LEARNING TO DETECT PHISHING URLs

Ram B. Basnet[1], Andrew H. Sung[2], Quingzhong Liu[3]

[1]*Colorado Mesa University, 1100 North Ave. Grand Jct. CO 81501, USA*
[2]*New Mexico Tech, 801 Leroy Pl. Socorro NM 87801, USA*
[3]*Sam Houston State University, Huntsville, TX 77341, USA*

Abstract

Phishing attacks have been on the rise and performing certain actions such as mouse hovering, clicking, etc. on malicious URLs may cause unsuspecting Internet users to fall victims of identity theft or other scams. In this paper, we study the anatomy of phishing URLs that are created with the specific intent of impersonating a trusted third party to trick users into divulging personal data. Unlike previous work in this area, we only use a number of publicly available features on URL alone; in addition, we compare performance of different machine learning techniques and evaluate the efficacy of real-time application of our method. Applying it on real-world data sets, we demonstrate that the proposed approach is highly effective in detecting phishing URLs with an error rate of 0.3%, false positive rate of 0.2% and false negative rate of about 0.5%, thereby improving previous results on the important problem of phishing detection.

Keywords— *Phishing URL, phishing websites, machine learning, web mining, phishing attack, URL classification*

----------***----------

1. INTRODUCTION

Phishing, according to AntiPhishing Working Group [1], is a criminal mechanism employing both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials. For example, a "phisher" sends out emails masquerading as a trustworthy person or a reputable institution, such as a bank, to a large number of random Internet users. The phishers trick users by employing social engineering tactics inducing them to click on a link to a forged site where the user is then asked to provide private information, e.g., password, bank account information, credit card number, social security number, etc. Once the Internet users are lured into a fraudulent website, even the experienced users are often fooled to fulfill the website's primary goal [11].

The means of distribution of phishing URLs include, among others, spam or phishing messages with links to the phishing site, Blackhat search engine optimization (SEO) techniques, Internet downloads, peer-to-peer (P2P) file sharing networks, social networking sites, visiting vulnerable web sites such as blogs, forums, comment accepting news portals, instant messaging (IM), Internet Relay Chat (IRC), etc.

Blacklisting is the most common anti-phishing technique used by modern web browsers. However, study shows that centralized, blacklist-based protection alone is not adequate enough to protect end users from new and emerging zero-day phishing webpages that appear in the thousands and quickly disappear every day. The study also shows that heuristics based phishing techniques outperform centralized blacklisting techniques used by most web browsers [27]. What are needed to address the shortcomings of blacklisting are methods that are discovery-oriented, dynamic, and semi-automated. This approach should not replace, but compliment the blacklist to provide defense-in-depth mechanism to effectively combat the phishing attacks.

To that end, we present a heuristic-based methodology for automatically classifying URLs as being potentially phishing in nature. This methodology could then be used to thwart a phishing attack by either masking the potentially phishing URL, or by alerting the user about the potential threat. Because of the focus on the URL itself, this approach can be applied anywhere that a URL can be embedded, such as in email, web pages, chat sessions, to name a few. We evaluate our approach on real-world data sets with more than 16,000 phishing and 31,000 non-phishing URLs. We experimentally demonstrate that our approach can obtain an error rate of less than 0.3% while maintaining about 0.2% false positive and 0.5% false negative rates. Featured with high accuracy rate, we believe that our light-weight approach can be used by individual users in their system for near real-time phishing URL detection.

The contributions of this paper are: 1) A demonstration that a phishing URL can be detected by using the information on the URL alone without looking at the actual web page contents and regardless of the context or medium the URL is distributed. 2) An examination of the importance of publicly available information on a URL in the evaluation of whether that URL is phishing. 3) A comparison of a number of publicly available machine learning classifiers to determine the best for classification of phishing URLs. 4) A demonstration that the proposed methodology can be used for near real-time application in detecting phishing URLs. 5) A demonstration that the properties of phishing URLs change over time and how the data drift can affect classifiers' performances.

2. BACKGROUND

In this section, we provide a definition of phishing URL and review some related works.

2.1 Definition of Phishing URL

Whittaker et al. [33] define a phishing web page as "any web page 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 a 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. Thus, a phishing URL is a URL that leads user to a phishing web page. Our study, by this definition, is therefore independent of the attack vector by which a phishing URL is distributed.

2.2 Related Work

This section provides related work in phishing attack detection and classification using machine learning and non-machine learning approaches.

2.2.1 Machine Learning Approaches

The work by Garera et al. [13] is the most closely related to our work. They use logistic regression over 18 hand-selected features to classify phishing URLs. The features include the presence of certain red flag key words in the URL, some proprietary features based on Google's PageRank and webpage quality guidelines. Even though they do not analyze the page contents to use as features, they use the pre-computed page based features from Google's proprietary infrastructure that they call *Crawl Database*. They achieve a classification accuracy of 97.3% over a set of 2,500 URLs. Direct comparison with our approach, however, is difficult without access to the same datasets or features. Though similar in goal, our approach differs significantly in both methodology (considering new publicly available features based on URLs alone and comparing several different machine learning algorithms) and scale (considering more features and an order-of-magnitude more samples).

Ma et al. propose a method to classify malicious URLs using variable number of lexical and host-based properties of the URLs. Using these features, they compare the accuracy of batch and online learning algorithms [18] and [19]. Among 4 batch algorithms, they determine that Logistic Regression performs the best for their problem. For large-scale application in detecting malicious URLs, they determine that Confidence-Weighted algorithm performs the best among 4 online algorithms. Though we use some similar features and classification models, our approach is different in a number of ways. First, the scope of our work is limited to detecting phishing URLs as opposed to detecting wide range of malicious URLs. Our techniques can certainly be extended to detecting and classifying wider range of malicious URLs. Secondly, we have a fixed set of smaller number of features. Thirdly, we do not use host-based properties of web pages such as WHOIS entries, connection speed, etc. Though WHOIS information can be very useful in determining the reputation of hosts and registrars and the reputation of the domains overall, it makes the repetition of the experiments difficult.

Whittaker et al. [33] describe the design and performance characteristics of a scalable machine learning classifier that has been used in maintaining Google's phishing blacklist automatically. Their proprietary classifier analyzes millions of pages a day, examining the URL and the contents of a page to determine whether or not a page is phishing. Their system classifies web pages submitted by end users and URLs collected from Gmail's spam filters. Though some URL based features are similar, we propose several new features and evaluate our approach with publicly available machine learning algorithms and public data sets. Unlike their approach, we do not use any proprietary and page content based features.

Zhang et al. [34] present CANTINA, content-based approach to detect phishing websites, based on the TF-IDF information retrieval algorithm and the Robust Hyperlinks algorithm. 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. The goal of our approach is to avoid downloading the actual web pages and thus reduce the potential risk of analyzing the malicious content on user's system. In order to achieve this goal, we evaluate only the features related to URLs.

A number of machine learning-based studies can be found in related contexts such as in detecting phishing emails. Fette et al. [12] use a set of 10 features extracted from email headers, WHOIS information on sender's domain, email contents, URL structures, etc. and apply Support Vector Machines (SVMs) to classify phishing emails from legitimate ham emails. We further improve the accuracy of Fette et al. by introducing groups of keyword based features from the email contents [3]. Using different classification models we achieve classification accuracy of 98%, while maintaining low false positive and negative rates. Fette et al. [12] hypothesized that phishing email classification appears to be simple text classification problem but, the classification is confounded by the fact that the class of "phishing" emails is nearly identical to the class of real emails. Motivated by the hypothesis, we base the phishing email classification problem as the text classification problem in our previous work [5]. Using Confidence-Weighted linear classifier, an online algorithm, and using only the email text contents as "bag-of-words" representation, we achieve a classification accuracy of 99%, maintaining false positive and false negative rates of less than 1% on public benchmark data sets.

