Phishing Sites Detection from a Web Developer's Perspective Using Machine Learning

Xin Zhou
Department of Computer Science
University of Houston
xzhou21@uh.edu

Rakesh M. Verma
Department of Computer Science
University of Houston
rverma@uh.edu

Abstract

The Internet has enabled unprecedented communication and new technologies. Concomitantly, it has brought the bane of phishing and exacerbated vulnerabilities. In this paper, we propose a model to detect phishing webpages from a web developer's perspective. From this standpoint, we design 120 novel features based on content from a webpage, four time-based and two search-based novel features, plus we use 34 other content-based and 11 heuristic features to optimize the model. Moreover, we select Random Committee (Base learner: Random Tree) for our framework since it has the best performance after comparing with six other algorithms: Hellinger Distance Decision Tree, SVM, Logistic Regression, J48, Naive Bayes, and Random Forest. In real-time experiments, the model achieved 99.4% precision and 98.3% MCC with 0.1% false positive rate in 5-fold crossvalidation using the realistic scenario of an unbalanced dataset.

1. Introduction

Over the last decade, many cyber attacks start with a poisoned link. Phishing websites try to hook Internet surfers into revealing their sensitive information including credentials, bank account, and other personal information. These new, short-lived phishing URLs can easily bypass signature-based detectors. To combat this problem, researchers have also used machine learning methods to detect phishing websites. Nevertheless, there is still no definitive solution with machine learning or another approach.

We propose an effective phishing detection system using machine learning, which is based on finding and classifying the differences of webpage source code between hand-coded, auto-generated, and toolkit-generated webpages. We also add other features selectively to optimize the framework. In the real world, a phishing website normally exists for a very short time. The no longer live phishing sites likely redirect to legitimate sites. Therefore, we conduct our experiments on both offline and real-time datasets.

For evaluating our work, we use accuracy, false positive rate, precision, recall, and F score. We also add MCC (Matthews correlation coefficient) and AUC (Area under the ROC curve) for comparison because these are better metrics for both balanced and unbalanced datasets, since they balance the classification performance between minority and majority class. *Precision* is defined as true positives divided by summation of true positives and false positives, i.e., the proportion of positive identifications that were actually correct. A phishing detection system must aim for high precision, since true and false positives are most important in a real world application. *Recall* is defined as true positives divided by the sum of true positives and false negatives, which is the proportion of actual positives that are identified correctly. Recall shows that how effectively the system can detect a phishing webpage. Formulas for the metrics are given in Aassal et al. [1].

We use crossvalidation, when the dataset is not too large, to avoid over-fitting issues. Crossvalidation is used to estimate how accurately a predictive model performs in practice. Normally, the model is trained on a training subset, then tested on a different subset of the data called test dataset. Using crossvalidation gives a more realistic idea of the performance, since the method is tested in K rounds on every dataset instance. In K-fold crossvalidation, the data is divided into K subsets. Then, the algorithm uses i^{th} subset as testing dataset in round i and the remaining subsets as training dataset.

In this paper, we focus on web content analysis and feature selection and make the following *contributions:* (i) We introduce many new features for this problem. (ii) In contrast to most previous work, we test our models on both balanced and unbalanced datasets, which is the more realistic scenario. (iii) We do both real-time and offline experiments. (iv) We use the right metrics for different scenarios, and (v) a new

URI: https://hdl.handle.net/10125/64536
978-0-9981331-3-3

classifier, the Hellinger Distance Decision Tree [2]. To our knowledge, this has never been used for phishing website or even any security challenge before. Our observations and ideas are as follows:

A hand-coded website, which is created by an attacker, is typically formatted poorly and has weird behaviors. A web developer optimizes the usage of HTML header tags for improving SEO (Search Engine Optimization) [3]. For example, the source code of real PayPal login page uses four <h1> tags and one <h2> tag, but a fake PayPal page does not use any <h1> tag and uses one <h2> tag (shown in Figure 1).

When a mimic site is just generated by a social engineering toolkit,[1] the links to social media sites or other links may not match the domain of current URL. A feature that checks the patterns on domain, title, and links can help the model to identify the discrepancies in source code. For example, Figure 2 shows a phishing website that is generated by SET (Social-Engineer Toolkit). Note that the links such as "http://facebook.com/legal/terms/update" and "http://facebook.com/about/privacy/update" are supposed to be internal links "/legal/terms/update" and "/about/privacy/update."

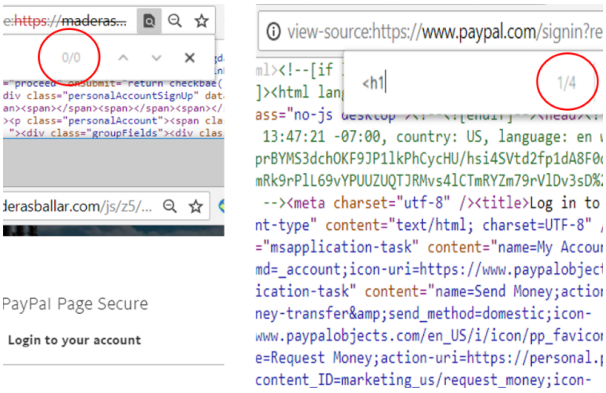

Figure 1. PayPal Login Page (Fake vs Real)

Normally, a business website has its own domain emails for business, service, support, or internal use, but many phishing sites do not have these types of emails, since it adds extra cost and their sites only exist for a very short time. We can extract domain emails based on top search results using Google, Bing, or other search engines to avoid missclassifying some legitimate websites. For example, a website "https://popuband.com" will own email addresses such as "support@popuband.com," "service@popuband.com," and so on, if they use email to communicate with customers or employees. Only

