Large-Scale Automatic Classification of Phishing Pages

Colin Whittaker
Google Inc.
cwhittak@google.com

Brian Ryner
Google Inc.
bryner@google.com

Marria Nazif
Google Inc.
marria@google.com

Abstract

Phishing websites, fraudulent sites that impersonate a trusted third party to gain access to private data, continue to cost Internet users over a billion dollars each year. In this paper, we describe the design and performance characteristics of a scalable machine learning classifier we developed to detect phishing websites. We use this classifier to maintain Google's phishing blacklist automatically. Our classifier analyzes millions of pages a day, examining the URL and the contents of a page to determine whether or not a page is phishing. Unlike previous work in this field, we train the classifier on a noisy dataset consisting of millions of samples from previously collected live classification data. Despite the noise in the training data, our classifier learns a robust model for identifying phishing pages which correctly classifies more than 90% of phishing pages several weeks after training concludes.

1 Introduction

Phishing is a social engineering crime generally defined as impersonating a trusted third party to gain access to private data. For example, an adversary might send the victim an email directing him to a fraudulent website that looks like a page belonging to a bank. The adversary can use any information the victim enters into the phishing page to drain the victim's bank account or steal the victim's identity. Despite increasing public awareness, phishing continues to be a major threat to Internet users. Gartner estimates that phishers stole $1.7 billion in 2008, and the Anti-Phishing Working Group identified roughly twenty thousand unique new phishing sites each month between July and December of 2008 [3], [17]. To help combat phishing, Google publishes a blacklist of phishing URLs and phishing URL patterns [7], [29]. The anti-phishing features in Firefox 3, Google Chrome, and Apple Safari use this blacklist. We provide access to the list to other clients through our public API [18].

In order for an anti-phishing blacklist to be effective, it must be comprehensive, error-free, and timely. A blacklist that is not comprehensive fails to protect a portion of its users. One that is not error-free subjects users to unnecessary warnings and ultimately trains its users to ignore the warnings. A blacklist that is not timely may fail to warn its users about a phishing page in time to protect them. Considering that phishing pages only remain active for an average of approximately three days, with the majority of pages lasting less than a day, a delay of only a few hours can significantly degrade the quality of a blacklist [2], [30].

Currently, human reviewers maintain some blacklists, like the one published by PhishTank [25]. With PhishTank, the user community manually verifies potential phishing pages submitted by community members to keep their blacklist mostly error-free. Unfortunately, this review process takes a considerable amount of time, ranging from a median of over ten hours in March, 2009 to a median of over fifty hours in June, 2009, according to PhishTank's statistics. Omitting verification to improve the timeliness of the data is not a good option for PhishTank. Without verification, the list would have many false positives coming from either innocent confusion or malicious abuse.

An automatic classifier could handle this verification task. Previously published efforts have shown that a classification system could examine the same signals a human reviewer uses to evaluate whether a page is phishing [13], [16], [20], [21], [35]. Such a system could add verified phishing pages to the blacklist automatically, substantially reducing the verification time and improving the throughput. With higher throughput, the system could even examine large numbers of questionable, automatically collected URLs to look for otherwise missed phishing pages.

This paper describes such an automatic phishing classifier that we built and currently use to evaluate phishing pages and maintain our blacklist. Since its activation in November, 2008, this system evaluates millions of potential phishing pages every day. To evaluate each page, the classifier considers features regarding the page's URL, content, and hosting information. We retrain this classifier daily using approximately ten million samples from classification data collected over the last three months. To provide

training labels for this data, we use our published blacklist, the most complete listing of known phishing pages we have available. Since the coverage of our published blacklist is not perfect, the training labels contain a number of misclassifications. Nevertheless, our process develops classification models that demonstrate excellent performance, maintaining a false positive rate well below 0.1% while maintaining high recall. During the first six months of 2009, our classifier evaluated hundreds of millions of pages, automatically blacklisting 165,382 phishing pages. For comparison, PhishTank evaluated 139,340 potential phishing pages, finding only 47,203 actual phishing pages, during the same timespan [25].

The contributions of this paper are: 1) A demonstration that a scalable machine learning classifier can be used to automatically maintain a blacklist of phishing pages. 2) A demonstration that a machine learning classifier for phishing pages can achieve a very high accuracy despite a noisy training set. 3) An examination of the value of certain features of a web page in the evaluation of whether that page is phishing.

In Section 2, we precisely define "phishing" and review previous work regarding anti-phishing tools and classifiers. Section 3 describes the classifier's design and how we operate it in our production environment. We provide an analysis of the performance of the classifier in Section 4 before concluding in Section 5.

2 Background

2.1 Definition of Phishing

PhishTank defines phishing as "a fraudulent attempt, usually made through email, to steal ... personal information" [25]. In order to protect end users against the broadest set of phishing scams, we use a somewhat more general definition of phishing than this. We define a phishing page as any web page that, without permission, alleges to act on behalf of a third party with the intention of confusing viewers into performing an action with which the viewer would only trust a true agent of the third party. Note that these actions include, but are not limited to, submitting personal information to the page. In a sense, our definition of phishing is closer to "web forgery," the phrase used in the Firefox user interface, than the traditional definition of phishing. This definition certainly covers the typical case of phishing pages displaying graphics relating to a financial company and requesting a viewer's login credentials. This definition also covers phishing pages which display a trusted company's logos and request that the viewer download and execute an unknown binary. Sites which claim to be able to perform actions through a third party once provided with the viewer's login credentials meet this broader definition as well. Sites claiming to unblock an email or chat account when given the login credentials fall under this last category. Note that if one of these sites is sanctioned by the third party, then it would be properly authorized and therefore not a phishing page.

2.2 Related Work

Garera et al. described an early prototype of our current system, using both Google's internal data, like PageRank [26], and features extracted from the URL to classify pages [16]. Our system continues to use some of the features they describe while expanding the feature set considerably. However, the original preparation of the training set involved manual labeling, which is not feasible for training sets as large as the ones we use now. We designed a new training system for this paper.

Other papers have also examined the problem of automatically classifying phishing web pages, but they have described experimental systems, not systems in active use. Also, none of these efforts used a noisy training dataset. Our system, on the other hand, provides classifications to an audience of over a hundred million web users in real time. We can publish these classifications without further review because our classifier makes fewer false positive classifications than the other systems despite our noisy training data.