2.2.2 Non-Machine Learning Approaches

Besides machine learning (ML) based techniques, there exist many other approaches in phishing detection. Perhaps, the most widely used anti-phishing technology is the URL blacklist technique that most modern browsers are equipped with [14] and [28]. Other popular methods are browser-based plug-in or add-in toolbars. SpoofGuard [9] uses domain name, URL, link, and images to evaluate the spoof probability on a webpage. The plug-in applies a series of tests, each resulting in a number in the range from 0 to 1. The total score is a weighted average of the individual test results. There has been an attempt to detect phishing attack using user generated rules [6]. Other anti-phishing tools include SpoofStick [30], SiteAdvisor [20], Netcraft anti-phishing toolbar [23], AVG Security Toolbar [2], etc.

3. OUR METHOD

In this section, we describe in detail our approach to detecting phishing URLs. We begin with an overview of the classification problem, followed by a discussion of the generation of our data sets, features we extract, and finally the set of machine learning classifiers we use to evaluate our methodology.

3.1 Method Overview

We propose a heuristic-based approach to classify phishing URLs by using the information available only on URLs. We treat the problem of detecting phishing URLs as a binary classification problem with phishing URLs belong to the positive class and benign URLs belong to the negative class. We first run a number of scripts to collect our phishing and benign URLs 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 various machine learning algorithms to build models from training data, which is comprised of pairs of feature values and class labels. Separate set of test data are then supplied to the models, and the predicted class of the data instance is compared to the actual class of the data to compute the accuracy of the classification models. Figure 1 provides the overview of graphical representation of phishing URL detection framework.

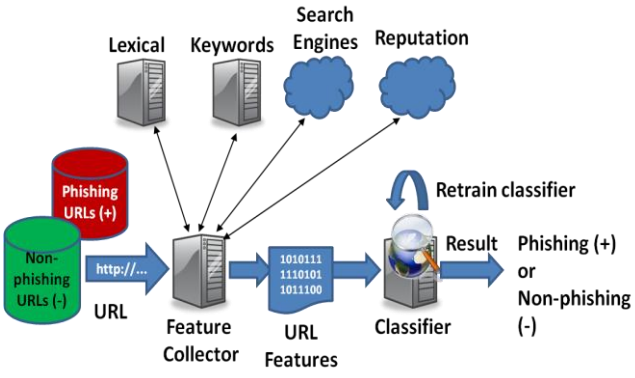

Fig. 1 Overview of phishing URL detection framework

3.2 Data Sets

For experiments, we collected our data from various credible sources that are also used by Ma et al. [18], Zhang et al. [34], and many others. For phishing URLs, we wrote Python scripts to automatically download confirmed phishing websites' URLs from PhishTank. PhishTank is a collaborative clearing house for data and information about phishing on the Internet [25]. After signing up, developers and researchers can download confirmed phishing URL lists in various file formats with an API key provided for free. A potential phishing URL once submitted is verified by a number of registered users to confirm it as phishing. We collected first set of 11,361 phishing URLs from June 1 to October 31 of 2010 and call it OldPhishTank data set.

Phishing tactics used by scammers evolve over time. In order to follow these evolving URL features and to closely mimic the real-world scenario, we collected second batch of 5,456 confirmed phishing URLs that were submitted for verification from January 1 to May 3, 2011. We call it NewPhishTank data set.

In order to address URLs that were produced using shortening services such as bit.ly, goo.gl, etc., we developed a Python library [4] to utilize the web service API provided by *longurl.org* to automatically detect and expand shortened URLs. The service currently supports about 333 popular shortening services. However, some short URLs – either from some new and unsupported shortening services or because the shortening services do not expand as the target URL has been reported as phishing or malicious – do exist in our data sets.

We collected the non-phishing URLs from two public data sources: Yahoo! directory and DMOZ Open Directory Project. We used Yahoo's server redirection service, *http://random.yahoo.com/bin/ryl*, which randomly selects a web link from Yahoo directory and redirects browser to that page. In order to cover wider URL structures, we also made a list of URLs of most commonly phished targets (using statistics of top targets from PhishTank). We then crawled those URLs, parsed the retrieved HTML contents, and harvested the hyperlinks therein to also use as non-phishing URLs. Those additional hyperlinks are assumed to be benign since they were extracted from a legitimate source. We use 22,213 legitimate URLs using these sources and call it Yahoo data set. These URLs were collected between September 15, 2010 and October 31, 2010. The other source of legitimate URLs, DMOZ, is a directory whose entries are vetted manually by editors. We use 9,636 randomly chosen non-phishing URLs from this source and call it DMOZ data set.

We then paired OldPhishTank and NewPhishTank data sets with non-phishing URLs from a benign source (either Yahoo or DMOZ). We refer to these data sets as the OldPhishTank-Yahoo (OY), OldPhishTank-DMOZ (OD), NewPhishTank-Yahoo (NY), and NewPhishTank-DMOZ (ND).

Table 1: Feature Categories and Number of Features in Each Category

Feature Category	Feature Count
Lexical based	24
Keyword based	101
Search Engine based	6
Reputation based	7

Anywhere from a handful to tens of thousands of features have been proposed and used in detecting phishing URLs. We developed our set of 138 features based on related works, drawing primarily from [3], [12], [13], [33], and [34]. Some of these features are modified to fit our needs, while others are newly proposed. The use of relatively small number of fixed set of features makes the decision boundaries less complex, and therefore less prone to over-fitting as well as faster to evaluate for most batch algorithms.

We group features that we gather into 4 broad categories. Table 1 summarizes each category and the number of features from that category that we use in our data sets for classifying phishing URLs. We briefly describe each feature category with their statistics from a randomly selected 80% of OldPhishTank-Yahoo training data set (we call it "Random Set") in the following sub sections.

3.2.1 Lexical-based Features

Lexical features, the textual properties of the URL itself, have been widely used in literatures [3], [12], [18], [19], [33], and [34] in detecting phishing attacks.

We examine various obfuscation techniques phishers may employ and derive a number of phishing like features to use in our classifiers. For instance, we check if there's a port number in a URL and check if the port belongs to the list of standard HTTP ports: 80, 8080, 21, 443, 70, and 1080. If the port number doesn't belong to the standard list, we flag it as a potentially phishing URL.

There are 24 features in this category. We summarize the real-valued and binary features separately in Table 2 and 3, respectively. The decimal numbers are rounded to 2 digits after the decimal point.

Table 2: Feature Categories and Number of Features in Each Category

Feature Description	URL Type	Max	Min	Mean	Med-ian
Length of Host	Phishing	240	4	21.38	19
	Non-phishing	70	5	18.77	18
Number of '.' in Host	Phishing	30	0	2.13	2
	Non-phishing	5	1	2.14	2
Number of'.' in Path	Phishing	18	0	0.86	1
	Non-phishing	13	0	0.25	1
Number of '.' in URL	Phishing	30	0	3.00	3
	Non-phishing	15	1	2.38	2
Length of Path	Phishing	380	0	24.55	15
	Non-phishing	360	0	10.74	1
Length of URL	Phishing	999	13	66.09	18
	Non-phishing	383	15	41.22	33

Table 3: Summary of Lexical-based Binary Valued Features and their Statistics

Feature Description	% Phishing	% Non-phishing
'-' in Host	2.02%	9.03%
Digit [0-9] in Host	30.06%	3.11%
IP Based Host	4.15%	0.00%
Hex Based Host	0.18%	0.00%
'-' in Path	15.82%	6.64%
'/' in Path	98.39%	96.18%
'=' in Path	4.58%	0.16%
';' in Path	0.07%	0.00%
',' in Path	0.15%	0.28%
Has Parameter Part	0.18%	0.77%
Has Query Part	0.07%	0.01%
'=' in Query Part	13.45%	10.43%
Has Fragment Part	0.18%	0.77%
'@' in URL	0.33%	0.08%
'Username' in URL	0.33%	0.08%
'Password' in URL	0.02%	0.00%
Has Non-Standard Port	0.01%	0.00%
'_' in Path	11.16%	8.41%

3.2.2 Keyword-based Features

Many phishing URLs are found to contain eye-catching word tokens (e.g., *login*, *signin*, *confirm*, *verify*, etc.) to attract users' attention. Garera et al. [13] selected 8 red flag keywords to use as features. Whittaker et al. [33] use every string token separated by non-alphanumeric characters out of the URL to use as features. However, they rely on the feature selection methods built into their machine-learning framework to incorporate only the most useful of these features into their classification models. Though our word based feature extraction technique is somewhat similar to theirs, the selection technique significantly differs as we employ a formal feature selection technique which we describe next.

