Learning to Detect Phishing Webpages

Ram B. Basnet*
Colorado Mesa University
Grand Junction, Colorado, USA
rbasnet@coloradomesa.edu

Andrew H. Sung
University of Southern Mississippi
Hattiesburg, Mississippi, USA
andrew.sung@usm.edu

Abstract

Phishing has become a lucrative business for cyber criminals whose victims range from end users to large corporations and government organizations. Though Internet users are generally becoming more aware of phishing websites, cyber scammers come up with novel schemes that circumvent phishing filters and often succeed in fooling even savvy users. Recent studies to detect phishing and malicious webpages using features from URLs alone show promise. The approach, however, may not be reliable and robust enough to detect evolving sophisticated phishing webpages. For examples, phishers can use URL shortening services to masquerade their phishing URLs, or use compromised legitimate websites to host their phishing campaign. Along with the features from URLs, we propose many novel content based features and apply cutting-edge machine learning techniques to demonstrate that our approach can detect phishing webpages with error rates 0.04-0.44%, false positive and false negative rates of 0.0-0.30% and 0.06-0.73% respectively on real-world data sets using Random Forests classifier, thereby improving previous results on the important problem of phishing detection.

Keywords: phishing attack, phishing webpages, content-based approach, batch learning, online learning

1 Introduction

In a typical phishing attack, a phisher sends out deceptive emails pretending to come from a reputable institution, e.g., a bank. The phisher urges the user to click on a link to fraudulent site where user is asked to reveal private information such as password, bank account information, credit card number, social security number, etc. Whittaker et al. [42] define a phishing webpage as "any webpage that, without permission, alleges to act on behalf of a third party with the intention of confusing viewers into performing an action with which the viewers would only trust a true agent of the third party." This definition, which is similar to the definition of "web forgery", covers a wide range of phishing pages from typical ones – displaying graphics relating to a financial company and requesting a viewer's personal credentials – to sites which claim to be able to perform actions through a third party once provided with the viewer's login credentials. In this paper, we use the same definition of phishing and propose a novel method to detect such phishing webpages.

Besides financial institutions and online payment services, phishers are increasingly targeting various other popular online brands such as gaming and social networking sites. Phishers targeted gaming site in the first half of 2010 reaching a high of 16.7% of all impressions in June. The percentage of active phishing sites that targeted social networks increased during the final months of the year accounting for 4.2% of active sites in December, despite receiving 84.5% of impressions that month, according to Microsoft Security Intelligence Report [2].

Journal of Internet Services and Information Security (JISIS), volume: 4, number: 3, pp. 21-39

*Corresponding author: Department of Computer Science, Mathematics and Statistics, Colorado Mesa University, Grand Junction, CO, 81501, USA, Tel: +1-970-248-1680, Web: myhome.coloradomesa.edu/~rbasnet

Due to the strategic position of the browser and the concentration of the browser market, web browsers play a key role in defending the end users against phishing by warning them directly and effectively. However, studies show that blacklists were ineffective when protecting users initially, as most of them caught less than 20% of phish at hour zero [29, 39]. It has been found that two tools using heuristics to complement blacklists caught significantly more phish initially than those using only blacklists [29]. Whitelist approach, on the other hand, maintains a list of known "good" URLs. It is, however, very difficult to maintain the list of a large number of variants of legitimate sites. Moreover, phishers may compromise a legitimate site and host their phishing campaign on an otherwise popular and benign website. The methods needed to address these shortcomings should be discovery-oriented, dynamic, and semi-automated.

To that end, we propose a set of heuristics that can be used to evaluate the "phishy" nature of a webpage. These heuristics are rooted in the evaluation of data commonly available from search engines, phishing and malware related web resources, hosting servers, and webpage contents. Some of these features are based on historical statistics and reports published by trustworthy and well-known online sources. While we borrow some popular features from existing research, we also propose a number of novel discriminative features based on text and HTML contents of phishing webpages. We ran our experiments on a datasets of more than 16,000 phishing and 24,000 non-phishing webpages comparing several batch and online learning algorithms. We experimentally demonstrate that our approach can obtain error rate of less than 0.5% while still maintaining low false positive rate of less than 0.3% and false negative rate of less than 0.7%.

2 Our Method

In this section, we provide a detailed discussion of our approach to detecting phishing webpages. We begin with an overview of the classification problem, followed by a discussion of the generation of our datasets, features we extract, and finally the set of machine learning classifiers we use in our experiments to evaluate our methodology.

2.1 Method Overview

We propose a machine learning based approach to classifying phishing webpages by using the information available on URLs, hosting servers, and page contents. We treat the problem of detecting phishing webpages as a binary classification problem where we classify phishing webpages from legitimate non-phishing ones. We first run a number of scripts to collect our phishing and non-phishing URLs, automatically fetch their page contents and create our data sets. Our next batch of scripts then extracts a number of features by employing various publicly available resources in order to classify the instances into their corresponding classes. We then apply machine-learning algorithms to build models from training data, which is comprised of pairs of feature assignments and class labels. Separate sets of test data are then supplied to the models, and the predicted class of the data instance (phishing or non-phishing) is compared to the actual class of the data to compute the accuracy and various other performance measures of the classification models. Figure 1 shows the graphical representation of our phishing webpage detection methodology.

2.2 Data Sets

Public and raw benchmark data sets for phishing websites seem to be scarce in literature. Hence, we decided to collect our own data from various popular and credible online sources. For phishing webpages, we wrote Python scripts to automatically download confirmed phishing URLs from PhishTank

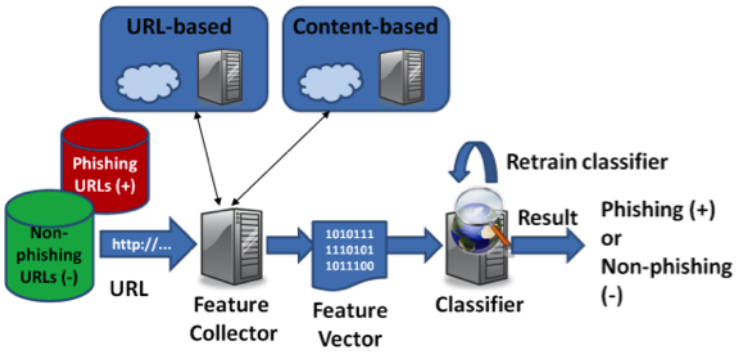

Figure 1: Graphical representation of phishing webpage detection methodology

[4]. PhishTank, operated by OpenDNS, is a collaborative clearinghouse for data and information about phishing on the Internet. A phish once submitted is verified by a number of registered users to confirm it as phishing. We collected first set of phishing URLs from June 1 to October 31, 2010. Phishing tactics used by scammers evolve over time. In order to investigate these evolving tactics and to closely mimic our experiments as the real world in the wild scenario, we collected second batch of confirmed phishing URLs that were submitted for verification from January 1 to May 3, 2011. We wrote scripts [5] to automatically detect and expand the shortened URLs provided by online service longurl.org.

