Learning to Detect Phishing Emails

Ian Fette
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
icf@cs.cmu.edu

Norman Sadeh
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
sadeh@cs.cmu.edu

Anthony Tomasic
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
tomasic@cs.cmu.edu

ABSTRACT

Each month, more attacks are launched with the aim of making web users believe that they are communicating with a trusted entity for the purpose of stealing account information, logon credentials, and identity information in general. This attack method, commonly known as "phishing," is most commonly initiated by sending out emails with links to spoofed websites that harvest information. We present a method for detecting these attacks, which in its most general form is an application of machine learning on a feature set designed to highlight user-targeted deception in electronic communication. This method is applicable, with slight modification, to detection of phishing websites, or the emails used to direct victims to these sites. We evaluate this method on a set of approximately 860 such phishing emails, and 6950 non-phishing emails, and correctly identify over 96% of the phishing emails while only mis-classifying on the order of 0.1% of the legitimate emails. We conclude with thoughts on the future for such techniques to specifically identify deception, specifically with respect to the evolutionary nature of the attacks and information available.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce—*Security*; H.4.3 [**Information Systems**]: Communications Applications—*Electronic mail*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Feature evaluation and selection*

General Terms

Security

Keywords

phishing, email, filtering, spam, semantic attacks, learning

1. INTRODUCTION

Phishers launched a record number of attacks in January 2006, as reported by the Anti-Phishing Working Group [3]. This is part of a very clear trend in which the number of attacks are increasing without showing any signs of slowing. These attacks often take the form of an email that purports to be from a trusted entity, such as eBay or PayPal. The email states that the user needs to provide information, such as credit card numbers, identity information, or login credentials, often to correct some alleged problem supposedly found with an account. Some number of users fall for these attacks by providing the requested information, which can lead to fraudulent charges against credit cards, withdrawals from bank accounts, or other undesirable effects.

The phishing problem is a hard problem for a number of reasons. Most difficulties stem from the fact that it is very easy for an attacker to create an exact replica of a good site, such as that of a bank, that looks very convincing to users. Previous work [25] indicates that the ability to create good-looking copies, as well as users' unfamiliarity with browser security indicators, leads to a significant percentage of users being unable to recognize a phishing attack. Unfortunately, the ease with which copies can be made in the digital world also makes it difficult for computers to recognize phishing attacks. As the phishing websites and phishing emails are often nearly identical to legitimate websites and emails, current filters have limited success in detecting these attacks, leaving users vulnerable to a growing threat.

Our overall approach, first described in [13], centers on extracting information that can be used to detect deception targeted at web users, which is accomplished by looking at features from each incoming email or potential attack vector. This process involves extracting data directly present in the email, as well as collecting information from external sources. The combination of internal and external information is then used to create a compact representation called a feature vector, a collection of which are used to train a model. Based on a given feature vector and the trained model, a decision is made as to whether the instance represents a phishing attack or not. We present a detailed description of our approach, which filters approximately 96% of phishing emails before they ever reach the user.

The remainder of this paper is organized in the following manner. Section 2 discusses previous approaches to filtering phishing attacks, while Section 3 gives an overview of machine learning and how we apply it to the task of classifying phishing emails, and how it could be used in a browser toolbar. Section 4 covers the results of empirical evaluation, as well as some challenges presented therein. Section 5 presents some concluding remarks.

2. BACKGROUND

2.1 Toolbars

The first attempts specifically designed to filter phishing attacks have taken the form of browser toolbars, such as

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

the Spoofguard [8] and Netcraft [23] toolbars. As reported in [10], most toolbars are lucky to get 85% accuracy identifying phishing websites. Accuracy aside, there are both advantages disadvantages to toolbars when compared to email filtering.

The first disadvantage toolbars face when compared to email filtering is a decreased amount of contextual information. The email provides the context under which the attack is delivered to the user. An email filter can see what words are used to entice the user to take action, which is currently not knowable to a filter operating in a browser separate from the user's e-mail client. An email filter also has access to header information, which contains not only information about who sent the message, but also information about the route the message took to reach the user. This context is not currently available in the browser with given toolbar implementations.