Using the *Random Set* (Section III-B), we tokenize each phishing URL by splitting it using non-alphanumeric characters. After applying Porter stemmer [22], we obtain 12,012 unique root tokens and their frequencies. While we could use every word token appearing on phishing URLs as a feature – an approach taken by Ma et al. [18] and [19] to detect malicious URLs – this many keyword based features plus other features per URL can burden our batch learning

algorithms without yielding any performance benefit. Instead, we discard all tokens with length < 3 such as *d*, *c*, *e*, *br*, *fr*, *it*, etc. Some single-character tokens have high frequencies, but offer no meaning. Two-character tokens are mostly the country code top-level domains (ccTLD). We also discard several common URL parts such as *http*, *www*, *com*, etc. and webpage file extensions such as *htm*, *html*, *asp*, *php*, etc. Since top target organizations such as *paypal*, *ebay*, *bankofamerica*, *wamu*, etc. are covered by top target list under reputation-based features, we discard them as well.A large number of tokens have frequency one, suggesting that they are not frequently used in phishing URLs. Most of these words either do not make sense as a whole or they have random characters such as *ykokejox*, *riversid*, *sxkretyvwufatnrmomgpqjdw*, *njghlfi*, etc. We discard these tokens as well. With this preliminary selection, we are left with 1,127 tokens.

We then apply feature selection technique commonly used in text classification. Feature selection serves two main purposes. First, it makes the process of training and applying a classifier more efficient by decreasing the size of the discriminative features. This is of particular importance for classifiers that, unlike Naïve Bayes, are expensive to train. Second, feature selection often increases classification accuracy by eliminating noise features. We compute mutual information (MI) of each term in phishing class. MI measures how much information the presence or absence of a term contributes to making the correct classification decision on a class [24]. MI is computed using the following equation:

$$MI(U;C) = \frac{N_{11}}{N} log_2 \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} log_2 \frac{NN_{01}}{N_{0.}N_{.1}} + \frac{N_{10}}{N} log_2 \frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N} log_2 \frac{NN_{00}}{N_{0.}N_{.0}} \quad (1)$$

where U is a random variable that takes values $e_t = 1$ (the URL contains term t) and $e_t = 0$ (the URL does not contain term t), C is a random variable that takes values $e_c = 1$ (the URL is in class c) and $e_c = 0$ (the URL is not in class c), N s are counts of URLs that have the values of e_t and e_c that are indicated by the two subscripts. For example, N_{10} is the number of URLs that contain t ($e_t = 1$) and are not in c ($e_c = 0$). $N_{1.} = N_{10} + N_{11}$ is the number of URLs that contain t ($e_t = 1$) and we count URLs independent of class membership ($e_c \in \{0, 1\}$). $N = N_{00} + N_{01} + N_{10} + N_{11}$ is the total number of URLs in the training set.

Terms with high MI values indicate that they are more relevant to the class and are good discriminative features, whereas terms with lower MI values indicate that they are less relevant. For brevity, Table IV shows only the top 10 terms based on MI along with the percentage of each term appearing in phishing and non-phishing URLs in the *Random Set*.

Table 4: Top 10 root Terms (Based on Mutual Information) and Their Statistics

Root Term	MI	% Phishing URLs	% Non-phishing URLs
log	0.1740	21.77%	1.71%
pay	0.1027	13.26%	0.50%
web	0.0778	14.90%	1.62%
cmd	0.6840	10.08%	0.37%
account	0.0559	7.86%	0.34%
dispatch	0.0390	5.69%	0.01%
free	0.0362	7.20%	0.48%
run	0.0331	4.89%	0.16%
net	0.0320	13.05%	5.05%
confirm	0.0292	3.42%	0.00%

Note that the statistical values of each feature among phishing and non-phishing URLs are rounded to 2 and MI values are rounded to 4 decimal digits. Because some keywords are so sparsely present among non-phishing URLs in the training dataset, their statistical values are rounded to 0.00% (for instance, term *confirm* in Table 4). The only feature term that is entirely absent among non-phishing URLs from all data sets is *config*.

By ordering the terms based on MI values from high to low, we then use these terms as binary features on data set OY. Using forward feature selection method, we train and test Naïve Bayes 1,127 times for each feature set size from 1 to 1,127 and record its error rate for each run. We choose Naïve Bayes because of its speed and its effectiveness in spam filtering application [29] – a problem similar to ours – that uses "spammy" word tokens. Figure 2 shows the error rates on feature size from 1 to 1,127.

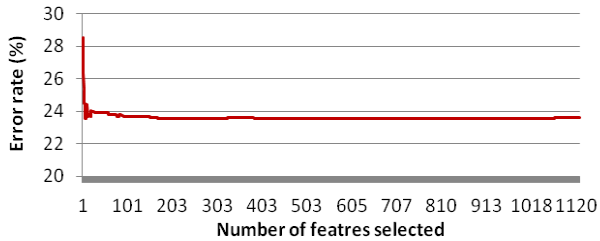

Fig 2 Effects of keyword feature set size on error rate using Naïve Bayes on OY dataset.

Initially, the error rate decreases significantly from ~29% to ~24% as the number of features increases. But after 100 features, the change in error rate is statistically insignificant (< 0.1%) until all the features are added. Thus, we decide to use the top 101 terms based on their MI as keyword based features.

3.2.3 Reputation-based Features

PhishTank produces various top 10 statistical reports on phishing websites every month. We downloaded 3 types of statistics: Top 10 Domains, Top 10 IPs, and Top 10 Popular Targets from the first batch of statistics published in October 2006 to October 2010. The idea behind this is to make use

of the historical data on top IPs and domains that host phishing websites. If a URL has many other phishing related heuristics and also its host belongs to top IP and/or top domain that has historic reputation of hosting phishing web pages, then we can increase our confidence level to classify the URL in question as phishing. There are 311 unique domains, 354 unique IPs, and 43 unique targets in the top 10 statistics during 4 years of period.

Features from PhishTank statistics may appear little biased to use against detecting phishing URLs from the same source. The idea, however, is to use features based on many statistical reports similar to these on phishing web pages. For instance, we include statistics from StopBadware.org, which we explain next. We plan to find more public statistics to use in our future work, and we hypothesize that these features help in detecting phishing URLs if included with many other discriminative features.

StopBadware.org works with its network of partner organizations such as Google, Sunbelt Software, etc. and individuals to fight back against viruses, spyware, etc. [31]. It produces top 50 IP address report from number of reported URLs. We check if the IP address of a URL belongs to this top 50 report and flag it as potentially phishing if it does.

To the best of our knowledge, this is the first work to use such publicly available historical statistics to detect phishing URLs.

