CANTINA+: A Feature-rich Machine Learning Framework for Detecting Phishing Web Sites

GUANG XIANG, JASON HONG, CAROLYN P. ROSE, LORRIE CRANOR
School of Computer Science
Carnegie Mellon University

Phishing is a plague in cyberspace. Typically, phish detection methods either use human-verified URL blacklists or exploit webpage features via machine learning techniques. However, the former is frail in terms of new phish, and the latter suffers from the scarcity of effective features and the high false positive rate (FP). To alleviate those problems, we propose a layered anti-phishing solution that aims at 1) exploiting the expressiveness of a rich set of features with machine learning to achieve a high true positive rate (TP) on novel phish, and 2) limiting the FP to a low level via filtering algorithms.

Specifically, we proposed CANTINA+, the most comprehensive feature-based approach in the literature including eight novel features, which exploits the HTML Document Object Model (DOM), search engines and third party services with machine learning techniques to detect phish. Moreover, we designed two filters to help reduce FP and achieve runtime speedup. The first is a near-duplicate phish detector that uses hashing to catch highly similar phish. The second is a login form filter, which directly classifies webpages with no identified login form as legitimate.

We extensively evaluated CANTINA+ with two methods on a diverse spectrum of corpora with 8118 phish and 4883 legitimate webpages. In the randomized evaluation, CANTINA+ achieved over 90% TP on unique testing phish and over 99% TP on near-duplicate testing phish, and about 0.4% FP with 10% training phish. In the time-based evaluation, CANTINA+ achieved over 87% TP on unique testing phish, about 95% TP on near-duplicate testing phish, and about 1% FP under 20% training phish with a two-week sliding window. Capable of achieving 0.4% FP and over 90% TP, our CANTINA+ has been demonstrated to be a competitive anti-phishing solution.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General – Security and Protection; H.3.3 [**Information Storage and Retrieval**]: Retrieval Models; I.5.1 [**Computing Methodologies**]: Pattern Recognition

General Terms: Algorithms, security, languages

Additional Key Words and Phrases: Anti-phishing, machine learning, information retrieval

1. INTRODUCTION

Phishing is a form of identity theft, in which criminals build replicas of target websites and lure unsuspecting victims to disclose their sensitive information like passwords, personal identification numbers (PINs), etc. According to one survey by the Gartner Group [McCall 2007], phishing attacks in the United States caused

Author's address: Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213.

$3.2 billion loss in 2007, with about 3.6 million victims falling for the attacks, a huge increase from the 2.3 million the year before. Moore et al [Moore and Clayton 2007] reported that the loss to consumers and businesses in 2007 in the US alone was around $2 billion. A significant proportion of those losses was caused by one particularly infamous group, known as the "rock phish gang", that uses toolkits to create a large number of unique phishing URLs, putting additional pressure on the timeliness and accuracy of blacklist-based anti-phishing techniques. Although there have been many advances in anti-phishing solutions in recent years, phishing still causes tremendous losses every year.

The exact definition of phish varies from paper to paper. Here, we define phish to be a webpage satisfying the following criteria:

(1) It impersonates a well-known website by replicating the whole or part of the target site, showing high visual similarity to its target.

(2) It has a login form requesting sensitive information such as a password.

Sometimes, phishing attacks are launched in multiple pages where users need to click a few buttons before arriving at the page with login forms. Though absent initially, login forms will appear eventually due to the nature of phishing activity.

Generally, mainstream anti-phishing methods either utilize human-verified URL blacklists, or go for phish via features using machine learning algorithms. While the former is intuitive, mostly involving URL matching, the latter is relatively complicated. The general idea of any feature-based method is to first find a set of discriminative features that could help distinguish phish and non-phish, then learn a machine learning model based on those features over a training set composed of phish and legitimate pages, and finally apply the model to classify a web page whose identity is to be examined. Among those steps, obtaining a high quality feature set is of primary importance. A variety of features have been proposed for phish detection in the literature, however, some of those features are not discriminative enough, and almost all of them only utilize the URL and HTML DOM to examine the discrepancy between phish and non-phish. In our work, we propose the most comprehensive feature-based approach using a rich set of resources including the URL, HTML DOM, third party services, and search engines to detect phish, which is demonstrated to be very effective against phishing attacks by our experiment and is the primary contribution of our work in this paper.

Though somewhat effective, existing solutions have apparent weaknesses. While human-verified blacklists are widely adopted in industry due to the very low FP, they do not generalize well to new attacks. For example, Sheng et al [Sheng et al. 2009] found that zero hour protection offered by major blacklist-based toolbars has a TP between 15% and 40%. Furthermore, human-verified blacklists can be slow to respond to new phishing attacks, and updating blacklists usually involves enormous human effort. For example, PhishTank, a community site where people can view and confirm phishing sites, posted statistics in March 2009 showing that it took on average 10 hours to verify a submitted URL [PhishTank a]. Sheng et al found that blacklists were updated at different speeds, and an estimated 47% - 83% of phish appeared on blacklists 12 hours after they were launched [Sheng et al. 2009]. Finally, human-verified blacklists can be easily overwhelmed by automatically generated URLs, which is a tactic in use already by rock phish.

On the other hand, while feature-based approaches offer a more general mechanism to detect novel attacks, not many effective features have been proposed. Furthermore, this type of methods tend to have a relatively higher FP. Concerns over liability for false positives have been a major barrier to deploying these technologies [Sheng et al. 2009]. To underscore this point, Sheng et al [Sheng et al. 2009] evaluated eight popular toolbars, all of which employ human verified blacklists to achieve an extremely low FP in spite of the amount of human labor required.

An ideal anti-phishing solution needs to have reasonable TP against new attacks with very low FP while involving minimum manual labor. The key to achieve a high TP is to design new features that are characteristic of phishing patterns, and the core ingredient leading to a very low FP is filtering via heuristics. With those in mind, we set as our goal in this paper contributing to the literature by addressing the weaknesses of both blacklists and feature-based methods in a unified framework. Specifically, we propose novel features to improve the TP and design filtering algorithms absent in the literature to reduce FP and human effort.

We name such a layered system CANTINA+, which exploits the generalization power of machine learning techniques and the expressiveness of a rich set of webpage features to detect phish variants. Our pipeline consists of three major modules. The first leverages the high similarity among phishing webpages due to the prevalent use of phishing toolkits, and examines a webpage's similarity to known phishing attacks via hashing to filter highly similar phish. The second exploits the property that phishing attacks usually utilize login forms to request sensitive information, and employs heuristics to filter webpages with no login forms prior to the classification phase. The third module, the core of our framework, utilizes 15 highly expressive features with machine learning algorithms to classify webpages. This module adopts the idea of extracting website ownership from our previous work [Xiang and Hong 2009] in building two features, and significantly extends our past work with CANTINA [Zhang et al. 2007] by eight novel discriminative features. Details about contributions of this paper over [Zhang et al. 2007] are given in section 2.

This paper makes the following three research contributions. First, we propose eight novel features capturing the intrinsic characteristics of phishing attacks using a wide spectrum of resources, including the HTML DOM, search engines, and third party services, obtaining superior classification performance. Second, our approach ameliorates the typical weakness of high FP of feature-based approaches by using a layered structure with login form filtering. Note that login form detection is quite nontrivial due to the flexibility of the HTML DOM, which will be explained in section 3. Third, the diversity of web pages in our corpus and the comprehensiveness of our evaluation methods all exceed the techniques in the literature.

In our experiment, we evaluate our approach on a rich corpus with web pages from six categories, and conducted a thorough experiment with randomized and time-based methodologies to inspect the generality of our method as well as its real-world performance. In the randomized evaluation, CANTINA+ achieved an over 90% TP on unique testing phish, an over 99% TP on near-duplicate testing phish, and an about 0.4% FP under 10% training phish with login form filtering. In the time-based evaluation, our method achieved an over 87% TP on unique testing phish, an about 95% TP on near-duplicate testing phish, and an about 1%

FP under 20% training phish with a two-week sliding window. Those phishing attacks whose timestamps fall in the sliding window will be used to train machine learning models, and by using such a length-adjustable moving window, we are able to incorporate the latest phishing variants into our training data and also achieve runtime speedup. There has not been any experimental evaluation as to what is acceptable for end users in the literature, and we have among the lowest FP rate of any feature-based detection techniques out there. It is possible to get even lower false positives using extremely conservative features, though this would significantly impact true positives.

The remainder of this paper is organized as follows. Section 2 presents a literature review, followed by a high-level description of the CANTINA+ architecture in section 3. Subsequently, we elaborate on our hash-based near duplicate phish detection and login form filtering algorithms in section 4 and section 5 respectively. Section 6 is dedicated to a detailed depiction of our feature set and machine learning algorithms. The experiment setup is sketched out in section 7, and the full experimental results using both evaluation methodologies are reported in section 8. In section 9, we present a detailed error analysis, portray potential exploitation techniques phishers may adopt to defeat our design, and point out some other limitations of our proposed approach. We wrap up with conclusions in section 10.

2. RELATED WORK

In terms of client applications, anti-phishing techniques can be categorized into phishing email detection and phishing webpage detection. One major work for the former is PILFER [Fette et al. 2007], in which Fette et al designed ten features exploiting the email HTML and employed random forest as the learning algorithm to classify emails. In another work, Abu-Nimeh et al [Abu-Nimeh et al. 2007] adopted the bag-of-words strategy and simply used a list of 43 most frequent words as features in a machine learning approach. Since our focus in this paper is on the latter and feature-based techniques dominate in the anti-phishing arena, we will focus the related works in feature-based phishing web page detection.

One area of work in the literature uses URL features to detect phishing webpages. Garera et al [Garera et al. 2007] categorized phishing URLs into four groups, each capturing a phishing pattern, and used a set of fine-grained features from the phishing URLs together with other features to detect phish. Applying a logistic regression model yielded an average TP of 95.8% and FP of 1.2% over a repository of 2508 URLs. Though interesting, this method has unstable performance in that URLs could be manipulated with little cost, causing the features to fail.

Researchers have also proposed approaches that guard users against phishing attacks by monitoring the information flow, mostly the password. Kirda et al implemented a system called AntiPhish [Kirda and Kruegel 2005], which watches the password field of the HTML form and searches the domain of the site being visited among a list of previous logins when identical password is found. AntiPhish warns users of potential attacks if a domain match is not found. One problem with this tool is that manual labor is usually involved, and also false positives could be raised if the same password is used on multiple sites, which is what users typically do. Moreover, no formal evaluation of the tool was conducted. Another work is

password hashing (PwdHash) by Ross et al in [Ross et al. 2005]. PwdHash sends a hash value computed from the user's password and the website domain, rather than the plain text password, to the server for authentication, rendering password stealing at the fake phishing site futile. However, as claimed by the authors, their technique suffers from spyware, DNS attacks, focus stealing, etc. Our approach in this paper, however, focuses on another perspective of the anti-phishing campaign, i.e., detecting a phishing site before users input their passwords. We believe that the safest way to protect the users is to identify fake sites and warn them not to give their passwords, rather than encrypt the passwords. In a recent study, Yue et al [Yue and Wang 2010] designed a client-side tool called BogusBiter, which sends a large number of bogus credentials to suspected phishing sites, hiding the real credential among the bogus ones. However, BogusBiter relies on web browsers' built-in components or third-party toolbars to detect phish and thus is more of a reaction rather than detection technique.