We collected our legitimate webpages from two public data sources. One is the Yahoo! Directory, the web links in which are randomly provided by Yahoo's server redirection service [7]. We used this service to randomly select a URL and download its page contents along with server header information. In order to cover wider URL structures and varieties in page contents, we also made a list of URLs of most commonly phished targets. We then downloaded those URLs, parsed the retrieved HTML pages, and harvested and crawled the hyperlinks therein to also use as benign webpages. We made the assumption, which we think is reasonable, to treat those webpages as benign, since their URLs were extracted from a legitimate sources. These webpages were crawled between September 15 and October 31 of 2010. The other source of legitimate webpages is the DMOZ Open Directory Project[1]. DMOZ is a directory whose entries are vetted manually by editors.

Based on the date on which phishing URLs were submitted to PhishTank for verification, we generated two data sets. The first data set, we refer to it as OldPhishTank, contains 11,240 phishing webpages submitted before October 31, 2010. The second data set, we refer to it as NewPhishTank, contains 5,454 phishing webpages submitted for verification between January 1st and May 3rd of 2011. Combining these phishing data sets with non-phishing webpages from 2 sources, we generate our 5 data sets, which are summarized in Table 1.

Features were collected as soon as a webpage was crawled and successfully downloaded. We discarded the URLs that were no longer valid as the page couldn't be accessed to extract features from their contents.

2.3 Features

Anywhere from a handful to tens of thousands of features have been proposed and used in classifying phishing webpages. We developed our set of 179 features based on related works, drawing primarily from existing literatures [12, 13, 14, 15, 23, 24, 30]. Nevertheless, we also propose many novel server and content-based highly relevant features. The use of relatively small number of fixed set of features

[1] http://www.dmoz.org

Data set	# Phishing webpages	# Non-phishing webpages	Total
OldPhishTank-Yahoo (OY)	11,240	21,946	33,186
NewPhishTank-DMOZ (ND)	5,454	9,635	15,089
OldPhishTank-DMOZ (OD)	11,240	9,635	20,875
NewPhishTank-Yahoo (NY)	5,454	21,946	27,400
All data sets (OYND)	16,694	31,581	48,275

Table 1: Summary of data sets

Feature category	Feature count
Lexical based	24
Keyword based	101
Search Engine based	6
Reputation based	8
Content based	40
Full set	179

Table 2: Feature category and number of features in each category

makes the decision boundaries less complex, and therefore less prone to over-fitting as well as faster to evaluate.

We group features that we gather into 2 broad categories: URL based and content based features. We briefly describe each feature category in the following subsections. We summarize feature categories with number of features in each category in Table 2.

2.3.1 URL-based Features

URL-based features are extracted from the webpage's URL and its meta-data. For clarity and to better understand the types of tactics used by phishers, URL-based features are further grouped into 4 broad categories and briefly describe them next. These URL-based features are similar to the ones we used in our previous work [15].

Lexical based features. Lexical features, the textual properties of the URL itself, have widely been used to detect phishing emails and malicious websites [23, 24, 30, 43]. Phishers usually obfuscate URLs to trick users into thinking that the malicious URL belongs to a legitimate website users are familiar with. For example, they take advantage of "typo squatting" – using a domain closely related to the actual target's Web domain or putting the domain name or the target name in URL's path or sub-domain.

Phishers also obfuscate URLs with certain characters such as '-', soft hyphen, Unicode, and visually similar looking characters. We try to identify these tactics and use them as binary features in our classifiers. For instance, we check if certain characters such as '-', '_', '=', '@', digits, and non-standard port, etc., are present in the URL. There are 24 real and binary features in this category.

Keyword based features. Phishing URLs are found to contain several eye-catching word tokens. For example, suggestive words such as "login" and "signin" are very commonly found in phishing URLs. After calculating the mutual information (MI) [32] of each unique term extracted from phishing URLs, we applied performance based feature selection technique to come up with 101 keyword-based features.

After ranking all the unique terms based on the MI values in descending order, we applied Naïve Bayes classifier to search the subset of features using forward selection technique that yielded the best accuracy result [26]. These are Boolean features that state if a root term is present or not in a URL. Keyword based features include terms such as 'log', 'pay', 'update', 'warn', 'cmd' 'free', 'access', 'bonus', 'bank', 'user', etc. 8 red flag keyword based features have also been used in [24].

Search engine based features. In order to gather search engine based features, we check if a URL and its domain exist in search engines' index. We employ top three search engines: Google, Bing, and Yahoo!. First, we search for the whole URL and retrieve top 30 results. We used top 30 results because experiments show that going beyond don't improve classification accuracy [43]. If the results contain the URL, we consider it as a potentially benign URL, phishing otherwise. We also check if the domain part of URL matches the domain part of the result links. Similarly, if there is a match, we flag the URL as a potentially legitimate URL. Otherwise, we query the search engine again with just the domain part of the URL. If none of the returned link matches the search query, we flag the webpage as potentially phishing. Garera et al. [24] and Whittaker et al. [42] have used Google's proprietary page rank and index tables to generate their features.

If both the URL and the domain do not exist in search engines' index, it is a high indication that the domain is a newly created one and thus more likely to be phishing. There are 6 search engine based binary features.

Reputation based features. The idea behind reputation based features is to make use of the historical statistical data on top IPs and domains that have the bad reputation of hosting the most phishing web sites. If a webpage has many other phishing related heuristics and also its host belongs to top IP and/or domain that has historic reputation of hosting most phishing web sites, then we can increase our confidence level to classify the URL at hand as a phishing one.

We use 3 types of statistics, Top 10 Domains, Top 10 IPs, and Top 10 Popular Targets published by PhishTank [4].

We also use top 50 IP address report produced by StopBadware.org [6]. We check if the IP address of a URL belongs to this top 50 report and flag it as potentially phishing if it does.

Besides, we use hpHosts to check if the domain of a URL exists in its database. hpHosts is a community managed and maintained hosts file that allows an additional layer of protection against access to ad, tracking and malicious websites [8].

We use Safe Browsing API [1] to check URLs against Google's constantly updated blacklists of suspected phishing and malware pages and use 3 binary features for membership in the blacklists provided by the API.

There are 8 reputation based binary features.

2.3.2 Content-based Features

Our preliminary experiments in classifying phishing websites using features based on URLs show some promising results. Can the content-based features help to improve accuracy in phishing webpage classification by reducing false positives and negatives? We try to find the answer to that question in this paper. An ingenious phishing webpage resembles the look and feel of the target legitimate website. In fact, it will be extremely hard for users to distinguish a phishing site from its target site without aid of other browser clues. Data indicates that some phishing attacks have convinced up to 5% of their recipients to provide sensitive information to spoofed websites [20].

Some content-based features have been previously used in related works [34, 42, 43]. Augmenting the features based on a URL, we propose several novel features based on the technical (HTML) – as

opposed to text-based features used in related works – contents of a phishing webpage. Besides page contents, our web crawlers download server header information and extract certain features which, we think, can help classifiers in better discriminating phishing webpages from non-phishing ones. Nevertheless, we also propose some novel and discriminative text based features to detecting phishing webpages.