Zhang et al. presented a system for using Google web search as a filter for phishing pages but used only 2519 active URLs in their most realistic dataset [35]. Also, their conservative classifier demonstrated a false positive rate of 3%, too high to be viable without further review. Fette et al. described a system for classifying phishing emails that shares many similarities with our system, despite not classifying web pages [13]. In fact, we extract many of the same features regarding page content as they extracted regarding email content. Ludl et al. also discussed a system for classifying phishing pages based on features of the page which inspired some of our page features [20]. Ma et al. published a pair of papers describing another system for identifying malicious URLs by examining lexical features of the URLs and features of the sites' hosting information [21], [22]. We use very similar features in our system, though we also use a large number of features describing page content. Also, they do not assign labels for their URLs dynamically with their classifier as we do.

Besides our published blacklist, a number of other anti-phishing solutions exist. For example, several vendors provide a blacklist similar to ours. PhishTank, described in Section 1, makes its data available in the form of a downloadable feed [25]. Netcraft also makes a feed of its blacklist data available to service providers and hosting companies [24]. Additionally, Netcraft supplies a toolbar that blocks the user from browsing to phishing pages

and displays additional data, like websites' hosting countries. Other anti-phishing toolbars, like SpoofGuard, highlight suspicious website characteristics based on a set of heuristics [9]. Microsoft combines a manually compiled blacklist with heuristic analysis in its latest version of Internet Explorer [23].

Wu et al. examined the value of a number of these tools in the context of a series of simulated phishing attacks, though they did not examine any clients using Google's blacklist [33]. They also question the effectiveness of these toolbars from a user interface perspective, though recent studies have found that the Firefox anti-phishing feature is indeed effective at protecting users when triggered [12]. Zhang et al. looked at the coverage of the anti-phishing tools including the built-in phishing protection of Firefox 2, at the time powered by our blacklist, along with many other similar tools [34]. They found that our blacklist was competitive with other products, although their sample size was relatively small, consisting of only 100 phishing URLs and 516 legitimate URLs. A recent study by Sheng et al. also compared tools based on Google's blacklist with other anti-phishing tools in two experiments, one in October, 2008 and one in December, 2008 [30]. Incidentally, their two experiments compare the quality of our blacklist before and after we enabled our automatic classification system in November, 2008. We discuss the findings of this paper in Section 4.3.

Highlighting the need for an anti-phishing solution, several papers have studied why Internet users fall for phishing attacks in the first place. Dhamija et al. conclude that the average unaided user lacks the knowledge to properly interpret the security signals built into web browsers, leaving them vulnerable to phishing [10]. Downs et al. explore the mental models used by Internet users to evaluate potential phishing pages [11]. Some of their subjects used incorrect strategies to analyze potential scams, leaving them at risk. At a more fundamental level, Fogg et al. studied the attributes of a web page that make it credible [14]. They find that many features of a page's appearance enhance its perceived credibility, a fact that phishers routinely exploit and that we, in turn, use to motivate parts of our feature set.

3 Phishing Classifier Infrastructure

3.1 Overall System Design

Our system classifies web pages submitted by end users and collected from Gmail's spam filters. To successfully identify a wide variety of phishing pages, our system extracts and analyzes a number of features regarding these pages. These features describe the composition of the web page's URL, the hosting of the page, and the page's HTML content as collected by a crawler. A logistic regression classifier makes the final determination of whether a page is phishing on the basis of these features [5]. The classification model used by the classifier is developed in an offline training process. The training process uses features collected by the classification system over the past three months labeled according to our published blacklist.

Using our published blacklist in this fashion introduces the risk of feedback loops, where an error in our published data propagates to classification models used to generate additional published data. To minimize this risk, we also examine the relatively small number of user submitted phishing pages and reported errors manually, allowing us to break any such loops. Note that we must manually review these submissions anyway to promptly correct any user reported errors in our blacklist. However, less than one percent of the input to our system receives a manual review, leaving our automatic system to handle the bulk of the analysis.

Since we use our published blacklist to label the training data, the resulting classifier effectively generalizes the blacklist. We find that this works well in practice, since many of the phishing pages not on our blacklist are similar to pages on our blacklist. By expanding on the common features of known phishing pages, the classifier can correctly identify new phishing pages.

3.1.1 Design Goals. First, the classifier must prioritize precision over recall to minimize the number of published false positives[1]. In order to find as many phishing pages as possible, we examine a large set of pages, most of which are not phishing. Given the relative scarcity of actual phishing pages in the set of pages examined, the false positive rate of the classifier must be extremely low.

Second, the classifier must achieve high recall. If the classifier fails to identify most of the phishing pages, it does not adequately replace a manual system.

Third, the classifier must tolerate noisy training data. Since we must train on the data and classifications we already have available, we need the classifier to pick up consistent patterns despite any existing misclassifications.

Finally, the classifier must process a large number of web pages with low latency. Our URL collection system obtains millions of new pages to examine every day, so the classifier must keep up with the load. The classifier may need to drop some likely non-phishing URLs early in the classification process to reduce the load on bottlenecks in the system and improve overall latency.

[1] Precision = number of true positive classifications / number of positive classifications.

Recall = True positive rate = number of true positive classifications / number of positive examples.

False positive rate = number of false positive classifications / number of negative examples. [15]

3.2 Classification Workflow

The workflow for classifying a web page as phishing divides the work into a number of separate processes, each handled by a pool of workers. The first process extracts features about the URL of the page. The second process obtains domain information about the page and crawls it. If the crawl succeeds, the third process extracts features from the page contents. The final process assigns the page a score based on the collected features representing the probability that the page is phishing. If the score is high enough, this process automatically adds the page to the blacklist published by our serving system. To save bandwidth, our serving system combines similar blacklisted URLs into blacklist patterns.

To help run this workflow, we create tasks to represent units of work for each workflow process. A task management system buffers these tasks for the various processes, assigns tasks to workers, and retries any tasks that fail. After a worker finishes with a task, it stores any generated data in a Bigtable database and adds a new task to the queue for the next process in the workflow [8]. If a task fails or times out, the task is returned to its queue and retried. Since each task is an independent unit of work, no inter-worker coordination is necessary. Therefore, if the workflow cannot handle the volume of collected URLs, we can increase the number of workers, and the overall throughput of our system scales linearly.

A detailed description of each of the workflow processes follows. Refer to Section 4.2 for a summary of the described features with statistics regarding their value to the classifier.

