shows, based on feature definition, phishing and legitimate websites are distinguishable with two mis-

classified samples. In addition, it shows that phishing samples are not related to any legitimate website.

Figure 1b shows our proposal approach. There are still two features, namely *feature 3* and *feature 4* in the y-axis and the x-axis respectively. There are three target websites in this graphs named *Target 1*, *Target 2*, and *Target 3*. All dots in the graph indicate phishing samples. Since features are defined to show the similarity of phishing instances to the target website, phishing websites in each group are close to each other, and this group represents individuals attacking a target website in the form of a cluster.

3.2 Fingerprint Definition

We define the fingerprint of a legitimate website as a mathematical representation of that website, which can uniquely distinguish a legitimate website from other legitimate websites. Moreover, comparing suspicious websites with this fingerprint would determine if it is similar to the target website or not, a vital sign when a phishing website attacks a target. If a suspicious website's similarity to a target website exceeds a given threshold, then this would be assumed as a phishing website.

For each given target site, we would consider both the visual and the textual characteristics of the target site. The process of extracting a fingerprint and then matching it with suspicious websites is independent of other legitimate websites. Consequently, the phishing detection process for each target would be independent of other targets. Thus, we have no issues with biased data.

We define the fingerprinting for any given legitimate website as follows. For each given legitimate website, there would be a set of features that,

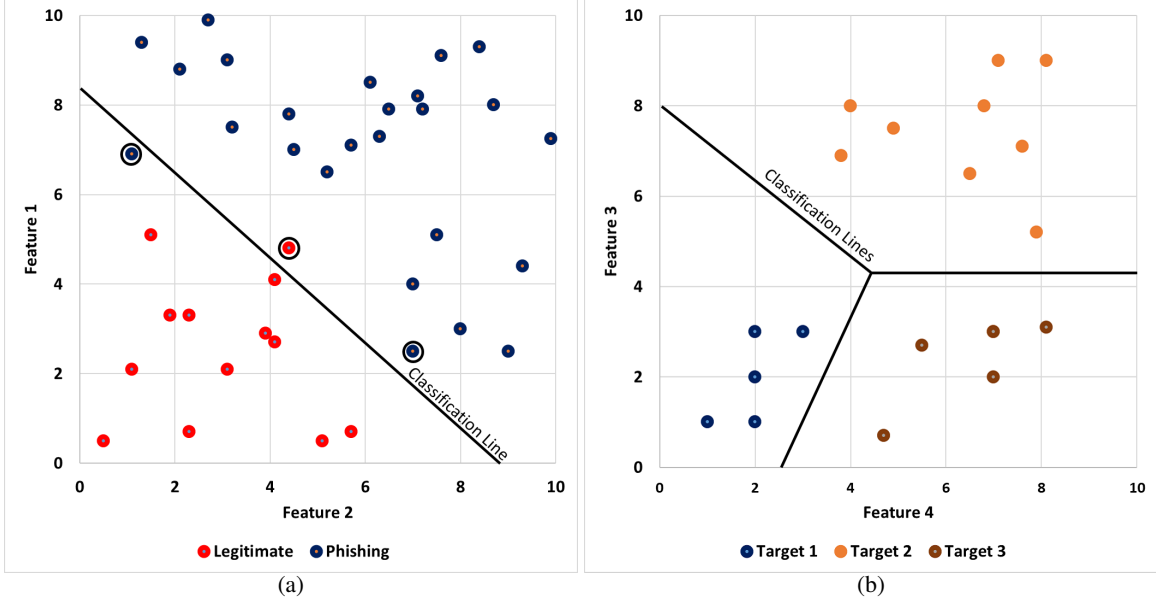
- Uniquely represent a website so that it can be distinguished from other legitimate websites, and which we call the *fingerprint* of the website, and
- Comparing the fingerprint of a website to any other given website will return the level of similarity between them.