We briefly describe our content-based features in the following paragraphs.

TITLE tag. Most webpages (phishing or non-phishing) usually contain a TITLE tag. Essentially, the TITLE tag's text is what is displayed as text for links returned from major search engines (Google, Yahoo and Bing). Zhang et al. select the top 5 words with highest TF-IDF value to generate lexical signature of a page [43]. They feed each lexical signature to Google search engine and check if the domain name of the current webpage matches the domain name of the top 30 results. If yes, they consider it to be a legitimate website. We use webpage's TITLE text as the signature of the page and search Google using the title of the page. We then check if the whole URL and domain part of the current webpage matches the URL and domain name of the top 30 results. If yes, we consider it to be a legitimate webpage. We also check if any of the top targets is present in the title. We stick to top 30 results because Zhang et al. showed that going beyond top 30 didn't improve the classification performance [43].

FORM tag. Most phishing webpages have a FORM tag for potential victims to submit their personal information. In fact 99.3% of phishing webpages had one or more FORM tag(s) in them. Phishers tend to bypass the hassle of setting up SSL/TLS and simply follow easy and insecure method to send the form data back to them. Using 'get' method to send form data is insecure way to send password or any sensitive data as the data are encoded by the browser into the URL and which in turn is logged by HTTP(S) server in plain text besides most likely being stored in client's browser history [27]. If a legitimate website is hacked, phishers can modify the form's target to send data to different domain that is under their control. Since online account password is one of the most common confidential information phishers are after, we check whether a page contains the keyword password as an input type attribute inside a form tag.

Server and client redirection. Cyber criminals may compromise a legitimate Web server and redirect the traffic to a malicious website using HTTP redirect or by Pharming. Pharming is a type of attack which is conducted either by changing the hosts file on a victim's computer or by exploitation of vulnerability in DNS server software [40]. Phishers may poison DNS entries for a target website on a given DNS server, replacing them with the IP address of a server they control. In order to circumvent anti-phishing technologies phishers may use META tag's refresh property to make the client's browser automatically load a phishing page that scammer controls. Moreover, phishers may use obfuscated JavaScript to dynamically generate the HTML page with just a META refresh tag with the URL of a phishing webpage as target [11].

In order to evade detection, phishers may setup a large number of webpages to redirect traffic from one page to another before taking potential victims to the final phishing page. Victims, as in other web based malware attacks, e.g., drive-by-download attacks [18], are often sent through a long chain of redirection operations making it more difficult to track down an attack, notify all the involved parties (e.g., registrars and providers), and ultimately, take down the offending sites. We anticipated this tactic and investigated it in our data set to use as features. We record the number of times the browser is redirected; for example, by the server or using META refresh tag. "Number and target of redirections" is one of the 10 features used in detecting drive-by-download attacks [18]. Since the META refresh tag's URL is the final destination of phishing scam, our web crawlers download the destination URL and check it against the Google's blacklists [1].

Other tags. Due to Cross Site Scripting vulnerability (XSS) [3] or other vulnerabilities in a website, hackers may be able to inject IFRAME tag and force victim's browser to unknowingly load a malicious webpage inside an otherwise legitimate page. We count anchor tags for both internal (within the same domain) and external (to other domain) links. The idea is that phishers, for convenience, usually copy just one page (normally a login page) of the target website and simply leave all other objects (such as CSS, scripts, images, etc.) pointing back to the target website. We, therefore, expected to have more external links than internal links in phishing Webpages.

IFRAMEs are the most prevalent type of attack that uses HTML and JavaScript [33, 2]. We count the number of IRFAME tags on a webpage and use them as a real valued feature. Furthermore, we check IFRAME's source URLs against the Google's blacklists and also check if they are indexed by Google search engine by querying search engine with the complete URL and checking if any of resulting links match with the searched URL. We found that most of the IFRAME tags in non-phishing webpages were for displaying banner ads while most of the IFRAME tags in phishing webpages were for displaying part of the webpage perhaps to invade filters. Interestingly, no IFRAME source URLs from both phishing and non-phishing webpages were in Google's blacklist.

Page structure. Most browsers are forgiving and usually have high tolerance for missing tags or non-standard and invalid markups. Even though phishing webpages need to be rendered well and must resemble very close to their targets visually, we anticipated that sophisticated phishers would exploit this browser feature. In order to break HTML parser and evade detection, phishers may deliberately churn poorly structured obfuscated HTML codes. We modified and used the Beautiful Soup parser [36] to parse HTML content and extract content-based features. One of the most powerful features of Beautiful Soup is that it won't choke on bad markup. It parses a (possibly invalid) XML or HTML document into a tree representation. We observed that our parser failed to parse 27.8% of phishing webpages and 21.9% of legitimate webpages. Surprisingly, relatively a large percentage of developers of legitimate websites were not following industry-standard best programming practices. We used regular expressions to collect features from webpages that the parser was unable to parse.

Using these URL based, server based, and content based features, we encode each webpage into a feature vector with 179 dimensions. Most of the lexical based and all of the keyword-based features are generated by the "bag-of-words" representation – the order of the word as they appear on the URL is not taken into consideration. All real-valued features are scaled to fall between 0 and 1, thus giving equal weight to each feature which makes the max and min value for each feature to be 1 and 0 respectively. The phishing examples are labeled as 1 and legitimate ones are labeled as -1.

2.3.3 Machine Learning Algorithms

Since no single classifier is perfect, we evaluate several popular supervised classifiers, specifically the ones that have been applied to problems similar to ours. Empirically comparing a large number of classifiers we want to find out the best suitable one for detecting phishing webpages. As researchers, we have no vested interest in any particular classifier. We simply want to empirically compare a number of classifiers based on their availability in implementation and determine the one that yields the best performance in terms of both training time and accuracy to the problem of detecting phishing webpages.

We evaluate a set of 6 batch learning algorithms: Support Vector Machines (SVM) [41] with rbf and linear kernels, Random Forests [16], Naïve Bayes [38], C4.5 [35], Logistic Regression (LR) [28] and a set of 5 online learning algorithms: updatable version of Naïve Bayes (NB-U), updatable version of LogitBoost (LB-U), Perceptron [37], Passive-Aggressive (PA) [22], and Confidence-Weighted (CW) [19] algorithms.

3 Empirical Evaluations

We used 10 times 10-fold cross-validation (unless otherwise stated) to estimate the test error and other classifier performance criteria. The experiments were run on a machine with 2 dual-core 2 GHz Intel processors with 4 GB memory. To conduct all the experiments, we used WEKA (Waikato Environment for Knowledge Analysis) [25] and CW libraries [19, 22]. C4.5 classifier is named as J48 in WEKA and we use the terms C4.5 and J48 interchangeably throughout this paper.

3.1 Classifier Performance Criteria

There are several metrics to measure the quality of binary classification models. We report the standard ones that are briefly described below.

Error rate. Error rate is the total number of incorrectly classified instances among all webpages available during the test.

False positive rate (FPR). FPR determines a classifier's test performance on classifying non-phishing webpages incorrectly as phishing among all non-phishing webpages available during the test.

