Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs

Justin Ma, Lawrence K. Saul, Stefan Savage, Geoffrey M. Voelker
Department of Computer Science and Engineering
University of California, San Diego

ABSTRACT

Malicious Web sites are a cornerstone of Internet criminal activities. As a result, there has been broad interest in developing systems to prevent the end user from visiting such sites. In this paper, we describe an approach to this problem based on automated URL classification, using statistical methods to discover the tell-tale lexical and host-based properties of malicious Web site URLs. These methods are able to learn highly predictive models by extracting and automatically analyzing tens of thousands of features potentially indicative of suspicious URLs. The resulting classifiers obtain 95–99% accuracy, detecting large numbers of malicious Web sites from their URLs, with only modest false positives.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; I.5.1 [**Pattern Recognition**]: Models—*Statistical*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Feature evaluation and selection*

General Terms

Algorithms, Security

Keywords

supervised learning, malicious Web sites, L1-regularization

1. INTRODUCTION

The Web has become a platform for supporting a wide range of criminal enterprises such as spam-advertised commerce (e.g., counterfeit watches or pharmaceuticals), financial fraud (e.g., via phishing or 419-type scams) and as a vector for propagating malware (e.g., so-called "drive-by downloads"). Although the precise commercial motivations behind these schemes may differ, the common thread among them is the requirement that unsuspecting users visit their sites. These visits can be driven by email, Web search results or links from other Web pages, but all require the user to take some action, such as clicking, that specifies the desired Uniform Resource Locator (URL).

KDD'09, June 28–July 1, 2009, Paris, France.

Clearly, if one could inform users *beforehand* that a particular URL was dangerous to visit, much of this problem could be alleviated. To this end, the security community has responded by developing blacklisting services — encapsulated in toolbars, appliances and search engines — that provide precisely this feedback. These blacklists are in turn constructed by a range of techniques including manual reporting, honeypots, and Web crawlers combined with site analysis heuristics. Inevitably, many malicious sites are not blacklisted — either because they are too new, were never evaluated, or were evaluated incorrectly (e.g., due to "cloaking"). To address this problem, some client-side systems analyze the content or behavior of a Web site *as* it is visited. But, in addition to run-time overhead, these approaches can expose the user to the very browser-based vulnerabilities that we seek to avoid.

In this paper, we focus on a complementary part of the design space: *lightweight* URL classification — that is, classifying the reputation of a Web site *entirely* based on the inbound URL. The motivation is to provide inherently better coverage than blacklisting-based approaches (e.g., correctly predicting the status of new sites) while avoiding the client-side overhead and risk of approaches that analyze Web content on demand.

In particular, we explore the use of statistical methods from machine learning for classifying site reputation based on the relationship between URLs and the lexical and host-based features that characterize them. Unlike previous work, we show that these methods are able to sift through tens of thousands of features derived from publicly available data sources, and can identify which URL components and meta-data are important without requiring heavy domain expertise. Indeed, our system automatically selects many of the same features identified by domain experts as being typical of "malicious" Web sites. But more importantly, our system selects new, non-obvious features that are highly predictive and yield substantial performance improvements. We evaluate this approach across 20,000 to 30,000 URLs drawn from different sources, and show that it can obtain an overall prediction accuracy of 95–99%, detecting a large fraction of malicious websites while maintaining a very low false positive rate.

We begin the remainder of this paper with an explanation of the formal basis for our approach.

2. APPROACH

In this section, we provide a detailed discussion of our approach to classifying site reputation. We begin with an overview of the classification problem, followed by a discussion of the features we extract, and finally the set of machine learning classifiers we use for the study.

For our purposes, we treat URL reputation as a binary classification problem where positive examples are malicious URLs and

negative examples are benign URLs. This learning-based approach to the problem can succeed if the distribution of feature values for malicious examples is different from benign examples, the training set shares the same feature distribution as the testing set, and the ground truth labels for the URLs are correct.

Significantly, we classify sites based only on the relationship between URLs and the lexical and host-based features that characterize them, and we do not consider two other kinds of potentially useful sources of information for features: the URL's page content, and the context of the URL (e.g., the page or email in which the URL is embedded). Although this information has the potential to improve classification accuracy, we exclude it for a variety of reasons. First, avoiding downloading page content is strictly safer for users. Second, classifying a URL with a trained model is a lightweight operation compared to first downloading the page and then using its contents for classification. Third, focusing on URL features makes the classifier applicable to any context in which URLs are found (Web pages, email, chat, calendars, games, etc.), rather than dependent on a particular application setting. Finally, reliably obtaining the malicious version of a page for both training and testing can become a difficult practical issue. Malicious sites have demonstrated the ability to "cloak" the content of their Web pages, i.e., serving different content to different clients [20]. For example, a malicious server may send benign versions of a page to honeypot IP addresses that belong to security practitioners, but send malicious versions to other clients. Nonetheless, we show in Section 4 that classifying on lexical features of the URL and features about the host are sufficient for highly accurate prediction.

2.1 Features

We categorize the features that we gather for URLs as being either lexical or host-based.

Lexical features: The justification for using lexical features is that URLs to malicious sites tend to "look different" in the eyes of the users who see them. Hence, including lexical features allows us to methodically capture this property for classification purposes, and perhaps infer patterns in malicious URLs that we would otherwise miss through ad-hoc inspection.

For the purpose of this discussion, we want to distinguish the two parts of a URL: the hostname and the path. As an example, with the URL `www.geocities.com/usr/index.html`, the hostname portion is `www.geocities.com` and the path portion is `usr/index.html`.