3.2.1 Candidate URL Collection. We receive new potential phishing URLs in reports from users of our blacklist and from spam messages collected by Gmail. We receive approximately one thousand user reports and five million URLs from spam emails each day. For URLs from spam emails, we take precautions to make sure that we do not accidentally fetch user-identifiable URLs. Primarily, we ensure that several unique Gmail users received a URL before we add that URL to our system.

3.2.2 URL Feature Extraction. We can often tell whether or not a web page is phishing simply by looking at the URL. Phishers commonly construct their URLs to confuse the viewer into believing that the URLs belong to a trusted party. To identify the telltale signs of these efforts, the first process in the workflow, the URL Feature Extractor, looks only at the URL of the page to determine features.

First, if the URL is improperly constructed or if it matches a whitelist of high profile, safe sites, then the URL Feature Extractor drops the URL from the workflow entirely. We manually compile this whitelist of 2778 sites, requiring that each site both have high traffic and not host arbitrary user-generated content. Sites on this list include citibank.com and cnn.com.

One feature this process extracts is whether the URL contains an IP address for its hostname. Using an IP address in this fashion effectively disguises the owner of the site from a casual viewer. It also prevents administrators from shutting down the site by disabling the domain name. However, the URL will break if the host computer changes its IP address. Fortunately, static IP address hosting is easy to detect. Since a few legitimate services, like the Google webcache, use an IP address as the URL host, this feature cannot be used in isolation.

Another feature this process extracts is whether the page has many host components. Phishers commonly use a long hostname, prepending an authentic-sounding host to their fixed domain name, to confuse viewers into believing that the page is legitimate [33]. An example of this is 9794.myonlineaccounts2.abbeynational.co.uk.syrialand.com. This is also easy to detect automatically by counting the number of host segments in the URL before the domain (e.g., 5 in the previous example.)

Besides manipulating the structure of their URLs, phishers often include characteristic strings in their URLs to mislead viewers. These can include the trademarks of the phishing target, like "abbeynational" in the example above, or more general phrases associated with phishing targets, like "login". The URL Feature Extractor extracts all string tokens separated by non-alphanumeric characters out of the URL to use as features rather than looking for specific character strings as in Garera et al. [16]. By including all of these tokens, our models can respond automatically to phishing attacks that use a common string in their URLs. The feature extractor transforms each of these tokens into a boolean feature, such as "The path contains the token 'login.'" Although each URL does not have many of these features, the number of these sparse, boolean features in the dataset increases the overall size of the feature space significantly. When combined with similar boolean features regarding the hosting and page content described in Section 3.2.4, the total number of features seen in one month can exceed one million. We rely on feature selection methods built into our machine learning framework to incorporate only the most useful of these features into our classification models [5].

The URL Feature Extractor also collects URL metadata, including PageRank, from Google proprietary infrastructure and constructs corresponding features [26]. We also use a domain reputation score computed by the Gmail antispam system as a feature. This score is roughly the percentage of emails from a domain which are not spam. Domains that send lots of non-spam email earn high reputation scores and are less likely to host phishing sites. Taylor

describes the exact method for calculating these reputation scores [32].

3.2.3 Fetching Page Content. After the URL Feature Extractor analyzes the URL, the Content Fetcher process crawls the page and gathers its hosting information. First, the Content Fetcher resolves the host and records the returned IPs, nameservers, and nameserver IPs. It also geolocates these IPs, recording the city, region, and country. Next, the Content Fetcher sends the URL to a pool of headless web browsers to render the page content. Rendering the page in a browser ensures that we mimic the environment that the user would experience as much as possible. After the browser renders the page, the Content Fetcher receives and records the page HTML, as well as all iframe, image, and javascript content embedded in the page.

The Content Fetcher rate limits fetches to each website as an extra safeguard against generating a high volume of traffic to popular sites. Based on the rate of recent fetches to the requested domain, the Content Fetcher may defer the task until later or drop the task to avoid creating a large backlog of fetches.

3.2.4 Hosting and Page Feature Extraction. While the URL of a phishing page may be manipulated by a phisher, the same is not true for the page's hosting information. The DNS entries for a phishing page must be accurate, otherwise potential victims cannot view the page. While the hosting information alone cannot prove conclusively that a page is phishing, this data can establish whether a page is hosted like other phishing sites. The Page Feature Extractor uses the page hosting data gathered by the Content Fetcher to generate features for this purpose.

To start, the Page Feature Extractor constructs features out of the autonomous system numbers to which the page's hosts and nameservers correspond using the routing data from the University of Oregon Route Views project [1]. Autonomous system numbers give a more accurate picture of IP address association than simply looking at IP address subnets. Also, they present a smaller range of features for the machine learning algorithms. The feature extractor also computes features based on the geolocations of the page's hosts and nameservers, taking into account their city, region, and country.

Even with a legitimate looking URL and reputable hosting, we can still tell when a page is phishing by looking at the page contents. To this end, the Page Feature Extractor also extracts features from the HTML collected by the Content Fetcher.

One of these features is the extent to which pages link to other domains in terms of both HTML hyperlinks and images. Links and images on phishing pages often point directly to the target website. For the links, they need to function correctly for the phishing page to look legitimate. In the case of the images, the phishers do not need to copy all of the target's images to their short-lived phishing pages if they link to the appropriate target images directly. These features are similar to ones used by the classifier constructed in Ludl et al. [20].

While we could construct features out of every term appearing in the text of a page, this many features per page would overburden our machine learning software. Instead, the feature extractor only makes features of the terms with the highest term frequency-inverse document frequency (TF-IDF) values [28]. The TF-IDF value of a term on a page is the frequency of the term in the given page (term frequency,) divided by the log of the frequency of the term in all pages (document frequency.) In this case, the document frequency is calculated based on terms found on pages in the same language as the evaluated page in the Google search index. Phishing pages often use terms from their targets prominently, and their highest valued TF-IDF terms reflect this. Non-phishing pages do not contain these target-related terms often enough to give them a high TF-IDF value. Zhang et al. also used terms with the highest TF-IDF values as a key component of their analysis [35].

Finally, the Page Feature Extractor constructs a feature indicating whether the page has a password field. Most standard phishing sites use a form with a password field to steal a viewer's login credentials, though nontraditional phishing pages may request that the viewer download a virus or key logger instead. Non-phishing pages that have password fields are usually easy to distinguish on the basis of their other features.

3.2.5 Page Classification. With all of the features collected, the classifier process scores the page on a scale of 0.0 being not at all phishing to 1.0 being definitely phishing. The score translates to the computed probability that the page is phishing. A separate training process, described in Section 3.3, computes models which the classifier reloads daily and uses to calculate these scores.