False negative rate (FNR). FNR determines a classifier's test performance on classifying phishing webpages incorrectly as non-phishing. When unsuspecting users rely on automated system to detect phishing webpages, it is normally preferred to have very low or rather 0 FNR. Users may rather visit a webpage with caution, rather than the false sense of security.

F1-measure. F1 score or F1-measure is the harmonic mean of precision and recall. In F1-measure, precision and recall are equally weighted. For better comparison, we show all three measures in our results.

Mathew's correlation coefficient (MCC). Since we have an unbalanced dataset (number of phishing webpages is not equal to the number of non-phishing webpages), we also present MCC. MCC takes into account true and false positive and negative and is generally regarded as a balanced measure that can be used even if the classes are of very different sizes [10]. MCC gives coefficient value between -1 and +1 where a coefficient of +1 represents a perfect prediction, 0 a random guess, and -1 an inverse prediction.

Balanced error rate (BER). BER, one of the main judging criteria in feature selection [17], is also used to evaluate classifiers' performance. BER is defined as the average of the error rate on positive class examples and the error rate on negative class examples. If there are fewer positive examples, the errors on positive examples will count more.

3.2 Classifier Evaluation

In these experiments, we evaluate the classifiers on each of our data sets using full features. Table 3 summarizes the performance results of all the classifiers. The non-shaded rows are performance results from batch learning algorithms while the shaded rows are results from the online learning algorithms.

The differences in performance results from all the classifiers are statistically insignificant. The best value for each performance metric is boldfaced. Overall, Random Forests (RF) performed the best

Classifier	OY			OD			NY			ND		
	Error	FPR	FNR	Error	FPR	FNR	Error	FPR	FNR	Error	FPR	FNR
RF	0.42	0.26	0.73	**0.04**	0.01	**0.06**	0.15	**0.07**	**0.44**	0.23	**0.00**	0.62
J48	**0.32**	**0.18**	**0.58**	0.06	0.02	0.10	0.35	0.16	1.10	0.25	0.01	0.68
SVM-lin	0.42	0.21	0.81	0.17	0.04	0.28	0.96	0.34	3.45	**0.19**	**0.00**	**0.53**
LR	0.47	0.30	0.81	0.26	0.13	0.37	0.72	0.29	2.44	0.44	0.31	0.66
SVM-rbf	0.86	0.52	1.54	0.35	**0.00**	0.65	1.48	0.48	5.50	2.74	0.65	6.42
NB	2.20	2.22	2.17	1.02	0.22	1.70	4.15	4.44	2.95	0.99	0.47	1.91
PA	0.57	0.37	0.96	0.17	0.11	0.21	1.22	0.48	4.18	0.25	0.04	0.62
LB-U	0.88	0.58	1.48	0.15	0.06	0.23	0.34	0.14	1.14	0.43	0.05	1.10
Perceptron	0.68	0.59	0.85	0.22	0.22	0.21	1.87	0.80	6.18	0.48	0.45	0.53
CW	0.57	0.46	0.78	0.24	0.26	0.22	1.86	1.33	4.00	0.23	0.03	0.57
NB-U	2.20	2.22	2.17	1.02	0.22	1.70	4.15	4.44	2.97	0.83	0.25	1.85

Table 3: Comparison of batch and online learning algorithms on the four data sets (OY, OD, NY, ND). The non-shaded rows are results from the batch and shaded ones are from the online-learning algorithms

followed by J48 and SVM-linear. Among batch classifiers, Naïve Bayes performed the worst followed by SVM-rbf and Logistic Regression.

In Table 4, we show performance metrics comparing batch and online learning algorithms on the combined data set (OYND). Among the online algorithms, overall Passive Aggressive (PA) algorithm yielded the best performance results followed by updatable version of LogitBoost (LB-U). When comparing between batch and online learning groups, most batch algorithms yield superior classification results though the differences are not that significant.

Additionally, in order to investigate the tradeoff between the accuracy and the time taken to build the model, we compare training time of each classifier on all the data sets in Table 5. Naïve Bayes is widely used in spam filters and related security applications partly because the training and testing time is fast. We see similar results in our experiments with Naïve Bayes showing the fastest time in most of the data sets to build models. Among online algorithms, Passive Aggressive (PA) and Perceptron took similarly least training time. Overall, online algorithms took less time compared to batch algorithms to build models.

Overall, we can argue that Random Forests (RF) has the best tradeoff for training time and overall accuracy in all the data sets.

3.3 Feature Evaluation

In these experiments, we explore the discriminative power and importance of each feature and feature category along with various combinations of feature sets. Our preliminary experimental results show that URL based features are effective in detecting phishing URLs. Particularly, as the focus of this paper is on the content-based features, we investigate the benefit of using content-based features along with the URL based features. Figure 2 compares the classification error rates on the four data sets using 6 different sets of features. For brevity, we only show detail results from Random Forests (RF) classifier, which yields overall both low error rates and also relatively faster training time (see section 3.2). However, the other classifiers also produced similar results on the data sets. Table 6 shows the total number of features in each feature set for the whole (OYND) data set.

Search engines based feature set yields the best error rates among all the individual feature categories in all the data sets. Content based feature set yields the second best results. Combined full feature set,

Classifier	Error	FPR	FNR	Precision	Recall	F-Measure	MCC	BER
J48	**0.41**	**0.24**	0.75	**99.55**	99.25	**99.40**	**0.991**	**0.49**
RF	0.44	0.30	**0.70**	99.43	**99.30**	99.36	0.990	0.50
LR	0.75	0.42	1.36	99.20	98.64	98.92	0.984	0.89
SVM-lin	0.77	0.38	1.50	99.28	98.50	98.89	0.983	0.94
SVM-rbf	1.17	0.54	2.38	98.97	97.62	98.29	0.974	1.46
NB	2.95	3.08	2.69	94.34	97.31	95.80	0.936	2.89
PA	**1.00**	**0.61**	**1.74**	**98.84**	**98.26**	**98.55**	**0.978**	**1.18**
LB-U	1.25	0.84	2.02	98.41	97.98	98.19	0.972	1.43
Perceptron	1.35	1.10	1.84	97.93	98.16	98.05	0.970	1.47
CW	1.36	1.00	2.03	98.10	97.97	98.04	0.970	1.52
NB-U	2.95	3.08	2.69	94.34	97.31	95.80	0.936	2.89

Table 4: Comparison of batch and online learning algorithms on the combined data set (OYND). The non-shaded rows are results from the batch and shaded ones are from the online learning algorithms

Classifier	OY	OD	NY	ND	OYND
J48	31.7 m	6.9 m	45 m	10.8 m	127.6 m
RF	10 m	3.7 m	9.6 m	3.9 m	25.7 m
LR	4.6 m	1.8 m	8.9 m	**0.6 m**	16.1 m
SVM-lin	3.2 m	**0.9 m**	5.7 m	14.5 m	12.5 m
SVM-rbf	6 m	1.8 m	7.6 m	21.6 m	18.1 m
NB	**2.8 m**	1.9 m	**3.1 m**	1.3 m	**6.1 m**
PA	**0.8 s**	**0.5 s**	**0.6 s**	0.4 s	**1.1 s**
LB-U	11.2 m	6.8 m	8.8 m	4.7 m	15.4 m
Perceptron	**0.8 s**	**0.5 s**	**0.6 s**	**0.3 s**	**1.1 s**
CW	2.7 s	1.7 s	2.1 s	1.2 s	4.2 s
NB-U	3.4 m	1.9 m	2.7 m	1.0 m	5.6 m