Lexical features are the textual properties of the URL itself (not the content of the page it references). We use a combination of features suggested by the studies of McGrath and Gupta [16] and Kolari et al. [13]. These properties include the length of the hostname, the length of the entire URL, as well as the number of dots in the URL — all of these are real-valued features. Additionally, we create a binary feature for each token in the hostname (delimited by '.') and in the path URL (strings delimited by '/', '?', '.', '=', '-' and '_'). This is also known as a "bag-of-words." Although we do not preserve the order of the tokens, we do make a distinction between tokens belonging to the hostname, the path, the top-level domain (TLD) and primary domain name (the domain name given to a registrar). More sophisticated techniques for modeling lexical features are available, such as Markov models of text. However, even with the bag-of-words representation, we can achieve very accurate classification results.

Host-based features: The reason for using host-based features is that malicious Web sites may be hosted in less reputable hosting centers, on machines that are not conventional web hosts, or through disreputable registrars. To an approximate degree, host-based features can describe "where" malicious sites are hosted, "who" own them, and "how" they are managed.

The following are properties of the hosts (there could be multiple) that are identified by the hostname part of the URL. We note some of these features overlap with lexical properties of the URL.

1. *IP address properties* — Is the IP address in a blacklist? Are the IPs of the A, MX or NS records in the same autonomous systems (ASes) or prefixes as one another? To what ASes or prefixes do they belong?
2. *WHOIS properties* — What is the date of registration, update, and expiration ? Who is the registrar? Who is the registrant? Is the WHOIS entry locked?
3. *Domain name properties* — What is the time-to-live (TTL) value for the DNS records associated with the hostname?

   Additionally, the following domain name properties are used in SpamAssassin Botnet plugin for detecting links to malicious sites in emails: Does the hostname contain "client" or "server" keywords? Is the IP address in the hostname? Is there a PTR record for the host? Does the PTR record inturn resolve one of the host's IP addresses?
4. *Geographic properties* — In which continent/country/ city does the IP address belong? What is the speed of the uplink connection (broadband, dial-up, etc)?

This list of features we use is not exhaustive, as there is always the possibility of generating or aggregating new meta-information about the URL such as popularity rankings in Netcraft, indexing in Google, etc. Nevertheless, the list is still extensive, as it represents many pieces of information about the URL and host (much of which is publicly available) that we can collect efficiently through the automated crawling tools at our disposal.

2.2 Classification Models

We use the features described in the previous section to encode individual URLs as very high dimensional feature vectors. Most of the features are generated by the "bag-of-words" representation of the URL, registrar name, and registrant name; binary features are also used to encode all possible ASes, prefixes and geographic locales of an IP address. The resulting URL descriptors typically have tens of thousands of binary features.

The high dimensionality of these feature vectors poses certain challenges for classification. Though only a subset of the generated features may correlate with malicious Web sites, we do not know in advance which features are relevant. More generally, when there are more features than labeled examples, we enter the regime in which statistical models are most prone to overfitting.

In this section, we briefly review the models that we studied for classification, as well as their relative strengths and weaknesses. For all these models, we adopt the following notation. We use n to denote the number of labeled examples in the training set and d to denote the dimensionality of the feature space (i.e., the number of features). We use $\mathbf{x} \in \Re^d$ to denote a feature vector and x_j to denote its jth component. Finally, we use $y \in \{0, 1\}$ to denote the label of an example, with $y = 1$ for malicious sites and $y = 0$ for benign ones.

We provide further details on the individual classifiers below. In particular, for each classifier, we briefly describe its testing process (how it predicts a label from the features of URLs) and its training process (how it learns the decision rule to predict these labels).

Naive Bayes: Commonly used in spam filters, this basic model assumes that for a given label, the individual features of URLs are

distributed independently of the values of other features [5]. Letting $P(\mathbf{x}|y)$ denote the conditional probability of the feature vector given its label, the model assumes $P(\mathbf{x}|y) = \prod_{j=1}^{d} P(x_j|y)$. Then, from Bayes rule, assuming that malicious and benign Web sites occur with equal probability, we compute the posterior probability that the feature vector $\mathbf{x}$ belongs to a malicious URL as:

$$P(y=1|\mathbf{x}) = \frac{P(\mathbf{x}|y=1)}{P(\mathbf{x}|y=1) + P(\mathbf{x}|y=0)}. \quad (1)$$

Finally, the right hand side of eq. (1) can be thresholded to predict a binary label for the feature vector $\mathbf{x}$.

A Naive Bayes classifier is most easily trained by computing the conditional probabilities $P(x_j|y)$ from their maximum likelihood estimates [5]. For real-valued features, we model $P(x_j|y)$ by a Gaussian distribution whose mean and standard deviation are computed over the jth component of feature vectors in the training set with label y. For binary-valued features, we estimate $P(x_j=1|y)$ as the fraction of feature vectors in the training set with label y for which the jth component is one.

The model parameters in the Naive Bayes classifier are estimated to maximize the joint log-likelihood of URL features and labels, as opposed to the accuracy of classification. Optimizing the latter typically leads to more accurate classifiers, notwithstanding the increased risk of overfitting.

Support Vector Machine (SVM): SVMs are widely regarded as state-of-the-art models for binary classification of high dimensional data. SVMs are trained to maximize the margin of correct classification, and the resulting decision boundaries are robust to slight perturbations of the feature vectors, thereby providing a hedge against overfitting. The superior generalization abilities of SVMs have been borne out by both theoretical studies and experimental successes [23].