Another area explores the visual and image elements to protect users from phishing attacks. To exploit visual similarity between webpages, Liu et al [Liu et al. 2005] proposed a method using three similarity metrics, i.e., block level similarity, layout similarity and overall style similarity, based upon webpage segmentation. A page is reported as phishing if any metric has a value higher than a threshold. Experiment on 320 official bank pages and 8 phishing pages showed a 100% TP and 1.25% FP. Though interesting, this work suffers from the following weaknesses: small corpus size, a high FP, and the potentially instability due to the high flexibility of the layout and style elements in the HTML documents. In [Dhamija and Tygar 2005], Dhamija et al proposed a new scheme named dynamic security skins, which authenticates the server by users' visual verification of the expected image and an image or "skin" generated by the server. Though interesting, this paper has no formal evaluation. SpoofGuard [Chou et al. 2004], a technique proposed by Chou et al., utilizes image check as one feature, examining the domain name and the existence of popular target site logos on a web page. However, [Chou et al. 2004] simply performed exact matches for all images on a page, which is easy to beat and very computationally prohibitive. Another work that exploits visual elements is [Medvet et al. 2008], in which Medvet et al compute a signature using the visible text, visible images and overall visual look-and-feel to compare the suspected pages with their legitimate counterparts. They conducted pairwise comparison along each dimension, which is costly even with some optimization. Moreover, finding the phishing target site is a very hard problem, and Medvet et al did not explain how they did that for a phishing attack impersonating any general brand. Recently, Chen et al [Chen et al. 2010] took a holistic view of the visual similarity between web pages, and applied compression algorithms on the pages as indivisible entities to detect phish. One problem of their approach is that it cannot handle novel attacks well.

Techniques that make use of features based on the HTML DOM to detect phish are also available. In [Rosiello et al. 2007], Rosiello et al came up with a layout-similarity-based approach, trying to overcome the problem in [Kirda and Kruegel 2005] that the same password for multiple sites tend to raise false alerts by AntiPhish. They added one extra layer of examination, checking the similarity of the HTML DOM between the web page currently being visited and the one visited be-

fore with the same password. This technique, however, is brittle in that the HTML DOM is very easy to manipulate without changing the layout of the web page. In [Ludl et al. 2007], Ludl et al applied a J48 decision tree algorithm on 18 features solely based on the HTML and URL, achieving a TP of 83.09% and a FP of 0.43% over a corpus with 4149 good pages and 680 phishing pages. However, features purely based on HTML DOM and URL are rather limited and may fail in capturing artfully designed phishing attacks. Studies [Zhang et al. 2007] have shown that search engines and third party services like WHOIS are effective in providing clues about the legitimacy of a webpage, and our research in this paper fully utilizes those tools in defining a rich set of high-level webpage features.

Another feature-based work exploring the HTML DOM is CANTINA [Zhang et al. 2007], in which Zhang et al proposed a content-based method using a simple linear classifier on top of eight features, achieving an 89% TP and a 1% FP on 100 phishing URLs and 100 legitimate URLs. In CANTINA+, we added significant new ingredients and achieved better results. Specifically, we used four of the CANTINA features, extended another one of them, and augmented CANTINA with ten more expressive features, eight of which are novel ones proposed by us. Instead of using a simple linear classifier, we adopted state-of-the-art machine learning algorithms to build detection models. Moreover, we conducted much more thorough evaluation on a much larger and richer corpus.

Aside from the methods introduced above that examine the degree of unusualness of a webpage via features, other techniques have been proposed that detect phish by directly discovering the target brand being phished. Pan et al [Pan and Ding 2006] proposed a method that extracts the webpage identity from key parts of the HTML via the χ^2 test, and compiled a list of features based on the extracted identity. Their method achieved an average FP of about 12%. However, the assumption that the distribution of identity-related words usually deviates from that of other words is questionable, which is indicated by their high FP. Even in DOM objects, the most frequent term often does not coincide with the web identity. Xiang et al [Xiang and Hong 2009] proposed a hybrid detection model that recognizes phish by discovering the inconsistency between a webpage's true identity and its claimed identity via search engine and information extraction techniques. Their full system achieved a TP of 90.06% and a FP of 1.95%.

The analog of identifying the target brand of phishing attacks in the email domain is the sender ID verification in the email authentication protocol, which aims at verifying that every email originates from the domain from which it claims to have been sent. However, while the latter has been solved well by industry solutions such as the sender ID framework, the former still remains a hard research problem. If the targeted brand were known a priori, the phishing attack could have be detected trivially and techniques as proposed in the literature are unnecessary.

Ideas trying to combine the merits of blacklist-based and feature-based methods in a unified framework also exist. In a recent work, Xiang et al [Xiang et al. 2010] proposed a content-based probabilistic approach, which leverages existing human-verified blacklists and applies the shingling technique, a popular near-duplicate detection algorithm used by search engines, to detect phish. Their algorithm achieved 0% FP with a TP of 67.15% using search-oriented filtering, and 0.03% FP and

73.53% TP without filtering.

In addition to the research reviewed above, anti-phishing toolbars based on different techniques are also available, many of which exploit blacklists to assure a low FP. Well-known products include Microsoft Internet Explorer (IE) 8, Firefox 3, McAfee SiteAdvisor, Symantec Norton 360, EarthLink, account guard by eBay, etc. The technical details of these products have not been made publicly available, however, their effectiveness has been measured in [Sheng et al. 2009].

3. SYSTEM ARCHITECTURE

Figure 1 shows the overall flow of CANTINA+. The feature extractor, shared by the training and testing phases, is the core of our hybrid framework, in which the values of the 15 features are extracted. Specifically, the goal of the training phase is to obtain the feature values for each instance of the training corpus, which is then used by the machine learning engine to build classifiers. The goal of the testing phase is to label real web pages as phish or not.

In the testing phase, we first apply two filters to web pages to reduce false positives and speed up runtime performance. The first filter is a hash-based filter that compares a web page against known phish. The second filter checks a given web page for a login field. We will describe the details of these two filters in the next two sections. If the web page is not detected as a near-duplicate of the existing phish and a login form is found in the HTML, we move on to extract the 15 features from the web page using the URL, HTML DOM and other resources, and apply a pre-trained model to classify its identity. In real-world scenarios, we can use a sliding window to include the most recent phishing attacks in the training data.

4. HASH-BASED NEAR-DUPLICATE PAGE REMOVAL

The growing use of toolkits [Cova et al. 2008] to create phish produces a massive volume of phishing webpages that are very similar or even identical to each other in terms of HTML. This observation led us to adopt page duplicate detection algorithms to identify pages that are extremely likely to be phish, by comparing a given page against known phish. In our previous work [Xiang et al. 2010], we proposed an adaptive probabilistic anti-phishing algorithm based on URL blacklists exploiting the high similarity among phishing attacks. In this work, we simply design a more rigorous hash-based filter to quickly recognize identical phish while mainly rely on the machine learning engine to detect other variants.

To detect duplicate pages, we used the SHA1 hash algorithm, a popular method for checking if two pieces of digital content are the same [NIST 1995]. SHA1 is a fast and secure procedure that produces 160-bit hash values and is applicable to content of any length with a low likelihood of collisions. To use SHA1, we first remove all spaces in the HTML. We also remove all default values in HTML input fields and replace them with empty strings. Our rationale here is that we have seen some phishing sites that insert random email addresses into such fields. We then compute a SHA1 hash on the processed HTML, which is then compared against a pool of hash values of known phishing web pages. Currently, we use PhishTank's verified blacklist as our known list of phishing sites.

We acknowledge that this hashing-based filter is easy to beat. However, it is

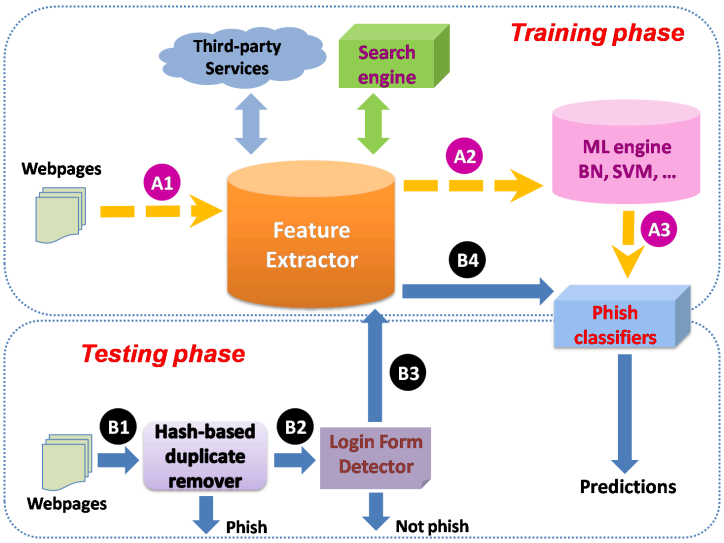

Fig. 1. System architecture. In the training stage, A1) 15 feature values are extracted from each instance in the training corpus; A2) the feature values are organized in proper format and forwarded to the machine learning engine; A3) classifiers are built for phish detection. In the testing stage, B1) the hash-based filter examines whether or not the incoming page is a near-duplicate of known phish based on comparing SHA1 hashes; B2) if no hash match is found, the login form detector is called, which directly classifies the webpage as legitimate if no login form is identified; B3) the webpage is sent to the feature extractor when a login form is detected; B4) the pre-trained learning models run on the features and predict a class label for the webpage.

highly effective against existing phish today, is fast in terms of runtime performance, and cheap to implement. Also, we only use it as a filtering step to remove near-duplicate phish, and mainly rely on machine learning approaches with our feature set for phish detection.

5. LOGIN FORM DETECTION

Almost all phishing attacks try to trick people into sharing their information through a fake login form. In this section, we present our algorithm using the HTML DOM to filter pages with no login forms prior to the classification step. On the surface, this sounds simple, yet finding a login form in practice is actually by no means trivial. Typically, a login form is characterized by three properties, i.e., 1) FORM tags, 2) INPUT tags, and 3) login keywords such as password, PIN, etc. INPUT fields are usually used to hold user input. Login keywords guarantee that we are actually facing a login form rather than other types of forms such as the common search form. We compiled 42 login keywords to allow flexibility in detecting various patterns such as “passcode”, “customer number”, etc.

Due to phishing and other unconventional design patterns, a login form does not always satisfy all three properties above, and to cope with such variations, we designed the following algorithm to declare the existence of a login form.

(1) We first handle the regular case in which form tags, input tags and login keywords all appear in the DOM. Login keywords are searched in the text nodes

as well as the alt and title attributes of element nodes of the subtree rooted at the form node. Return **true** if all three are found.