Table 5: Time taken to build model (training time) for batch and online learning algorithms on all data sets. The shaded classifiers are online learning and non-shaded ones are batch learning algorithms

however, yields the best overall error rates in all the data sets. Content-based features significantly help in improving error rates by improving both false positive and negative rates when combined with each individual URL based feature set. The combined URL based feature sets yield 0.67% error rate which is much better than each URL based feature set. All feature sets but search engine based feature category yields 2.23% error rate which is better than that given by content based feature set but worse than that given by the combined URL based features. This further emphasizes the importance of search engine based features. URL based features when combined with the content based features (full feature set), however, yield the best error rates along with the best false positive and negative rates, emphasizing the importance of content based features.

Next, in order to find the importance of each individual feature, we compute the F-score (Fisher score) of all the features. F-score is a simple but effective criterion to measure the discrimination between a feature and the label. Based on statistic characteristics, it is independent of the classifiers [17].

A larger F-score value indicates that the feature is more discriminative. One known disadvantage of F-score is that it considers each feature separately and does not say anything about the mutual co-relation

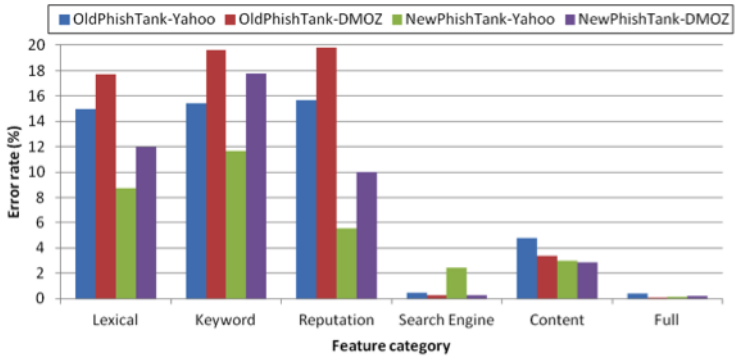

Figure 2: Error rates for RF classifier with five feature categories on each of the four data sets. Overall, using more features improves classification accuracy

Feature category	Feature count		Error rate	FPR	FNR
	Total	Relevant			
Lexical based	24	23	14.34%	8.00%	26.40%
Keyword based	101	98	16.42%	5.20%	37.60%
Reputation based	8	8	14.4%	1.50%	38.90%
Search Engine based	6	6	1.62%	1.00%	2.70%
URL (lexical+keyword+reputation+search engine)	139	135	0.67%	0.40%	1.20%
Content based	38	38	4.49%	3.60%	6.20%
Lexical + Content based	62	61	3.53%	2.90%	4.70%
Keyword + Content based	139	137	3.45%	2.50%	5.30%
Reputation + Content based	46	46	3.55%	2.70%	5.20%
Search Engine + Content based	44	44	0.75%	**0.05%**	1.20%
Lexical + Keyword + Reputation + Content	171	167	2.23%	1.60%	3.40%
Full features	177	173	**0.44%**	0.30%	**0.70%**

Table 6: Error rates for RF classifier with various combinations of features on the combined (OYND) data set

among features. For brevity, we list the top 20 features based on F-score in Table 7. The top 20 feature list is dominated by URL based features except for 5 content based features that are ranked 7, 9, 10, 18, and 19. Mostly keyword based features made up the bottom of the list.

3.4 Mismatched Data Sets

Features extracted by observing a particular data set can yield impressive low classification error rates when trained and tested on disjoint sets of the same data source using the right classifier. However, experiments results show that when training and testing sources are completely mismatched, a classifier's accuracy decreases significantly [30]. To investigate if this phenomenon holds in this context, we experiment with training and testing on various combinations of phishing and non-phishing sources of webpages. We use the abbreviations defined in Section 2.2 to refer to each combination of data sets, e.g., OY for Old PhishTank-Yahoo.

Table 8 shows classification results of training and testing on mismatched data sets using Random Forests (RF) classifier.

Rank	Feature description	F-score
1	Domain NOT in Google top results	8.48
2	URL NOT in Google top results	7.77
3	URL NOT in Yahoo top results	3.46
4	Domain NOT in Yahoo top results	2.47
5	URL NOT in Bing top results	1.95
6	Domain NOT in Yahoo top results	2.05
7	Title search NOT matching domain	0.68
8	URL in Google phishing blacklist	0.62
9	Title search NOT matching URL	0.31
10	Password field in page	0.26
11	Digit [0-9] in Host	0.22
12	IP in PhishTank top 10	0.20
13	Number of dots in URL path	0.11
14	PhishTank top target in URL	0.11
15	'log' in URL	0.10
16	PhishTank top domain in URL	0.10
17	Length of URL	0.08
18	Number of internal links	0.08
19	Has server redirection	0.08
20	'pay' in URL	0.07

Table 7: Top 20 features based on F-score using the combined OYND data set

As expected, when trained and tested RF classifier with same data set, the overall error rates are normally better compared to when trained and tested with mismatched – possibly different sources – data sets (see the diagonal values in Table 8). When newer phishing webpages are tested against the model trained from older phishing webpages, the error rate is 3.8% contributed mostly by false negatives. When only the non-phishing webpage source is mismatched (e.g., OD and OY), error rate increases due to higher false positives. This observation is similar to the ones observed by Ma et al. [30]. However, when only the phishing webpage data sets are mismatched (e.g., OD, ND), we do not see a big increase in error rates. The worst error rate in this category is 3.9% (NY and OY). This small increase in error rate may be due to the fact that the sources of phishing webpages are not technically different but they differ by the time when phishing URLs were submitted for verification. We look into this in detail in the next section. When classifier is trained with combination of any phishing data sets plus DMOZ (OD, ND), and tested with combinations of phishing and Yahoo data set (OY, NY), the error ranges between 9–38%, contributed mostly by high false positives. This is due to the fact that DMOZ data set is about half the size of Yahoo data set and Yahoo data set also includes non-phishing webpages from top targets, which look very similar to phishing webpages that DMOZ doesn't have. When trained with the combined data set, the model generalizes well and yields good performance results across all test data sets (last row) except for testing OY data set, which yields a higher error rate of 5.3%. This experiment concludes that the training data set needs to be selected properly that represent the actual test environment in the real world in order to achieve the best performance from a classifier.