Future work to more closely integrate a user's email environment with their browser could alleviate these problems, and would actually provide a potentially richer context in which to make a decision. As discussed later in this paper, there are some pieces of information available in the web browser and website itself that could help to make a more informed decision, especially if this information could be combined with the context from the initial attack vector, such as the email prompting a user to visit a given website. This is discussed in greater detail in Section 3.3.

The second disadvantage of toolbars is the inability to completely shield the user from the decision making process. Toolbars usually prompt users with a dialog box, which many users will simply dismiss or misinterpret, or worse yet these warning dialogs can be intercepted by user-space malware [2]. By filtering out phishing emails before they are ever seen by users, we avoid the risk of these warnings being dismissed by or hidden from the user. We also prevent the loss of productivity suffered by a user who has to take time to read, process, and delete these attack emails.

2.2 Email Filtering

Although there are clear advantages to filtering phishing attacks at the email level, there are at present not many methods specifically designed to target phishing emails, as opposed to spam emails in general. The most closely related prior attempt is [7], in which the authors use structural features of emails to determine whether or not they represent phishing attacks. The features are mostly linguistic, and include things such as the number of words in the email, the "richness" of the vocabulary, the structure of the subject line, and the presence of 18 keywords. Other examples include the filter built into Thunderbird 1.5 [21]. However, this filter is extremely simple, looking for only the presence of any one of three features, namely the presence of IP-based URLs, nonmatching URLs (discussed in Section 3.2.3), and the presence of an HTML "form" element. The Thunderbird built-in filter still only presents a warning to the user, and does not avoid the costs of storage and the user's time. In our implementation and evaluation, we seek to fill this gap in email-based phishing filters. Our approach is generalizable beyond email filtering, however, and we do note how it could be used and what changes would be required in the context of filtering web pages as opposed to emails.

Many people have proposed ways in which to eliminate spam emails in general, which would include phishing emails (see, for example, [17, 9, 16, 27, 26, 18]). A number of early attempts at combating spam emails were based on so-called "naïve" approaches, ranging from "bag-of-words", in which the features of an email are the presence or absence of highly frequent and rare words, to analysis of the entropy of the messages. While these approaches looking at the text of the email appear to do well for spam, in practice these approaches often fail to stop phishing emails. This makes sense, as phishing emails are designed to look as close as possible to a real, non-spam email that a legitimate company would (or already has) sent out. As such, it is our belief that to stop phishing emails, we need to look at features selected specifically to detect this class of emails.

Looking at class-specific features is not a new approach in email filtering. SpamAssassin [4], for instance, has a number of rules that try to detect features common in spam email that go beyond just the text of the email. Such tests include things like the ratio of pixels occupied by text to those occupied by images in a rendered version of the mail, presence of certain faked headers, and the like. Spamato [1] is another extensible filtering platform that ships with a number of advanced filters, such as Vipul's Razor [24] (a collaborative algorithm using both URLs and message hashes), that work in tandem to detect spam emails. Our contribution is a new approach focused on learning to detect phishing, or semantic attacks in general. We do this by extracting a plurality of features designed to highlight deception, utilizing both sources of information internal to the attack itself, as well as external sources to gain more information about the context of the attack. Our solution can easily be used in conjunction with existing spam filters. The solution significantly reduces the amount of phishing emails with minimal cost in terms of false positives (legitimate emails marked as phishing).

3. METHOD

3.1 Overall Approach

Our approach, PILFER, is a machine-learning based approach to classification [20]. In a general sense, we are deciding whether some communication is deceptive, i.e. whether it is designed to trick the user into believing they are communicating with a trusted source, when in reality the communication is from an attacker. We make this decision based on information from within the email or attack vector itself (an internal source), combined with information from external sources. This combination of information is then used as the input to a classifier, the result of which is a decision on whether the input contained data designed to deceive the user.

With respect to email classification, we have two classes, namely the class of phishing emails, and the class of good ("ham") emails. In this paper we present a collection of features that has been identified as being particularly successful at detecting phishing, given the current state of attacks. We expect that over time, as the attacks evolve, new sets of features will have to be identified combining information from both internal or external sources. The features currently used are presented in Section 3.2, with Section 3.3 discussing how these can be adapted for use in detecting phishing web pages. In Section 4 we present a method for evaluating the effectiveness of these features, as well as the results of such an evaluation.