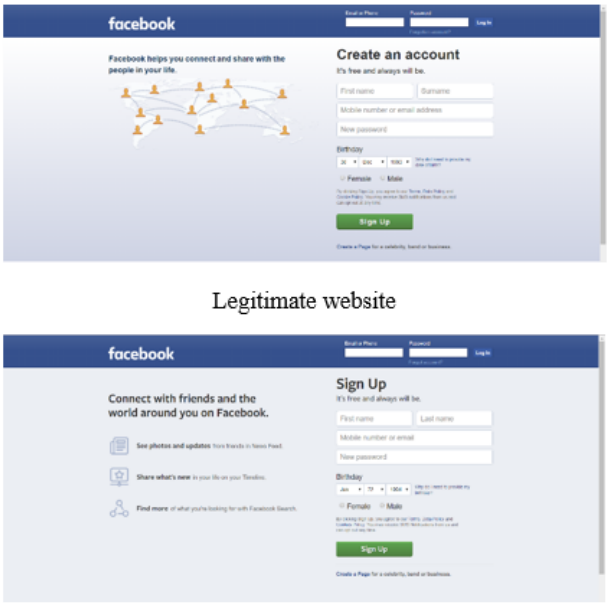

Figure 2. Legitimate VS Phishing

a few attackers will spend the extra money to run an email server. Even if they do, their emails may not show up in the top search results, since phishing websites are short-lived. Another hypothesis is that a phishing website sometimes only mimics a subset of the pages from a legitimate website, then its number of sub-domains will be lower than legitimate one. Overall, a legitimate website has many more sub-domains than a phishing website. We used the "theHarvester"[2] tool with the Bing search engine (Google sometimes blocks our IP) to extract these features, but it took from one to three seconds to extract. We understand that this could be a hindrance in a real world application, since most people will not wait for a few seconds to open a web page, but this can be used from the back-end to create and update a powerful whitelist.

Some content-based features have been used

[1] https://github.com/trustedsec/social-engineer-toolkit

[2] https://github.com/laramies/theHarvester

previously, e.g., [4, 5, 6, 7, 8, 9], but we should extract features more carefully. From a web developer's perspective, different types of links have totally different characteristics and purposes. For example, (1) using an external image will slow down the loading speed; (2) an external link of open source JavaScript library has less impact compared to those video and image links; (3) social share links normally are longer than others or constant length based on a specific hash function; (4) The length of links can vary, but the mean length and standard deviation for each type of link is within a certain range based on the development style, tools, and directory management.

We also observed that toolkit-generated pages have different characteristics compared to hand-coded or auto-generated web pages. A web developer will optimize the loading speed of a web page so as to make it load up as fast as possible, which means more customization and features will be added. A developer will download libraries like Bootstrap and JQuery, then save them into the server instead of taking the link from a provider's page. For example, one can directly use the JQuery link "https://code.jquery.com/jquery-3.3.1.slim.min.js" on tag "<script>," but downloading it to one's server is better.

SEO is super important to any website that needs more visitors from search engines over time. A mimic page usually contain weird links and headers, no concerns about SEO since a phishing link is normally sent by an email or embedded somewhere. Simply calculating the mean and standard deviation of all links does not make any sense, since a legitimate website could also use lots of external links, but it will be really meaningful if we group the links into different categories by their behaviors, usage, and tags. Therefore, we split all links into 30 sets based on HTML tags and link types. Based on these sets, we calculate the size, mean, and standard deviation for each set, thus extracting 90 features totally. More details are shown on L-Tree (Figure 3).

The rest of this paper is organized as follows. In Section 2 we describe the features, feature extraction and datasets. Section 3 details the pipeline for the experiment. We present the results and analysis in Section 4. Related works are discussed in Section 5 and Section 6 concludes the paper.

2. Features and datasets

In this section we explain our features and describe the datasets in detail.

2.1. Feature selection

Several features have been used by other researchers for phishing URL/website detection. We select only eight URL-based and three host-based features from two surveys [10, 11] to enhance our detecting system, since our goal is to improve the effectiveness of the model based on the differences between hand-coded, auto-generated, and toolkit-generated pages. We now describe the features we have implemented for our models. Most of them are novel, i.e., to our knowledge, they have not been used by other researchers in the phishing literature. In parentheses, next to the feature group title, we list the number of features for each category.

URL-based (8): (i) domain_Length: length of domain. (ii) hyphen_symbol: number of hyphens "-" in domain. (iii) consecutive_numbers: find all maximal sequences of digits in domain name and calculate the score $\sum(length(number))^2$ (e.g., for "amaz0n.666buy01.com", we get 1^2 + 3^2 + 2^2 = 14). (iv) digits_count: count the number of digits in the domain (e.g., digits_count(amaz0n.666buy01.com) = 6). (v) protocol: whether the default hypertext transfer protocol is used by the website (e.g., "https://google.com" uses https://). (vi) numberOfPeriods: number of periods in the URL. (vii) TLD: top level domain. If a URL contains a common TLD ("com," "net," "org," "edu," "mil," "gov," "co," "edu," "biz," "info," "me"), then we set value of this attribute to 1. Otherwise, it is 0. (viii) numberOfredirects: number of total redirects to reach the final destination.

Search-based (2 novel features): (i) Number of domain emails (novel): using "theHarvester" we extract domain emails that are found in top 50 search results from Bing. (ii) Number of hosts (novel): using "theHarvester" to extract sub-domains that are found in top 50 search results from Bing.

For example, we feed a domain "Microsoft.com" into the Bing search engine, then count the number of domain emails and number of sub-domains from top 50 search results. We choose 50 top search results because it is the best number for extracting features in reasonable time. Using more than 50 search results ends up taking more time for feature extraction, but yields no significant performance improvement.