The decision rule in SVMs is expressed in terms of a kernel function $K(\mathbf{x}, \mathbf{x}')$ that computes the similarity between two feature vectors and non-negative coefficients $\{\alpha_i\}_{i=1}^{n}$ that indicate which training examples lie close to the decision boundary. SVMs classify new examples by computing their (signed) distance to the decision boundary. Up to a constant, this distance is given by:

$$h(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i (2y_i - 1) K(\mathbf{x}_i, \mathbf{x}), \quad (2)$$

where the sum is over all training examples. The sign of this distance indicates the side of the decision boundary on which the example lies. In practice, the value of $h(\mathbf{x})$ is thresholded to predict a binary label for the feature vector $\mathbf{x}$.

SVMs are trained by first specifying a kernel function $K(\mathbf{x}, \mathbf{x}')$ and then computing the coefficients α_i that maximize the margin of correct classification on the training set. The required optimization can be formulated as an instance of quadratic programming, a problem for which many efficient solvers have been developed [6]. In our study, we experimented with both linear and radial basis function (RBF) kernels.

Logistic Regression: This is a simple parametric model for binary classification where examples are classified based on their distance from a hyperplane decision boundary [11]. The decision rule is expressed in terms of the sigmoid function $\sigma(z) = [1 + e^{-z}]^{-1}$, which converts these distances into probabilities that feature vectors have positive or negative labels. The conditional probability that feature vector $\mathbf{x}$ has a positive label $y=1$ is the following:

$$P(y=1|\mathbf{x}) \;=\; \sigma(\mathbf{w}\cdot\mathbf{x} + b), \quad (3)$$

where the weight vector $\mathbf{w} \in \Re^d$ and scalar bias b are parameters to be estimated from training data. In practice, the right hand side of eq. (3) is thresholded to obtain a binary prediction for the label of the feature vector $\mathbf{x}$.

We trained models for logistic regression using a regularized form of maximum likelihood estimation. Specifically, we chose the weight vector $\mathbf{w}$ and bias b to maximize the objective function:

$$\mathcal{L}(\mathbf{w}, b) = \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i) - \gamma \sum_{\alpha=1}^{d} |w_\alpha|. \quad (4)$$

The first term computes the conditional log-likelihood that the model correctly labels all the examples in the training set. The second term in eq. (4) penalizes large magnitude values of the elements in the weight vector $\mathbf{w}$. Known as ℓ_1-norm regularization, this penalty not only serves as a measure against overfitting, but also encourages sparse solutions in which many elements of the weight vector are *exactly* zero. Such solutions emerge naturally in domains where the feature vectors contain a large number of irrelevant features. The relative weight of the second term in eq. (4) is determined by the regularization parameter. We selected the value of γ in our experiments by cross validation.

We included ℓ_1-regularized logistic regression for its potential advantages over Naive Bayes and SVMs in our particular domain. Unlike Naive Bayes classification, the parameters in logistic regression are estimated by optimizing an objective function that closely tracks the error rate. Unlike SVMs, ℓ_1-regularized logistic regression is especially well suited to domains with large numbers of irrelevant features. (In large numbers, such features can drown out the similarities between related examples that SVMs expect to be measured by the kernel function.) Finally, because ℓ_1-regularized logistic regression encourages sparse solutions, the resulting models often have decision rules that are easier to interpret in terms of relevant and irrelevant features.

3. DATA SETS

This section describes the data sets that we use for our evaluation. For benign URLs, we used two data sources. One is the DMOZ Open Directory Project [19]. DMOZ is a directory whose entries are vetted manually by editors. The editors themselves go through a vetting process, with editors given responsibility for larger portions of the directory as they gain trust and experience. The second source of benign URLs was the random URL selector for Yahoo's directory. A sample of this directory can be generated by visiting `http://random.yahoo.com/bin/ryl`.

We also drew from two sources for URLs to malicious sites: PhishTank [21] and Spamscatter [3]. PhishTank is a blacklist of phishing URLs consisting of manually-verified user contributions. Spamscatter is a spam collection infrastructure from which we extract URLs from the bodies of those messages. Note that the malicious URL data sets have different scopes. PhishTank focuses on phishing URLs, while Spamscatter includes URLs for a wide range of scams advertised in email spam (phishing, pharmaceuticals, software, etc.). Both sources include URLs crafted to evade automated filters, while phishing URLs in particular may be crafted to visually trick users as well.

Table 1 summarizes the number and types of features in the data sets that we use in our evaluations. The four data sets consist of pairing 15,000 URLs from a benign source (either Yahoo or DMOZ) with URLs from a malicious source (5,500 from PhishTank and 15,000 from Spamscatter). We refer to these sets as the DMOZ-PhishTank (DP), DMOZ-Spamscatter (DS), Yahoo-PhishTank (YP), and Yahoo-Spamscatter (YS) sets. We collected URL features between August 22, 2008 – September 1, 2008.

We normalized the real-valued features in each feature set to

Feature type	DP	YP	DS	YS
DNS NS record	10818	4186	10838	3764
WHOIS info	9070	3959	8702	3525
Hostname	7928	3112	7797	2959
TLD + domain	5787	2186	5609	1980
DNS A record	5627	2470	5207	1862
Geographic	4554	2250	4430	1836
Path tokens	4363	8021	8390	11974
Last token of path	1568	3751	2892	5071
DNS MX record	1323	497	1422	559
TLD	319	284	102	51
Connection speed	31	29	30	30
DNS TTL	14	13	14	12
Blacklists	7	7	7	7
WHOIS dates	6	6	6	6
Spamassassin plugin	5	5	5	5
IP address misc.	4	4	4	4
Lexical misc.	3	3	3	3
All features	51427	30783	55458	33648

Table 1: Breakdown of feature types for data sets used in evaluations.

lie between zero and one, shifting and rescaling each real-valued feature so that zero and one corresponded to the minimum and maximum values observed in the training set. Values outside this range in the testing set were clipped to zero or one as appropriate. The normalization served to equalize the range of the features in each feature set, both real-valued and binary. Intuitively, it reflects our prior belief that the real-valued and binary-valued features are equally informative and therefore should be calibrated on the same measurement scales.