Training	Testing (Error rate)			
	OY	OD	NY	ND
OY	**0.42%**	0.44%	3.82%	4.27%
OD	9.84%	**0.04%**	38.30%	0.55%
NY	3.96%	19.11%	**0.87%**	1.53%
ND	11.52%	0.77%	16.34%	**0.45%**
All data sets (OYND)	5.28%	**0.00%**	**0.00%**	0.16%

Table 8: Overall error rates when training on one data set and testing on another (possibly different source) using RF classifier

3.5 Data Drift

As phishers change their attack tactics over time, so must the classifier's model employed in detecting phishing webpages. It is evidently clear from previous experiments that classifiers performance drifts over time (see Table 8) when classifiers are trained with older data set (OY) and tested with newer data set (NY). Moreover, measuring how performance drifts over time can help us determine how often we need to retrain our classifier to keep it up-to-the-minute. In order to experimentally demonstrate how our approach would work in the wild scenario if deployed in real world, we use OldPhishTank phishing data set and 22,480 (twice the number of phishing webpages) randomly selected non-phishing webpages from Yahoo and DMOZ data sets as our "base" training set. Figure 3 and Figure 4 show the comparison of batch and online learning algorithms respectively after training the classifiers on different intervals. The x-axis shows number of days in the experiment with the phishing webpages collected from January 1st to May 3rd of 2011, and the y-axis shows the cumulative error rates on testing the classifiers. To generate test data for non-phishing webpages, twice the number of phishing webpages was randomly selected from the remaining 9,101 non-phishing webpages from Yahoo and DMOZ data sets.

3.5.1 Comparison of Batch Algorithms

Though batch learning algorithms are ultimately limited by the training time and memory limitation in a large-scale real-time application, we experimented with the top 2 batch algorithms from the first experiment in Section 3.2 as our combined data set (OYND) is small enough to hold in the memory. Figure 3 compares classification accuracies of Random Forests (RF) and J48 using different training intervals. RF-once and J48-once curves represent training once on "base" training set and using those models for testing on all other days. RF-weekly and J48-weekly curves represent training the classifiers weekly with the data collected up to that week and testing on the data collected in the subsequent week. Similarly, RF-daily and J48-daily curves represent training and testing the models on the daily basis. Overall, RF outperforms J48 among all interval-based training with the best cumulative error approaching 1.5% when training the classifiers daily.

As most batch algorithms take anywhere from several seconds to minutes to train models, retraining them after every instance may not be practical in real-world applications where tolerably low turn-around time is expected for a successful implementation besides high accuracy rates. We, therefore, try to exploit the advantage of online algorithms in the next section.

3.5.2 Comparison of Online Algorithms

The major advantage of online algorithms is that they allow the model to update incrementally without having to retrain them from scratch which saves time and memory resources. Online algorithms

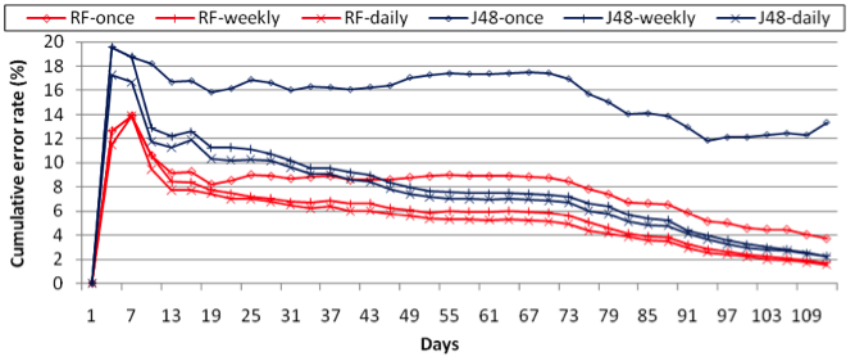

Figure 3: Error rates for batch algorithms using all features and various interval based training

have been shown to perform better compared to some batch algorithms when trained continuously with variable-feature sets in the context of detecting malicious URLs [31]. In order to investigate if the online algorithms achieve similar superior performances in our context, we compare classification performances of five online algorithms such as updatable version of Naïve Bayes (NB-U), Perceptron, Passive-Aggressive (PA), Confidence-Weighted (CW), and the updatable version of LogitBoost (LB-U) algorithms and show the performance results in Figure 4.

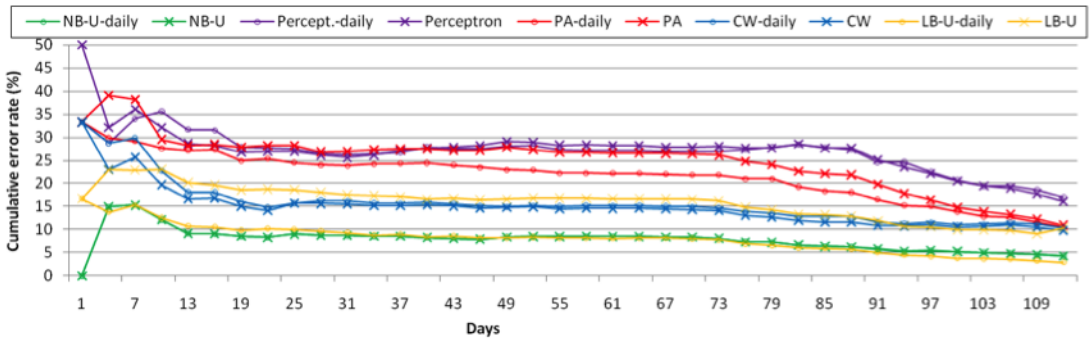

Figure 4: Error rates for online algorithms using all features and continuous and interval based training

When trained on a daily interval, the updatable version of LogitBoost (LB-U) performs the best among all the online learning algorithms with cumulative error approaching to 2.9%. Interestingly, LB-U performs worse when trained incrementally. When trained in interval or incrementally, the online version of Naïve Bayes, surprisingly, outperforms the remaining three online algorithms with the cumulative error approaching to 4.2% as compared to 16.1% by Perceptron, 10.4% by PA and 9.8% by CW. Naïve Bayes is also comparatively the most robust to data drift as its performances remain very similar (15-4%) when trained either daily or continuously. Though the cumulative error rates are much worse (more than 10%) for other three classifiers (PA, Perceptron and CW), they show improvements in accuracy results when trained continuously over training on the daily basis. Moreover, performances of these 3 classifiers are as bad as random guessing when retrained them weekly (the results are not shown). Nevertheless, all online classifiers were much faster in training models as opposed to all the batch algorithms.

Interestingly, in this context, all the online algorithms yield lower accuracies compared to the batch algorithms that are trained on the daily basis.

3.6 Top Targets

In these experiments, we investigate how our proposed method performs in detecting the legitimate webpages from top targets as non-phishing. We combine all the datasets (OYND) except the webpages of the top targets and use the combined 46,992 instances as a training data set. We use the rest 1,671 instances as a test data set. Random Forests (RF) yields a 13.76% error rate, meaning only 86.23% of legitimate webpages related to the top targets are correctly classified as non-phishing webpages and the rest are all misclassified as phishing webpages. This big error rate is due to the fact that these target webpages look very similar to the forged phishing webpages, and the model, perhaps, is not trained with the types of webpages that are seen during testing.