Host-based (3): We use a custom parser to extract these features based on the information from WHOIS Lookup: (i) Expiration minus creation date. (ii) Current minus updated date. (iii) Current minus creation date.

Content-based (90 (novel) + 12 + 22 + 30 (novel) = 154):

1. <a>, <link>, <script>, <video>, <img>, <meta>: split all links by these six HTML tags first, then divide again by five types: (i) any URL contains current domain, (ii) social network links ("Facebook," "YouTube," "Google," "Twitter," "Instagram," "Pinterest)", (iii) other https links, (iv) other http links, and (v) internal links. Calculate the size, mean, and standard deviation of these 30 sets. 30 * 3 = 90 novel features (Figure 3).

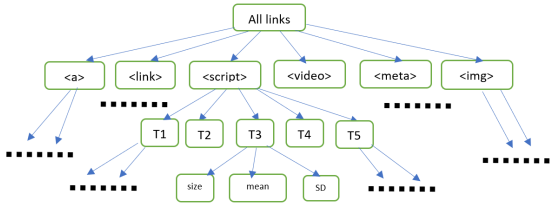

Figure 3. L-Tree

2. Display/content features (12): (i) <h1> header length, (ii) <h2> header length, (iii) title length, (iv) number of inputs, (v) number of paragraphs, (vi) number of spans, (vii) number of buttons, (viii) real domain name on title, (ix) real domain name on <h1>, and (x) real domain name on <h2>, (xi) Is it a login page? (xii) Is there a submit form?

3. Login form features (22): number of <input> forms by their types: (i) button (ii) checkbox (iii) color (iv) date (v) datatime-local (vi) email (vii) file (viii) hidden (ix) image (x) month (xi) number (xii) password (xiii) radio (xiv) range (xv) reset (xvi) search (xvii) submit (xviii) tel (xix) text (xx) time (xxi) url (xxii) week

4. Alexa Features ($5 * 3 * 2 = 30$ novel features): Alexa rank is normalized by the ranges. Convert [<1000, <10000, <100000, <500000, <1000000, <5000000, 5000000+, unranked] to [1, 2, 3, 4, 5, 6, 7, 8]. Group external links into five sets by HTML tags: <link>, <script>, <video>, <img>, <a>. For each set of links, compute: (i) Mean (5) and standard deviation (5) of global ranks of domains. (ii) Mean (5) and standard deviation (5) of country-based ranks of domains. (iii) Mean (5) and standard deviation (5) of country codes (ISO) in domains.

Time-based (4 novel features): A phishing website is normally hosted on a cheap server with cheap domain, which has poor performance. Therefore, we choose four elapsed-time features based on feature extraction: (i) Elapsed time for URL-based features. (ii) Elapsed time for Host-based features: the parser will take longer to find the information from WHOIS Lookup for poorer servers. (iii) Elapsed time for Content-based features: the loading speed of a website is based on its servers and content. (iv) Elapsed time for Search-based features: higher ranking webpages could be parsed much faster than lower ranking ones.

2.2. Datasets

We use two major sources for our datasets, both of them are publicly available and easy to access, and keep updating the dataset every time we perform a new experiment. (1) Phishing URLs from PhishTank.[3] (2) Legitimate URLs found by our URL-crawler.

Web crawler: We developed a URL crawler to create two legitimate pools. Pool A contains a total of 21369 URLs based on top 5000 websites from Alexa on 05/28/2019. Pool B contains a total of 202,056 URLs based on top 50,000 websites from Alexa on 06/01/2019. To ensure diversity of the dataset, the URL-crawler grabs at most five URLs from each website.

Offline dataset: We used phishing URLs that were collected by PhishTank from 01/01/2019 to 04/22/2019. Then we took 4090, 21369, and 202056 legitimate URLs from pools A, A, and B respectively to conduct three offline experiments.

Online dataset: We extracted the features of phishing URL in real time. We collected a total of 2916 phishing URLs from 04/23/2019 to 06/08/2019. Then we took 2916, 15263, and 202056 legitimate URLs from pools A, A, and B respectively to conduct three real-time experiments.

3. Pipeline

Figure 4 shows the pipeline of our system. It has the following components:

Whitelist filter: We use the final URL (the destination after all redirects) as our input URL. Sometimes, a no longer existing phishing URL from PhishTank may redirect to a common legitimate web homepage such as "google.com/ServiceLogin," "facebook.com," "google.com," etc. Therefore, we use a white list that contains five common legitimate websites $\{google, microsoft, facebook, airbnb, bing\}$ to filter out such URLs that used to be phishing, but now redirect to a legitimate website.

[3] https://www.phishtank.com/

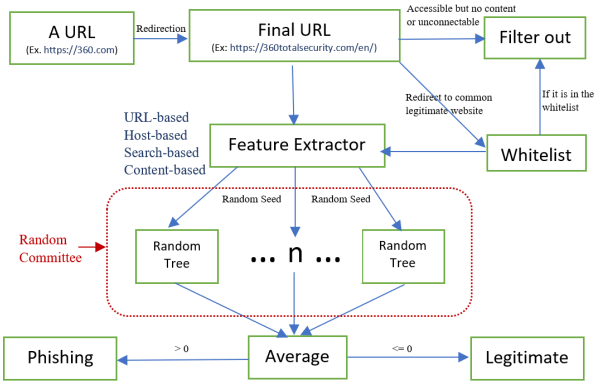

Figure 4. System Pipeline

Feature extractor: We build the feature extractor in Java with following libraries: Jsoup, Apache-commons-net, htmlparser-1.6, WEKA V.3.8.2, and theHarvester. For each URL, it takes on average 1.63 seconds to extract all the features on a Linux 3.10.0-862.6.3.el7 server with 2 x Eight-Core Intel Xeon Processor 3.20GHz, 25MB Cache, 8 x 64GB memory size, and dual 1-Gigabit Ethernet. **Offline Version:** We download phishing URLs from PhishTank and extract the features every time we start a new experiment. This will let some fake phishing URLs (they are still accessible but no longer being used as phishing) labeled as phish feed into the machine learning model. **Online Version:** The feature extractor automatically extracts features from fresh phishing URLs as they are verified by PhishTank.