Each legitimate website has at least one screenshot. We use f_i for feature i and denote F_j^A as all fingerprint features of j^{th} screenshot of legitimate website A . Thus, if screenshot j has L_j number of features, F_j^A would be:

$$F_j^A = \{ f_0 \cup f_1 \cup \dots f_{L_j} \} \quad (1)$$

If $|A|$ is the total number of screenshots for legitimate website A , then the fingerprint of legitimate website A would be called as F^A and calculated as follows:

Figure 1: Two different ways of modeling the phishing problem with regards to feature definition. The left image shows features that are defined based on similarity among phishing websites and that among legitimate websites. The right image shows features that have been defined based on their similarity to target websites. Samples that are attacking a target are clustered together.



$$F^A = \{ F_0^A \cup F_1^A \cup \dots F_{|A|}^A \} \quad (2)$$

Screenshots are captured images that have been shown on the end-users display. These captures are taken from login pages of legitimate websites or login pages of older versions of legitimate website. If visual or textual traits of a legitimate website change over time, we need to add new screenshots and update the fingerprint to capture these changes to detect new phishing attacks.

4 OUR METHODOLOGY

Our approach consists of the following steps. We are given the *benign set* that consists of the set of legitimate websites that we are trying to protect from the phishing attacks. For each legitimate website in the benign set, we perform the following three steps.

[Step 1: Extracting Fingerprints from Legitimate Websites] We create a fingerprint using textual and visual features.

[Step 2: Creating and Labeling Dataset] We create a labeled dataset. We assign a label of 1 if the data is a phishing sample that is targeting the legitimate website. Otherwise, we assign it a label of 0.

[Step 3: Create a Machine Learning Model] We create a machine learning model corresponding to the legitimate website.

4.1 Extracting Fingerprints

For each given legitimate website, we prepare one or more screenshots of the target website. In some cases, if the website has multiple login pages that are not visually identical, we prepare more than one screenshot, then extract textual and visual features. The visual and textual characteristics of each page define the unique identity of each legitimate website.

Textual Sector

Textual elements are the text that is visible by the end-user. Examples include text that asks users to enter their username and password or terms and conditions of using the services. Graphical designers use these characteristics to create a uniquely distinguishable webpage. Thus, we use those characteristics to create fingerprints in this study.

For the textual feature gathering, we use an Optical Character Recognition (Optical Character Recognition (OCR)) algorithm, which uses machine learning to extract text word by word as an object which we can use in programming. For our OCR algorithm, we used the web-based Google Cloud Vision API which is one of the best algorithms available. Google OCR hides technical details from end-users,

but it includes five steps: *Text Detection*, *Direction Identification*, *Script Identification*, *Text Recognition*, and *Layout Analysis*.

Text Detection uses a Conventional Neural Network (CNN)-based model, to detect and localize the line of text that generates a set of bounding boxes. *Direction Identification* classifies direction per bounding box and *Script Identification* identifies script per bounding box. *Text Recognition* recognises the text from the image. It includes a character-based language model, inception style optical model, and custom decoding algorithm. Finally, *Layout Analysis* determines reading order and distinguishes titles, header, etc. (Ma, 2019).

The extracted text provided by Google OCR is cleaned. We ignore punctuation and stop words and make all texts lower-case. We fix misspelled words if there is any. Misspelled words are relatively rare in legitimate websites but pretty common for fake websites. The cleaned list of extracted words create the *textual sector* of the fingerprint.

Visual Sector

Visual elements include, but are not limited to, brand logo and other graphical iconic elements related to the website. These are elements that make a login page unique compared to other legitimate webpages. We used a segmenting algorithm to detect, localize, and save many of these segments found in the legitimate web pages. The segments generated were then manually scrutinized to eliminate the ones that were considered not relevant to the site’s fingerprint.

Figure 2 shows a screenshot of *Yahoo*, captured from the legitimate website. The parts in black rectangles are the visual segments and the parts in red rectangles are the textual segments that constitute the fingerprint.

4.2 Creating Dataset

We create a dataset for each legitimate website. The phishing samples that target the legitimate website are labeled as 1 and anything else is labeled as 0.