Several blacklists have been used by Ma et al. [18] and [19]. We use Safe Browsing API [14] to check URLs against Google's constantly updated blacklists of suspected phishing and malware pages and use 3 binary features for membership in those blacklists. Essentially, these blacklists are also used by Google Chrome and Mozilla Firefox to warn users of potentially malicious websites.

Table 5 summarizes the distribution of reputation-based features in phishing and non-phishing URLs.

Table: 5 Reputation-based Features and Their Statistics

Feature Description	% Phishing URLs	% Non-phishing URLs
PhishTank Top 10 Domain in URL	20.98%	4.87%
PhishTank Top 10 Target in URL	32.65%	14.21%
IP in PhishTank Top 10 IPs	17.30%	0.87%
IP in StopBadware Top 50 IPs	2.31%	1.37%
URL in Phishing Blacklist	42.41%	0.00%
URL in Malware Blacklist	0.45%	0.05%
URL in RegTest Blacklist	0.16%	0.00%

3.2.4 Search Engine-based Features

Google search engine has been used by Garera et al. [13], Whittaker et al. [33], and Zhang et al. [34]. Whittaker et al. use PageRank from Google proprietary infrastructure. Garera et al. use Google's proprietary technologies such as PageRank, page index, and page quality scores. These are pre-computed during Google's crawl phase and are stored in a table, which they call Crawl Database. Though the features generated from the Google proprietary infrastructure seem plausible, it makes the repetition and validation of experiments extremely difficult if not impossible. On the other hand, our search engine based feature gathering technique uses either publicly available APIs or mimics users using search engines to gather information on a URL.

Zhang et al. select the top 5 words with highest TF-IDF value to generate lexical signature of a page. They feed each lexical signature to Google search engine and check if the domain name of the current web page matches the domain name of the top 30 results. If yes, they consider it to be a legitimate website. Though the goal is similar, we utilize search engines in different ways. Instead of using the query terms, we use the URL and its domain part.

We check if the URL exists in the search engines' index in the following manner. First, we search for the whole URL and retrieve the top 30 results. Our preliminary experiments showed that retrieving top 10 results was enough to check if a URL has been indexed. We use the top 30 results, even so, to be on the safe side as the work by Zhang et al. show that retrieving more than 30 results doesn't yield any performance improvements. If the results contain the URL, we consider it as a potentially benign URL, phishing otherwise. We also check if the domain part of the URL matches the domain part of any links in the results. 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 a URL. If none of the returned links matches the query URL, we flag the URL as potentially phishing. 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 the URL in question is more likely to be phishing. Hence, we believe that these features also compliment the 'age of domain' feature based on WHOIS used by most of the related works.

Our heuristic, however, makes certain assumptions that the leading top 3 search engines index the vast majority of legitimate websites and that legitimate sites usually live longer and hence, the search engine crawlers will index them sooner or later. On the other hand, the average time a phishing site stays online is 4.5 days or even less [27], [34]. Moreover, there won't be that many links pointing to the phishing web site. Because of the low life span and lack of links pointing to the phishing web site, we assume that search engines crawlers may not get to the site before they are taken down. We employ 3 major search engines with the strong reason that at least one of them may have indexed legitimate website if not all. Furthermore, search engines may try to filter out known malicious links from the search

results using their proprietary technologies. Thus, the heuristic effectively leverages on top search engines crawling resources and proprietary filtering techniques.

To the best of our knowledge, this is the first work to utilize search engines in this manner to detect phishing URLs.

Search engine based features and their statistics are summarized in Table 6.

Table: 6 Search Engine-based Features and Their Statistics

Feature Description	% Phishing URLs	% Non-phishing URLs
URL NOT in Google Top Results	98.71%	4.85%
Domain NOT in Google Top Results	98.27%	2.64%
URL NOT in Bing Top Results	96.95%	34.63%
Domain NOT in Bing Top Results	96.34%	12.77%
URL NOT in Yahoo Top Results	98.93%	17.74%
Domain NOT in Yahoo Top Results	98.71%	13.95%

3.3 Classification Models

Since no single classifier is perfect, we evaluate several supervised batch-learning classifiers. As researchers, we have no vested interest in any particular classifier. These classifiers are chosen mostly because they have been applied to problems similar to ours, such as in detecting: spam and phishing emails, phishing and malicious URLs, phishing webpages, etc. 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 and testing time and accuracy to the problem of detecting phishing URLs.

We evaluate the following 7 classifiers implemented in WEKA (Waikato Environment for Knowledge Analysis) library [15] with their default parameter values:
1) Support Vector Machines (SVMs with rbf kernel) [32]
2) SVMs with linear kernel
3) Multilayer Perceptron (MLP) [16]
4) Random Forest (RF) [8]
5) Naïve Bayes (NB) [17]
6) Logistic Regression (LR) [7]
7) C4.5 [26] – which is implemented as J48 in WEKA.

4. EMPIRICAL EVALUATIONS

Using the features described in Section III-C, we encode each individual URL into a feature vector with 138 dimensions. We scale the real-valued features, available mostly in lexical based features, to lie between 0 and 1. Scaling equalizes the range of the features in real-valued and binary features further emphasizing that we are treating each feature as equally informative and important.

In order to evaluate our methodology, we perform 5 major experiments. We use 10 times 10-fold cross-validation (unless otherwise stated) to evaluate the classifiers. The experiments are run on a machine with 2 dual-core 2 GHz Intel processors and 4 GB memory.

4.1 Experiment 1- Classifier Evaluation

In this experiment, we evaluate classification performance of 7 classifiers on all data sets using the whole feature set. Figure 3, 4, and 5 compare the overall error rates, false positive rates (FPR), and false negative rates (FNR) respectively.

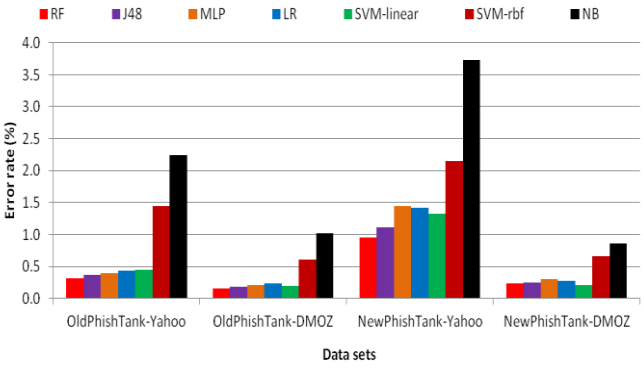

Fig. 3 Overall error rates of all classifiers on each of four data sets using all features.

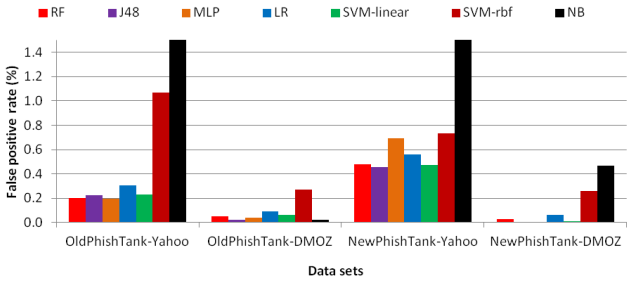

Fig. 4 False positive rates of all classifiers on each of four data sets using all features

Note NB has 2.24% and 3.62% false positive rates on OldPhishTank-Yahoo and NewPhishTank-Yahoo data sets respectively.