One further complication arises due to undefined, or missing, features. Many real-valued features are undefined for large numbers of examples in the data set (e.g., DNS time-to-live values). We handled missing values using the following heuristic: for each real-valued feature, we defined an extra binary feature indicating whether the feature was defined. This heuristic enables the classifiers to learn how to treat missing features from the way they appear in the training set.

4. EVALUATION

In this section, we evaluate the effectiveness of the classifiers on identifying URLs to malicious sites. Specifically, we want to answer the following questions: Does using more features lead to more accurate classification? What is the most appropriate classification model to use? What is the impact on accuracy if we tune the classifier for lower false positives? Can we effectively classify data from once source with a model that was trained on a different data source? What trends do we see among the relevant features? And what do the misclassified examples have in common?

4.1 Methodology

We start by describing our experimental methodology. For each feature set and data set in our experiments, we perform classification over 10 random splits of the data, and the splits are 50/50 between training and testing. We learn a decision threshold t that will minimize the overall classification error (see Section 2.2 for methodology of learning t). We show the average classification results of those 10 splits.

We ran our experiments on a machine with 2 dual-core 2.33 GHz Xeon processors with 4 GB memory. Memory exhaustion was not an issue, but typical usage was on the order of a few hundred megabytes. We implemented a Naive Bayes solver in MATLAB. For the SVM solvers, we used the `.mex` implementations of LIBSVM [6] and LIBLINEAR [7] that interface with MATLAB. We

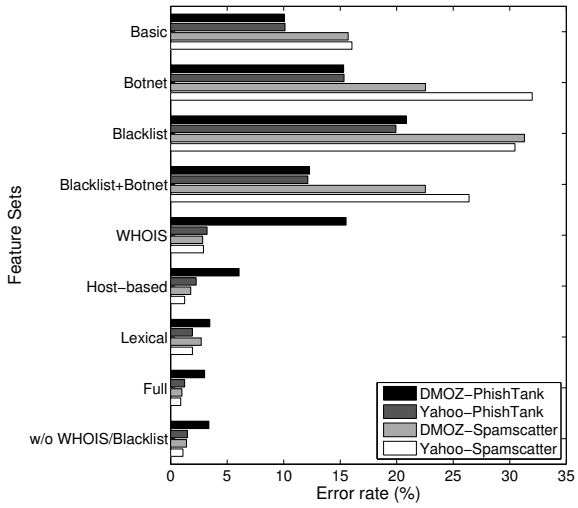

Figure 1: Error rates for LR with nine features sets on each of the four URL data sets. Overall, using more features improves classification accuracy.

implemented a custom optimizer for ℓ_1-regularized logistic regression in MATLAB using multiplicative updates [24].

4.2 Feature Comparison

Our first experiments on URL classification were designed to explore the potential benefits of considering large numbers of automatically generated features as opposed to small numbers of manually chosen features. Fig. 1 compares the classification error rates on the four data sets described in Section 3 using nine different sets of features. The different feature sets involve combinations of features reflecting various practical considerations. For brevity, we only show detailed results from ℓ_1-regularized logistic regression (LR), which yields both low error rates (see Section 4.3) and also highly interpretable models (see Section 4.6). However, the other classifiers also produce qualitatively similar results.

Table 2 shows the total number of features in each feature set for the Yahoo-PhishTank data set. For these experiments, we also report the number of *relevant* features that received non-zero weight from ℓ_1-regularized logistic regression. (The other three data sets yield qualitatively similar results.) The feature sets are listed in ascending order of the total number of features. In what follows, we describe the detailed compositions of these feature sets, as well as their effects on classification accuracy.

We start with a "Basic" feature set that corresponds to the set of URL-related (not content-related) heuristics commonly chosen by various anti-phishing studies, including Fette et al. [8], Bergholz et al. [4] and CANTINA [28]. This set consists of four features: the number of dots in a URL, whether the hostname contains an IP address, the WHOIS registration date (a real-valued feature), and an indicator variable for whether the registration date is defined. Under this scheme, the classifier achieves a 10% error rate.

The next three feature sets (Botnet, Blacklist, and Blacklist + Botnet) represent the use of current spam-fighting techniques for predicting malicious URLs. The features for "Botnet" are from the SpamAssassin Botnet plugin (Section 2.1). These consist of five binary features indicating the presence of certain client- or server-specific keywords, whether the hostname contains an IP address, and two more features involving the PTR record of the host. With these features, we obtain a 15% error rate at best.

Feature set	# Features Total	Relevant	Err%
Basic	4	4	10.12
Botnet	5	5	15.34
Blacklist	7	6	19.92
Blacklist+Botnet	12	10	12.14
WHOIS	3967	727	3.22
Host-based	13386	1666	2.26
Lexical	17211	4488	1.93
Full	30597	3891	1.24
w/o WHOIS/Blacklist	26623	2178	1.48

Table 2: Total and relevant number of features for the LR classifier and the Yahoo-PhishTank data set. We reference the overall error rate from Figure 1.

The "Blacklist" feature set consists of binary variables for membership in six blacklists (and one white list) run by SORBS, URIBL, SURBL, and Spamhaus. These lists combined provide at best a 20% error rate. When we combine the blacklists with the SpamAssassin Botnet plugin features in the "Blacklist+Botnet" feature set, the error rate improves to 12%.