To compute this score, the classifier first creates a set of feature values for the page. For boolean features, "true" becomes a value of 1.0, and "false" becomes a value of 0.0. All continuous features are scaled to be between 0.0 and 1.0. Any features absent from the page are assumed to have a value of 0.0. To compute the score for the page in log odds, the classifier combines these values using a logistic regression defined by the classification model [5]. Finally, the classifier transforms the score in log odds to the final output score:

$$score = \frac{e^{logodds}}{1 + e^{logodds}}$$

If the score is greater than 0.5, the page is more likely

than not phishing, and the classifier marks it as such within our review system. Before the classifier automatically blacklists the page, it checks to make sure that the page does not have a high PageRank [26]. If it does, the page is popular, unlikely to be phishing, and therefore a possible classification error. Popular pages require a final manual review before being added to the blacklist to prevent high impact false positive classifications.

3.2.6 Aggregation and Serving. The final process in the workflow, the Blacklist Aggregator, prepares the blacklist to be served to clients. The Blacklist Aggregator reads in the set of blacklisted URLs and transforms them into host suffix/path prefix expressions required by the SafeBrowsing protocol specification [7].

The Blacklist Aggregator next applies a broadening algorithm to the expressions, to reduce the number of patterns on our blacklist. If a number of blacklisted URLs match some broader expression, the Blacklist Aggregator may blacklist the broader expression instead. Phishing attacks often use URLs that vary only by a portion of the host or path, so broadening reduces these URLs to a single blacklist entry per attack. This reduction minimizes the size of the list served to our clients. It also improves our blacklist coverage by blocking URLs that are part of a phishing attack but were missed by our classification system. To avoid acting too aggressively, the broadening algorithm avoids patterns that match sites with a high PageRank [26]. For example, this safeguard prevents it from blacklisting the top-level URL of a hosting site even though the site may contain several phishing pages.

Once pattern broadening finishes, the Blacklist Aggregator removes any overlapping expressions and assembles the remaining expressions into the binary format served to clients using the SafeBrowsing protocol.

3.3 Training Process

Training the classifier is an offline process that runs once per day to pick up new phishing trends. As a training dataset, we use a sample of roughly ten million URLs analyzed by the classification workflow over the past three months along with the features obtained at the time. We include each URL only once in our training set, even if we received it both from a user report and from a Gmail spam message. We also limit the number of URLs from any single domain to 150 per week to prevent a single domain from having too much weight in determining our classification models. We find this limit prevents concentrated phishing attacks from dominating our training data while still allowing us to include a representative sample of large websites with many legitimate pages. We use our published blacklist as a noisy source of true phishing classifications. We assume that pages not in our blacklist are not phishing. As we discover more phishing pages or identify errors in our published blacklist, we update our training data for subsequent training runs.

When we train classification models, we build six separate models: five cross-validation models, each leaving out a different part of the training data, and a candidate model trained on all of the data. Each of the cross-validation models excludes a different set of URLs according to each URL's first entry time in our database. The first model excludes the earliest data, the last model excludes the latest, etc. This sharding scheme aims to have one of the validation models entirely exclude each individual phishing attack, possibly consisting of a large number of similar URLs. This separates attacks quite well, since phishing attacks rarely last more than a few days while our shards each span almost three weeks [2], [30]. Without this type of separation, each cross-validation model would train on samples of every attack, leading us to believe that the candidate would perform better than it would in reality. We train the models using a proprietary, but general purpose, implementation of the online gradient descent logistic regression learning algorithm, which we run over the entire dataset during training [5]. This implementation examines blocks of the training data serially to find potentially useful features to include in the classification model. Features which do not contribute to the model are simply omitted. This system can handle millions of training instances, each with dozens of features and a sparse feature space containing over a million unique features. Since we use an online learning algorithm, training on larger datasets means takes longer but does not require significantly more computing resources.

We chose to use this proprietary machine learning system because it is convenient to operate in our computing environment, not because it performs unusually well. Machine learning systems which can handle noisy training data and a very large feature set should perform similarly to our proprietary system. For example, we experimented with using a classifier based on random forests [6]. Since random forests do not dynamically select features, we preselected 3000 features from the training set using information gain and document frequency feature selection algorithms [15]. With these features, we found that a random forest classifier consisting of 100 trees each trained on a random quarter of the training set performed similarly to our proprietary classifier. Demonstrating another approach, Ma et al. showed that generic online learning algorithms performed well on a dataset similar to ours with two million instances and hundreds of thousands of sparse features [22]. These findings suggest that both random forests and other online learning implementations would adequately substitute for our proprietary learning system.

Once the training finishes, we test each cross-validation model against the training data left out during its construction. Because the last model excluded the latest training data, it only trained on older data. Therefore, the performance of this model provides an estimate of how the candidate model will perform on new, live data. We also test the candidate model on all of the training data to make sure that the model contains no obvious flaws.

For our target model performance, we require that models exhibit better than 90% precision and 90% recall. If the cross-validation models average above this level, the cross-validation fold tested on the last part of training data is above this level, and the candidate model performs above this level on the full training set, then the candidate model is pushed to the live classifiers.

3.4 Potential Adversarial Attacks

Phishers can attempt to bypass our system in a few ways. The following attacks represent real vulnerabilities in our classifier, which we expect phishers to attempt to exploit. However, we believe that the possible attacks on our system are either limited or expensive.

Phishers can try to bypass our system by disguising their pages as non-phishing pages. However, in order for phishing pages to operate correctly, they must both appear visually like a third party and request that the victim perform some action. Both of these characteristics are possible to identify automatically. To make these features less suspicious, the phishers could try to trick our classifier by giving their page a high PageRank. However, PageRank is designed to make manipulation of the ranking scores difficult [26]. Consequently, altering the PageRank of a phishing page requires a significant investment, which reduces the potential profit of the phisher. Alternately, the phishers could try to post their content on a reputable host which already has a high PageRank. A phisher could accomplish this by exploiting a site with a high PageRank and using it to display their phishing page. Administrators of sites with high PageRanks typically remove malicious pages under their control promptly, though, limiting the potential audience and profitability of phishing pages posted to their sites.