3.2 Features as used in email classification

Some spam filters use hundreds of features to detect unwanted emails. We have tested a number of different features, and present in this paper a list of the ten features that are used in PILFER, which are either binary or continuous numeric features. As the nature of phishing attacks changes, additional features may become more powerful, and PILFER can easily be adapted by providing such new features to the classifier. At this point, however, we are able to obtain high accuracy with only ten features, which makes the decision boundaries less complex, and therefore both more intuitive and faster to evaluate. We explain these features in detail below. While some of these features are already implemented in spam filters (such as the presence of IP-based URLs), these features are also a useful component of a phishing filter.

3.2.1 IP-based URLs

Some phishing attacks are hosted off of compromised PCs. These machines may not have DNS entries, and the simplest way to refer to them is by IP address. Companies rarely link to pages by an IP-address, and so such a link in an email is a potential indication of a phishing attack. As such, anytime we see a link in an email whose host is an IP-address (such as `http://192.168.0.1/paypal.cgi?fix_account`), we flag the email as having an IP-based URL. As phishing attacks are becoming more sophisticated, IP-based links are becoming less prevalent, with attackers purchasing domain names to point to the attack website instead. However, there are still a significant number of IP-based attacks, and therefore this is still a useful feature. This feature is binary.

3.2.2 Age of linked-to domain names

Phishers are learning not to give themselves away by using IP-based URLs. Name-based attacks, in which a phisher will register a similar or otherwise legitimate-sounding domain name (such as playpal.com or paypal-update.com) are increasingly common. These domains often have a limited life, however. Phishers may register these domains with fraudulently obtained credit cards (in which case the registrar may cancel the registration), or the domain may be caught by a company hired to monitor registrations that seem suspicious. (Microsoft, for instance, watches for domain name registrations involving any of their trademarks.) As such, the phisher has an incentive to use these domain names shortly after registration. We therefore perform a WHOIS query on each domain name that is linked to, and store the date on which the registrar reports the domain was registered. If this date is within 60 days of the date the email was sent, the email is flagged with the feature of linking to a "fresh" domain. This is a binary feature.

3.2.3 Nonmatching URLs

Phishers often exploit HTML emails, in which it is possible to display a link that says paypal.com but actually links to badsite.com. For this feature, all links are checked, and if the text of a link is a URL, and the HREF of the link is to a different host than the link in the text, the email is flagged with a "nonmatching URL" feature. Such a link looks like `<a href="badsite.com"> paypal.com</a>`. This is a binary feature.

3.2.4 "Here" links to non-modal domain

Phishing emails, often contain text like "Click here to restore your account access". In many cases, this is the most predominantly displayed link, and is the link the phisher intends the user to click. Other links are maintained in the email to keep the authentic feel, such as the link to a privacy policy, a link to the user agreement, and others. We call the domain most frequently linked to the "modal domain" of the email. If there is a link with the text "link", "click", or "here" that links to a domain other than this "modal domain", the email is flagged with a "here" link to a non-modal domain feature. This is a binary feature.

3.2.5 HTML emails

Most emails are sent as either plain text, HTML, or a combination of the two in what is known as a multipart/alternative format. The email is flagged with the HTML email feature if it contains a section that is denoted with a MIME type of text/html. (This includes many multipart/alternative emails). While HTML email is not necessarily indicative of a phishing email, it does make many of the deceptions seen in phishing attacks possible. For a phisher to launch an attack without using HTML is difficult, because in a plain text email there is virtually no way to disguise the URL to which the user is taken. Thus, the user still can be deceived by legitimate-sounding domain names, but many of the technical, deceptive attacks are not possible. This is a binary feature.

3.2.6 Number of links

The number of links present in an email is a feature. The number of links is the number of links in the html part(s) of an email, where a link is defined as being an `<a>` tag with a href attribute. This includes mailto: links. This is a continuous feature.

3.2.7 Number of domains

For all URLs that start with either `http://` or `https://`, we extract the domain name for the purpose of determining whether the email contains a link to a "fresh" domain. For this feature, we simply take the domain names previously extracted from all of the links, and simply count the number of distinct domains. We try to only look at the "main" part of a domain, e.g. what a person actually would pay to register through a domain registrar. It should be noted that this is not necessarily the combination of the top- and second-level domain. For instance, we consider the "main" part of `www.cs.university.edu` to be `university.edu`, but the "main" part of `www.company.co.jp` would be `company.co.jp`, as this is what is actually registered with a registrar, even though technically the top-level domain is `.jp` and the second-level domain is `.co`. This feature is simply the number of such "main" domains linked to in the email, and is a continuous feature.