Textual Sector Matching

For each given input website, we use the previously discussed OCR algorithm to get all the text out of the screenshot. Then we match each word in the fingerprint with words of the given input. If the word in the fingerprint does not match any word in the website, we assign that feature as -1 . If the word in the fingerprint does match that of the word in the input, then that corresponding feature is assigned a value that reflects the importance of the word in the input website. The TF-IDF algorithm statistically reflects the importance of a word in a corpus. The corpus

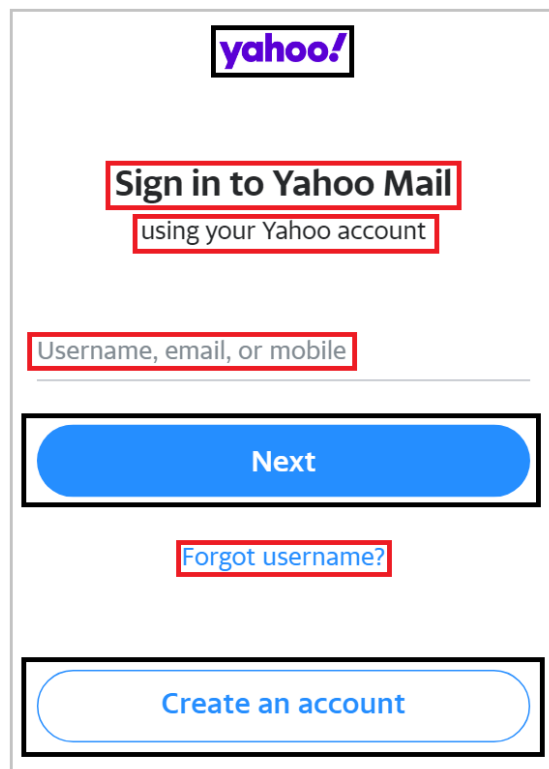


Figure 2: Legitimate screenshot from Yahoo.com. Black colored boundaries specify visual segments and red colored boundaries specify parts with texts returned by OCR algorithm.

consists of all of the words in the legitimate websites. Thus, for each word of a given website, we use TF-IDF to evaluate its importance in the context of the website, if it matches with a word in the fingerprint. In such a case, the TF-IDF score is assigned to the corresponding feature. For example, the company’s name will have a higher value than a word like *login* in the learning vector and makes it more meaningful for the machine learning algorithm.

Visual Sector Matching

In order to determine if the visual characteristics of an input image match that of our target website, we need to check if the legitimate website’s segments exist in the input image. This fact introduced many complications, as image comparison is often times challenging. We also had to consider what an adversary might do to bypass a comparison algorithm. We decided to leverage the concept of homography to counter simple rotations and deformations. This was achievable by using the key-points of both the segments and the input image. In the next step, we determined the quality of the projective transformation of the input image for each legitimate segment. If no homography transformation is found, we con-

sider that segment to be absent from the input image. Furthermore, if a homography does exist, we compare the resulting image with the segment to determine the validity of this homography. For the comparison algorithm, we implemented a custom algorithm called *BF-Matcher*, which compares the key points using brute force key-point matching and the dot product between these matches. If the comparison value is above a threshold we consider the segment to exist in the input image. The feature value of this part of the fingerprint matching is the result of the comparison algorithm.

For every given input, we will have a vector that specifies the similarity between the given website and the fingerprint of the target site. We have created our dataset after calculating these features.

5 EXPERIMENTS AND RESULTS

We discuss the datasets used and machine learning metrics, and then elaborate on the two experiments we have conducted and the results in this section.

5.1 Created Dataset

(Dalgic et al., 2018) gathered screenshots of phishing attacks and made it publicly available. This dataset includes labeled phishing samples of 14 brands. While (Dalgic et al., 2018) only has the phishing samples, we need screenshots of the legitimate websites to create a fingerprint for them. Thus, we captured screenshots of these legitimate websites and added them to our dataset. We also manually double-checked all of the phishing samples and their relationship to the claimed target website to find discrepancies and fixed a few of them.