Alternatively, the phishers could attempt to manipulate our classification models. If they could pollute our training set, our classifier would make more user-visible mistakes which would reduce the value of our blacklist. Specifically, phishers could try to expand the set of pages deemed non-phishing by introducing non-phishing pages with some phishing attributes into our training set, similar to the attack described by Barreno et al. [4]. Theoretically, they could alter the training set enough so that trained classifiers would mistake phishing pages for non-phishing pages. To combat this, we use a large training set collected over a long period of time and limit the number of times a given domain can appear in the training set as described in Section 3.3. Since we weight all data equally, phishers do not have an opportunity to target higher weighted segments of our training set. Despite our safeguards, we still see phishing campaigns that make up a significant portion of our training data (see Section 4.1.) However, setting up similar campaigns of non-phishing pages to possibly influence our classification models would be expensive, again reducing the phisher's gains.

Phishers could attempt to use the rate limiting described in Section 3.2.3 to prevent our system from crawling their pages by putting many phishing pages on one domain. However, if they put enough different pages on the same domain to slow down our fetches, our URL aggregation algorithm will block their entire domain (see Section 3.2.6.)

Finally, phishers can try to bypass our system by hiding their phishing pages from us. They could avoid sending phishing emails to Gmail users to avoid our automatic URL collection system. However, many users forward email from other accounts into Gmail, so this tactic would not always work. Instead, phishers could exploit our privacy safeguards, described in Section 3.2.1, to keep their URLs out of our system. Phishers could also try to serve our crawlers non-phishing content different from what they serve their intended victims to escape automatic phishing classifications.

Even if a phishing page defeats our automatic classification system, the more people it reaches, the higher the likelihood that a victim will report the page to us. This allows us to correctly classify the phishing page manually, regardless of how the page appears to our classification system. Because of this, defeating our automatic classifier does not mean that the page will never appear on our blacklist. Instead, phishers can only hope to increase the length of time before their pages are blacklisted.

4 Evaluation

In our evaluation, we intend to demonstrate the overall quality of our classification system and the value of the features we examine. To evaluate the quality of the classification system, we first train a full classification model along with cross-validation models as described in Section 3.3. Next, we evaluate the models' performance on both their training data and on a validation dataset. We intend that these tests closely emulate the environment in which our classifier normally operates and therefore provide a realistic evaluation of the performance of our system. To examine the usefulness of the features, we compare the feature values in phishing pages and in non-phishing pages. We use previously collected classification data to determine feature values, and we use our published blacklist to determine

which pages are phishing. While using our published blacklist as ground truth in these evaluations allows us to examine a large quantity of data, errors in our blacklist introduce some noise into the analysis. Based on the relatively small number of misclassifications reported to us by our users, we do not expect this noise to impact our analysis of the features significantly.

4.1 Evaluation Dataset

In our evaluation, we use two sets of data. The first set, a copy of a training set used by our system, contains data collected between April 16, 2009 and July 14, 2009 with labels from July 15, 2009. We use this set to examine our selected features and train our evaluation models. We use the second set, collected during the first two weeks of August, 2009, as a validation dataset. Besides the different size, this second set differs from the first set in that we updated its labels with our blacklist data from August 24, ten days after the end of data collection period. Consequently, the second set contains fewer labeling errors, since we had more than a week to respond to reported errors and correct any mislabelings. This dataset allows us to show the performance of the classification model with as little noise as possible. Both datasets exclude URLs dropped from the system prior to final classification, since our data regarding these URLs may be incomplete. For example, our database does not include page or hosting features for URLs dropped due to rate limiting (see Section 3.2.3.) Also, both datasets limit the number of URLs from any one domain to 150 per week, just like we do during our training process, and do not contain duplicate URLs. Table 1 details the statistics of these two datasets.

Note that these datasets do not contain a random subset of web pages. As described in Section 3.2.1, the only URLs analyzed are either submitted as a potential phishing page or collected from a spam email. From these, URL feature extraction filters out approximately half of the pages as either invalid or whitelisted (see Section 3.2.2.) The majority of these filtered pages are URLs for high profile domains added to spam emails to enhance their appearance of legitimacy. The remaining pages in the training sets are already quite suspect. However, most of these pages are not phishing. Since our datasets contain roughly one percent phishing web pages, our classifier must demonstrate a false positive rate of 0.1% to achieve precision of 90%.

Despite skipping many URLs from overrepresented domains through rate limiting and capping the number of URLs coming from any single domain in our datasets, one phishing campaign is overweighted in our data. Roughly 51,000 of the phishing pages in the April–July dataset, hosted across hundreds of domains, match a particular bank phishing template.

	Apr 16–Jul 14	Aug 1–14
Total URLs Received	446,152,060	74,816,740
User Submitted URLs	75,048	14,490
Gmail Spam URLs	446,093,814	74,805,549
URLs Skipped: Invalid/Whitelisted	226,636,463	44,806,767
URLs Skipped: Rate Limited	202,727,905	27,228,220
URLs Skipped: Error (e.g., Host Not Found)	3,606,765	521,093
URLs with Complete Data	13,180,927	2,260,660
Dataset URLs (limited # URLs/domain)	9,388,395	1,516,076
Phishing Dataset URLs	103,684 (1.1%)	16,967 (1.1%)
Dataset URLs From User Submission	46,682	9,830
Dataset URLs From Gmail Spam	9,348,783	1,507,682

Table 1. Dataset statistics. Note that URLs may come from both user submissions and Gmail spam. Refer to Section 3.2.2 for more information on skipping invalid and whitelisted URLs and Section 3.2.3 for more on skipping rate limited URLs.

All of their URLs are in the form http://www.bankofamerica.com.srv_<random>.<random>.<random top-level domain>/customerservice/securedirectory/cform.do/cform.php, and all their page contents are identical. Because of the large number of randomized domains, we suspect that a botnet using domain flux, like the Torpig botnet analyzed by Stone-Gross et al., hosts this phishing campaign [31]. While the weighting of this attack in our dataset is not ideal for our evaluation, it accurately reflects the widespread nature of this particular phishing campaign. We highlight the situations where this attack impacts the statistics we present in the following analysis.

4.2 Evaluation of Features

In this section, we analyze the value of the assorted features in our feature set. For a summary of the feature set we described in Section 3.2, see Table 2. Note that since phishing pages make up 1.1% of our dataset, if 1.1% of the URLs with a feature are phishing, then that feature is completely uncorrelated with phishing. If the feature appears more often than that on phishing pages, then it is correlated with phishing.