In order to address this, we randomly select 66% of legitimate webpages from top targets and include them in the training set and retrain the model. We test the newly trained model with the remaining 569 instances of webpages. As expected, the results improve significantly, yielding a 1.05% error rate, misclassifying only 6 non-phishing webpages as phishing. This experiment further emphasizes the importance of judicious selection of training data set with proper representation of all the possible phishing and non-phishing webpages the system will be tested against in real world.

Unlike related works [24, 30, 34, 43, 42], we explicitly test the validity of our method on the actual phishing targets and show that our method yields good accuracy results in detecting not only the phishing webpages but also in detecting the legitimate webpages of the actual targets.

3.7 False Positives and Negatives

We were motivated with a hypothesis that including features from webpage contents could lower false positives and negatives in classifying phishing websites. We experimentally confirmed our hypothesis that including the content-based features lowered both the false positive and false negative rates (see Table 6). Despite low false positive and false negative rates achieved by this approach, examining misclassified webpages may help us to understand the limitations of our approach and possibly reveal trends that can suggest refinements to our current approach. We examined the test results by Random Forests (RF) on the combined (OYND) data set.

Non-phishing webpages that we downloaded from the links harvested from our initial seed URLs of most targeted brands contributed to some false positives. These URLs were long in nature containing some of the red-flagged keywords. Interestingly, these URLs also didn't exist in search engines' indexes. Also, some poorly written legitimate webpages that didn't use HTTPS to transmit FORM data with password field were classified as phishing webpages. Though these groups of non-phishing webpages are not phishing, but they sure are insecure in the sense that unencrypted confidential data can be sniffed during the transmission.

Phishing webpages hosted on free legitimate hosting services that have been around for a while, phishing URLs that didn't have any of the red flagged keywords but their domain were in search results were classified as legitimate webpages contributing to false negatives.

One of the advantages of using machine-learning approach is that their learned models allow us to tune tradeoff between false positives and negatives. Due to enterprise's policy reason, or individual Internet users' personal preferences, focusing to minimize the overall error rate may not be always desirable. Rather, the practitioners may want to tune a threshold t to have low false positives at the expense of more false negatives or vice versa.

Figure 5 shows the results of this experiment as an ROC graph with respect to threshold t over an instance of the combined data sets using (OYND) using RF classifier. By tuning false positives to a very low 0.1%, we can achieve 1.95% false negative rate. By tolerating slightly higher false positives of 0.3%, however, we can achieve significantly lower false negative rate of 0.7%.

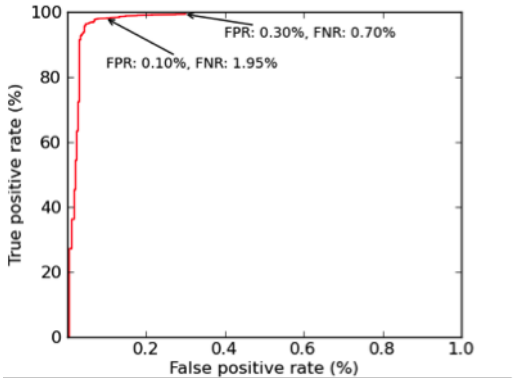

Figure 5: ROC showing tradeoff between false positives and false negatives using RF on the combined data set (OYND). Note that the false positive rate on the x-axis ranges between 0 and 1%

4 Related Work

The work by Whittaker et al. [42] is most closely related to our work. They describe the design and performance characteristics of a scalable machine learning classifier to detect phishing websites, which has been used in maintaining Google's phishing blacklist automatically. Their proprietary classifier analyzes millions of pages a day, examining URL and the contents of a webpage to determine whether or not a page is phishing. Their system classifies webpages submitted by end users and URLs collected from Gmail's spam filters. Though some features are similar, we propose several new URL and webpage content based features and evaluate our approach using publicly available batch and online learning algorithms on public data sets. While they use features from the terms appearing in the text of a page, most of our content-based features are based on HTML tags of a webpage. Unlike their approach, we also do not use DNS entries and geo-locations of pages' host and nameservers.

Zhang et al. present CANTINA, content based approach to detecting phishing web sites, based on the TF-IDF information retrieval algorithm [43]. By using a weighted sum of 8 features (4 content-related, 3 lexical, and 1 WHOIS-related), they show that CANTINA can correctly detect approximately 95% of phishing sites. Though similar in motivation, our approach differs significantly in both methodology (comparing a number of batch and online learning algorithms) and scale (considering a large number of features and an order-of-magnitude more training examples).

Miyamoto et al. [34] used AdaBoost algorithm to 8 heuristics proposed by Zhang et al. [43] and achieved 97% accuracy (a 2% improvement) in detecting phishing websites. Our approach differs in the same way as it differs from Zhang et al. In [14] we proposed a rule based approach to detect phishing webpages and showed that the proposed approach was effective in classifying phishing webpages from non-phishing ones yielding classification accuracy of more than 99%.

Dong et al. propose a method based on analysis of users' online behaviors to detect phishing websites [21]. Other papers have attempted to automatically classify malicious and phishing URLs from using the information and meta-data on URLs alone [15, 24, 30]. The approach of relying on URL properties alone to detect malicious and phishing webpages is very intriguing. Because of the focus on the URL itself, this approach can be applied anywhere that a URL can be embedded, such as in email, webpages, chat sessions, to name a few. Depending on the number and type of meta-data and heuristics used, the approach may be efficient as it doesn't require the need to download the actual webpage. Since most phishing webpages are also rigged with malware – virus, spyware, Trojans, adware, and all kinds of badware – it is deemed unsafe to download and analyze those pages on users system. Nevertheless,

we think this approach may not be reliable and robust enough as phishers can easily circumvent such a system. For example, phishers can use homegrown URL shortening services [9] to masquerade their phishing URLs or use compromised legitimate websites to host their phishing campaign. In the ever-changing threat landscape, security counter-measures need to be adapted and tuned constantly in order to protect Internet users from plethora of social engineering based attacks. Moreover, in this paper we show that adding content-based feature does improve classification accuracies by reducing false positives and negatives compared to only using URL based features.

Similar to spam filters, several phishing email filters have been proposed, see for e.g., [12, 13, 23].

5 Conclusion

In this paper, we proposed many new URL based, server based, and content based features for classifying phishing webpages. We demonstrated that the proposed features are highly relevant to the automatic discovery and classification of phishing webpages. We tested our approach on real-world temporal data sets using a number of popular batch and online learning methods. We showed that the proposed approach can detect phishing webpages (in some data sets) with an accuracy of as high as 99.9%, false positive rate of as low as 0.00% and false negative rate of 0.06% using features from URLs, web servers, and the contents of the webpages.

With a goal to ultimately constructing a near real-time phishing webpage detection system, we evaluated batch and online learning algorithms for our application to study their benefits and tradeoffs. Most classifiers showed statistically similar performance results. Overall, among the examined batch classifiers, Random Forests (RF) performed the best in terms of classification performance and training time in this context. Online algorithms, however, showed poorer classification performance compared to batch algorithms. As expected, the classification performances of all the classifiers degraded while training them with older and testing them with newer data sets. It is, thus, recommended to retrain classifiers with newer data sets as and when they become available if deployed in real world.