In the next step, we run our fingerprint extraction algorithm to create a fingerprint for each legitimate website. We evaluated the learning vector based on the similarity to the fingerprint extracted for the targeted website.

For each target website, we created a separate dataset and for each given screenshot, we evaluated whether the text or visual segments exist in the fingerprint or not which was used to encode the feature vector. If the instance is attacking this target website, we label it as 1, otherwise we label it as 0.

Table 1 lists all target sites we used in our experiments with the number of items in both textual and visual segments of the fingerprint and number of phishing instances for each target. We created 14 separate datasets for the 14 respective target websites. The machine learning classifier will be trained on the corresponding dataset. This helps to relate the phishing

Table 1: List of used target websites and number of textual, visual and total features used for fingerprint, and total number of phishing samples for each target.

Website	#Fingerprint Segment			Samples
	Tex.	Vis.	Tot.	
Adobe	26	26	52	70
Alibaba	82	22	104	76
Amazon	24	14	38	29
Apple	34	31	65	64
BOA ¹	182	39	221	111
Chase	127	99	226	111
DHL	48	33	81	109
Dropbox	45	16	61	115
Facebook	84	34	118	144
Linkedin	77	28	105	38
Microsoft	15	10	24	117
Paypal	27	14	41	214
Wellsfargo	166	42	208	134
Yahoo	14	12	26	114

¹ Bank of America

Table 2: Data Definitions

Metric	Definition
A	Samples attack current target
A'	Samples classified that attack
nA	Samples do not attack current target
nA'	Samples classified that do not attack
$A \rightarrow A'$	Attack samples classified <i>correctly</i>
$A \rightarrow nA'$	Attack samples classified <i>incorrectly</i>
$nA \rightarrow nA'$	Non-Attack samples classified <i>correctly</i>
$nA \rightarrow A'$	Non-Attack samples classified <i>incorrectly</i>

samples to a specific target site, instead of relating all phishing websites to all legitimate websites, as we discussed it in (Alexa, 2020).

5.2 Machine Learning and Scores

In our experiments, we used five different classifiers available in Scikit-learn toolkit (Pedregosa et al., 2011) and then we selected the best one. We used Random Forest (RF), Gradient Boosting (GB), and Support Vector Machine (SVM) with two different kernels: Linear (l) and Gaussian Radial-basis function kernel (g). In addition, we used a Majority Voting (MV) as another classifier that acts as a voter among all of the fitted classifiers. We ran each experiment 10 times and reported the average of the results with five-fold cross-validation to avoid issues of over-fitting. We tested the performance of each learning model against unseen samples.

Evaluating the effectiveness of an algorithm only by relying on accuracy in imbalanced datasets may be misleading. Because a majority group with a large

Table 3: Definition of performance metrics

Score	Formula	Description
TPR	$\frac{ A \rightarrow A' }{ A }$	correctly classified attacking
PPV	$\frac{ A \rightarrow A' }{ A \rightarrow A' + nA \rightarrow A' }$	correctly over total predicted attacking
f1	$2 * \frac{TPR * PPV}{TPR + PPV}$	harmonic average of TPR and PPV
ACC	$\frac{ A \rightarrow A' + nA \rightarrow nA' }{ A + nA }$	classified correctly in total

margin can dominate the accuracy result. We have an imbalanced dataset and thus reporting accuracy may be misleading. To address this limitation, F1 score which is a harmonic average of precision and recall has been proposed and widely accepted. We report F1 score to show how effective is our algorithm. We also reported accuracy score for further comparison with other studies.

5.3 Performance of Classifiers

For each dataset of the target site, we split the dataset into three sub-sets: one set with only textual features, the other set with entirely visual features, and the last one with both textual and visual features. We then trained and tested all of five classifiers for each sub-set ten times and reported the average.

Figure 3 highlights these results. It shows MV with the highest f1 score of 97.62% among all of the classifiers; thus, we selected this classifier for further experiments. Furthermore, it gives the best accuracy as well.