These small features sets are reasonably effective, but including features that result in very large feature sets significantly improve classification accuracy.

The WHOIS registration date was a popular feature in previous studies [8][4][28]. Inspired by these examples, we also create a WHOIS feature set that includes the registration date as well additional WHOIS properties. The "WHOIS" set we evaluated includes the registration, update, and expiration dates, as well as a bag-of-words representation of the registrar and registrant. The feature set grows to nearly 4,000 on the YP data set (Table 2.1), while reducing the error to 3–15% across the data sets. WHOIS features can provide useful information for differentiating malicious and benign URLs, since creators of legitimate sites are likely to adopt different registration patterns than creators of malicious sites (whose sites are taken down more frequently, or who want to anonymize their registration information).

WHOIS features, however, are just a subset of the available host-based features. If we also include the remainder of our host-based features — IP, DNS, and geographic properties as well as membership in a blacklist and SpamAssassin Botnet plugin features — the feature sets from our data sets grow to over 10,000 features. And these additional features are beneficial: the LR classifier has an even lower error rate of 1.2–6%. Using host-based features provides a richer set of properties for the classifier to capture phenomena such as malicious sites being hosted from a particular IP prefix, ISP, country and the like.

In addition to host-based features, we can also use lexical properties of the URLs themselves as potentially strong features (again, we are not considering the Web page content). That is, if the URLs themselves "look" malicious, then a lexical feature set can help us automatically learn the tell-tale strings that characterize a malicious or benign URL. For example, malicious sites may have legitimate-looking tokens appended to the sub-domain of the hostname, or have them embedded in the path of the URL for obfuscation purposes. In the "Lexical" set, we consider a "bag-of-words" representation of the tokens in the URL augmented with additional properties such as the length of the hostname, the length of the entire URL and the number of dots. This is indeed an effective feature set, with a resulting error is between 1.9% and 3.5%.

Combining the host-based and lexical features into the "Full" feature set, we obtain the lowest classification error among all feature sets at 0.9–3.0%. For YP, the FP/FN rates are 0.8%/2.6%.

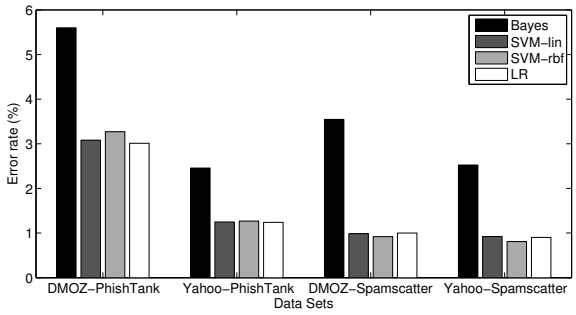

Figure 2: Overall error rates for the "Full" feature set using different classification algorithms and data sets. The SVM and LR classifiers do well with large feature sets such as "Full" as compared to Naive Bayes.

Finally, blacklists and WHOIS information provide us known high-quality information for classifying URLs. But do the low classification error rates of the "Full" feature set critically depend on those particular features? The results for "w/o WHOIS/Blacklist" show the error rates without WHOIS and blacklist features are 1.1–3.4%. Since the error rates across this feature set remain close to those of the "Full" set, we find that the remaining features provide sufficient information for achieving accurate classification results — using high-quality WHOIS and blacklist information is not necessary for excellent results.

Having reviewed the composition and results for each feature set, we point out two noticeable trends: (1) Before "WHOIS," the DMOZ-PhishTank set has similar error rates to Yahoo-PhishTank, but starting with "WHOIS" the Yahoo-trained set has lower error. This happens because the distribution of WHOIS and lexical features between DMOZ and Yahoo are distinct enough that certain URL tokens and certain WHOIS properties receive more weight in one data set than the other. (2) Prior to "WHOIS," the PhishTank-trained sets have better error rates because PhishTank Web sites have more undefined WHOIS registration dates (hence better "Basic" results), more IP addresses in the hostname (better "Botnet" results), and more URLs in blacklists (better "Blacklist" results). But starting with "WHOIS," the Spamscatter-trained sets do better because the introduction of registrar/registrant names provides a wealth of information for differentiating malicious and benign sites, and the distribution of lexical tokens in Spamscatter is distinct enough from benign sources to provide traction for accurate classification results.

These results demonstrate that different data sets provide different feature distributions for distinguishing malicious and benign URLs. Rather than manually discovering and adjusting the decision rules for different data sets, machine learning techniques can adapt to these differing distributions by learning the appropriate decision rules automatically.

4.3 Classifier Comparison

Next we compare the accuracy of the four classifiers described in Section 2, each of which has different tradeoffs between model complexity and training execution time. We use the "Full" feature set from Section 4.2 for comparison.

Figure 2 compares the results of the classifiers: Naive Bayes (Bayes), SVM with a linear kernel (SVM-lin), SVM with an RBF kernel (SVM-rbf), and ℓ_1-regularized logistic regression (LR). The bars show the overall error rates for each classifier on each of our four data sets. Additionally, Table 3 shows the training and testing

times for each classifier. The cross-validation time refers specifically to the cross-validation process used in LR to choose a regularization constant. Although expensive, it can be effectively amortized when training over time.

	Bayes	SVM-lin	SVM-rbf	LR
Train	51 s	50 s	63 s	30 min
Test	87 s	91 s	170 s	90 s
Cross-validation	—	—	—	5 hrs

Table 3: Training and testing times for the Yahoo-PhishTank data set.