(2) We then handle the case where form and input tags are found, but login keywords exist outside the subtree rooted at the form node f. First, we examine whether the form f is a search form by searching for keyword "search" in the same scope as in step 1. If f is not a search form, we traverse the DOM tree up for K levels[1] to ancestor node n, and search login keywords under the subtree rooted at n in the same scope as in step 1. Return **true** if a match is found.

(3) We then capture the phishing pattern in which forms and inputs are detected, but phishers put login keywords in images and refrain from using text to avoid being detected. Check the subtree rooted at f for text and images, and return **true** if no text is found and only images exist.

(4) Finally, we handle the case where phishers only use input fields and leave out form tags on purpose. Search login keywords and image patterns in a similar fashion, but in the scope of the whole DOM tree r, and return proper results.

This algorithm covers most of the login form variants, with a 98.06% TP on our phishing corpus as shown in section 8.1. The features in this algorithm may flag a form as a login form when it actually is not. However, this slightly larger coverage on the one hand helps prevent falsely filtering a phishing page prior to the content analysis stage, and on the other still removes the vast majority of pages with no login forms from consideration, thus reducing false positives and significantly accelerating the detection process.

6. A FEATURE-RICH MACHINE LEARNING FRAMEWORK FOR ANTI-PHISHING

When the hash-based filter finds no matches with existing phish and the login form filter detects a login form in the HTML, our approach relies on the machine learning engine to capture phishing variants. The set of high-level features is the major contribution of this paper, and in this section we will elaborate on the design and rationale of our feature set, as well as the machine learning algorithms. In particular, features 2, 3, 4, 12 are taken from CANTINA, feature 5 and 13 are taken from [Garera et al. 2007], feature 11 is a variant of one feature in CANTINA, and features 1, 6, 7, 8, 9, 10, 14, 15 are the novel ones we proposed.

It is necessary to point out that false positives or false negatives might be caused by each feature. However, the combination of all the features will make up for the inadequacy of individual features, and will yield better performance.

6.1 High-level Webpage Features

We have organized our features into three categories. The first (features 1 through 6) deals with the URL of the web page. The second (features 7 through 10) inspects the HTML content of the web page. The third (features 11 through 15) involves searching the web for information about that web page. Specifically, feature 1, 6, 7, 8, 9, 10, 14, 15 are novel ones proposed by us in CANTINA+.

URL-based Features

[1]We took $K = 2$ in our current implementation.

(1) ***Embedded domain***. This feature examines the presence of dot separated domain/hostname patterns such as "www.ebay.com" in the path part of the webpage URL. URLs corresponding to legal websites usually show these patterns in the hostname segment of the URL, while phishers sometimes add their target's domain/hostname in the path segment to trick users into trusting their phishing sites. To capture more of such phishing URLs, we avoid hard-coded domains and instead search in the path segment of the URL with a pre-compiled regular expression that seeks dot-separated string segments[2].

(2) ***IP address***. This feature checks if a page's domain name is an IP address. Attackers often employ IP address in the URL to disguise a webpage's malicious nature, while legitimate websites almost always use domain names instead of IP addresses due to their easy memorability.

(3) ***Number of dots in URL***. This feature counts the number of dots in the URL. Phishing pages tend to use more dots in their URLs than the legitimate sites.

(4) ***Suspicious URL***. This feature checks if a page's URL contains an "at" (@) or the domain name has a dash (-). An @ symbol in a URL causes the string to the left to be disregarded, with the string on the right treated as the actual URL for retrieving the page. Combined with the limited size of the browser address bar, this makes it possible to write URLs that appear legitimate within the address bar, but actually cause the browser to retrieve a different page. Dashes are also not often used by legitimate sites.

(5) ***Number of sensitive words in URL***. In [Garera et al. 2007], Garera et al summarized a set of eight sensitive words[3] that frequently appear in phishing URLs, and we create this feature counting the number of the eight sensitive words that are found in a page URL. This is a numeric feature with a range of 0 to 8.

(6) ***Out-of-position top level domain (TLD)***. This feature checks if a TLD appears in an unusual position in the URL. An example is `http://cgi.ebay.com.ebaymotors.732issapidll.private99dll.qqmotorsqq.ebmdata.com`, in which we see the TLD "com" in a position usually not for TLDs in the hostname.

HTML-based Features

(7) ***Bad forms***. Phishing attacks are usually accomplished through HTML forms. This feature checks if a page contains potentially harmful HTML forms. To satisfy our definition of harmful, a webpage is required to have all of the following: 1) an HTML form, 2) an <input> tag in the form, 3) keywords related to sensitive information like "password" and "credit card number" or no text at all but images only within the scope of the HTML form, 4) a non-https scheme in the URL in the action field or in the webpage URL when the action field is empty.
Login form recognition here is realized by the SAX parser, a sequential access parser API to read data from XML documents, rather than the complicated DOM-based

[2]Three constraints must be met for a dot-separated string to be eligible for an embedded domain. First, at least three segments must exist. Second, each segment must have two or more characters. Third, each segment is composed of letters, numbers and underscores only.
[3]The sensitive words include "secure", "account", "webscr", "login", "ebayisapi", "signin", "banking", "confirm".

process as in the login form filter in section 5. Specifically, we defined 39 login-related keywords to narrow down our focus to forms that truly request user private information. In some hard cases, attackers remove all the textual content in the HTML form and only use images of text to avoid text-based detection. We do regard such forms as harmful, though we currently do not have a reliable way of detecting them. We also note that attackers may craft login form patterns that render this feature futile. However, the combination of our 15 features may still capture other phishy characteristics of a web site.

(8) ***Bad action fields***. From an engineering point of view, placing the authentication scripts of the whole website in one location facilitates the development and maintenance of the code, and legitimate websites tend to adopt this practice. Accordingly, the authentication methods in the script on legitimate websites are usually called via absolute URLs in the action field of the HTML form. However, phishing sites are usually ephemeral and the design principle is often to make everything as simple as possible, as a result of which the authentication code is usually placed in the current directory and the action field of the HTML form is typically a simple file name. As such, this feature is set to 1 if the action field is empty or a simple file name, or points to a domain different from the webpage domain. The last branch is to identify cross-domain scripting, which is sometimes deemed harmful for web applications. The HTML forms in which the action fields are scrutinized in this feature denote those login forms, i.e., those that meet the first three requirements in the bad forms feature above. Though we have found exceptions from both phishing and good pages that break the above formulation, this feature is still demonstrated to be among the top performing ones in our feature set.

(9) ***Non-matching URLs***. This feature examines all the links in the HTML, and checks if the most frequent domain coincides with the page domain. The rationale behind this feature is that links on phishing sites are usually meaningless and thus noisy values such as "#", "index.html", URLs of the target legitimate sites, etc., are often seen especially when the attacks are automatically created by toolkits, leading to inconsistency between the page domain and the most frequent domain in the links. Sometimes, the links on a phishing page point back to various parts of the phishing site, however, phishers do not very often use a different absolute URL for each such link but rather stick to similar URLs. To catch that pattern, we count the percentage of highly-similar links[4] in the HTML, and set the value of this feature to 1 if any single pattern occurs more often than a threshold. In addition, we also count the percentage of empty or ill-formed links in the HTML, and apply thresholding to set corresponding feature values.

(10) ***Out-of-position brand name***. The vast majority of companies put their brand name into their domain name. Typically, the brand name appears in the domain string as the second-level or third-level domain. Phishing sites, however, are always hosted on compromised or newly registered domains. To make these sites look trustworthy, attackers sometimes include brand names or domain names of the victim sites in their phishing URLs, causing an out-of-position brand name.

[4]Highly-similar links in this paper are defined to be those that are either identical or differ only in the fragment component of the URL.

The example in the 6th feature above still applies here, in which ebay, the target brand, appears in an unusual position in the hostname.
However, since we have no a priori knowledge about the brand name of a web page, we follow the analysis in the 9th feature above and use the most frequent domain keyword in the HTML links as the website brand name. With this estimated brand name, we remove the page domain keyword as well as the string to its right from the URL, and search in the remaining portion for the brand name. If a match is found, the page under investigation is suspicious and the feature value is set to 1.

Web-based Features

(11) ***Age of domain***. This feature checks the age of the webpage domain name. Many phishing sites are hosted on recently registered domains, and as such have a relatively young age. To exploit that property, this feature measures the number of months since the domain name was first registered. Specifically, we perform WHOIS lookups to retrieve the domain registration date, and if the domain registration entry is not found on the WHOIS server, this feature will simply return -1, deeming it suspicious. Note that this feature does not account for phishing sites launched on compromised legal domains, nor does it account for phishing sites hosted on otherwise legitimate domains.

(12) ***Page in top search results***. This feature was originally used in CANTINA [Zhang et al. 2007]. Specifically, we extract the top K words from the page content ranked by the term frequency and inverse document frequency (TF-IDF) metric, and search those top terms plus the webpage domain keyword[5] in Google. The webpage is deemed legitimate if the page domain matches the domain name of any of the top N search results; otherwise, it is regarded as being phishy. The intuition behind this feature is that search engines are more likely to index legitimate websites, while phishing sites have much less chance of being crawled. According to the experimental findings in [Zhang et al. 2007][Xiang and Hong 2009], we took $K = 5, N = 30$ in this work. We used Google as the collection corpus in this work, and estimated the document frequency of a term w by the number of result entries obtained by searching w in Google.

(13) ***PageRank***. PageRank is a link analysis algorithm first used by Google, in which each document on the web is assigned a numerical weight from 0 to 10, with 0 indicating least popular and 10 meaning most popular. In our method, we add another value -1 when the PageRank value for a particular webpage is not available. An intuitive rationale behind this feature is that phishing webpages usually have very low PageRank scores due to their ephemeral nature and few incoming links pointing to them, while legitimate cases tend to have higher PageRank values. This feature was also used in [Garera et al. 2007].

(14) ***Page in top results when searching copyright company name and domain***. This and the following feature complement the 12th feature by directly seeking the webpage on the web without analyzing the terms in the page content. Generally, the TF-IDF feature above may not work well for two cases. First, some

[5]The domain keyword is the segment in the domain representing the brand name, which is usually the non-country code second-level domain such as "Paypal" for "paypal.com" or the third-level domain such as "ebay" in http://www.ebay.com.au/.

terms with high TF-IDF scores may not be relevant in searching the intended webpage, and as a result, the query may not return the expected webpage domain in top N entries. Second, due to company affiliations, two closely related domains are sometimes literally different such as "blogger.com" and "blogspot.com", which renders straightforward string matching inadequate.

This feature uses as query phrase the page domain plus the copyright company name that is usually found on the bottom of a webpage showing a website's brand name, and treats a webpage as suspicious if its domain is absent from the top N search results and legitimate otherwise. This brand recognition idea was taken from our previous work [Xiang and Hong 2009]. In this paper, we only employed the copyright field to extract brand names instead of searching the page title and using the whole page content via named entity recognition.