5.1 Acknowledgment

The authors would like to acknowledge the generous support received from ICASA (the Institute for Complex Additive Systems Analysis), a division of New Mexico Tech. The authors are thankful to Max Planck and Rudra Kafle for providing valuable comments and feedback for improvement.

References

[1] Google: Safe browsing api. http://code.google.com/apis/safebrowsing/.

[2] Microsoft: Security intelligence report. http://www.microsoft.com/security/sir/default.aspx.

[3] OWASP: Cross-site Scripting (XSS) - OWASP. http://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

[4] PhishTank: Out of the net, into the tank. http://www.phishtank.com/developer_info.php.

[5] PyLongURL - Python library for LongURL.org. http://code.google.com/p/pylongurl/(2010).

[6] StopBadware: IP Address Report - Top 50 by number of reported URLs. http://stopbadware.org/reports/ip.

[7] Yahoo! Inc.: Random Link - random. http://random.yahoo.com/fast/ryl.

[8] hpHosts: Online - Simple, Searchable and FREE! http://hosts-file.net/, 2005.

[9] Symantec cloud: Messagelabs intelligence reports. http://www.messagelabs.com/mlireport/MLI_2011_05_May_FINAL-en.pdf, 2011.

[10] P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5):412–424, February 2000.

[11] R. Basnet. Interpreting obfuscated JavaScript. `http://sites.google.com/site/rambasnet/anatomy-of-phishing-attacks/javascript-deobfuscation-technique`, 2010.

[12] R. Basnet, S. Mukkamala, and A. Sung. Detection of phishing attacks: a machine learning approach. In B. Parsad, editor, *Soft Computing Applications in Industry, Studies in Fuzziness and Soft Computing*, volume 226, pages 373–383. Springer, 2008.

[13] R. Basnet and A. Sung. Classifying phishing emails using confidence-weighted linear classifiers. In *Proc. of the 2010 International Conference on Information Security and Artificial Intelligence, Chengdu, China*, pages 108–112, 2010.

[14] R. Basnet, A. Sung, and Q. Liu. Rule-based phishing attack detection. July 2011.

[15] R. Basnet, A. Sung, and Q. Liu. Learning to detect phishing URLs. *IJRET: International Journal of Research in Engineering and Technology*, 3(6):11–24, June 2014.

[16] L. Breiman and A. Cutler. Random forests. `http://stat-www.berkeley.edu/users/breiman/RandomForests/cc_home.htm`.

[17] Y.-W. Chen and C.-J. Lin. Combining SVMs with various feature selection strategies. `http://www.csie.ntu.edu.tw/~cjlin/papers/features.pdf`.

[18] M. Cova, C. Kuregel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Proc. of the International World Wide Web Conference (WWW'10), Releigh, North Carolina, USA*, pages 281–290. ACM, 2010.

[19] K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, March 2006.

[20] R. Dhamija, J. Tygar, and M. Hearst. Why phishing works. In *Proc. of the 2006 SIGCHI Conference on Human Factors in Computing Systems (CHI'06), Montreal, Quebec, Canada*, pages 581–590. ACM, April 2006.

[21] X. Dong, J. Clark, and J. Jacob. User behavior based phishing websites detection. In *Proc. of the 2008 International Multiconference on Computer Science and Information Technology (IMCSIT'08), Wisla, Poland*, pages 783–790. IEEE, October 2008.

[22] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *Proc. of the International Conference on Machine Learning (ICML'08), Helsinki, Finland*, pages 264–271. IEEE, July 2008.

[23] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In *Proc. of the 16th International Conference on World Wide Web (WWW'07), Alberta, Canada*, pages 649–656. ACM, 2007.

[24] S. Garera, N. Provos, M. Chew, and A. Rubin. A framework for detection and measurement of phishing attacks. In *Proc. of the 5th ACM Workshop on Recurring Malcode (WORM'07). Hilton Alexandria Mark Center, VA, USA*, pages 1–8. ACM, 2007.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):1–18, June 2009.

[26] M. A. Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

[27] J. Korpela. Methods GET and POST in HTML forms - What's the difference? `http://www.cs.tut.fi/~jkorpela/forms/methods.html`, 2003.

[28] S. le Cessie and J. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41:191–201, 1992.

[29] C. Ludl, S. McAllister, E. Kirda, and C. Kruegel. On the effectiveness of techniques to detect phishing sites. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 20–39. Springer, 2007.

[30] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In *Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09), Paris, France*, pages 1245–1254. ACM, June-July 2009.

[31] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious URLs: an application of large-scale online learning. In *Proc. of the 26th Annual International Conference on Machine Learning (ICML'09),*

Montreal, Quebec, Canada, pages 681–688. ACM, 2009.

[32] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[33] E. Messmer. iFrame attacks surge, security firm says. `http://www.networkworld.com/news/2008/042408-iframe-attacks-surge.html`, 2008.

[34] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi. A proposal of the Adaboost-based detection of phishing sites. In *Proc. of the 2nd Joint Workshop on Information Security (JWIS'07), Waseda University, Tokyo, Japan*, August 2007.

[35] J. R. Quinlan. *C4.5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[36] L. Richardson. Beautiful soup. `http://www.crummy.com/software/BeautifulSoup/`.

[37] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[38] I. Rush. An empirical study of the naive bayes classifier. `http://www.research.ibm.com/people/r/rish/papers/RC22230.pdf`, November 2001.

[39] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang. An empirical analysis of phishing blacklists. In *Proc. of the 6th International Conference on Email and Anti-Spam (CEAS'09), Mountain View, California, USA*, 2009.

[40] S. Stamm and Z. Raman. Drive-by pharming. `http://www.cs.indiana.edu/pub/techreports/TR641.pdf`, 2006.

[41] V. Vapnik. *The nature of statistical learning theory*. Springer, 2000.

[42] C. Whittaker, B. Ryner, and M. Nazif. Large-scale automatic classification of phishing pages. In *Proc. of the 17th Annual Network and Distributed System Security Symposium (NDSS'10), California, USA*, February 2010.

[43] Y. Zhang, J. I. Hong, and L. F. Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proc. of the 16th International Conference on World Wide Web (WWW'07), Alberta, Canada*, pages 639–648. ACM, May 2007.

Author Biography

Ram B. Basnet is an assistant professor of the Computer Science, Mathematics and Statistics Department at Colorado Mesa University.He received his B.S.in Computer Science from Colorado Mesa University in 2004, and M.S. and Ph.D. in Computer Science from New Mexico Tech in 2008 and 2012, respectively. His research interests are information assurance, phishing detection, machine learning, and data mining.

Andrew H. Sung is professor and director of School of Computing at the University of Southern Mississippi. He received his B.S. in Electrical Engineering from National Taiwan University in 1976, M.S. in Mathematical Sciences from the University of Texas at Dallas in 1980, and Ph.D. in Computer Science from the State University of New York at Stony Brook in 1984. His research interests are computational intelligence, information security, bioinformatics, data mining, petroleum exploration and reservoir modeling.