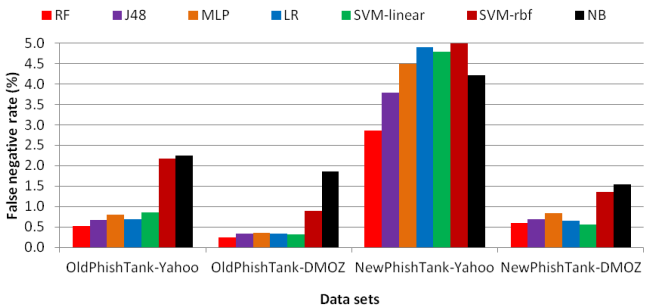

Fig. 5 False negative rates of all classifiers on each of four data sets using all features

Note SVM-rbf has 7.92% and AB has 8.98% false negative rates on NewPhishTank-Yahoo data set.

The differences in overall error rates on all the classifiers are not significant on each data set. Random Forest (RF) performs the best in all performance metrics followed by J48 on each of four data sets. Naïve Bayes (NB) consistently performs the worst followed by SVM-rbf on all data sets. Classifiers yield worst performance on NewPhishTank-Yahoo data set mostly due to high false negatives that range between 2.8–8.9%. RF's error rate ranges between 0.16–0.95%, whereas the NB's error rate ranges between 0.86–3.74%. J48 yields 0% false positives on NewPhishTank-DMOZ data set. False positives rates are normally better than false negative rates for all classifiers on each of the four data sets.

We choose RF classifier in the rest of the experiments primarily because its training and testing times are reasonably fast (see Section IV-C-2) with best overall classification accuracies.

4.2 Experiment 2 – Feature Evaluation

In this experiment, we compare various combinations of feature sets to evaluate how effective each feature category is in detecting phishing URLs. Specifically, we compare individual feature category and combine it with the lexical based feature category – the most commonly used feature category in phishing detection. We use RF classifier on OldPhishTank-Yahoo (OY) data set as it has sufficiently good number of phishing and non-phishing URLs with varieties in URL structures covering most feature categories. Results on these experiments are displayed in Table 7.

Table 7: Search Engine-based Features and Their Statistics

Feature Category	Feature Count	Error Rate	FPR
Lexical based	24	14.62%	8.22%
Keyword based	101	15.39%	5.36%
Lexical + Keyword based	125	9.25%	5.73%
Reputation based	7	15.71%	2.69%
Lexical+ Reputation based	31	8.37%	4.74%
Lexical + Keyword + Reputation	132	5.88%	3.37%
Search Engine based	6	0.47%	**0.15%**
Lexical + Search Engine based	30	0.37%	0.20%
All Features	138	**0.31%**	0.20%

When using lexical based feature type alone, RF classifier achieves an error rate of 14.62%. Similarly, keyword based feature type, which has 101 features, yields a 15.39% error rate. When combined lexical with keyword-based features, the error rate improves to 9.25%. Reputation based feature set with 7 features provide the worst error rate of 15.71%. What is interesting about these feature sets is that each set provides better true negative rate than true positive rate. This indicates that the absence of these features can detect non-phishing URLs with good accuracy, but their presence doesn't necessarily result in higher accuracy in detecting phishing URLs. When the first three feature categories are combined, the error rate significantly improves to 5.88%.

Search engine based feature set alone provides the lowest error rate of 0.47%. It also provides the highest true positive and true negative rates. The experiment on the data set with all feature categories combined gives the best performance results across all the metrics. The combined features provide at best a 0.31% error rate. Search engine based features provide the lowest false positive rate of 0.15% indicating that very few non-phishing URLs are misclassified as phishing using this feature set alone.

As feature selection technique is applied to reduce the size of keyword based feature set (see Section III-C-2), we also experiment using all 12,012 keyword-based features in combination with the rest of the feature set. Using Random Forests (RF) classifier on the OY data set, we obtain slightly worse error rate of 0.75%. Higher error rate was expected given so many uninformative keyword based noise features. On the other hand, the training and testing time on RF was noticeably slower. This emphasizes the importance of feature selection to improve classifier's performance as and when necessary. It would be interesting to see the results of feature ranking and selection on the whole feature set. We leave this for our future work.

These results demonstrate that search engine based features are the most discriminative features in detecting phishing URLs. Combining all the features, however, provide the best accuracy confirming the importance of each feature category.

4.3 Experiment 3 – Time Analysis

Next we investigate the feasibility of real-time application of the proposed approach. In order to prevent Internet users from clicking on phishing URLs in real-time, such a proposed system needs to be highly accurate (very low false positive and negative rates) and needs to have tolerably low response time. In this experiment, we analyze time taken by our prototype system in classifying whether a given URL feed is phishing by considering time taken from generating a feature vector to testing and getting the final verdict.

4.3.1 Feature Collection Time

We expect the system to take the most time in accessing the web and collecting the proposed search engine and some reputation-based features. In our prototype experimental system, the single-threaded feature collector takes about 3.78 seconds in average to generate feature vector from a phishing URL and 3.2 seconds in average from a non-phishing URL. Thus, in general, it takes about 3.49 seconds to collect all the features and generate the feature vector in the format readable by the classifiers. The lower time taken for non-phishing URLs is because most non-phishing URLs are usually present in search engines results and as such the feature collector doesn't have to query search engines again

for just the domain part of the URLs. We believe that we can significantly lower the time to collect features by employing parallel processing and local caching techniques that can access the web and gather various features simultaneously. We leave this as our future work when we build a robust, scalable system to test in the real world.

4.3.2 Training and Testing Times

To provide better comparisons of time taken by the classifiers to train and test the models and to find out the time taken to classify an instance of URL, we use 80/20 percentage split on the OldPhishTank-Yahoo data set with all features for training and testing purposes respectively. We summarize the average results in Table 8 after running the experiments for 5 times. The cross-validation time is the time taken by classifiers in Experiment 1 on NewPhishTank-Yahoo data set.

Table 8: Training and Testing Time Taken by all Classifiers on OY Data Set Using all Features

Average Time	Total Samples	RF	J48	MLP	LR	SMV-lin	SVM-rbf	NB
Train	26860	1.05 m	2.28 m	1.69 h	0.56 m	1.65 m	1.56 m	**0.15 m**
Test	6714	0.03 s	**0.003 s**	88.08 s	0.03 s	2.0 s	2.24 s	0.20 s
Test	1	0.021 ms	**0.003 ms**	13.12 ms	0.03 ms	1.33 ms	2.01 ms	0.19 ms
Cross-validation	27669	14.6 m	36.2 m	20.9 h	6.3 m	14.1 m	12.7 m	**1.7 m**

Naïve Bayes is widely used in spam filters partly because the training and testing times are fast. We see similar result with NB taking the fastest 0.15 minutes to build the model. However, it also performs the worst in terms of error rate. MLP has the worst training time (in hours) and also worst testing time. We can observe that Random Forests (RF) has one of the best tradeoff between train and test time and overall accuracy. RF's training time is second best and comparatively very close to that of NB's, and its overall performance results are the best among all the classifiers (see Experiment 1).

Though training time is generally higher compared to testing time, training may be done offline and less frequently. It's the test time that is crucial in providing the real-time service of detecting phishing URLs.Time taken to test a URL, once the model is trained, by most of the classifiers is in very low milliseconds and negligible compared to time taken to collect features. Overall, our single-threaded experimental system can classify a URL in less than 3.5 seconds in average. We understand that this time may not be acceptable in real-time classification system. Nonetheless, we believe that there's a lot of room for improvements during the design and implementation of the system for large-scale deployment to effectively provide near real-time service. Furthermore, users may likely tradeoff some speed for its high accuracy when it comes to detecting potentially dangerous phishing URLs.

4.4 Experiment 4

In these experiments, we demonstrate how a classifier's performance varies when using mismatched data sources and temporal-based data sets.