Machine learning frameworks: We use the WEKA library[4] to construct our classification and prediction models. We build the models based on the following machine learning (ML) algorithms: Hellinger Distance Decision Trees, SVM, Random Forest, Naive Bayes, J48, Logistic Regression, and Random Committee (Random Tree). This framework is convenient and efficient in detecting phishing URLs. We developed this framework for detecting phishing URL/website, but it is flexible enough for detecting phishing emails, if an email contains link(s).

Table 1: Performance Comparison of ML Algorithms

Alg.	Acc.	FP.	Pre.	Recall	F1	MCC
R. C.	0.978	0.022	0.978	0.979	0.978	0.957
R. F.	0.971	0.027	0.973	0.970	0.971	0.943
J48	0.948	0.054	0.946	0.950	0.948	0.897
Logistic	0.928	0.074	0.926	0.930	0.928	0.856
SVM	0.929	0.037	0.859	0.811	0.834	0.790
HDDT	0.865	0.130	0.868	0.860	0.864	0.729

[4]https://www.cs.waikato.ac.nz/ml/weka/downloading.html

4. Analysis and results

4.1. Model

Based on the results shown in Table 1 of different algorithms using an offline balanced dataset (4090:4090) in 5-fold crossvalidation, we can see R. C. (Random Committee) with base learner Random Tree has the best performance compared to five other models. Therefore, we implement our detecting system using Random Committee for further investigation. All following results are based on this method.

4.2. Random Committee

Random Committee is a type of ensemble learning method that builds an ensemble of randomized base classifiers. Each base classifiers is built using a different random number seed (but based on the same data and the same base learner). The final prediction is based on an average of the predictions generated by the individual base classifiers.

4.3. Evaluation

Detailed evaluation results for precision, recall, FP, and MCC using balanced and unbalanced datasets are given next. Tables 2 to 5 give offline results in 5-fold crossvalidation and Tables 6 to 9 give online results with 5-fold crossvalidation. We also test the model with combined features minus features from a specific category (shown in Table 5 and 9). For example, NU means the model with all features *except* URL-based features. In real-world deployment of a phishing detection system, the model must perform with relatively high precision and recall. These tables show that our model has achieved high precision, recall, and MCC in both balanced and unbalanced datasets in 5-fold crossvalidation. Comparing our offline and online experiments, the best results were achieved with all features (Table 7): 99.4% precision, 98.3% MCC and 0.1% false positive rate on an unbalanced dataset (15263:2916) in a real-time experiment.

We also gave a *tough* test to our method in real time using a much larger dataset (202056:2916) in 5-fold crossvalidation. As Table 8 shows, the model still achieves 98.2% precision and 93.5% MCC with 0.00025% false positive. For search-based features only, the performance degrades with larger unbalanced datasets (compare Tables 6, 7 and 8), because many legitimate websites are not among the top 50 retrieved results. Results without content-based features on Table 5 & 9 are also lower, proving their importance. However, we should mention that the

content-based features are developed based on personal web development experience of the authors and the pyschology of the lazy phisher. We also perform an online experiment without retraining to study their effectiveness on unseen data.

It is hard to directly compare with other methods, since the datasets are different. Nevertheless we note that CANTINA+ [12], Tan et al. [13], and Shirazi et al. [14] claim good results without using crossvalidation on smaller datasets (details are shown in Table 10). Huang et al. [15] uses a large dataset, but do not report false positive rate in their paper and 97% *accuracy* is not so high for a large unbalanced dataset. Samuel et al. [16] propose an impressive framework and their model has achieved relatively high accuracy, high MCC, and low false positive rate, but they use train/test instead of crossvalidation on a smaller dataset (150000:1216). The model proposed by Ma et al. [17] also achieves good results, but their dataset is unbalanced in an un-realistic scenario, i.e., more phishing URLs than legitimate ones.

Table 2: Balanced (4090:4090) - Offline

Feat.	Acc.	FP	Pre.	Rec.	F1	MCC	AUC
URL	0.824	0.174	0.825	0.821	0.823	0.647	0.866
Host	0.773	0.403	0.702	0.948	0.807	0.583	0.868
Time	0.873	0.125	0.874	0.872	0.873	0.747	0.875
Search	0.786	0.170	0.814	0.743	0.777	0.575	0.829
Content	0.952	0.046	0.954	0.950	0.952	0.905	0.989
All	0.978	0.022	0.978	0.979	0.978	0.957	0.997

Table 3: Unbalanced (21369:4090 $\approx$ 5:1) - Offline

Feat.	Acc.	FP	Pre.	Rec.	F1	MCC	AUC
URL	0.924	0.028	0.824	0.674	0.741	0.702	0.885
Host	0.916	0.008	0.926	0.522	0.667	0.657	0.868
Time	0.944	0.025	0.856	0.785	0.819	0.787	0.880
Search	0.877	0.076	0.613	0.626	0.620	0.546	0.855
Content	0.977	0.009	0.950	0.904	0.927	0.913	0.992
All	0.987	0.005	0.973	0.948	0.961	0.953	0.996