There are many advantages to this feature. First, search engines are more likely to have entries in their index for legitimate sites, and searching the page domain and the website brand name directly has a higher chance of returning the intended page in top positions, thus remedying the first problem of the 12th feature. Second, this feature alleviates the second weakness of the 12th feature as discussed above in that related domains tend to be all returned when searching the copyright brand name without other irrelevant query terms thanks to the broad coverage of modern search engines. Third, copyright fields may not show up in every page, and once they are missing, we simply query the page domain in search engines. Again, the argument in the first benefit of this feature explained above applies here, and we eschew false positives even if we misclassify a phish under this scenario.

The copyright field on a webpage typically shows some patterns like "Copyright © 1995-2008 eBay Inc. All Rights Reserved.", and we defined 11 regular expressions to extract brand names. Some pages have more than one copyright field, and we prefer the one with word overlap with the page domain, with keywords like "Inc.", "Ltd.", or simply the last copyright field. Since this and the following feature seek the intended webpage directly via search engines, we impose a relatively stricter constraint on N and examine the top $N = 10$ search results for the webpage domain.

(15) ***Page in top results when searching copyright company name and hostname***. This feature is identical to the 14th feature except that we use the hostname instead of the domain name in the query, which is useful especially when the domain name is too short to introduce noisy results in top result entries.

6.2 Machine Learning Algorithms

We compare six state-of-the-art learning algorithms in training the phish detector, including Support Vector Machines (SVM) [Burges 1998], Logistic Regression (LR), Bayesian Network (BN) (a probabilistic graphical model that makes inferences via a directed acyclic graph), J48 Decision Tree, Random Forest (RF) and Adaboost, with the primary goal of evaluating the effectiveness of our feature set. All the ML algorithm implementations were taken from the Weka package [Witten and Frank 2005]. We found through extensive experiment that BN performed consistently within the top performing group of algorithms. None of the other algorithms significantly outperformed it, and it was frequently ranked No.1. Therefore, we only report the performance of BN in this paper.

7. EXPERIMENTAL SETUP

7.1 Evaluation Metrics

In our experiment, we adopted *true positive rate* and *false positive rate* as the main evaluation metrics. We also used the F1 measure, which integrates both TP and FP with equal weights into one summary statistic. In tuning the machine learning models, we adopted the concept of Receiver Operating Characteristics (ROC) curves [Fawcett 2006] and employed the area under the ROC curve (AUC) [Cortes and Mohri 2003] metric, which, as a standard approach to evaluate binary classification performance, portrays the trade-off between TP and FP. Statistically, the AUC equals the probability that given a randomly generated positive instance and negative instance, a classifier will rank the positive one higher than the negative one, and thus is a good summary statistic for model comparison.

7.2 Webpage Corpus

Our webpage collection consists of phishing cases from PhishTank, and legitimate webpages from five sources. To eliminate the impact of language heterogeneity on our content-based method, we only included English web pages in our corpus. Our legitimate collection mainly focuses on popular sites, commonly spammed sites, common phishing target sites, etc. Although our corpus is not representative of what users would experience in their every day browsing, by evaluating CANTINA+ on these hard cases, we actually provide the worst case performance statistics, which is more beneficial for an objective evaluation of our method and its real-life application that follows.

Phishing sites are usually ephemeral, and most pages will not last more than a few days typically because they are taken down by attackers to avoid tracking. To fully study our approach over a larger corpus, we downloaded the phishing pages when they were still live and conducted our experiment in an offline mode. Our downloader employed Internet Explorer to render the webpages and execute Javascript, so that the DOM of the downloaded copy truly corresponds to the page content and thus gets around phishing obfuscations. We also downloaded images to allow us to use CANTINA [Zhang et al. 2007] for comparison.

For phishing instances, we used the phish feed of Phishtank [PhishTank b], a large community-based anti-phishing service with $35,849$ active accounts and $489,397$ verified phish [PhishTank a] by November 20, 2009. The phish corpus in our experiment was collected over two periods, with the researchers of this paper manually removing nonexisting URLs. Phish set 1 was collected starting in early May of 2008, and 6943 phishing webpages were downloaded during a five-month period. Phish set 2 was initiated in late February of 2009 and a total of 1175 phish were garnered from February 27, 2009 to April 2, 2009. The purpose of two-separate web crawls is to roughly examine whether or not phishers tend to reuse phishing sites built a while ago.

To thoroughly test the FP, we collected the same five sets of legitimate web pages in two separate crawlings, the details of which are given in Table I, with legitimate corpus 1 for randomized evaluation and legitimate corpus 2 for time-based evaluation. The missing pages in legitimate corpus 2 compared with the legitimate corpus 1 were due to broken links. Fetterly et al discovered through

large-scale web crawling that webpage content was fairly stable over time [Fetterly et al. 2003], and based on that finding, we did not download legitimate corpus 2 at each time point but rather downloaded only once the whole set at a time later than all the phishing timestamps in phish set 2.

Table I. Legitimate collections from 5 sources. The size column marked by "1" gives the corpus sizes for randomized evaluation, while that marked by "2" gives the corpus sizes for time-based evaluation. We use legitimate corpus 1 and legitimate corpus 2 to refer to the two collections. Legitimate corpus 2 was downloaded on April 2, 2009, and was a subset of legitimate corpus 1.

Source	Size (1)	Size (2)	Crawling Method
Top 100 English sites from Alexa.com	1023	958	Crawling homepages to a limited depth
3Sharp [3sharp report 2006]	101	87	Downloading webpages that still existed at the time of downloading
Generic banks on Yahoo directory	985	878	Crawling the homepages for a varying number of steps within the same domains
Other categories of Yahoo directory	371	330	Same as the generic bank category
The most common phishing targets	81	69	Saving login pages of those sites

7.3 Evaluation Methodology

For anti-phishing algorithms, two typical evaluation methodologies exist, i.e., *randomized evaluation* and *time-based evaluation*. The former is mainly to inspect the overall performance on all the available data, while the latter is to examine the performance under more real-world scenarios, training models on the past data and applying the models to future cases. To fully evaluate our approach, we adopted both methodologies in our experiment.

In light of a significant percent of near-duplicate phish with high similarity in terms of content due to the use of toolkits, it is necessary to see how our approach performs on the testing data with unique phish and with near-duplicates of the training phish respectively. Ideally, the TP on the testing set with unique phish should be reasonably high, and the TP on the testing set with near-duplicate phish of the training set should be even higher, if not 100%. Accordingly, we conducted two series of experiments under each evaluation methodology, using unique testing phish and near-duplicate testing phish respectively. This unique and near-duplicate dichotomy is important also because learning models with repetitive patterns in the training data tends to decrease the effectiveness of our machine learning approach.

In the randomized evaluation, we utilized both phish set 1 and phish set 2, and legitimate corpus 1 in this evaluation. We adopted the standard train, validation and test methodology, which is a common practice in machine learning. All the train/test splits were performed randomly. We reserved 70% percent of the legitimate set as the testing set, and used the remaining 30% for model training and tuning. To see the impact of the percentage of phish p in the training data on the detection performance, we built a series of randomly selected training sets varying p from 10% to 70%. In optimizing the algorithm parameters, those training sets with different p were further divided via stratified sampling into a training portion and

Table II. Statistics of near-duplicate phish detection. In detecting near-duplicate phish in set 1, we only examined the phish set 1 for hash matches, while for phish set 2, we scanned through both phishing sets. There is only one common duplicate phish between two sets, suggesting that phishers do not replicate phishing sites created long ago for future attacks.

	Phish set 1	Phish set 2
Download time	May 2008 – Sep 2008	Feb 27, 2009 – Apr 2, 2009
Total size	6943	1175
#unique webpages	1595	624

a validation portion. Stratification ensures that the class distribution is preserved between the training and validation parts. In performing the final tests with the optimal model parameters, the whole training sets were used to train the classifiers. To reduce random variation and avoid lucky train/test splits, we used the average statistics over 10 runs in all our experiments.

In the time-based strategy specifically, we utilized the timestamps of the phishing attacks and simulated real-life scenarios by training our models on the past data and testing them on future data via a sliding window mechanism. We used legitimate corpus 2 and phish set 2 in this experiment. Specifically, we moved a sliding window of length L (in terms of days) step by step along the time line and applied our detection algorithm to the webpages with timestamp T_i (day in our current evaluation) using models learned on a training set composed of the phishing data with time labels falling in window $[T_{i-L}, T_{i-1}]$ and a subset of randomly selected legitimate webpages from legitimate corpus 2 in Table I. Varying the percentage of phish p in the training set from 20% to 70% controls how many legitimate cases to be chosen for each sliding window. After a subset of legitimate pages are randomly selected into the training set for each sliding window, the rest of the legitimate corpus together with the phishing instances on time point T_i make their way into the testing set for the evaluation of TP and FP on time point T_i. The reported TP and FP are the mean of the TP_i and FP_i at all time points.

8. EXPERIMENTAL RESULTS

8.1 Hash-based Near-duplicate Phish Detection

Our goal is to see the extent to which phishing toolkits are employed to produce phishing site replicas, and thus we compute the hash values for all phish in our corpus using the algorithm given in section 4 and explicitly keep only one copy among the webpages with identical SHA1. Table II shows that 72.67% phish are replicas according to our hash-based filtering algorithm, suggesting the effectiveness of this hashing-based filter in capturing near-duplicate phish and their simple variants.

8.2 Login Form Detection

As shown in Table III, we successfully detected 98.06% phishing pages with login forms, and filtered a significant percentage of good pages from other categories. For the remaining 1.94% phishing pages, they either do not have a login form (very rare in our phish corpus), use login keywords not in our list such as "serial key", or organize the form/input tags in a way our method misses.

This step contributes to the reduction of FP in that a certain portion of legitimate pages as shown in Table III are removed from being processed by the feature

Table III. Statistics of login form detection. 98.06% phishing pages with login forms were successfully detected.

Corpus	Phishtank	Alexa	3Sharp	Banks	Yahoo	Prominent
#total pages	2219	1023	101	985	371	81
#detected with login forms	2176	263	31	229	77	73
% good pages filtered		74.29	69.31	76.75	79.25	9.88

extraction and classification modules. Note that we actually did not particularly train the models on pages without forms in the whole layered system. Some legitimate pages for training purposes happen to contain no login forms. If the training set consists solely of pages with login forms, the TP and FP will both be higher.

8.3 Randomized Evaluation

The main goal of randomized evaluation is to inspect thoroughly the overall performance of CANTINA+ on all our data via stratification and multiple run averaging, which is a standard practice in machine learning. In this section, we show the performance of our layered method under the smallest percentage of training phish, i.e., 10%, mainly due to two reasons. First, this setting is more realistic because in the real-world scenario, the volume of legitimate cases typically far outnumbers phishing attacks. Second, our approach did not manifest drastic difference under various percentages of training phish. For the same reason, we only give the experimental result under 20% training phish in the time-based evaluation in section 8.4. This number is different from the 10% in the randomized evaluation in that we only used legitimate corpus 2 in the time-based evaluation and for some sliding window, the volume of the training phish is large enough such that we do not have 90% legitimate pages for training.