3.2.8 Number of dots

There are a number of ways for attackers to construct legitimate-looking URLs. One such method uses subdomains, like `http://www.my-bank.update.data.com`. Another method is to use a redirection script, such as `http://www.google.com/url?q=http://www.badsite.com`. To the user (or a naïve filter), this may appear to be a site hosted at google.com, but in reality will redirect the browser to badsite.com. In both of these examples, either by the inclusion of a URL into an open redirect script or by the use of a number of subdomains, there are a large number of dots in the URL. Of course, legitimate URLs also can contain a number of dots, and this does not make it a phishing URL, however there is still information conveyed by this feature, as its inclusion increases the accuracy in our empirical evaluations. This feature is simply the maximum number of dots ('.') contained in any of the links present in the email, and is a continuous feature.

3.2.9 Contains javascript

JavaScript is used for many things, from creating popup windows to changing the status bar of a web browser or email client. It can appear directly in the body of an email, or it can be embedded in something like a link. Attackers can use JavaScript to hide information from the user, and potentially launch sophisticated attacks. An email is flagged with the "contains javascript" feature if the string "javascript" appears in the email, regardless of whether it is actually in a `<script>` or <a> tag. This might not be optimal, but it makes parsing much simpler, especially when dealing with attacks that contain malformed HTML. This is a binary feature.

3.2.10 Spam-filter output

Many mail clients already have a spam filter in place, and as such it seems natural to leverage the ability of existing solutions in combating the phishing problem. We therefore include as a feature the class assigned to the email by SpamAssassin - either "ham" or "spam". This is a binary feature, using the trained version of SpamAssassin with the default rule weights and threshold.

3.3 Features as used in webpage classification

Although we have focused primarily on detecting attacks at the email level, most of the features discussed in Section 3.2 also can be applied towards classifiying a webpage in a browser environment. A spam filter cannot generally be run on a webpage, so the last feature is not applicable, but the other features can still be evaluated with slight modification. For instance, the number of dots could be turned into two features - the number of dots in the URL of the current page, and the maximum number of dots in all URLs linked to in the current page. The same extension could be applied to the evaluation of the domain age feature. One could likewise split the presence of deceptive links into two features - deceptive links on the current page, and deceptive links on the previous page. This technique may enable the use of additional context, and would be especially useful if the user is coming to the attack site from a message in their web-based email interface. Additionally, one might be able to make use of additional context available in the browser and its history in features such as the following.

3.3.1 Site in browser history

As phishing sites are short-lived and located at a number of different URLs, the presence or absence of the current website in the browser's history would provide information for the classification process. A site never previously visited (not in the history) is more likely to be a phishing website than a site already visited for the following simple reason: a user would have no reason to have previously visited that particular spoof of the legitimate site during its short lifetime. This feature could be used in a binary fashion (present in history or not) if that's all that were available, but if the history included the number of times the page was visited, that would be even more valuable. A large number of visits would establish some existing relationship with the site, which likely indicates some level of legitimacy.

3.3.2 Redirected site

A site can be reached in a number of different ways, including redirection from another site. When a user goes to a web page, either by clicking a link or typing in a URL, that web page can redirect the browser to a different page. Redirection has many legitimate uses, but one could imagine an attacker using a redirection service such as TinyURL [15] to hide the phishing site's URL in the email (or other attack vector). The browser is explicitly instructed to redirect to a new page, and as such it would be possible to create a feature out of whether or not the browser was redirected to the present page, or whether the user went to the current page explicitly. This information would be available in the context of a browser, but might not be available if only analyzing the source email.

3.3.3 tf-idf

"tf-idf", or term frequency-inverse document frequency, is a measure of importance of a term. One can use tf-idf to attempt to identify key terms of a page, and subsequently determine whether the current page is a copy of a more popular page. In general, this involves searching for the key terms on a page and checking whether the current page is present in the result. This method and its accuracy are discussed in more detail in [30].