4.4.1 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, experiment results from study by Ma et al. [18] show that when training and testing sources are completely mismatched, a classifier's performance decreases significantly. To investigate if this phenomenon holds in our case, we experiment with training and testing on various combinations of phishing and non-phishing sources of URLs. There's one caveat, however, in our phishing URLs data sources. As we couldn't find the second credible source for phishing URLs, we generate two phishing data sets (separated by two whole months of collection time) from the same source PhishTank as described in Section III-B. Even though it may not make a strong case for different sources, we argue that it does make a very strong case for investigating data drift (evolving features in phishing URLs, see Section IV-D-2).We use the abbreviations defined in Section III-B to refer to each combination of data sets, e.g., OY for Old PhishTank-Yahoo.

Table 9: Training and Testing Time Taken by all Classifiers on OY Data Set Using all Features

Training	Testing (Error Rate)			
	OY	OD	NY	ND
OY	0.31%	0.08%	2.69%	4.96%
OD	10.92%	0.09%	13.39%	0.33%
NY	0.47%	0.79%	0.87%	0.21%
ND	10.45%	0.27%	12.54%	0.33%
All data sets (OYND)	0.33%	0.06%	0.10%	0.16%

Table 9 shows classification results of training and testing on mismatched data sets using Random Forest classifier.

As expected, when trained and tested RF classifier with the 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 IX). When new phishing URLs are tested

against the model trained from old phishing URLs, the error rate is 2.69% contributed mostly by false negatives.When only the non-phishing URL source is mismatched (e.g., OD and OY), error rate increases due to higher false positives. This observation is similar to the one observed by Ma et al. [18]. However, when only the phishing URL 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 2.69% (OY and NY). This small increase in error rate may be due to the fact that the source of phishing URLs are not technically different but they differ by the time when phishing URLs were submitted for verification. We look more into this 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 data sets and Yahoo data set (OY, NY), the error ranges between 10–13%, contributed mostly by high false positives. When trained with URLs from all the data sets, the model generalizes well and yields good performance results across all test data sets (last row). This experiment concludes that the classifiers trained using URL data from one data source may not generalize well while testing data from different data source. Furthermore, it shows that 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 in the important problem of detecting phishing URLs.

Though instances in data sets may be different, we can compare these results with those in Ma et al. [18] in cases where the data sources are similar. Their data sets include 5,500 phishing URLs from PhishTank and all of their benign URLs come from Yahoo and DMOZ.When trained and tested with the various combinations of Yahoo, DMOZ, and PhishTank data sources, our method achieves errors in the range 0.06–13.39%, while their approach report errors in the range 1.24–33.54% on the same data sources. They achieve at best 0.90% on the split of the Yahoo-Spamscatter data source showing that their approach is better at detecting spam URLs than at detecting phishing URLs. We argue that our approach achieves superior performance in detecting phishing URLs.

4.4.2 Concept Drift

Phishing tactics and URL structures keep changing continuously over time as attackers come up with novel ways to circumvent the existing filters. As phishing URLs evolve over time, so must the classifier's trained model to improve its performance. Ma et al. [19] conclude that retraining algorithms continuously with new features is crucial for adapting successfully to the ever evolving malicious URLs and their features. An interesting future direction would be to find the effect of variable number of features using online algorithms in detecting phishing URLs.

We use OldPhishTank data set and 22,722 (twice the number of phishing URLs) randomly selected non-phishing URLs from Yahoo and DMOZ data sets as our "base" training set. Figure 6 shows the classification error rates for classifiers after training them only once using the "base" training set. The *x*-axis shows number of weeks in the experiment with the phishing URLs collected from January 1st week to May 1st week of 2011, and the *y*-axis shows the error rates on testing the classifier with phishing URLs collected each week. For non-phishing URLs, to generate each week's test data, twice the number of phishing URLs is randomly selected from Yahoo and DMOZ data sets.

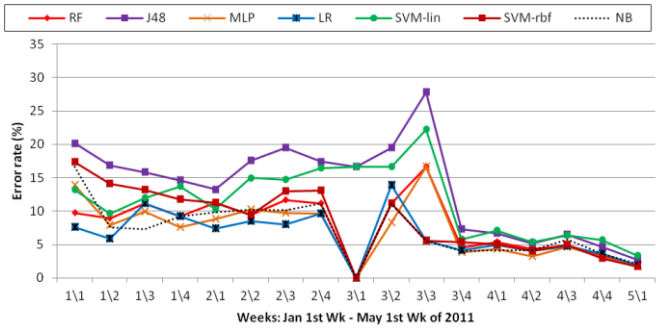

Fig 6 Error rates for all classifiers after training them once and testing them on weekly data.

While most classifiers perform poorly in this experiment, J48 performs the worst for most of the weeks with error rate reaching as high as 28% when testing with data from 3rd week of March (3\3). Random Forests (RF) and Logistic Regression (LR), in general, perform better with error ranging from 0–13%. For all the classifiers, the high error rate is due to high false negatives, which get as high as 60–80% for some weeks. For week 3\1 data, most classifiers yield a 0% error rate, as there are only 2 phishing URLs and 4 non-phishing URLs for the week's test data. These high error rates due to high false negative rates suggest that the model must be retrained on fresh data to account for new combinations of features on phishing and non-phishing URLs over time in order to keep the model fresh and achieve better performance.

To address this issue, we retrain classifiers every week with training data collected up to that week and test on the data from the following week. We show these results in Figure 7. To test the models for data collected on January 1st week (1\1), for example, we train classifiers using "base" training set. To test data collected on week 1\2, we retrain all the classifiers with "base" training set plus all data collected up to the previous week (1\1 in this case). As a result, the error rates decrease over time due to significant improvement in false negative rates week after week. For RF classifier, error rate decreases from 9% on the first week to 0.4% for the last week. Interestingly, we see the most improvement in performance of J48. Its error rate starts with the highest 20% for the first week and it gradually improves to 0.6% on the last week. Although fresh data eventually helps most classifiers improve over previous experiment where the classifiers are trained only once, we feel that a week's training data is still insufficient. By looking at the trends in our experiments and as observed by Ma et al. [19], training on daily data or perhaps using incremental training with individual instance using online algorithms may improve the results on classification of continuously changing phishing

URLs. But due to lack of enough data for a lot of individual days in our data sets, we leave these experiments for future work when we'll collect phishing URLs from several feeds on a large scale.

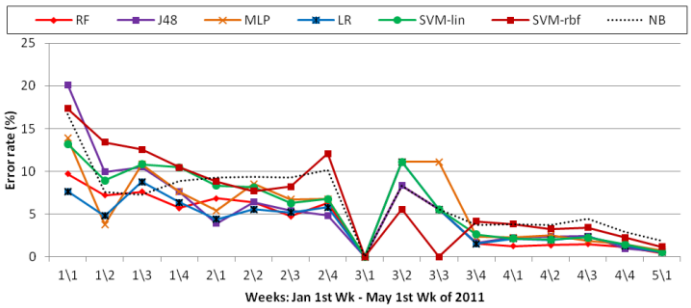

Fig. 7 - Error rates for all classifiers after training them every week.

4.5 Experiment 5 - Top Targets

In our OldPhishTank-Yahoo (OY) data set (see Section III-B), we include legitimate URLs of top online brands that are frequently targeted by phishers. Since our goal is to detect phishing URLs that forge legitimate URLs, we try to investigate how our proposed method performs against the URLs from legitimate top targets. To that end, we combine all the data sets (OYND) except the top targets' URLs and use the combined 46,992 instances as a training data set. We use the rest 1,674 instances as a test data set. When trained and tested Random Forest (RF) with these data sets, it yields a 19.35% error rate, meaning only 80.65% of legitimate URLs related to the top targets are correctly classified as non-phishing URLs and the rest are all misclassified as phishing URLs. This large error rate is due to the fact that these target URLs look very similar to the forged phishing URLs, and the model, perhaps, is not trained with the types of URLs that are seen during testing.