The next best classifier after MV is GB. It has an average f1 score of 97.29% and an accuracy of 99.68%, which is slightly more than RF. These results show that both GB and RF were able to significantly detect phishing attacks against all 14 websites.

5.4 Effectiveness of Model

Figure 4 reports the accuracy of the model for all 14 targeted websites that we ran the experiments for, and figure 5 reports the f1 scores with separation based on textual and visual sections of the dataset. For this experiment, we used GB as it gave the best results in the previous experiment. Figure 4 shows the results of three sub-sets: only textual features, only visual features, and both together. The accuracy was over 98% for all cases when we used both the visual and textual features. It also shows that the visual sector alone does not give good results for the following websites: *Adobe*, *Amazon*, *Microsoft*, and *Yahoo*.

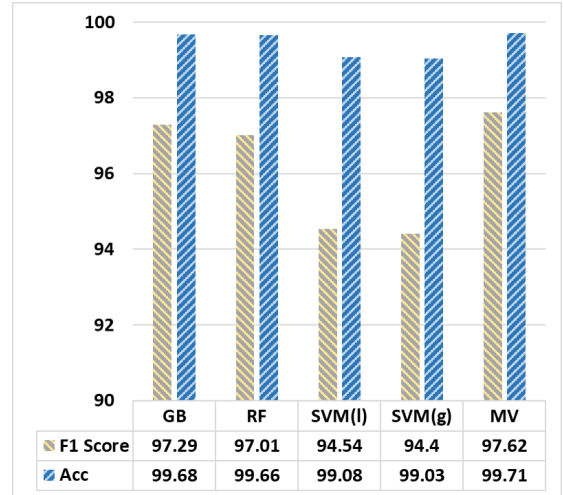


Figure 3: F1 score of trained model for different classifiers.

Our dataset is a highly imbalanced dataset, and we reported the f1 score to balance between *precision* and *recall*. Having a high f1 score guarantees both precision and recall have high values. In this case, the classifier does not ignore one class with a lower volume of data to increase total accuracy. Figure 5 clearly shows the model gives a high f1 score for all of the websites. *Chase* website with the f1 score of 99%75 has the highest score, and *Microsoft* with 88.42% has the lowest score when we used both visual and textual features. The reason that *Microsoft* has the lowest score among the targeted website is because the fingerprint process could not find enough segments to create the fingerprint.

In addition, our model gives exceptionally high scores among all of the websites regardless of their popularity. While *Amazon* and *Yahoo* are among the top twenty most visited websites (Alexa, 2020) but they have the same accuracy or f1 score as *DHL*, which has a popularity rank of 1248 (Alexa, 2020). This demonstrates that our experiments were free from the bias stemming from the popularity of the website in contrast to current approaches.

6 CONCLUSION

In this paper, we propose an approach that detects whether a phishing website is attacking a target legitimate website. We generate fingerprints for legitimate websites using visual and textual characteristics and detect phishing websites based on how closely their features match these fingerprints. Our approach is not biased towards more popular websites and can be adapted for new attacks. We demonstrated our approach on 14 different target websites with varying

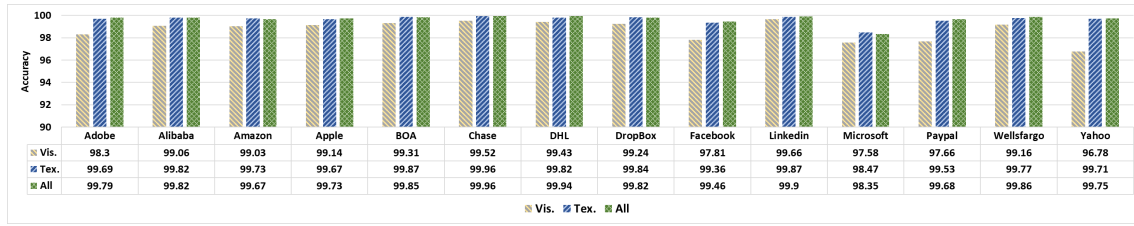


Figure 4: Accuracy of trained model for targeted websites.