4. EMPIRICAL EVALUATION

4.1 Overview

In this section, we present the details of our implementation used in evaluation of PILFER (Section 4.2) and in evaluating SpamAssassin (Section 4.3). The dataset of emails used to perform the evaluation is described in Section 4.4. Certain challenges are present when trying to do post-hoc analysis of phishing attacks, the specifics and impact of which are discussed in Section 4.5. Section 4.6 introduces some terminology, and Section 4.7 shows our results in classifying the dataset.

4.2 Machine-Learning Implementation

In order to test our model, we first run a set of scripts to extract all the features listed in Section 3.2. Once the features are extracted, we train and test a classifier using 10-fold cross validation. (The dataset is divided into ten distinct parts. Each part is then tested using the other nine parts of the data as the training data. This ensures that the training data is separate from the test data, and is called "cross-validation".) For our reference implementation of PILFER, we use a random forest [6] as a classifier. Random forests create a number of decision trees (in our case, 10), and each decision tree is made by randomly choosing an attribute to split on at each level, and then pruning the tree. The exact workings of the classifier are beyond the scope of this paper. We evaluated a number of other classifiers as well, including SVMs [11], rule-based approaches, normal decision trees, and Bayesian approaches, but the overall accuracies of most of the classifiers were not different with statistical significance. Accuracies for some of these other classifiers are shown in appendix A. For a complete discussion of classifiers and text classification in general, the reader is directed to a machine learning text such as [20] or [11].

4.3 Testing SpamAssassin

SpamAssassin is a widely-deployed freely-available spam filter that is highly accurate in classifying spam emails. For comparison against PILFER, we classify the exact same dataset using SpamAssassin version 3.1.0, using the default thresholds and rules. The results reported for "untrained" SpamAssassin are obtained by simply treating the entire dataset as a test set, and not training on any emails. This represents an out-of-the-box install of SpamAssassin. (To be sure that SpamAssassin was untrained, we deleted the .spamassassin directory where learned data is stored before testing the emails). The results reported for the "trained" SpamAssassin are from the same version, except that we now use 10-fold cross validation, where before each fold we clear SpamAssassin's learned data (by deleting ∽/.spamassassin). We then train on all the emails in the train part of the fold and test on those in the test part of the fold. To get realistic results from SpamAssassin, we disable online tests (blacklist lookups, mostly). Since the dataset we are using is slightly older and publically available, it is probable that the blacklists have much more information about the senders of the emails in the dataset at the time of testing than was available at the time the emails were sent, and so including these tests would artifically inflate the accuracy. By disabling these online tests, we hope to more closely approximate the information available at the time the attacks are first sent out. It is not a perfect approximation, but it is the closest we can come.

4.4 Datasets

Two publicly available datasets were used to test our implementation: the ham corpora from the SpamAssassin project [5] (both the 2002 and 2003 ham collections, easy and hard, for a total of approximately 6950 non-phishing non-spam emails), and the publicly available phishingcorpus [22] (approximately 860 email messages). We use a series of short scripts to programmatically extract the features from Section 3.2, and store these in a database for quick reference. We label emails as being non-phishing if they come from the SpamAssassin ham corpora, and as phishing if they come from the phishingcorpus. For these experiments we used the entire dataset and did not re-label any of its contents.

4.5 Additional Challenges

There are a number of challenges posed by doing post-hoc classification of phishing emails. Most of these challenges apply mainly to the phishing emails in the dataset and materialize in the form of missing information, which has the net effect of increasing the false negative rate. Without the challenges outlined below, which are mostly artifacts of testing after the fact as opposed to live in a real system, even better accuracy should be possible.

The age of the dataset poses the most problems, which is particularly relevant with the phishing corpus. Phishing websites are short-lived, often lasting only on the order of 48 hours [12]. Some of our features can therefore not be extracted from older emails, making our tests difficult. For instance, in one of our features, we are interested in the age of domains linked to. We perform a WHOIS query to determine the date a domain was registered, and subtract this date from the date the email was sent according to its headers to determine its age. In many cases of phishing attacks, however, these domains are no longer live at the time of our testing, resulting in missing information. The disappearance of domain names, combined with difficulty in parsing results from a large number of WHOIS servers returning results in non-standardized formats resulted in only being able to programmatically extract registration dates for 505 of a total of 870 distinct domain names referenced in the dataset at the time of writing.