In order to address this, we randomly select 75% of legitimate URLs from top targets and include them in the training set and retrain the model. We test the newly trained model with the remaining 418 instances of top target URLs. As expected, the results improve significantly, yielding a 1.67% error rate, misclassifying only 7 non-phishing URLs 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 URLs the system will be actually tested against.
Unlike related works ([13], [18], [19], [33]), 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 URLs but also in detecting the legitimate URLs of the actual targets.

5. DISCUSSIONS

5.1 Limitations

Despite offering low error rates, there are some limitations to our study. First, all of our phishing URLs came from a single source PhishTank; therefore the URLs received may not be representative of all phishing URLs. However, PhishTank not only automatically collects potential phishing URLs from users' email applications, it also allows registered users to manually submit a URL (perhaps received from various attack vectors besides email) for verification.

Search engine based features contribute to a major performance bottleneck due to the time lag involved in querying search engines. Also the service providers may either permanently block our system citing potential of service (DoS) attack originating from our system or temporarily limit the access to their services to manage throttle. Using local caching, our system must manage not to flood search engines with requests.

As our reputation based features rely on blacklists and other historical statistics provided by third parties, we have no control over the quality and reliability of the services and data provided by them. Though absence of these features may not significantly hurt the performance of our approach, we certainly can look for alternative sources.

PyLongURL script [4] couldn't automatically expand some shortened phishing URLs mostly because the legitimate shortening services blocked and removed them after receiving reports of phishing attacks. Even though these short URLs lack all the lexical and keyword based features, most of them are correctly classified because of the search engines and reputation-based features. This is very less likely to happen on legitimate URLs shortened using legitimate shortening services as the script will be able to expand them to their final URLs. Nevertheless, we experimented on OldPhishTank-Yahoo data set after removing the shortened URLs that couldn't be automatically expanded. The differences in classifiers' performance results, however, were statistically insignificant.

5.2 Error Analysis

We examine URLs that contributed to false positives and false negatives. This can further reveal limitations of the approach and possibly provide potential improvements to the current approach. We examine the test results on the whole data set (ONYD) with all features using Random Forest classifier with 80/20 percentage split test option. Some internal links that we harvested from legitimate highly targeted web sites have properties similar to phishing URLs, for examples: *http://www.standardbankbd.com/pages_details.php?id=35&phpsessid=39327de0de6269760dc6a1f3fb630*, *http://moneysense.natwest.com/natwest/adults/makingbankingsimple/loans.asp?page=MONEYSENSE/ADULTS/MAKING_BANKING_SIMPLE/LOANS*, etc. Perhaps newly generated links or due to the way search engines index URLs, the search engines didn't have these links indexed. Because some target names and some keywords such as login, signoff, etc. are present in them and they are usually lengthy as seen in examples, these non-phishing URLs resemble more like phishing URLs. However, this is as much an issue with the selection of training data as it is with

the methodology. Perhaps a second and much shorter work could mitigate these false positives using enhanced benign URL harvesting methods. Similarly, our method may flag new legitimate web pages as phishing in particularly those that have not yet been crawled and indexed by search engines.

Some URLs such as *http://whoblocksyou.net*, *http://www.checkmessenger3.net/en/*, *http://you-areblocked.com*, *http://www.yahooblockchecker.info/*, etc. and their variants are classified as non-phishing. This group of web pages promises users to find if any contacts in MSN, Yahoo or any common instant messenger have blocked you once you provide your email and password. These websites seem to be around for a while, some from as far back as 2007. Even though they have been confirmed as phishing sites by the PhishTank community long back, interestingly, the sites still exist. Moreover, Chrome, Firefox, and Internet Explorer didn't block these web sites (as of August 10, 2011) implying that these links were not in their blacklists.
Other group of web pages such as *http://home.comcast.net/~mikeskipper/*, *http://habbomatanza.galeon.com/*, etc. hosted on free legitimate hosting services contributed to more false negatives. These URLs do not have any red flag keywords and their domain and in some instances even the whole URLs were in search results and were not in browsers blacklists.

We believe that by looking into the page contents of URLs, some of these false negatives can be mitigated.

5.3 Tuning False Positives and Negatives

In case of detecting phishing URLs, false positives may be tolerated more than false negatives or vice versa. With false positive URLs, users have to be extra vigilant while loading the URL and manually confirm if the webpage is legitimate before submitting any sensitive personal information. False negatives, on the other hand, may provide false sense of security and users may end up giving up their personal information to a forged webpage. Instead of minimizing the overall error rate, for policy reasons or personal security preferences, users may want to tune the decision threshold to minimize the false negatives at the expense of more false positives or vice versa.

Figure 8 shows the tradeoff between false negative and false positive rates as an ROC graph for Random Forest over an instance of whole data set (ONYD) using all the features. The highlight on the figure shows that if the false positives are tuned to 0.15%, the model achieves a false negative rate of 3.16%. If we can tolerate a little higher false positive rate to 0.4%, however, we can achieve a lower false negative rate of 1.05%.

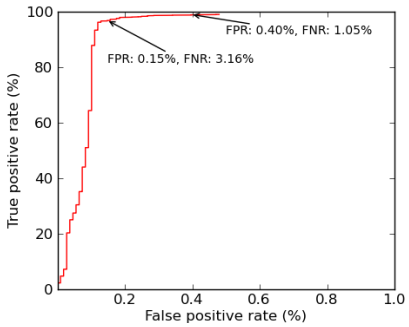

Fig. 8 ROC graph showing tradeoff between false negatives and false positives. Note that the x-axis ranges between 0-1%

5.4 Potential Adversarial Attacks

As search engine based features are highly discriminative, attackers may try to launch distributed denial of service (DDoS) attack on search engines. It is not very likely to happen on all three of them simultaneously, however. Though quality and index size of a search engine plays a big role in our approach to correctly classify phishing and legitimate URLs, we can look for an alternative as there are plenty to choose from. Using Blackhat SEO techniques, adversaries can get their phishing websites crawled in a short period of time. Though page ranking is not an issue, our technique may provide a large number of false negatives if phishing links are simply found in search engines' indexes. Adversaries may buy established domain names or establish a new web site hosting benign contents for a while. Once the major search engines index the web site, they may exploit this fact and start hosting phishing web pages effectively avoiding search engine based features. Consequently, buying a domain name, hosting a website for a long time, employing Blackhat SEO to alter the PageRank of a phishing page require significant investment, which reduces the potential profit from the phishing campaign.

Phishers may try to evade our heuristic-based approach by using new URL shortening services that we are either not aware of or do not know how to utilize the service automatically. Study shows that scammers are now establishing their own fake URL-shortening services [21]. Under this scheme, shortened links created on these fake URL-shortening services are further shortened by legitimate URL-shortening sites. These links are then distributed via phishing emails, blogs, micro-blogs, and social networking websites. Though we didn't observe it in our data sets, we anticipate this tactic to be used against our proposed system and see a need to address it in our future work. We can always enhance the capabilities of our URL expanding script by actively looking out for new shortening services and incorporating them into our script. Moreover, in order to successfully evade our approach, phishers not only have to use a rare shortening service, but also have to work hard to get those short links indexed by search engines.

Adversaries may try to reduce the information content in the lexical and keyword based features of URLs thus effectively reducing the phishing like features and making their links

more legitimate. Another approach is to leverage well-known infrastructure such as hosting phishing page on a legitimate popular domain such as free webhosting services or by breaking into legitimate web sites or by exploiting common Cross-site Scripting (XSS) vulnerabilities [10]. We can overcome this drawback by looking into the contents of the web pages. This drawback, however, is not particular to our approach, but to all the approaches that rely only the URL metadata and structures to detect potential maliciousness.