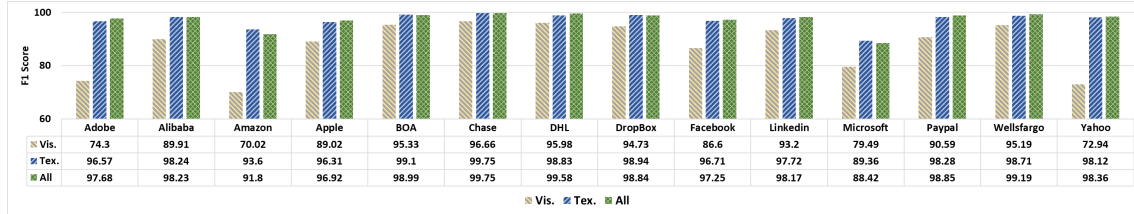


Figure 5: F1 score for trained model for targeted websites.

popularity. Our model achieved an accuracy of 99% for all of them with cross-validated data. Furthermore, we employed a one-vs-all technique and created an imbalanced dataset; we reported an accuracy of more than 98% among all of the websites, which is surprisingly high. It may possible that through adversarial machine learning attackers generate phishing samples that match the fingerprint of legitimate websites. Our future work will investigate how to protect against such attacks.

REFERENCES

- Abdelhamid, N., Ayesha, A., and Thabtah, F. (2014). Phishing Detection Based Associative Classification Data Mining. *Expert Systems with Applications*, 41(13).
- Alexa (2020). Competitive Analysis, Marketing Mix and Traffic.
- Cui, Q., Jourdan, G.-V., Bochmann, G. V., Couturier, R., and Onut, I.-V. (2017). Tracking Phishing Attacks Over Time. In *Proc. of IW3C2*.
- Cui, Q., Jourdan, G.-V., Bochmann, G. V., Onut, I.-V., and Flood, J. (2018). Phishing Attacks Modifications and Evolutions. In *Proc. of ESORICS*.
- Dalgic, F. C., Bozkir, A. S., and Aydos, M. (2018). Phishiris: A new approach for vision based brand prediction of phishing web pages via compact visual descriptors. In *Proc. of ISMSIT*.
- Gutierrez, C. N., Kim, T., Della Corte, R., Avery, J., Goldwasser, D., Cinque, M., and Bagchi, S. (2018). Learning from the Ones That Got Away: Detecting New Forms of Phishing Attacks. *IEEE Transactions on Dependable and Secure Computing*, 15(6).
- Ho, G., Cidon, A., Gavish, L., Schweighauser, M., Paxson, V., Savage, S., Voelker, G. M., and Wagner, D. (2019). Detecting and Characterizing Lateral Phishing at Scale. In *Proc. of USENIX*.
- Jain, A. K. and Gupta, B. B. (2018). Towards detection of phishing websites on client-side using machine learning based approach. *Telecommunication Systems*, 68(4).
- Ma, E. (2019). Secret of Google Web-Based OCR Service.
- Marchal, S., François, J., Engel, T., et al. (2012). Proactive Discovery of Phishing Related Domain Names. In *RAID*.
- Mohammad, R. M., Thabtah, F., and McCluskey, L. (2012). An Assessment of Features Related to Phishing Websites Using an Automated Technique. In *Proc. of IC-ITST*.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12.
- Shirazi, H., Bezawada, B., and Ray, I. (2018). “Kn0w Thy DomaIn Name”: Unbiased Phishing Detection Using Domain Name Based Features. In *Proc. of SACMAT*.
- Shirazi, H., Bezawada, B., Ray, I., and Anderson, C. (2019). Adversarial Sampling Attacks Against Phishing Detection. In *Proc. of DBSec*.
- Tan, C. L. (2018). Phishing Dataset for Machine Learning: Feature Evaluation.
- Van Der Heijden, A. and Allodi, L. (2019). Cognitive Triage of Phishing Attacks. In *USENIX*.
- Zou, J. and Schiebinger, L. (2018). AI Can Be Sexist and Racist—It’s Time to Make It Fair.