8.3.1 *Machine Learning Model Tuning.* Machine learning algorithms use different strategies to regulate the learning process. Normally, a variety of factors are considered by an algorithm. We chose to tune only the most important factors for each algorithm due to combinatorial effects. We tuned our models on the validation set and used the optimal parameters for our final classifiers.

For each algorithm, we found the optimal parameter with the best AUC [6] via 10-run tuning. Specifically, the optimal settings always improved the algorithms over the default parameters, mostly with less than 2% improvement in AUC. Typically, when the amount of training phish is insufficient, the AUC on the validation set is undesirable due to the low TP, and as the proportion of phish in the training set increases, AUC gradually amplifies and then possibly declines at the point where the degradation on FP outweighs the improvement on TP. In terms of the model complexity parameter, we see that mostly the classifiers achieved optimality when the amount of regularization is just appropriate. The major reason is that we are tuning the classifiers on a separate validation set, and overly small penalization leads to overfitting, while the other extreme yields undertrained models.

In the testing phase of randomized evaluation, we assigned the models with the optimal parameters, and tested them on a separate testing set. We felt that this was a reasonable and feasible approach, since in a real deployment, one could tune

[6] Area under the ROC curve, introduced in section 7.1

Table IV. Performance (10-run average) of CANTINA+ using Bayesian Network and CANTINA under randomized evaluation. For all cases in the table, CANTINA+ was trained with 10% phish in the training set. The legitimate testing sets are the same for the evaluations on both unique testing phish and near-duplicate testing phish, and therefore, the FPs remain the same. Overall, CANTINA+ significantly outperforms CANTINA. CANTINA has no explicit training phase, and is not influenced by the percentage of phish in the training set.

		Type of phish in the testing set					
		Unique			Near-duplicate		
Algorithm	Login filtering	TP (%)	FP (%)	F1	TP (%)	FP (%)	F1
CANTINA+ (with BN)	Yes	92.54	0.407	0.9592	99.63	0.407	0.9961
	No	93.47	0.608	0.9632	99.64	0.608	0.9952
CANTINA	Yes	71.47	0.335	0.8320	93.17	0.335	0.9630
	No	72.15	0.714	0.8348	93.19	0.714	0.9612

the models offline and then employ the optimal setup for online scenarios.

8.3.2 *Testing on the Holdout Data with Unique Phish.* Multiple run averaging is a standard practice in machine learning, and the goal of this experiment is to examine the performance of our feature set trained on randomly selected phish and tested on the remaining unique phish. Specifically, we used all the good URLs in legitimate corpus 1 and all the unique phishing webpages in our collection, i.e., 1595 from phish set 1 and 624 from phish set 2. In Table IV, we show the performance of CANTINA+ in this experiment with 10% phish in the training data.

As shown in Table IV, CANTINA+ achieved a high TP of 92.54% and 93.47% with a low FP of 0.407% and 0.608% with and without login form filtering respectively. The filtering step makes the TP significantly worse. For all cases including CANTINA, FP filtering via login form detection significantly improves FP because a certain number of legitimate pages (Table III) are detected with no login forms.

8.3.3 *Testing on the Holdout Data with Near-duplicate Phish.* The goal of this experiment is to show that learning with our feature set performs very well, if not perfectly, on the near-duplicate of the phish in the training set under the randomized evaluation setting. This is critical in demonstrating the power of our machine learning approach since we might as well directly use the simpler hash-based filtering if our proposed approach performs poorly on near-duplicate testing phish. In this experiment, we used a subset of good URLs in legitimate corpus 1 and unique phishing URLs from phish set 1 and phish set 2 for training, and tested our models on the near-duplicate phish from phish set 1 and phish set 2, i.e., a total of 5899 near-duplicate phish. In Table IV, we show the TP and FP of CANTINA+ in this experiment with 10% phish in the training data.

With only 10% training phish, CANTINA+ was able to achieve a very high TP of 99.63% and 99.64% with and without FP filtering respectively due to the strong similarity between the phish in the training and testing data, manifesting the power of our feature set in capturing the characteristics of phishing attacks. The experiment result also shows no significant effect of the FP filter. Since our legitimate corpus contains no replicated instances, choosing unique or near-duplicate testing phish only influences TP and the FP remains the same, as shown in the table.

Table V. Performance (10-run average) of CANTINA+ using Bayesian Network under time-based evaluation with a two-week sliding window. For all cases in the table, CANTINA+ was trained with 20% phish in the training set. The legitimate testing sets are the same for the evaluations on both unique testing phish and near-duplicate testing phish, and therefore, the FPs remain the same. CANTINA+ achieves a high F1 of over 0.93 under all cases.

		Type of phish in the testing set					
		Unique			Near-duplicate		
Algorithm	Login filtering	TP (%)	FP (%)	F1	TP (%)	FP (%)	F1
CANTINA+ (with BN)	Yes	88.56	1.320	0.9328	95.28	1.320	0.9693
	No	90.47	1.870	0.9407	95.65	1.870	0.9685

8.4 Time-based Evaluation

The randomized experiment in the previous section aims at evaluating CANTINA+ using the standard machine learning practice in which the training and testing data are randomly selected with multiple runs to reduce variance. The main goal of time-based evaluation, however, is to inspect the performance of CANTINA+ in real-world scenarios, in which we train our models using historical data and apply the models to future data. Since the overhead of stepwise parameter tuning for each sliding window is prohibitive, we did not explicitly tune the model parameters in this experiment and simply used the default values. In addition, because the original CANTINA has no training process, its performance should remain identical and we therefore did not compare our approach with CANTINA again in this evaluation. The experimental result of this time-based evaluation using a two-week sliding window with 20% training phish in each sliding window is shown in Table V.

8.4.1 *Result on Unique Testing Phish.* The goal of this experiment is to evaluate our approach on real-world phish stream with models trained on historic phishing attacks that have no overlap with the phish in the testing set. In this experiment, the positive corpus comes from the 624 unique phishing URLs from phish set 2, and the negative data set uses all the pages in legitimate corpus 2.

Table V shows that CANTINA+ achieved a high TP of 88.56% and 90.47% with and without FP filtering respectively with only two weeks' worth of phish in the training set for each sliding window. Login form filtering makes the TP significantly worse yet benefits the FP significantly.

We see that the TPs on the unique testing phish in the randomized evaluation in Table IV are better, mainly because for two-week long sliding windows, the number of training phish in each sliding window is usually smaller than its counterpart under the randomized evaluation, leading to slightly inferior TPs as a result. Another subtlety that causes this discrepancy is that the TP on the phishing data of the first day in our phish corpus is zero. In addition, the number of training phish with different sliding windows varies significantly, from a minimum of 19 to a maximum of 398, causing considerable variations in the resultant TPs across days, which is confirmed by the result that the maximum TP did not always occur under the setting with 70% training phish for each algorithm.

Table V also shows that with 20% phish in the training set in each sliding window, the FP of CANTINA+ with no login form filtering is 1.870%, which drops to 1.320% with login form filtering, both worse than the counterpart in the randomized experiment. The gap in FPs under the two evaluation methodologies can

be attributed to the following observations. First, we conducted 10 experiments and averaged the resultant statistics for each setting in the randomized evaluation, which helped reduce random variations in the performance. On the other hand, we could not perform 10-run averaging in the time-based evaluation due to the nature of this experimental strategy, since the training phish in the sliding window prior to the current time point are fixed and could not be randomized. This one-time random selection of legitimate pages might cause unlucky train/test split, leading to variable FPs. Second, learning models are optimized in the randomized experiment, while default parameter values are used in the time-based analysis.

A breakdown on the performance by days shows that the FP of our approach on the second day in our corpus is significantly worse than the other days, and after that the FP stays relatively stable. This is caused by the fact that we only have one day's worth of training phish in evaluating our model on the web pages of the second day, leading to a very small training set and therefore the undesirable FP.

8.4.2 *Result on Near-duplicate Testing Phish.* The goal of the experiment in this section is to demonstrate the effectiveness of our feature set on real-world phish stream with near-duplicate attacks of the training phish. Specifically, we utilized part of the 624 unique phishing URLs from phish set 2 and part of legitimate corpus 2 for training, and evaluated the models on the 550 near-duplicate phish from phish set 2 and the remaining URLs of legitimate corpus 2.

Overall, the pattern in the result of this experiment in Table V is similar to that of the experiment on unique testing phish in section 8.3.2. Particularly, the TP of CANTINA+ is around 95%, higher than the statistics in the experiment on unique testing phish, which is what we expected since the testing phish here highly resemble the training phish. However, the 95% TP here is lower than the over 99% TP (Table IV) in the randomized evaluation, and we offer two explanations regarding this gap. First, we have more training phish in the randomized experiment. Second, the testing phish set in the randomized evaluation is much larger (5348 vs 551), subsuming more near-duplicate instances.

8.5 Result under Various Percents of Training Phish

In deploying our system for real-world applications, we need to build a training set and train our approach in advance, for which the percentage of phish in the training data is a key parameter. To examine the impact of the ratio of the two types of web pages in the training data on the performance of our approach, we varied its value and evaluated the performance of CANTINA+ under both the randomized and time-based evaluation. The experimental result is given in Fig. 2 and Fig. 3.

In Fig. 2, a significant positive correlation is seen between TP and the percent of phish in the training data in all cases but the time-based evaluation on near-duplicate phish. This trend is self-evident in that machine learning models are able to detect more phish with more phishing patterns in the training set. Fig. 3 illustrates a similar story in terms of FP, with a significant negative correlation between FP and the percent of phish in the training data. Although the FP of CANTINA+ deteriorated as more training phish were added, the ratio between opposite classes in the training data, among other parameters, is of our choosing, and we can always optimize our approach to guarantee the best generalized performance.

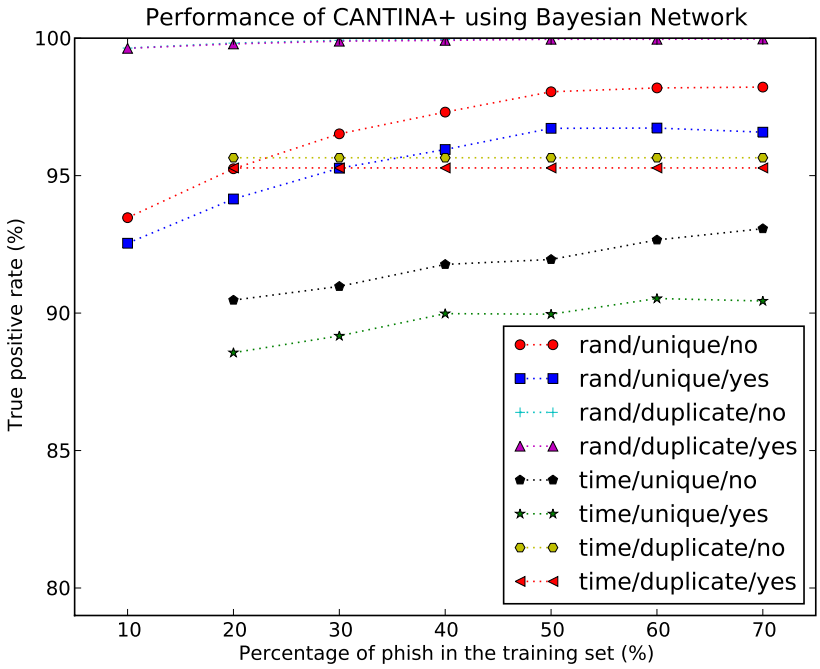

Fig. 2. TPs of CANTINA+ using Bayesian Network. Eight curves are shown, corresponding to all the combinations of evaluation methodology, type of testing phish and the use of login form filtering. A significant positive correlation is seen between TP and the percent of training phish. With other settings being identical, CANTINA+ always performs better on near-duplicate testing phish than on unique testing phish, and login form filtering makes the TP significantly worse. The two curves corresponding to the randomized evaluation on near-duplicate testing phish with and without login form filtering almost coincide.