For our URL features, we find that IP address hosting is several times more common in phishing pages than in non-phishing pages and that a very small percentage of non-

Feature	Type	Visualization
URL Features (Section 3.2.2)		
IP Address for Hostname	boolean	Table 3
Long Hostname	boolean	Table 3
Tokens from URL	booleans	Table 4
PageRank	float	Figure 1
Gmail Reputation	float	Figure 2
Hosting Features (Section 3.2.4)		
Autonomous System Numbers	booleans	Table 5
Geolocations	booleans	Table 6
Page Features (Section 3.2.4)		
Password Field	boolean	Table 7
Top TF-IDF Terms	booleans	Table 8
External Linking Frequency	float	Figure 3

Table 2. Summary of feature set. Features of type "boolean" consist of a single boolean value. The type "booleans" indicates a sparse set of binary features which are always "true" if present. For example, we construct one such feature for each token in the URL, and we construct one feature for each autonomous system number on which the URL is hosted.

phishing sites have unusually long host names (see Table 3.) Some tokens in the URL, like "signin" in the host, are far more common in phishing pages, whereas others, like "album" in the path, are more common in non-phishing pages (see Table 4.) Confirming the findings of Garera et al., we also find that no phishing pages have high PageRank scores, as shown in Figure 1 [16]. We find that the Gmail reputation scores, shown in Figure 2, follow a similar pattern.

	# Phishing URLs	# Non-phishing URLs	% Phishing
IP Address Host	2,341	33,208	6.6%
# Host segments before domain > 3	62,383	934	98.5%

Table 3. Statistics for URL characteristics for data collected Apr 16–Jul 14. Note that URLs belonging to the overrepresented phishing template have a long hostname. "% Phishing" is the fraction of URLs that are phishing out of all URLs with the feature.

Looking at our hosting features, we find that some autonomous systems, especially those belonging to residential ISPs, have very high levels of phishing, as shown in Table 5. Additionally, Table 6 shows that some areas are highly associated with phishing activity. While the United States hosts the largest number of phishing pages, a large

	# Phishing URLs	# Non-phishing URLs	% Phishing
Host contains "www"	71,034	4,872,888	1.4%
Host contains "signin"	148	3	98.0%
Host contains "blog"	25	33,824	0.1%
Path contains "custom-erservice"	54,749	349	99.4%
Path contains "com"	8,971	46,416	16.2%
Path contains "album"	31	83,184	0.04%

Table 4. Statistics for selected URL tokens for data collected Apr 16–Jul 14. Note that the URLs from the overweighted phishing template have "www" in the host and "custom-erservice" in the path.

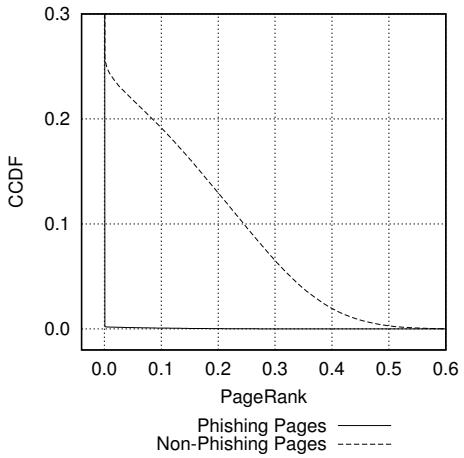

Figure 1. Complimentary cumulative distribution function (CCDF) of PageRank for phishing and non-phishing pages for data collected Apr 16–Jul 14.

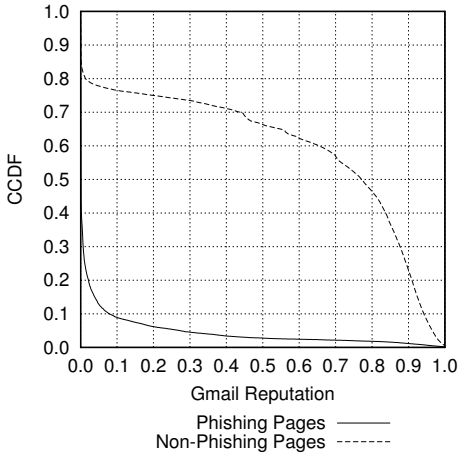

Figure 2. CCDF of Gmail reputation score for phishing and non-phishing pages for data collected Apr 16–Jul 14.

fraction of pages hosted in some Eastern European countries are phishing.

Turning to features from page HTML, Table 7 shows that a high proportion of phishing pages have a password field. The number of phishing pages without a password field indicates that nontraditional phishing attacks are becoming somewhat more common. When looking at the terms with the highest TF-IDF scores for a page, some terms are highly indicative of phishing, like "online banking" and "pin," whereas others, like "islands," are relatively safe (see Table 8.) Also, Figure 3 shows that relatively few legitimate pages link heavily to other domains.

4.3 Classifier Performance

To evaluate the performance of our classification system, we examine the classifier trained on data we collected between April 16 and July 14, described in Section 4.1. As outlined in Section 3.3, we train five cross-validation models, leaving out a different portion of the data set each time, and a candidate model, leaving out nothing. Figure 4 shows the performance of the classifier in a precision-recall curve for the average cross-validation performance, the final cross-validation fold by itself, the candidate model trained and tested on the whole dataset, and the candidate model tested on data collected between August 1, 2009 and August 14, 2009. To generate these curves, we first calculate the classification scores using the trained models. Next, as we vary a threshold from 0.01 to 0.99, we determine the precision and recall obtained by classifying pages with a score greater than that threshold as phishing pages. We add a point marking a threshold score of 0.5, the point at which

ASN	# Phishing URLs	# Non-phishing URLs	% Phishing
6739 (Cableuropa - ONO)	42,283	883	98.0%
8708 (RCS & RDS)	35,537	7,777	82.0%
6830 (UPC Broadband)	34,921	678	98.1%
7132 (AT&T Internet Services)	29,497	4,829	85.9%
9116 (Golden Lines Main)	26,645	5,280	83.5%
19262 (Verizon Internet Services)	26,174	1,855	93.4%
16338 (Cableuropa - ONO)	22,323	6,260	78.1%
9141 (UPC Poland)	22,219	19	99.9%
5089 (NTL Group Limited)	18,213	1,146	94.1%
5617 (Polish Telecom (Commercial))	18,054	3,437	84.0%

Table 5. ASNs hosting the most phishing pages for data collected Apr 16–Jul 14. Many phishing pages, including those from the widespread phishing template, are hosted on many different ASNs. Each URL may be hosted on multiple ASNs, which is why the sum of the phishing pages hosted by these ASNs exceeds the total number of phishing pages in the dataset.