It is not clear whether this dataset is representative of normal people's email inboxes or not, but to date it is the best data we have been able to find. We are currently planning a follow-up study where we will be having users label every email coming into their inbox as either legitimate, spam, or phishing. This future work will provide us with a dataset more representative of real users' inboxes.

4.6 False Positives vs. False Negatives

It is important to note that misclassifying a phishing email may have a different impact than misclassifying a good email, so we report separately the rate of false positives and false negatives. The false positive rate corresponds to the proportion of ham emails classified as phishing emails, and false negative rate corresponds to the proportion of phishing emails classified as ham. Let us denote the number of ham emails classified as ham (correctly classified) as ham_{ham}, the number of ham emails classified as phishing as ham_{phish}, the number of phishing emails classified as ham as $phish_{ham}$, and the number of phishing emails classified as phishing as $phish_{phish}$. We then define fp, the false positive rate, as

$$fp = \frac{ham_{phish}}{ham_{phish} + ham_{ham}}$$

and fn, the false negatives rate, as

$$fn = \frac{phish_{ham}}{phish_{ham} + phish_{phish}}$$

Given this definition, $fp = 0.1$ would correspond to one of every ten good emails being classified as phishing, and $fn = 0.2$ would correspond to two of every ten phishing emails being classified as good. We will use the terms fp and fn in this manner in the evaluations presented in the rest of the paper.

4.7 Results

On our dataset, we are able to more accurately classify emails using PILFER than by using a spam filter alone. PILFER achieves an overall accuracy of 99.5%. with a false positive rate fp of approximately 0.0013. PILFER's false negative rate fn on the dataset is approximately 0.035, which is almost one fourth the false negative rate of the spam filter by itself. These results are compared in detail with those of SpamAssassin in Table 1. As seen in the table, the inclusion of the result of a spam filter as a feature to PILFER makes for a significant reduction in phishing emails that get by. While PILFER without the spam filter's input has comparable accuracy to the spam filter, the accuracy obtained by providing the spam filter's decision as an input to PILFER,

Table 1: Accuracy of classifier compared with baseline spam filter

Classifier	False Positive Rate fp	False Negative Rate fn
PILFER, with S.A. feature	0.0013	0.036
PILFER, without S.A. feature	0.0022	0.085
SpamAssassin (Untrained)	0.0014	0.376
SpamAssassin (Trained)	0.0012	0.130

Table 2: Percentage of emails matching the binary features

Feature	Non-Phishing Matched	Phishing Matched
Has IP link	0.06%	45.04%
Has "fresh" link	0.98%	12.49%
Has "nonmatching" URL	0.14%	50.64%
Has non-modal here link	0.82%	18.20%
Is HTML email	5.55%	93.47%
Contains JavaScript	2.30%	10.15%
SpamAssassin Output	0.12%	87.05%

i.e. the combination of the two, improves the accuracy to be much better than either one alone. This result suggests that the features present in the two are catching different subsets of the phishing emails, and shows that a phishing filter and a spam filter can work well as complementary parts of an overall solution.

Table 2 shows the exact percentages of emails (by class) matching each of the seven binary features. All of the binary features are matched more frequently by phishing emails than by nonphishing emails. For the three non-binary features, their averages and standard deviations per-class are shown in Table 3. These features have higher mean values for phishing emails.

In summary, PILFER can be either deployed in a standalone configuration without a spam filter to catch a large percentage of phishing emails with very few false positives, or in conjunction with an existing spam filter such as SpamAssassin for even higher accuracy. If a filter like SpamAssassin is already deployed, then adding PILFER has the advantage of significantly reducing the number of phishing emails making it to the user, while having no significant effect on the number of emails erroneously caught by the filtering system.

5. CONCLUDING REMARKS

In this paper, we have shown that it is possible to detect phishing emails with high accuracy by using a specialized filter, using features that are more directly applicable to phishing emails than those employed by general purpose spam filters. Although phishing is a subset of spam (after all, who asks to receive emails from a person pretending to be their bank for the purpose of fraud and identity theft?), it is characterized by certain unique properties that we have identified.