8.6 Comparing CANTINA+ vs CANTINA

Zhang et al proposed CANTINA, a content-based method, which performed competitively in their experiment against two state-of-the-art toolbars, SpoofGuard and Netcraft [Zhang et al. 2007]. We implemented an offline version of CANTINA, and evaluated our hierarchical CANTINA+ with CANTINA on the same testing sets in the randomized evaluation. Table IV shows that our CANTINA+ always outperformed CANTINA by a huge margin in terms of TP, with a comparable FP.

In the experiment on unique testing phish, CANTINA+ outperformed CANTINA with a huge margin of over 0.12 in terms of F1, a statistically significant result indicating the superiority of CANTINA+. In particular, CANTINA+ gained an over 20% improvement over CANTINA on TP. Notably, our experiment shows a TP of about 71% for CANTINA in Table IV, drastically different from the TP of 89% in our original CANTINA paper [Zhang et al. 2007]. Three factors mainly caused this discrepancy. First, one feature in CANTINA assumes phishing mostly focuses on nine target sites and examines the inconsistency between the nine logos and the page domain. As phishing attacks evolve, however, the distribution of the most phished

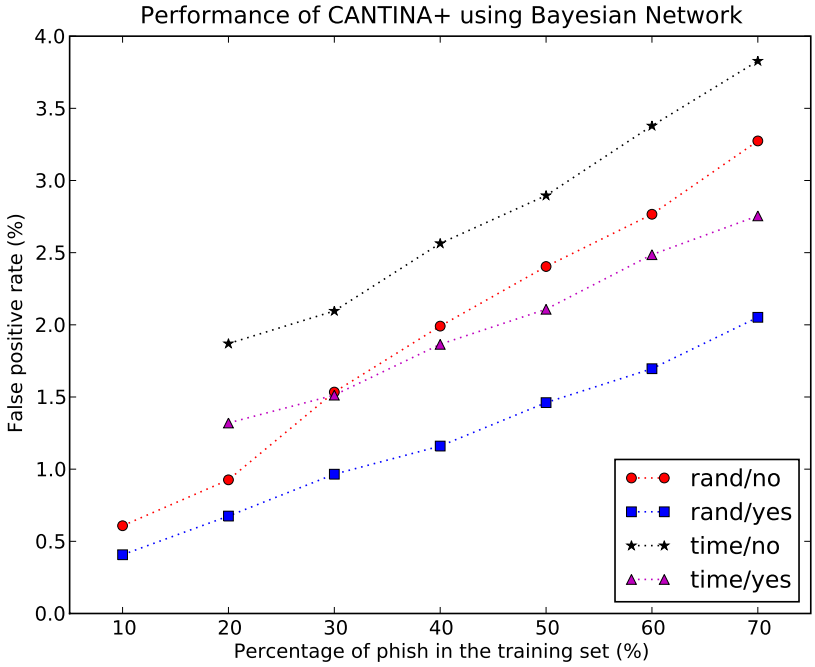

Fig. 3. FPs of CANTINA+ using Bayesian Network. The four curves correspond to the performance of our approach on unique testing phish. FP rises as the percentage of phish in the training data increases. Login forming filtering makes the FP significantly better.

brands changes, and this feature in CANTINA often fails. Second, CANTINA learns its feature weights on a rather limited 200 URLs, leading to significantly undertrained model. For instance, the "IP address" feature in CANTINA almost never fires alarms, but has a non-trivial weight. Third, we evaluated CANTINA in this paper on a much larger collection with harder testing cases, compared with the 200 testing URLs from three categories in [Zhang et al. 2007].

Particularly, CANTINA has no explicit training stage, which explains the phenomenon that the TPs of CANTINA only show slight fluctuations under an increasing value of the percentage of training phish. Furthermore, the testing legitimate sets remain the same under various ratios of training phish, and thus the FPs of CANTINA are thus constant.

In another experiment comparing CANTINA+ with CANTINA, we evaluated both methods on the 5899 near-duplicate phish from phish set 1 and phish set 2, and report the result in Table IV. The statistics indicate that CANTINA+ still outperformed CANTINA on this data set with a F1 of around 0.99 versus 0.96. Particularly, CANTINA+ beat CANTINA by far in terms of TP, with a roughly 6% margin. We observe that the TP of CANTINA is about 93% in this experiment, significantly better than its 71% TP on the unique testing phish. The cause of this is that we utilized a much larger testing set with substantial near-duplicate of the

training phish in this experiment.

8.7 Learning with Individual Features

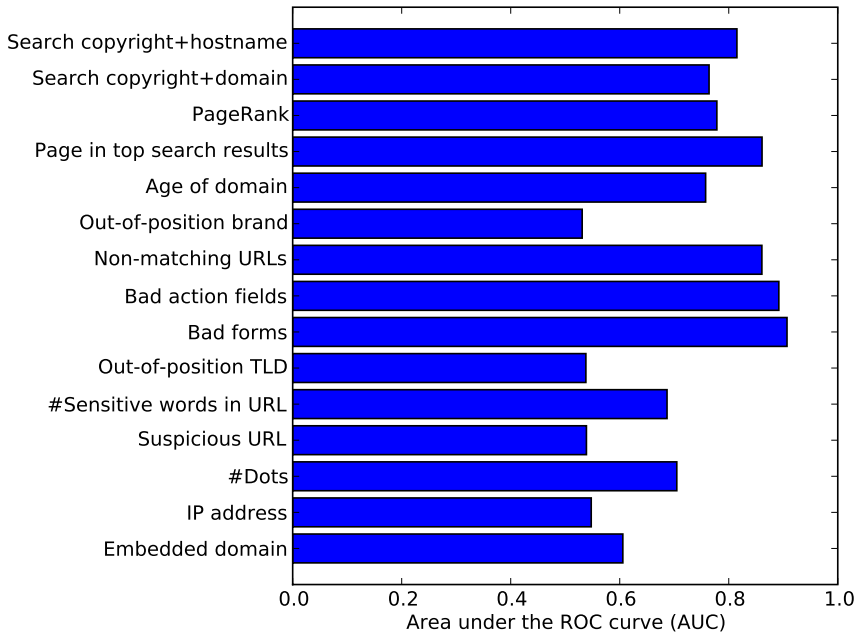

Fig. 4. Area under the ROC curve (AUC) of BN with each single feature (1-dimensional input space) under 10% training phish. Top-performing features include bad forms, bad action fields, non-matching URLs, and page in top search results, with over 0.85 AUC.

The statistics in the previous sections were obtained by using the whole feature set (15 features in total), and in this section, we evaluate the contribution of each single feature to the overall performance. We refrain from using TP/FP and instead stick to the summary statistic AUC in measuring the performance of each individual feature, because the separability of opposite classes in the 1-dimensional input space is prone to the impact of the quality of training data, and TP/FP may exhibit high variance since they only capture a single aspect of the big picture.

Across the learning algorithms, BN, Adaboost, RF and LR perform comparably, with all significantly better than J48, which in turn significantly outperforms SVM. We find through our experiment that the correlation between the AUC of each feature and the percent of phish in the training data is almost zero, and therefore, we only report the result with BN under 10% training phish in Fig. 4.

Fig. 4 shows that a few features clearly stand out from the others with over 0.85 AUC, including "bad forms", "bad action fields", "non-matching URLs", and "page in top search results". Besides, "age of domain", "search copyright company name

plus domain", "search copyright company name plus hostname" and "PageRank" also perform fairly with over 0.75 AUC, though less stellar than the above four superstars. The remaining features are apparently inferior, with under 0.7 AUC or even close to 0.5 AUC, almost amounting to random guessing.

9. DISCUSSION

9.1 Analysis on the Performance of Individual Features

In this section, we give a thorough analysis of possible errors our features might make. We also offer some insight on how phishers could defeat our feature set, as well as brief discussion about potential improvements to defeat attackers' attempt.

(1) ***Embedded domain***. This feature leads to below 0.61 AUC for all learning algorithms, suggesting an overall poor performance when used alone. Typically, legitimate domains/hostnames rarely show this pattern, but there are exceptions such as URL `http://groups.google.co.za/group/misc.invest.options/about?hl=st`. A remedy for such cases is to impose stricter constraints in the regular expressions demanding the dot-separated patterns to correspond to genuine domains/hostnames. In response, phishers could remove such embedded patterns, however, the principle that phishing URLs have to look somewhat legitimate often requires the domain of the victim site to appear in the phishing URL, which will be caught by this embedded domain feature.

(2) ***IP address***. This feature alone yields an undesirable AUC of less than 0.56 in our evaluation, indicating a poor performance slightly better than random guessing. Legitimate entities rarely, if not never, publish their websites via IP addresses, while a lot of phishing sites employ IP address to disguise their suspicious URLs. However, we did find a legitimate page using IP address in our corpus, which significantly deteriorated the performance of this feature. Since the FP of this feature is very low, an alternative usage is to employ it as a filter rather than a feature in the machine learning framework.

(3) ***#dots in URL***. Though not very good (AUC below 0.74), this feature outperforms the first two by a considerable margin. It is likely that legitimate URLs contain slightly more dots in some cases, however, phishing URLs typically cannot have this number reduced significantly in that attackers usually have to attach the target domain/hostname in the phishing URL as a masquerade.

(4) ***Suspicious URL***. This feature performs poorly, with an AUC of less than 0.55. The "at" (@) sign is never encountered in the webpage URLs in our corpus, partially because criminals have moved away from using @ since modern web browsers check for this feature. Dash "-" are seen often in both our phishing and legitimate repositories.

(5) ***#sensitive words in URL***. With under 0.73 AUC, this feature is based on the empirical finding of [Garera et al. 2007]. Attackers could simply eliminate those words or use variants of them in the URLs, thus defeating this feature. However, since phishing attacks usually target the login pages of famous websites, phishing URLs tend to manifest a certain word pattern, and though a specific set of words may lose effectiveness over time, the general idea still holds and discovering new words will enhance this feature accordingly.