Country	# Phishing URLs	# Non-phishing URLs	% Phishing
United States	76,124	4,675,812	1.6%
Romania	54,638	76,711	41.6%
Spain	53,703	179,682	23.0%
Poland	45,611	93,734	32.7%
Hungary	35,968	60,598	37.2%
Russia	34,391	327,142	9.5%
Israel	28,003	29,568	48.6%
United Kingdom	25,148	307,708	7.6%
Mexico	24,478	10,432	70.1%
Netherlands	22,025	614,726	3.5%

Table 6. Countries hosting the most phishing pages for data collected Apr 16–Jul 14. Each URL may be hosted in multiple countries.

	# Phishing URLs	# Non-phishing URLs	% Phishing
Password Field	90,072	1,620,184	5.3%
No Password Field	13,612	7,664,527	0.2%
% With Password Field	86.9%	17.5%	

Table 7. Password field statistics for data collected Apr 16–Jul 14.

Term among Top TF-IDF Scores	# Phishing URLs	# Non-phishing URLs	% Phishing
online banking	55,326	587	99.0%
pin	3,354	1,256	72.8%
paypal	6,683	5,509	54.8%
online	1,353	22,100	5.8%
email	259	52,437	0.5%
islands	223	107,166	0.2%

Table 8. Statistics for pages with selected terms among the highest TF-IDF scored terms in data collected Apr 16–Jul 14. Note that "online banking" was one of the top scored terms for pages from the overrepresented phishing template.

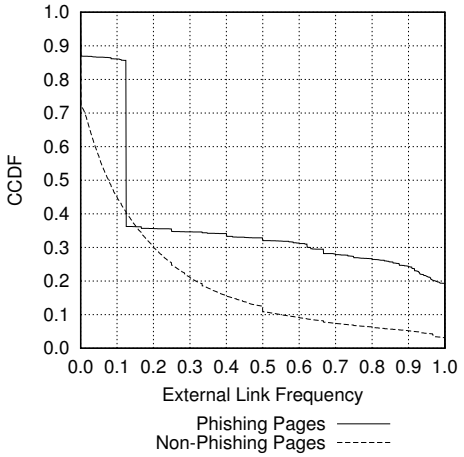

Figure 3. CCDF of external link frequency breakdown for phishing and non-phishing pages for data collected Apr 16–Jul 14. Note that the overrepresented phishing template has 12.5% of its links pointing to external targets, accounting for the large jump in the CCDF for phishing pages.

our classifier exhibits high precision while not sacrificing much recall, to each of these curves. Table 9 shows exact statistics for the classifier in cross-validation and on the August dataset for this threshold.

The precision-recall curve of the candidate classifier on the August dataset roughly matches the curve generated during cross-validation. This shows that even a month later, our classification model remains highly predictive. However, the point corresponding to a threshold of 0.5 shows higher precision and lower recall relative to the similar statistics from cross-validation. The improved labeling of the August dataset explains this difference. Most of the corrections to the phishing labels mark previously unidentified phishing pages properly. This raises the precision of the classifier, since some of the original classification false positives become true positives. This also lowers the recall, as some of the original true negatives become false negatives.

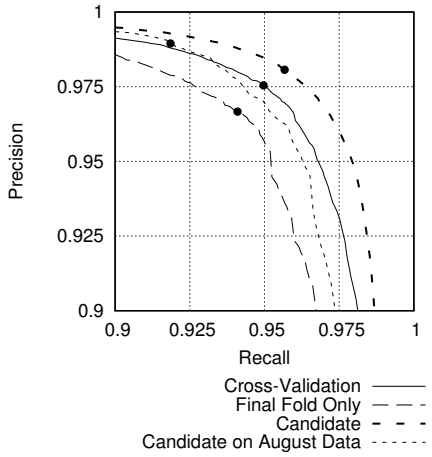

Figure 4. Precision-recall curves for classifier trained on data collected Apr 16–Jul 14. Note that the graph is zoomed in to better show the differences between the curves.

	Cross-Validation	Candidate on August Data
True Positives	98,467	15,585
False Positives	2,479	166
False Negatives	5,217	1,382
True Negatives	9,282,232	1,498,943
True Positive Rate/Recall	0.9497	0.9185
Precision	0.9754	0.9895
False Positive Rate	0.0003	0.0001

Table 9. Statistics for classifier trained on data collected Apr 16–Jul 14 using a threshold of 0.5.

Figure 5 shows how our classifier scores phishing and non-phishing pages during cross-validation. The vast majority of the classification scores are near the extremes. Over 99.9% of non-phishing pages score below 0.1, and 90% of phishing pages score over 0.9, indicating our feature space provides sufficient separation between phishing and non-phishing pages to allow for correct classification in most situations.

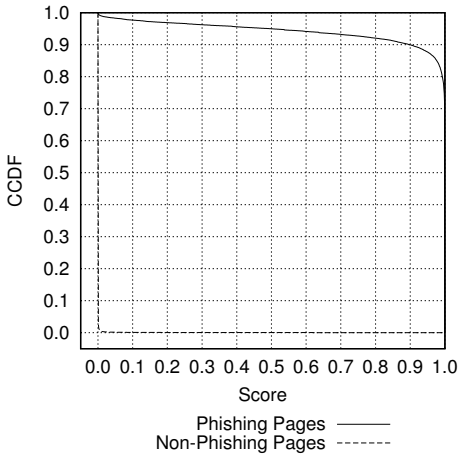

Figure 5. CCDF of classification score for phishing and non-phishing pages during cross-validation on data collected Apr 16–Jul 14.

The work of Sheng et al. illustrates the ultimate impact of our automatic classification system [30]. Between their two experiments, we began using our classifier to automatically update our published blacklist. Their results suggest that the coverage of our blacklist improved by 150% during the first two hours of a phishing attack due to our automatic classification system. By the end of this time period, our blacklist included 97% of the phishing pages examined. These findings highlight how much automatic classification has added to the comprehensiveness and timeliness of our blacklist.

4.4 Per-URL Classification Latency

Figure 6 shows the distribution of latency between when our classification workflow received URLs and when the classifier wrote out scores for data we collected between mid-April and mid-July. The workflow processed the majority of tasks quickly with a median of 76 seconds. However, a few tasks took up to a few hours to process, a delay attributed to the throttling of page fetches described in Section 3.2.3. Nonetheless, this represents a considerable improvement over PhishTank, which has not reported a median verification time under 10 hours for any month in the first half of 2009 [25].