The SVM and LR classifiers have at least half of the error of Naive Bayes, which is not surprising given the models that we chose. Naive Bayes is a classifier that sees wide-spread use in spam filters and related security applications, in part because the training and testing performance of Naive Bayes is fast. However, the benefit of reduced training time is outweighed in this case by the benefit of using classifiers whose explicit goal is to minimize errors. This tradeoff is particularly worthwhile when we are dealing with a large feature set.

As mentioned in Section 2.2, SVM classifiers can perform poorly if irrelevant features in a large feature set make the kernel functions poor measures of similarity. Given that the difference in error between SVM and LR is so small, though, this problem did not materialize in our data set. As such, we continue to show the results for LR for the remaining experiments because of its interpretability, which is useful for understanding how the model performs (Section 4.6) and how it might be improved (Section 4.7).

4.4 Tuning False Positives & Negatives

An advantage of using these models is that they can tradeoff false positives and false negatives. We have seen in the previous sections that classifying with the "Full" feature set yields very low overall error rates. For policy reasons, however, we may not want to choose a decision threshold t to minimize the overall error rate. Instead, we may want to tune the threshold to have very low false positives at the expense of more false negatives (or vice versa).

Consider blacklisting as a motivating example. Blacklisting has the intrinsic advantage that it will have very low false positives. Suppose a network administrator wants to take advantage of the benefits of classifying over a full feature set while maintaining the low false positives of a blacklist-only policy as applied to URL classification. To do so, we can select a threshold for the full feature set that will yield a false positive rate competitive with blacklisting while having a much lower false negative rates.

Figure 3 shows the results of this experiment as an ROC graph with respect to the decision threshold t. We see that even if we tune the decision threshold of the full-featured classifier to the same false positives as a blacklist-only classifier, the full-featured classifier still predicts malicious URLs with much better accuracy than with blacklists alone.

4.5 Mismatched Data Sources

Coupling the appropriate classifier with a large feature set can yield a highly accurate classifier when trained and tested on disjoint sets of the same data source. But do these results hold when training and testing examples are drawn from different data sources? To answer this question, we experiment with training and testing on various combinations of benign and malicious sources of URLs. (We use the abbreviations defined in Section 3 to refer to each combination of data sources, e.g. YP for Yahoo-PhishTank).

Table 4 shows classification results using ℓ_1-regularized logistic

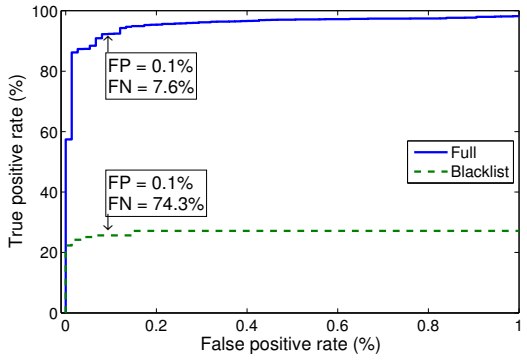

Figure 3: Tradeoff between false positives and false negatives: ROC graph for LR over an instance of the Yahoo-PhishTank data set using (1) the "Full" feature set and (2) blacklists only. We highlight the points where the false positives are tuned to 0.1%.

Training	Testing			
	YS	YP	DS	DP
YS	0.90%	5.66%	9.38%	17.73%
YP	21.33%	1.24%	44.02%	33.54%
DS	13.55%	31.57%	1.00%	13.70%
DP	22.95%	3.06%	22.92%	3.01%
All sources (YDSP)	0.95%	2.40%	1.43%	3.19%

Table 4: Overall error rates when training on one data source and testing on another (possibly different) data source using the LR classifier.

regression. Each row represents a pairing of benign and malicious URL training sources, and each column represents a pairing of benign and malicious URL testing sources. As expected, the entries along the NW-SE diagonal yield the lowest classification errors because they represent matching similar data sources (these numbers are repeated from previous experiments). When only the benign URL source is mismatched (e.g., YS and DS), error increases due to higher false positives. And if only the malicious URL source is mismatched (e.g., YS and YP), error increases due to higher false negatives. Nevertheless, we note that training on the Spamscatter set — which includes URLs advertising a wide range of scam sites — generally performs better on a mismatched source (YS-YP at 6%, DS-DP at 14%) than the PhishTank set (YP-YS at 21%, DP-DS at 23%) — which focuses only on phishing sites (Section 3).

Finally, if the training and testing sources are completely mismatched (SW-NE diagonal), the error ranges between 18–44%. Although better than random, the disparity in accuracy emphasizes the judicious selection of training data. This selection is of particular importance for use in a deployment: the training data should reflect the "testing" environment in which the system is used. To that end, if we use all four training sources, the generalization ability of the classifier is strong across all testing sources (last row). Thus, in a deployed system practitioners will want to pay special attention to collecting training data that is representative.

4.6 Model Analysis

Besides high classification accuracy, LR has the advantages of performing automatic feature selection as well as providing an interpretable linear model of the training data. In particular, because the output of a linear model depends on the weighted sum of the features, the sign and magnitude of the individual parameter vector coefficients can tell us how individual features contribute to a "malicious" prediction or a "benign" prediction. Positive coefficients

Feature type	Total	Relevant	Sign(coeff.) -	+
Path tokens	8021	419	95	324
DNS NS record	4186	1021	468	553
WHOIS info	3959	633	284	349
Last token of path	3751	76	17	59
Hostname	3112	315	123	192
DNS A record	2470	407	173	234
Geographic	2250	692	329	363
TLD + domain	2186	237	97	140
DNS MX record	497	70	34	36
TLD	284	40	15	25
Connection speed	29	19	9	10
DNS TTL	13	9	6	3
Blacklists	7	7	1	6
WHOIS dates	6	6	1	5
Spamassassin plugin	5	5	3	2
IP address misc.	4	3	3	0
Lexical misc.	3	3	0	3
All feature types	30783	3962	1658	2304