(6) ***Out-of-position TLD***. Due to the prevalent use of phishing toolkits, the out-of-position TLD pattern is not rare. Moreover, TLDs are clearly defined and easily extractable. In terms of false positives, a significant portion is caused by the use of top level domains as regular strings in the non-TLD positions of the hostname. We manually define a set of 11 TLDs[7] that often appear in a non-TLD position of the hostname in hope of eliminating corresponding false positives. However, the incompleteness of this summarization still led to a number of false positives, the most typical a few of which include TLD "travel" as in `travel.yahoo.com`, TLD "web" as in `web.sourceforge.com`, and TLD "fi" as in `fi.netlog.com`.
This feature could be improved in many ways. Defining a more complete list of multi-purposed TLDs will help in controling FP. This feature can also be used as a filter rather than a feature for machine learning in light of its low FP.

(7) ***Bad forms***. This feature is among the best in the whole set, achieving an AUC of over 0.89 in almost all cases. To spoof this feature, attackers could use login keywords beyond our dictionary, put the login keyword outside the FORM tag in the HTML, or rely on images of text, etc. In responding to phishers' manipulation, we can design new constraints or relax existing ones in this feature to capture more phishing patterns, such as adding more entries in the login keyword list, expanding the search scope, etc. On the other hand, certain legitimate webpages may meet the four conditions of this feature and are thus deemed as malicious. Though a login form is incorrectly detected in such cases, the page is unlikely to show suspicious patterns on the features, and we will rely on the features to make correct classification.

(8) ***Bad action fields***. This is another top-performing feature, with an AUC of more than 0.89 in almost all cases. This feature examines three criteria in a disjunctive fashion when looking for bad action fields. One trick to spoof the features in this feature is to use an absolute URL or a relative URL with directory hierarchy in the action field to refer to the authentication code. As a countermeasure, we can extend this feature by checking the scheme part of the URL in the action field. Since phishing pages seldom use "https", this simple enhancement is able to catch this phishing exploitation regardless of absolute or relative URLs being used.

(9) ***Non-matching URLs***. This is another top-ranked feature, with over 0.85 AUC. Extracting values for this feature is complicated, and two types of errors may occur due to the enormous ways of assigning link values in the HTML. An example for false positive is a webpage on the "live.com" domain, whose most frequent domain in the HTML links is "msn.com". On the contrary, attackers could try to evade detection by designing links in the HTML in a way that this feature misses. For example, a different URL[8] can be assigned to each link in the HTML, and the most frequent domain is chosen to point to the webpage domain. However, this process significantly magnifies the cost of building a massive volume of phishing sites via toolkits, and may not be feasible for phishers in reality.

[7]This set of country code and non-country code top level domains include mu, bt, info, ci, il, tv, fr, gm, my, it, us.
[8]By "different" in this context, we mean two URLs that differ not only in the fragment component of the URL.

(10) ***Out-of-position brand name***. False negatives of this feature are caused by a number of factors. The first is that most phishing pages do not include the target brands in the URL domain names. The second issue is that the set of heuristics in extracting the dominant domain keyword in the HTML only cover common patterns and may fail to discover the genuine target brand name under some occasions.
On the other hand, raising warnings on legitimate sites is rare for this feature, since it is not common practice for legitimate websites to insert the brand name before the webpage domain. The only a few false positives in our experiment share one property, i.e., the dominant domain keyword equals the page domain keyword, which is in turn a substring of the segment to its left in the hostname, such as `www.chaseonline.chase.com`. As a remedy, we can add one simple heuristic removing such unintended matches to reduce potential false positives.
Though the AUC of this feature is only slightly better than that of random guessing (an AUC of 0.5), the enhanced version of this feature as analyzed above will have extremely low FP, indicating that out-of-position brand name can be used as a filter rather than a feature in the machine learning framework with trivial contributions. Moreover, more patterns can be encoded into this feature to cover more phishing URL variants, magnifying the TP of this feature when used alone.

(11) ***Age of domain***. Mostly with an AUC between 0.7 and 0.84, this feature behaves fairly. Attackers sometimes upload phishing sites onto compromised legal domains, or host phishing attacks on otherwise legitimate domains, in which cases the age of domain is much older than typical phishing domains. However, there is a limit on the extent attackers could go in compromising legitimate sites given the fact that the cyberspace is paying increasingly intensive attention to security. On the other hand, newly registered legal domains tend to be incorrectly regarded as being phishy. However, other features of such new legitimate sites may exhibit typical benign patterns, and a correct prediction is thus still likely.

(12) ***Page in top search results***. One of the best features in the whole set, this feature has an AUC of over 0.85. In terms of false negatives, this feature will fail if a phishing domain is returned among top N results. However, real-time applications of our algorithm do not suffer from this problem as much. An informal explanation for that has two main points. First, when a new phish just comes out of attackers' workshop, few, if any, backlinks exist for it, and search engines are unlikely to return its domain in a top entry; second, search engines might index the phish as time progresses when more links on the web begin referring to it, however, the phish may have already become unavailable due to the short-lived nature of phishing activity.
For false positives, one cause is the selection of irrelevant query words via the TF-IDF metric. As a remedy, extending TF-IDF with other IR measures in selecting query words may help boost the intended domain to top result positions. Another possible cause of false positives is that the top search results subsume domains pointing to the same entity as the webpage domain, yet with different literal representations, such as "live.com" and "msn.com". Such related domains pose a threat to the current design of this feature, and feature 14 and 15 below are introduced to ameliorate this problem. One last hard situation is new legitimate websites, which

have not been covered by web crawlers, and thus might be mistakenly warned by this feature. For such cases, we rely on other features to make correct classifications.

(13) ***PageRank***. Mostly with an AUC between 0.76 and 0.8, this feature performs fairly in distinguishing the two types of webpages. Exploitation techniques exist for manipulating search engine algorithms to boost PageRank, however, major search engines are constantly improving their techniques to provide safer and better services, rendering phishers' attempt largely ineffective.
On the other hand, classifying a webpage based on PageRank only is not a simple task when the PageRank value is not available. Given an astronomical number of pages on the web, it is impossible that each has a valid PageRank value, especially for the gigantic number of unpopular webpages. In addition, though hosted on well-known websites, certain specific pages deep in the URL directory hierarchy may have zero or missing PageRank. Therefore, it will be more effective to harness other informative features to remedy the weakness of PageRank, and our other features serve that purpose.

(14) ***Search copyright company name plus domain***. This is another well-performing feature, with an AUC between 0.76 and 0.8. Aiming at finding the intended webpage directly by searching the copyright brand name and page domain, this feature is able to augment the TF-IDF feature under a certain circumstances. For instance, for legitimate URL `http://help.blogger.com/bin/static.py?page=start.cs` which the TF-IDF feature misclassifies, searching the copyright brand name "Google" plus the page domain successfully brings up "blogger.com" in top 10 entries.
In terms of false positives, one source of error comes from the process of extracting copyright brand names. The current regular expressions in this feature do not capture all the copyright variants and thus may fail in finding a brand name, or return noisy strings in addition to the intended copyright name. Searching such queries sometimes fails to retrieve the page domain among top 10 positions. Another possible cause of false positives is that the copyright name and page domain sometimes coincide. Executing search with two identical brand names in the query may bring up irrelevant results in top positions. On the other hand, false negatives are mostly caused by the cases in which phishing pages are hosted by compromised legal domains with no copyright fields.
Multiple ways exist in improving this feature. The first is to design more robust and effective brand name detectors, so that the recognized copyright name indeed subsumes the brand name with minimal noise. Another way is to remove duplicate domains in the query to boost the ranking of the intended page domain.

(15) ***Search copyright company name plus hostname***. This feature is based on a similar rationale as feature 14, with a better AUC in general (over 0.8), and accordingly, the resultant two types of errors are also analogous. However, since domain is only a substring of the hostname, this feature imposes stricter query criteria and thus has lower FP and TP than feature 14.

In addition to the AUC of each feature under the machine learning framework, we also give a more straightforward evaluation of the binary features among our feature set in Table VI. In this table, TP is simply the percentage of phishing attacks that the value of each feature indicates phishy (1 or 0 depending on the

Table VI. Performance of the binary features without using machine learning algorithms. TP in this table is computed as the ratio between the number of pages on which the feature values indicate phishy and the total number of phishing attacks. FP in this table is computed as the ratio between the number of pages on which the feature values indicate phishy and the total number of legitimate pages. We used all 2219 unique phishing attacks (phish set 1 + phish set 2) and 2561 legitimate pages (legitimate corpus 1) in this experiment. Top-performing features include page in top search results, bad forms, bad action fields, and non-matching URLs, which agree very well with the feature-wise result by AUC.

Feature name	TP (%)	FP (%)
Embedded domain	20.05	0.16
IP address	10.50	0.04
Suspicious URL	11.99	3.59
Out-of-position TLD	7.80	0.27
Bad forms	91.80	11.52
Bad action fields	88.01	9.64
Non-matching URLs	79.68	5.58
Out-of-position brand name	7.08	0.27
Page in top search results	92.92	21.44
Search copyright brand plus domain	65.57	9.76
Search copyright brand plus hostname	73.41	9.84

specific features), and FP is the percentage of the legitimate web pages that the value of each feature indicates phishy, with neither involving the learning step. Statistics in Table VI show that page in top search results, bad forms, bad action fields, and non-matching URLs perform the best in distinguishing two types of web pages, coinciding with the result by AUC above.

9.2 Runtime Performance

Our framework is composed of a training phase and a testing phase (Figure 1). The training phase can be done offline, and so users do not experience any time delay in this stage. When deployed and put into real-time usage, the testing phase is conducted in an online fashion as each webpage arrives, and the modules with high time complexity to a great extent determine the runtime performance and user experience. Since a certain information for each webpage in our corpus such as WHOIS records has been extracted and stored locally apriori, we only report the approximated running time (microseconds) of CANTINA+ in this section. All our experiment was conducted on a computer with a 1.73GHz processor and 2G RAM.

For the hash-based duplicate removal component, the SHA1 hash can be computed efficiently on the HTML, causing no apparent delay in the work flow. The average run time of this module is $58992ms \pm 28231ms$ per webpage in our corpus (mean $\pm$ standard deviation). Detecting login forms only involves traversing the HTML DOM, with an average run time of $151979ms \pm 722565ms$ per page. Once the feature values have been extracted, applying the pre-trained machine learning models also consumes an amount of time that is trivial. The module with critical time issue is the feature extractor, and we will give a thorough analysis on the time complexity of this module based on each individual feature.

The runtime performance of all our features is given in Table VII. Features in the first category (feature 1 through 6) simply probe the URLs via hard-coded patterns or simple regular expressions, and are thus the least time intensive in the feature

Table VII. Average running time and standard deviation of each feature (microseconds). The "bad forms" feature has a huge standard deviation of 2.56e+5ms due to its use of regular expressions on the page content to extract copyright company names and the high irregularity of HTML documents. The "age of domain" feature has a big standard deviation of 1.02e+6ms due to the irregular response times of the WHOIS service.