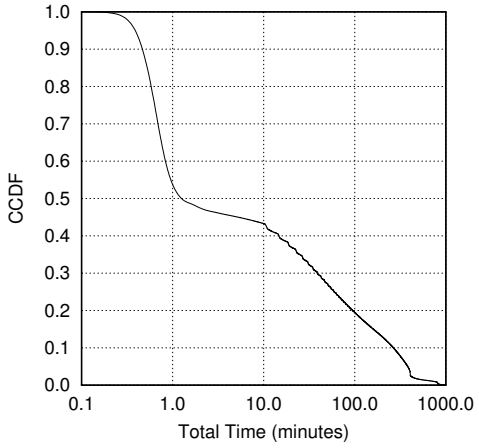

Figure 6. CCDF for end-to-end classification time for classifications made Apr 16–Jul 14.

5 Conclusion

In this paper, we describe our large-scale system for automatically classifying phishing pages which maintains a false positive rate below 0.1%. Our classification system examines millions of potential phishing pages daily in a fraction of the time of a manual review process. By automatically updating our blacklist with our classifier, we minimize the amount of time that phishing pages can remain active before we protect our users from them.

Even with a perfect classifier and a robust system, we recognize that our blacklist approach keeps us perpetually a step behind the phishers. We can only identify a phishing page after it has been published and visible to Internet users for some time. However, we believe that if we can provide a blacklist complete enough and quickly enough, we can force phishers to operate at a loss and abandon this type of Internet crime.

6 Acknowledgements

We would like to thank Garrett Casto, Noe Lutz, Eric Grosse, Bradley Taylor, Ian Fette, and Moheeb Abu Rajab. We give special thanks to Niels Provos for his guidance and insightful comments.

References

[1] Advanced Network Technology Center, University of Oregon. University of Oregon Route Views Project. http://www.routeviews.org/, Jan. 2005.

[2] Anti-Phishing Working Group. Phishing Activity Trends Report for the Month of January, 2008. http://www.antiphishing.org/reports/apwg_report_jan_2008.pdf, 2008.

[3] Anti-Phishing Working Group. Phishing Activity Trends Report, H2/2008. http://www.antiphishing.org/reports/apwg_report_H2_2008.pdf, Mar. 2009.

[4] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, Mar. 2006.

[5] J. Bem, G. Harik, J. Levenberg, N. Shazeer, and S. Tong. Large scale machine learning and methods. US Patent 7222127, 2007.

[6] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[7] G. Casto, O. Fisher, R. Moll, M. Nazif, and D. Born. Client specification for the Google Safe Browsing v2.1 protocol. http://code.google.com/p/google-safe-browsing/wiki/Protocolv2Spec, 2007.

[8] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pages 205–218, Nov. 2006.

[9] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. Mitchell. Client-side Defense against Web-Based Identity Theft. In *NDSS '04: Proceedings of the 11th Annual Network and Distributed System Security Symposium*, Feb. 2004.

[10] R. Dhamija, J. D. Tygar, and M. Hearst. Why Phishing Works. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590, Apr. 2006.

[11] J. S. Downs, M. B. Holbrook, and L. F. Cranor. Decision Strategies and Susceptibility to Phishing. In *SOUPS '06: Proceedings of the second symposium on Usable privacy and security*, pages 79–90, July 2006.

[12] S. Egelman, L. F. Cranor, and J. Hong. You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1065–1074, Apr. 2008.

[13] I. Fette, N. Sadeh, and A. Tomasic. Learning to Detect Phishing Emails. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 649–656, May 2007.

[14] B. J. Fogg, J. Marshall, O. Laraki, A. Osipovich, C. Varma, N. Fang, J. Paul, A. Rangnekar, J. Shon, P. Swani, and M. Treinen. What Makes Web Sites Credible?: A Report on a Large Quantitative Study. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 61–68, March-April 2001.

[15] G. Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[16] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A Framework for Detection and Measurement of Phishing Attacks. In *WORM '07: Proceedings of the 2007 ACM workshop on Recurring malcode*, pages 1–8, Nov. 2007.

[17] Gartner, Inc. Gartner Says Number of Phishing Attacks on U.S. Consumers Increased 40 Percent in 2008. http://www.gartner.com/it/page.jsp?id=936913, Apr. 2009.

[18] Google. Google Safe Browsing API. http://code.google.com/apis/safebrowsing/, 2009.

[19] C. Herley and D. Florencio. A Profitless Endeavor: Phishing as Tragedy of the Commons. In *NSPW '08: Proceedings of the 2008 workshop on New security paradigms*, Sept. 2008.

[20] C. Ludl, S. McAllister, E. Kirda, and C. Kruegel. On the Effectiveness of Techniques to Detect Phishing Sites. In *DIMVA '07: Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 20–39, July 2007.

[21] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254, June–July 2009.

[22] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying Suspicious URLs: An Application of Large-Scale Online Learning. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 681–688, June 2009.

[23] Microsoft. Internet Explorer 8 Security: SmartScreen Filter. http://www.microsoft.com/security/filters/smartscreen.aspx, 2009.

[24] Netcraft. Netcraft Toolbar. http://toolbar.netcraft.com/, 2004.

[25] OpenDNS. Phishtank. http://www.phishtank.com/, 2009.

[26] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[27] 'RSnake'. Phishing Social Networking Sites. http://ha.ckers.org/blog/20070508/, May 2007.

[28] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1983.

[29] F. Schneider, N. Provos, R. Moll, M. Chew, and B. Rakowski. Phishing Protection Design Documentation. http://wiki.mozilla.org/Phishing_Protection:_Design_Documentation, 2006.

[30] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang. An Empirical Analysis of Phishing Blacklists. In *CEAS 2009: Sixth Conference on Email and Anti-Spam*, July 2009.

[31] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, Nov. 2009.

[32] B. Taylor. Sender Reputation in a Large Webmail Service. In *CEAS 2006: Third Conference on Email and Anti-Spam*, July 2006.

[33] M. Wu, R. C. Miller, and S. L. Garfinkel. Do Security Toolbars Actually Prevent Phishing Attacks? In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610, Apr. 2006.

[34] Y. Zhang, S. Egelman, L. F. Cranor, and J. I. Hong. Phinding Phish: Evaluating Anti-Phishing Tools. In *NDSS '07: Proceedings of the 14th Annual Network and Distributed System Security Symposium*, February–March 2007.

[35] Y. Zhang, J. I. Hong, and L. F. Cranor. CANTINA: A Content-Based Approach to Detecting Phishing Web Sites. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 639–648, May 2007.