Table 5: Breakdown of features for LR for an instance of the Yahoo-PhishTank data set. We show total number of features in each feature type category, the number of relevant (non-zero) features, as well as the number of selected features with negative (benign) and positive (malicious) coefficients.

correspond with malicious features while negative coefficients correspond with benign features. Additionally, the training phase for ℓ_1-regularized logistic regression yields a *sparse* parameter vector $\mathbf{w}$ where the number of non-zero coefficients is much smaller than the total number of features, making it easier for us to focus on a smaller number of relevant features (a zero coefficient means that the corresponding feature will not contribute to the prediction outcome). These properties simplify the analysis of trends in malicious and benign features.

In this section, we analyze the model that LR constructed for one of the ten random splits of the Yahoo-PhishTank data set, and discuss trends that we observe in the model and correlate them with real-world phenomena. This example has 3,962 active (non-zero) features out of a total of 30,783 features. There are 1,658 negative (benign) coefficients and 2,304 positive (malicious) coefficients in the weight vector. Table 5 shows a breakdown of the different types of features selected in this LR model. Overall, we find that the automatic feature selection of certain feature groups are specific to the data set at hand, while other feature types indicate broader trends in malicious sites.

Not surprisingly, the weights learned for the blacklist features are all non-zero because the presence of domain names and IPs in one of the six blacklists, or its absence in the whitelist, is an accurate indicator of maliciousness.

Additionally, the weights selected for the "SpamAssassin plugin" features match their purpose as an indicator. For example, the model learned negative weights for having server words in the hostname, PTR records, and full-circle records. And it learned positive weights for having client words or an IP address in the hostname.

The miscellaneous IP address properties indicate whether the A record of a host shares the same prefix or AS as its MX or NS record. These features are considered benign on all counts, which makes sense because hosting the Web site and email server in the same data center reflects a more orthodox hosting setup that a customer would purchase through legitimate means, as opposed to a hosting infrastructure acquired by compromising a scattered array of machines.

Next, the classifier selected 40 out of the 284 top-level TLD features. Generic TLDs like '.gov', '.edu', '.com', '.org', and country-level TLDs like '.ca' and '.se' are the top-6 benign features by weight. By contrast, '.info', '.kr', '.it', '.hu', and '.es' form the top-6 malicious features by weight, domains correlated with some malicious sites.

For hostname tokens that do not include the TLD, the model selected 315 out of 3,112 features. Some interesting examples of malicious hostname tokens include the appearance of 'com' in the name, resulting from URLs to malicious sites that prepend the tokens of legitimate hostnames (e.g., from banks) as an obfuscation measure. And while it is no surprise to see that the presence of 'www' is considered a benign (albeit spoofable) feature, 'members' receives the most negative weight because of its co-occurrence with benign sites hosted on 'members.aol.com', 'members.tripod.com' and other free hosting services.

Although the URL's path tokens contribute the most features (about 8,000), the model selects 419 relevant path tokens during training, 324 of which are malicious. Among the automatically selected malicious features are 'account', 'webscr', 'login', 'ebayisapi', 'signin', 'banking', and 'confirm' — for comparison, these tokens the model learned automatically are also 7 out of the 8 manually chosen path tokens that were considered tell-tale phishing features in the study by Garera et al. [9]. Additional selected path tokens include 'images', 'com', 'www', and 'paypal'. Tokens like these indicate attempts by the site authors to spoof legitimate sites by including domain names within the URL path ('www.example.com'). Even 'http:' is considered a malicious feature, aiding in the prediction of URLs that attempt to hide legitimate-looking URLs in the path component (the case of using a legitimate domain's redirection facilities was more rare).

Out of the 3,959 WHOIS information features (mainly tokens in the names of the registrar and registrant of the domain name), the classifier selected 633 as relevant. As for dates in the WHOIS record, missing any of the three WHOIS dates (registration, update, expiration) is considered a malicious feature. Moreover, having a recent registration or update date is considered malicious — this corresponds with the behavior of malicious sites having newer registration dates because they are taken down more frequently. Having a newer expiration date is the only benign WHOIS date feature.

For connection speed features, the model selected 19 out of 29. Having a T1 speed for the DNS A and MX records are the top-2 benign features, while a residential cable, DSL or satellite connection hosting an address in the A, MX or NS record of an entry is considered malicious. These features reflect the phenomenon of malicious sites hosted on compromised machines in residential ISPs.

Finally, the selected DNS A/MX/NS record features and geographic features correspond to hosting activity associated with various regions of the Internet address space. For example, IP ranges belonging to Google, Yahoo and AOL are treated as benign features. Also, having DNS NS records in IP ranges belonging to major registrars such as RIPE (the network information center for Europe) are considered benign features. However, having an NS record in one of the IP prefixes run by GoDaddy is considered a malicious feature.

Overall, the classifier was able to automatically select malicious and benign features for which domain experts had prior intuition. Perhaps more importantly, the classifier automatically selected new, non-obvious features that were highly predictive and yielded additional, substantial performance improvements.

When faced with such a reputation system, the ability of adversaries to evade detection depends on their ability to avoid conforming to the trends and selections determined by the classifier; we discuss the issue further in Section 5.