Feature name	Average runtime (msec)	Standard deviation
Embedded domain	78	1099
IP address	72	1065
Number of dots in URL	32	699
Suspicious URL	32	704
Number of sensitive words in URL	46	846
Out-of-position TLD	68	1027
Bad forms	240049	2562551
Bad action fields	62	977
Non-matching URLs	7552	10946
Out-of-position brand name	88	1172
Age of domain	606266	1020873
Page in top search results	639736	440999
PageRank	500473	422601
Search copyright brand plus domain	414202	228470
Search copyright brand plus hostname	408619	206084

set. Features in the second category (feature 7 through 10) generally run slower than those in the first category due to the use of HTML DOM. However, DOM maneuver is essentially tree traversal with a certain extra string operations and is thus still sufficiently fast. Actually, multiple tasks are completed in one traversal of the HTML DOM in our implementation, such as finding the most frequent domain keyword via heuristics, recording all the forms and their action fields, etc., which significantly expedites the running of feature extraction. In contrast, the third category (feature 11 through 15) involves web search in addition to playing with the HTML DOM, yielding multiple round-trip web traffic, and thus to a large degree determines the time performance of our approach.

Various measures exist in improving the time performance of the feature extraction module. An essential strategy is caching. The age of domain feature benefits from caching the most, since the domain registration date is invariant and typical browsing behavior keeps most people on a small group of favorite websites to them. Also, the domain is shared throughout the whole site, and WHOIS lookups only need to be performed once. Similarly, caching the query and result pairs for the three search engine based features helps improve runtime performance. Particularly, the two features involving copyright search will be accelerated tremendously thanks to users' surfing habits and the fact that the copyright field is usually homogeneous within the same website. Another way to speedup the search-oriented features is to directly cache the URLs that have already been checked.

Besides proper caching, login form detection also plays a critical role in lowering the time complexity of our system. In reality, the vast majority of legitimate pages do not have login forms and thus will be directly filtered with a "legitimate" label. Only a tiny percent of the whole web with login forms will have to go through the feature extraction process, and users probably visit these webpages to access their accounts, which involves a fair amount of typing and checking and thus gives

our approach some extra time in determining the legitimacy of the corresponding pages. As long as users are warned before their personal credentials are submitted, the latency mostly caused by web search is acceptable since user experience is barely impacted and more importantly, no harm has been done to the web users.

9.3 Limitations of the Current CANTINA+

There are a certain phishing variants that our current CANTINA+, as well as almost all existing anti-phishing solutions, cannot deal with properly, which in turn makes our work and further research effort worthwhile.

Cross site scripting (XSS) is a type of typical security vulnerability in web applications by which criminals may launch a variant of phishing attack. Our anti-phishing solution is certainly not a panacea, but it is not designed to deal with this type of attacks in the first place. In this case, we will rely on other techniques such as cookie protection to assist our approach in detecting phish. Considering the fact that almost all phishing attacks are targeting on well-known financial sites, we have every reason to believe that those sites would provide solutions to mitigate XSS vulnerabilities.

Another difficult scenario is that attackers compromise legitimate domains and host phishing attacks on those servers. Though other features of CANTINA+ might fail in this case, the following features still work: "page in top search results", "PageRank", "page in top results when searching copyright company name and domain", "page in top results when searching copyright company name and hostname", and our anti-phishing algorithm therefore still has a chance catching such attacks.

Attackers sometimes build phishing web pages purely made up of images, leaving our algorithm no text to analyze. Although text-based technique is infeasible here, the sheer fact that there are only a bunch of images and a login form without text at all on a web page is a good indicator of a phishing attack, and we could train models using this as a feature.

10. CONCLUSIONS

In this paper, we presented CANTINA+, a layered solution for phishing web page detection with two goals: 1) to catch the constantly evolving novel phishing attacks, and 2) to control the false positive rate under an acceptable level. Exploiting the generalization power of machine learning techniques, CANTINA+ achieves the former with a classification engine that fully enjoys the expressiveness and effectiveness of a rich set of high-level features of our design, which is the major contribution of our work in this paper. Based on the observation that all of the current phishing attacks utilize login forms to hold sensitive user information, CANTINA+ realizes the latter by a login form detection algorithm that filters webpages with no login forms prior to the feature extraction step. Not only does this filter reduce the FP to a much lower level, it also significantly speeds up the running of our system. Moreover, exploiting the fact that a substantial percent of current phishing sites are created automatically by toolkits with high similarity, CANTINA+ employs a hash-based filtering module to inspect web pages in the beginning of the layered pipeline to identify near-duplicate attacks in a fast and highly reliable fashion.

Significantly overcoming the weaknesses of the mainstream URL blacklists and

feature-based phish detection approaches, CANTINA+ is demonstrated in our extensive experiment over a diverse spectrum of corpus to be an effective anti-phishing solution. Specifically, in the randomized evaluation where training and testing phish are randomly selected, CANTINA+ achieved over 90% TP on unique testing phish, over 99% TP on near-duplicate testing phish, and about 0.4% FP under 10% training phish with login form filtering to slash false positives. In the time-based evaluation methodology where earlier phish were used as training data to build classification models for later attacks, our approach achieved over 87% TP on unique testing phish, about 95% TP on near-duplicate testing phish, and about 1% FP under 20% training phish with a two-week sliding window. Capable of achieving 0.4% FP and over 90% TP, our CANTINA+ has the potential to be a competitive anti-phishing solution.

Our CANTINA+ achieved comparable performance under six state-of-the-art learning algorithms in the detection engine, which manifests that the superior performance comes from the efficacy of our feature set and hybrid design rather than a particular pick of machine learning algorithms. This also suggests that in real-world deployment of CANTINA+, practitioners could choose their favorite learning algorithms as necessary.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation grant CCF-0524189 ("Supporting Trust Decisions") and by Army Research Office grant DAAD19-02-1-0389 ("Perpetually Available and Secure Information Systems"). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation or the U.S. government.

REFERENCES

3SHARP REPORT. 2006. Gone phishing: Evaluating anti-phishing tools for windows. `http://www.3sharp.com/projects/antiphishing/gone-phishing.pdf`.

ABU-NIMEH, S., NAPPA, D., WANG, X., AND NAIR, S. 2007. A comparison of machine learning techniques for phishing detection. In *Proceedings of the anti-phishing working groups (APWG) 2nd annual eCrime researchers summit*. 60–69.

BURGES, C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery 2,* 2, 121–167.

CHEN, T.-C., DICK, S., AND MILLER, J. 2010. Detecting visually similar web pages: Application to phishing detection. *ACM Transactions on Internet Technology (TOIT) 10,* 2.

CHOU, N., LEDESMA, R., TERAGUCHI, Y., AND MITCHELL, J. C. 2004. Client-side defense against web-based identity theft. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS'04)*.

CORTES, C. AND MOHRI, M. 2003. Auc optimization vs. error rate minimization. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS'2003)*.

COVA, M., KRUEGEL, C., AND VIGNA, G. 2008. There is no free phish: An analysis of 'free' and live phishing kits. In *Proceedings of the 2nd USENIX Workshop on Offensive Technologies (WOOT'08)*.

DHAMIJA, R. AND TYGAR, J. D. 2005. The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 symposium on Usable privacy and security (SOUPS'2005)*. 77–88.

FAWCETT, T. 2006. An introduction to roc analysis. *Pattern Recognition Letters 27*, 861–874.

Fette, I., Sadeh, N., and Tomasic, A. 2007. Learning to detect phishing emails. In *Proceedings of the 16th international conference on World Wide Web (WWW'07)*. 649–656.

Fetterly, D., Manasse, M., and Najork, M. 2003. On the evolution of clusters of near-duplicate web pages. In *Proceedings of the 1st Conference on Latin American Web Congress (LA-WEB'03)*. 37–45.

Garera, S., Provos, N., Chew, M., and Rubin, A. D. 2007. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM Workshop on Recurring Malcode (WORM'07)*. 1–8.

Kirda, E. and Kruegel, C. 2005. Protecting users against phishing attacks with antiphish. In *Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05)*. Vol. 1. 517–524.

Liu, W., Huang, G., Liu, X., Zhang, M., and Deng, X. 2005. Detection of phishing webpages based on visual similarity. In *Special interest tracks and posters of the 14th international conference on World Wide Web (WWW'05)*. 1060–1061.

Ludl, C., McAllister, S., Kirda, E., and Kruegel, C. 2007. On the effectiveness of techniques to detect phishing sites. *Lecture Notes in Computer Science (LNCS) 4579*, 20–39.

McCall, T. 2007. Gartner survey. `http://www.gartner.com/it/page.jsp?id=565125`.

Medvet, E., Eurecom, E. K., and Kruegel, C. 2008. Visual-similarity-based phishing detection. In *Proceedings of the 4th international conference on Security and privacy in communication networks (SecureComm'08)*. 30–36.

Moore, T. and Clayton, R. 2007. Examining the impact of website take-down on phishing. In *Proceedings of the Anti-phishing Working Groups (APWG) 2nd annual eCrime Researchers Summit*. 1–13.

NIST. 1995. Secure hash standard. *Federal Information Processing Standards Publication 180-1*. National Institute of Standards and Technology (NIST).

Pan, Y. and Ding, X. 2006. Anomaly based web phishing page detection. In *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC'06)*. 381–392.

PhishTank. `http://www.phishtank.com/stats.php`.

PhishTank. `http://data.phishtank.com/data/online-valid/`.

Rosiello, A. P. E., Kirda, E., Kruegel, C., and Ferrandi, F. 2007. A layout-similarity-based approach for detecting phishing pages. In *Proceedings of the 3rd International Conference on Security and Privacy in Communication Networks (SecureComm'07)*. 454–463.

Ross, B., Jackson, C., Miyake, N., Boneh, D., and Mitchell, J. C. 2005. Stronger password authentication using browser extensions. In *Proceedings of the 14th conference on USENIX Security Symposium*. 17–32.

Sheng, S., Kumaraguru, P., Acquisti, A., Cranor, L., and Hong, J. 2009. Improving phishing countermeasures: An analysis of expert interviews. In *Proceedings of the 4th APWG eCrime Researchers Summit*.

Sheng, S., Wardman, B., Warner, G., Cranor, L., Hong, J., and Zhang, C. 2009. An empirical analysis of phishing blacklists. In *Proceedings of the 6th Conference on Email and Anti-Spam (CEAS'09)*.

Witten, I. H. and Frank, E. 2005. *Data Mining: Practical machine learning tools and techniques*, second ed. Morgan Kaufmann.

Xiang, G. and Hong, J. 2009. A hybrid phish detection approach by identity discovery and keywords retrieval. In *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*. 571–580.

Xiang, G., Pendleton, B. A., Hong, J. I., and Rose, C. P. 2010. A hierarchical adaptive probabilistic approach for zero hour phish detection. In *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*. 268–285.

Yue, C. and Wang, H. 2010. Bogusbiter: A transparent protection against phishing attacks. *ACM Transactions on Internet Technology (TOIT) 10,* 2.

Zhang, Y., Hong, J., and Cranor, L. 2007. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)*. 639–648.