Table 4: Unbalanced (202056:4090 $\approx$ 49:1) - Offline

Feat.	Acc.	FP	Pre.	Rec.	F1	MCC	AUC
URL	0.984	0.003	0.698	0.376	0.489	0.506	0.805
Host	0.985	0.003	0.742	0.412	0.530	0.547	0.785
Time	0.981	0.009	0.543	0.520	0.531	0.522	0.757
Search	0.980	0.000	0.500	0.000	0.001	0.015	0.714
Content	0.993	0.001	0.953	0.704	0.810	0.816	0.976
All	0.995	0.001	0.961	0.800	0.873	0.874	0.988

Table 5: Unbalanced (202056:4090 $\approx$ 49:1) - Offline

Feat.	Acc.	FP	Pre.	Rec.	F1	MCC	AUC
NU	0.994	0.001	0.942	0.780	0.853	0.855	0.981
NH	0.995	0.001	0.945	0.782	0.856	0.858	0.982
NT	0.995	0.001	0.956	0.795	0.868	0.870	0.976
NS	0.995	0.001	0.953	0.807	0.874	0.875	0.985
NC	0.987	0.006	0.697	0.690	0.694	0.687	0.842

Table 6: Balanced (2916:2916) - Online

Feat.	Acc.	FP	Pre.	Rec.	F1	MCC	AUC
URL	0.849	0.126	0.868	0.823	0.845	0.698	0.891
Host	0.810	0.344	0.737	0.963	0.835	0.650	0.881
Time	0.870	0.123	0.875	0.862	0.869	0.739	0.739
Search	0.838	0.103	0.883	0.778	0.827	0.680	0.894
Content	0.970	0.038	0.963	0.978	0.971	0.941	0.993
All	0.987	0.015	0.985	0.987	0.986	0.972	0.998

Table 7: Unbalanced (15263:2916 $\approx$ 5:1) - Online

Feat.	Acc.	FP	Pre.	Rec.	F1	MCC	AUC
URL	0.929	0.022	0.856	0.672	0.752	0.720	0.895
Host	0.913	0.005	0.949	0.486	0.642	0.643	0.882
Time	0.956	0.013	0.918	0.796	0.853	0.830	0.892
Search	0.899	0.061	0.684	0.694	0.689	0.629	0.892
Content	0.986	0.004	0.976	0.939	0.957	0.949	0.997
All	0.995	0.001	0.994	0.978	0.986	0.983	0.999

Table 8: Unbalanced (202056:2916 $\approx$ 69:1) - Online

Feat.	Acc.	FP	Pre.	Rec.	F1	MCC	AUC
URL	0.998	0.002	0.650	0.282	0.393	0.423	0.762
Host	0.989	0.002	0.736	0.373	0.495	0.520	0.801
Time	0.986	0.007	0.508	0.490	0.499	0.492	0.742
Sear.	0.986	0.000	1.000	0.002	0.003	0.041	0.746
Cont.	0.997	0.00034	0.971	0.785	0.868	0.871	0.986
All	0.998	0.00025	0.982	0.892	0.934	0.935	0.996

Table 9: Unbalanced (202056:2916 $\approx$ 69:1) - Online

Feat.	Acc.	FP	Pre.	Rec.	F1	MCC	AUC
NU	0.998	0.001	0.957	0.874	0.914	0.914	0.988
NH	0.997	0.001	0.951	0.874	0.911	0.911	0.988
NT	0.998	0.00034	0.974	0.898	0.935	0.935	0.986
NS	0.998	0.00038	0.971	0.898	0.934	0.933	0.989
NC	0.991	0.004	0.680	0.648	0.664	0.659	0.822

4.4. Feature ranking

To evaluate feature efficacy, we only chose the top 20 features to train and test. The results shown in Table 11

Table 10: Comparison with Other Methods from Research Literature (NR = Not Reported)

Technique	Dataset (legit:phish)	Evaluation	Lang.	Acc.	FP	Pre.	Rec.	F1	MCC	AUC
Table 6	2916:2916 in real time	5-fold C.V.	Several	0.987	0.015	0.985	0.987	0.986	0.972	0.998
Table 7	15263:2916 in real time	5-fold C.V.	Several	0.995	0.001	0.994	0.978	0.986	0.983	0.999
Table 8	202056:2916 in real time	5-fold C.V.	Several	0.998	0.0003	0.982	0.892	0.934	0.935	0.996
Samuel et al. [16]	4531:1036 - PhishTank & Intel Security.	5-fold C.V.	English	0.990	0.001	0.991	0.957	0.974	0.968	0.998
Samuel et al. [16]	4531:1036 (train) and 150000:1216 (test)	train/test	Several	0.998	0.001	0.857	0.958	0.904	0.905	NR
Bahnsen et al. [18] R.F.	1M:1M - web crawl & PhishTank	3-fold C.V.	Several	0.935	0.064	0.936	0.933	0.934	0.869	0.984
Bahnsen et al. [18] LSTM	1M:1M - web crawl & PhishTank	3-fold C.V.	Several	0.987	0.014	0.986	0.989	0.987	0.975	0.999
Thomas et al. [19]	500K:500K - Monarch, Twitter, etc.	n-fold C.V.	Several	0.866	0.003	0.961	0.734	0.832	0.725	NR
CANTINA+ [12]	1868:940 - several different sources	train/test	Several	0.970	0.013	0.964	0.955	0.959	0.940	NR
Ma et al. [17]	15000:20500 - DMOZ	n-fold C.V.	Several	0.955	0.001	0.998	0.924	0.960	0.913	NR

are based on a balanced dataset (2916:2916) using Random Committee in 5-fold crossvalidation. Using Information Gain, PCA, correlation, or Symmetrical Uncertainty feature selection method to pick top 20 features, the model still achieved good performance. Therefore, reducing the number of attributes to achieve fast detection is achievable. For abbreviations of content-based features, please see this page.[5]

4.5. False positives

We checked the false positives manually in a small testing sample, which contains 100 legit and 100 phish URLs. We observed one out of three false positives ("https://www.paypal.com/signin?") was because many phishing URLs (ex. "http://dinas.tomsk.ru/language/") only existed for a very short time, then they redirected to the legitimate PayPal login page. Even if we created a whitelist to filter out these "legitimate" URLs from PhishTank, some of them redirected to uncommon or unknown legitimate websites, which were not included in our whitelist.