To make reputation based features less suspicious, phishers may try to host their contents in domains and IPs that do not have historically bad reputation of hosting malicious websites. Sites with good reputation, however, are either too difficult to exploit or their administrators typically remove malicious pages under their control promptly, thus limiting the potential audience and profitability of phishing campaign hosted in their web servers.

Furthermore, since we provide equal weight to all the features, phishers do not have an opportunity to target higher weight features in order to invade our classifiers.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed new search engines, reputation, and statistically mined keyword based features for classifying phishing URLs. We empirically demonstrated that the proposed features are highly relevant to the automatic discovery and classification of phishing URLs. We evaluated our approach on real-world public data sets by comparing performance results of several popular supervised learning methods. Experimental results showed that the proposed anti-phishing solution was able to detect phishing URLs with an accuracy of more than 99.4% while maintaining false positive and false negative rates of less than 0.5%. We showed that our experimental prototype, once trained, could classify a given URL as phishing or non-phishing with a turnaround time of about 3 seconds.

Most classifiers except Naïve Bayes showed statistically similar performance metrics. For our problem, Random Forest (RF) classifier, however, provided the best tradeoff between the classification performance and the training and testing time. RF consistently outperformed all other classifiers in most experiments. We have shown that the steps of selecting representative training set and retraining algorithms continuously with fresh data are crucial aspects in adapting successfully to the ever-evolving stream of URLs and their features.

As our future work, we plan to develop a framework using this approach and deploy it for a large-scale real-world test. We hope to improve the feature collection time by employing parallel processing and local caching. We will also investigate the effectiveness of online algorithms as they have been found to outperform traditional batch algorithms in problem similar to ours [19]. We believe that by looking into the contents of web pages, we can further improve false positives and negatives. We're currently investigating this matter as well.

ACKNOWLEDGMENTS

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. Thanks go to Max Planck, Darryl Ackley, and Tenzin Doleck for providing helpful feedbacks on the paper.

REFERENCES

[1] APWG. 2010 1st Quarter Report, http://www.antiphishing.org/reports/apwg_report_Q1_2010.pdf, accessed on January 17, 2011.

[2] AVG Security Toolbar, http://www.avg.com/product-avg-toolbar-tlbrc#tba2, accessed on July 10, 2011.

[3] R.B. Basnet, S. Mukkamala, A.H. Sung, Detection of phishing attacks: a machine learning approach, In: Bhanu Prasad (Ed.), Studies in Fuzziness and Soft Computing, Springer, 2008, pp. 373-383.

[4] R.B. Basnet, PyLongURL - Python library for longurl.org, software available at: http://code.google.com/p/pylongurl/, 2010.

[5] R.B. Basnet, A.H. Sung, Classifying phishing emails using confidence-weighted linear classifiers, In: Proc. Int. Conf. Information Security and Artificial Intelligence, ISAI'10, Chengdu, China, 2010, pp. 108-112.

[6] R.B. Basnet, A.H. Sung, Q. Liu, Rule-based phishing attack detection, In: Proc. Int. Conf. Security and Management , SAM'11, Las Vegas, NV, USA, 2011.

[7] M.T. Brannick, Logistic regression, http://luna.cas.usf.edu/~mbrannic/files/regression/Logistic.html, accessed on February 27, 2011.

[8] L. Breiman, Random forests, http://oz.berkeley.edu/users/breiman/randomforest2001.pdf, 2001, accessed on February 20, 2011.

[9] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, J. Mitchell, Client-side defense against web-based identity theft, In: Proc. 11th Network and Distributed System Security Symposium, NDSS'04, San Diego, CA, USA, 2004.

[10] Cross-site Scripting (XSS), http://www.owasp.org/index.php/Cross-site_Scripting_(XSS), accessed on March 12, 2011.

[11] R. Dhamija, J.D. Tygar, Hearst M, Why phishing works, In: Proc. Int. Conf. Human-Computer Interaction, CHI'06, Montreal, Quebec, Canada, 2006, pp. 581-590.

[12] I. Fette, N. Sadeh, A. Tomasic, Learning to detect phishing emails, In: Proc. Int. Conf. World Wide Web, WWW'07, Banff, Alberta, Canada, 2007, pp. 649-656.

[13] S. Garera, N. Provos, M. Chew, A.D. Rubin, A framework for detection and measurement of phishing attacks. In: Proc. 5th ACM Workshop on Recurring Malcode, WORM'07, ACM, New York, NY, USA, 2007, pp. 1-8.

[14] Google Safe Browsing API - Google Code, http://code.google.com/apis/safebrowsing/, accessed on June 12, 2010.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, SIGKDD Explorations, 11 (2009) 10-18.

[16] K. Hornik, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359-366.

[17] G.H. John, P. Langley, Estimating continuous distributions in Bayesian classifiers, In: Proc. 11th Conf. Uncertainty in Artificial Intelligence, San Mateo, CA, USA, 1995, pp. 338-345.

[18] J. Ma, L.K. Saul, S. Savage, G.M. Voelker, Beyond blacklists: Learning to detect malicious web sites from suspicious URLs, In: Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, Paris, France, 2009, pp. 1245-1254.

[19] J. Ma, L.K. Saul, S. Savage, G.M. Voelker, Identifying suspicious URLs: an application of large-scale online learning, In: Proc. 26th Annual Int. Conf. Machine Learning, ICML'09, Montreal, Quebec, Canada, 2009, pp. 681-688.

[20] McAfee SiteAdvisor Software – Website Safety Ratings and Secure Search, http://www.siteadvisor.com, accessed on July 15, 2011.

[21] Microsoft Security Intelligence Report, Vol.10, 2011, http://www.microsoft.com/security/sir/default.aspx, accessed on June 5, 2011.

[22] Natural Language Toolkit (NLTK), http://www.nltk.org, accessed on July 15, 2011.

[23] Netcraft Anti-Phishing Toolbar, http://toolbar.netcraft.com, accessed on June 23, 2011.

[24] H. Peng, F. Long, C. Ding, Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy, IEEE Transaction on Pattern Analysis and Machine Intelligence 27 (2005) 1226-1238.

[25] PhishTank. Out of the net, into the tank, http://www.phishtank.com, accessed on June 18, 2010.

[26] J. R. Quinlan, C4.5 programs for machine learning, Morgan Kaufmann Publishers, San Mateo, CA, USA, 1993.

[27] S. Sheng, B. Wardman, G. Warner, L.F. Cranor, J. Hong, C. Zhang, An empirical analysis of phishing blacklists, In: Proc. 6th Int. Conf. Email and Anti-Spam, CEAS'09, Mountain View, California, USA, 2009.

[28] SmartScreen Filter – Microsoft Windows, http://windows.microsoft.com/en-US/internet-explorer/products/ie-9/features/smartscreen-filter, 2011.

[29] SpamBayes: Bayesian anti-spam classifier written in Python, http://spambayes.sourceforge.net, accessed on July 27, 2011.

[30] SpoofStick Home, http://www.spoofstick.com, accessed on July 25, 2011.

[31] StopBadware, IP Address Report – Top 50 by number of reported URLs, http://stopbadware.org/reports/ip, accessed on June 12, 2010.

[32] V.N. Vapnik, The nature of statistical learning theory, Springer, 1995.

[33] C. Whittaker, B. Ryner, M. Nazif, Large-scale automatic classification of phishing pages, In: Proc. 17th Annual Network and Distributed System Security Symposium, NDSS'10, San Diego, CA, USA, 2010.

[34] Y. Zhang, J. Hong, L. Cranor, CANTINA: a content-based approach to detecting phishing web sites, In: Proc. 16th Int. Conf. World Wide Web, WWW'07, Banff, Alberta, Canada, 2007, pp. 639-648.