One might be inclined to think that phishing emails should be harder to detect than general spam emails. After all, phishing emails are designed to sound like an email from a legitimate company, often a company with which the attacker hopes the user has a pre-existing relationship. Models based on "naïve" assumptions, such as certain words like "Viagra" being indicative of a class of un-desirable emails, no longer hold when the attackers are using the same words and the same overall "feel" to lure the user into a false sense of security. At the same time, phishing emails present unique opportunities for detection that are not present in general spam emails.

In general spam emails, the sender does not need to misrepresent their identity. A company offering to sell "Viagra" over the Internet does not need to convince potential buyers that they are a pharmacy that the user already has a relationship with, such as CVS or RiteAid. Instead, a spammer can actually set up a (quasi-)legitimate company called Pharmacy1283, and identify themselves as such, with no need to try to convince users that they are receiving a communication from their bank, or some other entity with which they have an established relationship. It is this misrepresentation of sender identity that is key to the identification of phishing emails, and further work in the area should concentrate on features to identify this deceptive behavior.

As the phishing attacks evolve over time to employ alternate deceptive behaviors, so does the information available to combat these attacks. The approach used is flexible, and new external information sources can be added as they become available. These sources could take the form of web services, or other tagged resources, to provide additional information to the decision making process. For instance, many phishing attacks include copies of corporate logos, and if one could map a logo back to its legitimate owner's website, that would be valuable information in determining the authenticity of a website or email displaying that logo. As image sharing and tagging services such as Flickr [29] are increasing in use, it is not unreasonable to think that some day in the near future, one might actually be able to search with an image and get back a description as a result.

There are a number of emerging technologies that could greatly assist phishing classification that we have not considered. For instance, Sender ID Framework (SIDF) [19] and DomainKeys [28], along with other such sender authentication technologies, should help to both reduce false positives and make detection of spoofed senders much simpler in the time to come. Looking farther into the future, deeper knowledge-based models of the user and the types of prior

Table 3: Mean, standard deviation of the continuous features, per-class

Feature	μ_{phishing}	σ_{phishing}	$\mu_{\text{non-phishing}}$	$\sigma_{\text{non-phishing}}$
Number of links	3.87	4.97	2.36	12.00
Number of domains	1.49	1.42	0.43	3.32
Number of dots	3.78	1.94	0.19	0.87

relationships she may or may not have with different sites or organizations could also help fend off more sophisticated phishing attacks. Such techniques would likely build on ongoing research on federated identities and semantic web technologies [14]. In the meantime, however, we believe that using features such as those presented here can significantly help with detecting this class of phishing emails. We are currently in the process of building a live filtering solution based around PILFER, which we will start making available to users for testing for further validation.

6. ACKNOWLEDGEMENTS

The work reported herein has been supported in part under the NSF Cyber Trust initiative (Grant #0524189) and in part under ARO research grant DAAD19-02-1-0389 ("Perpetually Available and Secure Information Systems") to Carnegie Mellon University's CyLab. The authors would also like to thank Lorrie Cranor, Jason Hong, Alessandro Acquisti, Julie Downs, Sven Dietrich, Serge Egelman, Mandy Holbrook, Ponnurangam Kumaraguru, and Steve Sheng. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

[1] K. Albrecht, N. Burri, and R. Wattenhofer. Spamato - An Extendable Spam Filter System. In *2nd Conference on Email and Anti-Spam (CEAS), Stanford University, Palo Alto, California, USA*, July 2005.

[2] A. Alsaid and C. J. Mitchell. Installing fake root keys in a pc. In *EuroPKI*, pages 227–239, 2005.

[3] Anti-Phishing Working Group. Phishing activity trends report, Jan. 2005. `http://www.antiphishing.org/reports/apwg_report_jan_2006.pdf`.

[4] Apache Software Foundation. Spamassassin homepage, 2006. `http://spamassassin.apache.org/`.

[5] Apache Software Foundation. Spamassassin public corpus, 2006. `http://spamassassin.apache.org/publiccorpus/`.

[6] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[7] M. Chandrasekaran, K. Karayanan, and S. Upadhyaya. Towards phishing e-mail detection based on their structural properties. In *New York State Cyber Security Conference*, 2006.

[8] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell. Client-side defense against web-based identity theft. In *NDSS*, 2004.

[9] W. Cohen. Learning to classify English text with ILP methods. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 124–143. IOS Press, 1996.