4.6. False negatives

We observed seven false negatives in one of our testing models because these phishing webpages are built on legitimate domains. For example, one of these phishing URLs was "https://sites.google.com/view /im-not-a-robot" based on legitimate domain "google.com."

[5]https://github.com/xzhou29/F_E/blob/master/dataset/list_of_content_based_features.txt

4.7. Content and search-based features

The precision, recall, and MCC shown in Tables 2 to 8 have proved that features we use from web development perspective actually work as well with Random Committee algorithm. Without using URL-based and host-based features, the models still achieve comparable performance. Since the purpose of this paper is to prove the efficiency of these content-based features, therefore we only use a few URL-based and host-based features. We believe our model could be further enhanced by adding more URL-based features, e.g., KL-Divergence, KS-Distance from [21] and [22] and others.

5. Related works

An influential search-based framework, CANTINA [23], uses TF-IDF scores of each term on the web page, then generates a lexical signature by taking the five terms with highest TF-IDF weights to feed into a search engine (Google). Detection is based on whether a domain of the current web page matches the domains of the top 30 search results. In the real world, there are many types of legitimate and phishing websites. For example, many new legitimate websites exist, which use very generic terms in their website content, e.g., non-profit websites do this frequently, and have no logos in the web content. Such domains may not be easy to find, *if that website is not popular*. Therefore, such

Table 11: Feature Ranking (see [20])

Method	Top features	FP	Pre.	Rec.	MCC	AUC
Information Gain	CF_6, CF_137, CF_1, numberOfHosts, CF_11, time_4, expiration_minus_creation, current_minus_creation, CF_93, CF_16, CF_3, CF_21, CF_140, CF_8, time_1, CF_99, CF_143, CF_136, CF_149, CF_15	0.035	0.966	0.974	0.939	0.970
Principle Component Analysis	CF_68, CF_83, CF_23, CF_21, CF_38, CF_27, CF_22, CF_2, CF_8, CF_12, CF_39, CF_19, CF_34, CF_24, CF_29, CF_96, CF_93, CF_65, CF_5, CF_110	0.032	0.842	0.892	0.840	0.942
Correlation	protocol, time_1, CF_99, domain_length, current_minus_creation, expiration_minus_creation, CF_136, CF_6, CF_141, CF_21, CF_68, hyphen_symbol, CF_38, CF_83, CF_7, CF_23, CF_93, CF_142, count_digits, CF_1	0.005	0.973	0.926	0.940	0.961
Symmetrical Uncert	CF_1, CF_21, CF_11, CF_6, protocol, time_1, numnberOfHosts, CF_99, CF_136, CF_117, CF_137, current_minus_creation, expiration_minus_creation, hyphen_symbol, CF_143, CF_81, CF_16, CF_3, CF_155, CF_76	0.006	0,967	0.947	0.949	0.974

methods tend to have a relatively high false positive rate, and must be complemented with other features. Although our search-based features are inspired by [13, 23], they are novel, since we look for domain emails and subdomains rather than keywords from the content. Another search-based method by Tan et al. [13] focuses on identifying brand names in HTML content to detect phishing websites, since phishers normally place brand names in different parts of a URL. They claim many effective features based on the result, but the dataset used is really small and legitimate URLs are collected manually. If a detector uses this method only, it will not work, since an attacker has full control on sub-domains that can increase importance of the words on URL weighting system. For example, if we build a phishing website with uncommon strings or patterns as domain name and then add these patterns on the web content, it will lead search engines to show the domain on top search results easily.

A hybrid framework CANTINA+ [12] uses a new method that leverages the high similarity among phishing web pages due to the prevalent use of phishing tool-kits, and examines a web page's similarity to known phishing attacks via hashing to filter highly similar phish. Their work tries to detect phishing pattern with a low false positive rate by filtering via heuristics. They also report run-time speedup because of this, but it is still slower compared to other methods especially in the feature extraction part. They implemented two filters to reduce false positive rate: a hash-based filter that compares a web page against known phish, and a login filter that checks a login field. As the researchers mentioned, hash-based filter actually is easy to beat, if an attacker modifies something on the web page. Login filter is based on three properties: Form Tag, Input Tag, and login keywords. These still can be beaten, if an attacker knows the rules of this filter. Other parts of this work are based on [23]. Buder et al. [24] use some machine learning algorithms and also add some visual similarities features with the help of natural language processing techniques.

There are more URL-based features such as patterns, character frequency and character distances in [15, 25, 21, 22] and visual similarity-based features in [26, 27, 28]. Another hybrid framework by Samuel et al. [16] uses 84 features based on calculation of the mean and standard deviation of URL length, mld length (mld = domain - top level domain), count of terms in URL, etc. Some of their features lack good justifications. However, this inspired our hypothesis that perhaps mean and standard deviation of different types of links could be used as features to predict phishing webpages, since a social engineering toolkit keeps the same format or uses external links all the time. Another hybrid approach by Dadkhah et al. [29] is based on classification algorithms that are capable of identifying different types of phishing page. Harshal et al. [30] investigate the effectiveness of three logistic regression classifiers using URL-based features based on bag-of-X representations. Ding et al. [31] use seven heuristic rules for detecting URL obfuscation and identifying phishing websites.