[10] L. Cranor, S. Egelman, J. Hong, and Y. Zhang. Phinding phish: An evaluation of anti-phishing toolbars. Technical report, Carnegie Mellon University, Nov. 2006.

[11] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods.* Cambridge University Press, New York, NY, USA, 2000.

[12] FDIC. Putting an end to account-hijacking identity theft, Dec. 2004. `http://www.fdic.gov/consumers/consumer/idtheftstudy/identity_theft.pdf`.

[13] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. Technical Report CMU-ISRI-06-112, Institute for Software Research, Carnegie Mellon University, June 2006. `http://reports-archive.adm.cs.cmu.edu/anon/isri2006/abstracts/06-112.html`.

[14] F. L. Gandon and N. M. Sadeh. Semantic web technologies to reconcile privacy and context awareness. *Journal of Web Semantics*, 1(3):241–260, 2004.

[15] Gilby Productions. Tinyurl, 2006. `http://www.tinyurl.com/`.

[16] P. Graham. Better bayesian filtering. In *Proceedings of the 2003 Spam Conference*, Jan 2003.

[17] B. Leiba and N. Borenstein. A multifaceted approach to spam reduction. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.

[18] T. Meyer and B. Whateley. Spambayes: Effective open-source, bayesian based, email classification system. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.

[19] Microsoft. Sender ID framework, 2006. `http://www.microsoft.com/senderid`.

[20] T. M. Mitchell. *Machine Learning.* McGraw-Hill Higher Education, 1997.

[21] Mozilla. Mozilla thunderbird, 2006. `http://www.mozilla.com/thunderbird/`.

[22] J. Nazario. phishingcorpus homepage, Apr. 2006. `http://monkey.org/%7Ejose/wiki/doku.php?id=PhishingCorpus`.

[23] Netcraft Ltd. Netcraft toolbar, 2006. `http://toolbar.netcraft.com/`.

[24] V. V. Prakash. Vipul's razor, 2006. `http://razor.sourceforge.net`.

[25] M. H. Rachna Dhamija, Doug Tygar. Why phishing works. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590. ACM Special Interest Group on Computer-Human Interaction, January 2006.

[26] I. Rigoutsos and T. Huynh. Chung-kwei: a pattern-discovery-based system for the automatic

Table 4: Average Accuracy of different classifiers on same features over 10 runs, with standard deviations

Classifier	fp	σ_{fp}	fn	σ_{fn}
Random Forest	0.0012	0.0013	0.0380	0.0205
SVM, C = 10	0.0024	0.0019	0.0408	0.0225
RIPPER	0.0025	0.0019	0.0383	0.0204
Decision Table	0.0022	0.0018	0.0555	0.0242
Nearest Neighbor w/ Generalization	0.0017	0.0022	0.0414	0.0265
1R	0.0012	0.0012	0.1295	0.0333
Alternating Decision Tree	0.0020	0.0018	0.0405	0.0229
Decision Stump	0.0012	0.0012	0.1295	0.0333
Pruned C4.5 Tree	0.0019	0.0017	0.0414	0.0235
Hybrid tree w/ Naïve Bayes leaves	0.0022	0.0017	0.0412	0.0209
Random Tree (1 random attribute/node)	0.0016	0.0015	0.0398	0.0200
AdaBoosted C4.5 tree	0.0019	0.0017	0.0414	0.0235
AdaBoosted Decision Stump	0.0016	0.0016	0.0748	0.0355
Voted Perceptron	0.0122	0.0053	0.0942	0.0311
Bayes Net	0.0384	0.0082	0.0689	0.0244
Naïve Bayes	0.0107	0.0030	0.0608	0.0248

identification of unsolicited e-mail messages (spam). In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.

[27] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[28] Yahoo. Domainkeys, 2006. `http://antispam.yahoo.com/domainkeys`.

[29] Yahoo. Flickr homepage, 2006. `http://www.flickr.com/`.

[30] Y. Zhang, J. Hong, and L. Cranor. Cantina: A content-based approach to detecting phishing web sites. In *WWW*, 2007.

APPENDIX

A. ACCURACIES OF OTHER CLASSIFIERS

Table 4 shows the accuracies in terms of false positive and false negative rates when different classifiers are used instead of the random forest used in PILFER. For almost all of the high-performing classifiers, the difference in accuracy is not statistically significant, and none are statistically significantly better.