A content-based framework PhishMon [4] uses features based on capturing various characteristics of legitimate web applications as well as their underlying web infrastructures. Thakur et al. [5] focus on the fundamental characteristics of phishing web sites and

decompose the classification task for a phishing web site into URL classifier and content-based classifier. Yuan et al. [8] propose to extract features from URLs and webpage links to detect phishing website. Sahingoz et al. [7] also analyze the hyperlinks found in the HTML source code of the website. Li et al. [6] propose a stacking model to detect phishing webpages using URL and HTML features. Mao et al. [9] extract many features from the Cascading Style Sheet (CSS) of webpages, then select an effective feature set for similarity.

A URL-based classifier by Xue et al. [32] uses URL correlation, which is based on the similarity of URLs with the list that they create. Furthermore, two efficient URL-based classifiers by Verma et al. [21, 22] focus on characters on URL. Page et al. [33] uses URL shortener click analytics to compare the life cycle of phishing and malware attacks. Additionally, Yang et al. [34] propose a phishing detection approach using deep learning. Ludl et al. [35] report their findings on analyzing the effectiveness of two popular blacklist-based anti-phishing solutions. Anand et al. [36] train text generative adversarial network with URLs in the minority class. Bahnsen et al. [18] use recurrent neural network approach without the need of manual feature creation. Liang et al. [37] uses Bidirectional LSTM and report superior performance than other machine learning methods. Nagaraj et al. [38] use an ensemble machine learning model (RF_NN) for classifying phishing websites. Dong et al. [39] use public key certification to detect phishing websites in real time. Rao et al. [40] use key discriminative features extracted from the source code of the website to detect phishing websites and enhance the blacklist. Williams et al. [41] investigate the end users behavior when faced with phishing websites, then show a proof of concept computer model for simulating human behaviors. Zouina et al. [42] propose a lightweight SVM system to detect phishing websites. A novel method by Hara et al. [43] analyzes visual similarity among web pages without a priori knowledge. Mehdi et al. [44] propose a nonlinear regression algorithm with a feature selection method, harmony search. Furthermore, there are two useful surveys [10, 11] on phishing website detection.

6. Conclusions

We propose an effective phishing detection system using an ensemble learning method, Random Committee, based on a total of 126 novel features from web development perspective plus 45 other features. Although, some features are inspired by previous work, most are based on our observations and understanding. We compared the models using different machine learning techniques including Hellinger Distance Decision Trees, SVM, Naive Bayes, Logistic Regression, J48, Random Forest, and Random Committee (Random Tree). The ensemble learning method Random Committee with base learner Random Tree had the best performance, thus we use it for further study. The model achieved solid performance in a difficult test, proving the effectiveness of our features.

Acknowledgments. We thank S. Baki and the reviewers for insightful comments. Research supported in part by NSF grants CNS 1319212 and DGE 1433817, and by ARO grant W911NF-16-1-0422.

References

[1] A. El Aassal, L. Moraes, S. Baki, A. Das, and R. Verma, "Anti-phishing pilot at ACM IWSPA 2018: Evaluating performance with new metrics for unbalanced datasets," in *Proceedings of 1st Anti-Phishing Pilot IWSPA-AP*, July 2018.

[2] D. A. Cieslak, T. R. Hoens, N. V. Chawla, and W. P. Kegelmeyer, "Hellinger distance decision trees are robust and skew-insensitive," *Data Mining and Knowledge Discovery*, pp. 136–158, 2012.

[3] QUICKSPROUT, "34 ways to improve SEO rankings in 2019." quicksprout.com [Online; posted 2019].

[4] A. Niakanlahiji, B. Chu, and E. Al-Shaer, "Phishmon: A machine learning framework for detecting phishing webpages," in *2018 IEEE International Conference on Intelligence and Security Informatics*, IEEE, 2018.

[5] T. Thakur and R. Verma, "Catching classical and hijack-based phishing attacks," in *International Conference on Information Systems Security*, pp. 318–337, Springer, 2014.

[6] Y. Li, Z. Yang, H. Y. X. Chen, and W. Liu, "A stacking model using url and HTML features for phishing webpage detection," in *Future Generation Comp. Syst.*, pp. 27–39, 2019.

[7] O. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, pp. 345–357, 2018.

[8] H. Yuan, X. Chen, Y. Li, Z. Yang, and W. Liu, "Detecting phishing websites and targets based on URLs and webpage links," in *24th International Conference on Pattern Recognition, ICPR 2018*, pp. 3669–3674, 2018.

[9] J. Mao, W. Tian, P. Li, T. Wei, and Z. Liang, "Phishing website detection based on effective CSS features of web pages," in *WASA*, pp. 804–815, 2017.

[10] G. Varshney, M. Misra, and P. K. Atrey, "A survey and classification of web phishing detection schemes," *Security and Communication Networks*, October 2016.

[11] D. Sahoo, C. Liu, and S. C. Hoi, "Malicious URL detection using machine learning: A survey," *CORR*, vol. abs/1802.02871, 2018.

[12] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "CANTINA+: A feature-rich machine framework for detecting phishing web sites," in *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, ACM, 2011.

[13] C. L. Tan, K. L. Chiew, and S. N. Sze, "Phishing website detection using URL-assisted brand name weighting system," in *ISPACS*, IEEE, 2014.

[14] H. Shirazi, B. Bezawada, and I. Ray, "Kn0w thy doma1n name: Unbiased phishing detection using domain name based features," in *SACMAT '18 Proceedings of the 23rd ACM on Symposium on Access Control Models and Technologies*, pp. p. 69–75, ACM, 2018.

[15] D. Huang, K. Xu, and J. Pei, "Malicious URL detection by dynamically mining patterns without pre-defined elements," in *World Wide Web - Internet and Web Information Systems*, vol. 17, pp. 1375–1394, 2014.

[16] S. Marchal, K. Saari, N. Singh, and N. Asokan, "Know your phish: Novel techniques for detecting phishing sites and their targets," in *36th International Conference on Distributed Computing Systems*, IEEE, August 2016.

[17] J. Ma, L. K. Saul, S. Savage, , and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," in *in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1245–1254, 2009.

[18] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. Gonzlez, "Classifying phishing URLs using recurrent neural networks," in *2017 APWG Symposium on Electronic Crime Research (eCrime)*, IEEE, 2017.

[19] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time URL spam filtering service," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy (SP)*, pp. 447–462, 2011.

[20] J. Alamelu Mangai, V. Santhosh Kumar, and S. Appavu alias Balamurugan, "A novel feature selection framework for automatic web page classification," *International Journal of Automation and Computing*, pp. 442–448, 2012.

[21] R. Verma and A. Das, "What's in a URL: Fast feature extraction and malicious URL detection," in *IWSPA '17 Proceedings of the 3rd ACM International Workshop on Security and Privacy Analytics*, ACM, 2017.

[22] R. Verma and K. Dyer, "On the character of phishing URLs: Accurate and robust statistical learning classifiers," in *CODASPY '15 Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pp. p. 111–122, ACM, 2015.

[23] Y. Zhang, J. Hong, and L. Cranor., "CANTINA: A content-based approach to detecting phishing web sites," in *WWW '07*, pp. 639–648, ACM, 2007.

[24] E. Buber, B. Diri, and O. K. Sahingoz, "NLP based phishing attack detection from URLs," in *17th International Conference on ISDA, Delhi, India*, pp. 608–618, 2017.

[25] A. Subasi, E. Molah, F. Almkallawi, and T. J. Chaudhery, "Intelligent phishing website detection using random forest classifier," in *2017 International Conference on Electrical and Computing Technologies and Applications*, IEEE, 2017.

[26] T. Chen, T. Stepan, S. Dick, and J. Miller, "An anti-phishing system employing diffused information," *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 4, pp. 16:1–16:31, 2014.

[27] S. Sarika and V. Paul., "Agenttab: An anti-phishing framework to defend tabnabbing attack," in *International Conference on Security and Authentication*, pp. 132–135, 2014.

[28] J. Mao, P. Li, K. Li, T. Wei, and Z. Liang, "Baitalarm: detecting phishing sites using similarity in fundamental visual features," in *Intelligent Networking and Collaborative System*, pp. 790–795, 2013.

[29] M. Dadkhah, S. Shamshirband, A. Wahid, and A. Wahab, "A hybrid approach for phishing web site detection," in *The Electronic Library*, pp. 927–944, 2016.

[30] H. Tupsamudre, A. K. Singh, and S. Lodha, "Everything is in the name a URL based approach for phishing detection," in *Cyber Security Cryptography and Machine Learning*, pp. 231–248, Springer International Publishing, May 2019.

[31] Y. Ding, N. Luktarhan, K. Li, and W. Slamu, "A keyword-based combination approach for detecting phishing webpages," *Computers & Security*, p. 256275, May 2019.

[32] Y. Xue, Y. Li, Y. Yao, X. Zhao, J. Liu, and R. Zhang, "Phishing sites detection based on URL correlation," in *4th International Conference on Cloud Computing and Intelligence Systems*, IEEE, 2016.

[33] S. Page, G. Jourdan, G. con Bochmann, J. Flood, and I. Onut, "Using URL shorteners to compare phishing and malware attacks," in *eCrime*, pp. 1–13, IEEE, 2018.

[34] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," *IEEE Access*, Jan 2019.

[35] C. Ludl, S. McAllister, E. Kirda, and C. Kruegel, "On the effectiveness of techniques to detect phishing sites," in *DIMVA : Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 20–39, 2007.

[36] A. Anand, K. Gorde, J. Moniz, N. Park, T. Chakraborty, and B. Chu, "Phishing URL detection with oversampling based on text generative adversarial networks," in *BigData*, pp. 1168–1177, IEEE, 2018.

[37] Y. Liang, D. Bao, and J. Cui, "Bidirectional LSTM: An innovative approach for phishing url identification," in *Intelligent Technologies and Robotics*, pp. 326–337, Springer, Cham, June 2019.

[38] K. Nagaraj, B. Bhattacharjee, A. Sridhar, and G. S. Sharvani, "Detection of phishing websites using a novel twofold ensemble model," in *Systems and IT 20(3)*, pp. 321–357, 2018.

[39] Z. Dong, A. Kapadia, J. Blythe, and L. J. Camp, "Beyond the lock icon: real-time detection of phishing websites using public key certificates," in *2015 APWG Symposium on Electronic Crime Research (eCrime)*, May 2015.

[40] R. Rao and A. Pais, "An enhanced blacklist method to detect phishing websites," in *ICISS*, pp. 323–333, 2017.

[41] N. Williams and S. Li, "Simulating human detection of phishing websites: An investigation into the applicability of the ACT-R cognitive behaviour architecture model," in *CYBCONF*, pp. 1–8, 2017.

[42] M. Zouina and B. Outtaj, "A novel lightweight URL phishing detection system using svm and similarity index," *Human-centric Computing and Information Sciences*, vol. 7, p. 17, 2017.

[43] M. Hara, A. Yamada, and Y. Miyake, "Visual similarity-based phishing detection without victim site information," in *Symposium on Computational Intelligence in Cyber Security*, IEEE, 2009.

[44] M. Babagoli, M. Pourmahmood, Aghababa, and V. Solouk, "Heuristic nonlinear regression strategy for detecting phishing websites," in *Soft Computing*, pp. 4315–4327, Springer Berlin Heidelberg, June 2019.