4.7 Error Analysis

A machine learning approach using a full feature set, consisting of both lexical and host-based features, yields a URL reputation classifier that has low false positive and low false negative rates. Despite the high accuracy of this approach, in this section we examine examples of misclassified URLs to reveal trends that can suggest refinements to our current approach, and also help us understand the limitations of using just URL features for classification (as opposed to additionally including features from Web page content). For our analysis, we examine one of the instances of the Yahoo-PhishTank data sets from the LR classifier, chosen randomly, which had 60 false positives and 71 false negatives (other instances resulted in similar conclusions).

The most common cause of false positives (benign sites mistakenly classified as malicious) was due to sites hosted at disreputable ISPs. This is a case of guilt by association — our classifier penalizes legitimate sites hosted in the same AS or IP prefix as malicious sites. This set of false positives could be mitigated by examining the content of a site. Also, if reputation systems become more prevalent, such false positives imply that it would behoove legitimate site owners to seek service providers that have a reputation for cracking down on malicious sites hosted on their networks.

The false negatives (undetected malicious sites) fell into the following categories: (1) URLs to sites with benign tokens in the URL, (2) malicious sites using free hosting services (such as 'geocities.com' or 'tripod.com'), (3) compromised sites, (4) redirection services, (5) sites hosted in reputable geographic regions (such as the US), and (6) sites with international TLDs hosted in the US. The first category shows that if certain lexical features have substantial weights in classification, false negatives can result. Eliminating this false negative could be accomplished by more careful vetting of which URL tokens become features.

The last five categories are related because they reflect the situation where host-based features of a malicious URL *appear* to be non-malicious. However, the remedies for mitigating these kinds of false negatives differ. For example, eliminating malicious sites in category (3) is a matter of encouraging sites to maintain up-to-date patches of software, and category (5) reflects the practice of purchasing services from a reputable ISP; such remedies, though, are out of the control of a reputation system. However, the closely-related category (6) sites could be addressed by adding features that indicate whether the country of a site's TLD corresponds to the country where the ISP is located. Finally, mitigating false negatives in (2) and (4) may require examining the content of a site.

5. EVASION

If deployed and effective, adversaries will naturally try to invent methods for evading this statistical modeling approach. As we discussed in Section 2, the effectiveness of statistical modeling depends on a few key assumptions. When hosting malicious sites, adversaries can try to violate these assumptions to evade detection.

One approach for evasion is to reduce the information content in the lexical features of URLs. For example, using redirection services such as TinyURL produces fewer distinguishing lexical features (e.g., an identical host name coupled with a random token). Another approach is to acquire many benign host-based features, such as hiding behind the well-provisioned infrastructure provided by legitimate free hosting services (e.g., 'geocities.com' or 'tripod.com'). These aliasing techniques could provide adversaries methods for misclassifying their sites as benign.

As discussed in Section 4.6, the weights assigned to features in our classifiers reflect trends in the features of malicious and benign URLs. An adversary evading detection can strive to craft a site that can spoof sufficient benign features with high weights, and avoid acquiring sufficient malicious features, to appear as a benign site to the detection algorithm. For instance, '.org' is a benign TLD according to our classifier, yet costs only $10/year to register.

There are, however, functional and cost-related limits to an adversary's ability to forge features. For instance, older WHOIS registration dates have high weights as benign features, but are difficult to casually obtain for malicious sites. Further, whether a site appears on a blacklist is out of its control. Whether such features are sufficient in the long term for differentiating benign from malicious sites remains an open question. As with any security approach facing an adaptable adversary, though, we envision having to evolve a deployment in response over time.

6. RELATED WORK

This section surveys related approaches in URL classification, related applications of statistical modeling, and other techniques.

6.1 Classification of URLs

The work by Garera et al. is the most closely related to our study [9]. They use logistic regression over 18 hand-selected features to classify phishing URLs. The features include the presence red flag key words in the URL, features based on Google's Page Rank and Google's Web page quality guidelines. Although a direct comparison with our approach is difficult without access to the same URLs or features, they achieve a classification accuracy of 97.3% over a set of 2,500 URLs. Though similar in motivation and methodology, our approach differs significantly in both scope (aiming to detect other types of malicious activity) and scale (considering an order-of-magnitude more features and training examples).

McGrath and Gupta do not construct a classifier but nevertheless perform a comparative analysis of phishing and non-phishing URLs [16]. For examples, they compare non-phishing URLs drawn from the DMOZ Open Directory Project [19] to phishing URLs from PhishTank [21] and a non-public source. The features they analyze include IP addresses, WHOIS thin records (containing date and registrar-provided information only), geographic information, and lexical features of the URL (length, character distribution, and presence of pre-defined brand names). We build on their work by incorporating similar sources and features into our approach.

Provos et al. perform a study of drive-by exploit URLs and use a patented machine learning algorithm as a pre-filter for VM-based analysis [22]. Unlike our approach, they extract content-based features from the page, including whether IFrames are "out of place," the presence of obfuscated javascript, and whether IFrames point to known exploit sites.

CANTINA classifies phishing URLs by thresholding a weighted sum of 8 features (4 content-related, 3 lexical, and 1 WHOIS-related) [28]. Among the lexical features, it looks at dots in the URL, whether certain characters are present, and whether the URL contains an IP address. The WHOIS-related feature CANTINA examines is the age of the domain. We use similar features in our approach, but entirely different models of classification.

6.2 Machine Learning in Related Contexts

Fette et al. use statistical methods in machine learning to classify phishing emails [8]. Their classifiers examine the properties of URLs contained within a message (e.g., the number of URLs, number of domains, and number of dots in a URL), but unlike our approach they also consider features of the email structure and content. Bergholz et al. further improve the accuracy of Fette et al. by introducing models of text classification to analyze email con-

tent [4]. Abu-Nimeh et al. compare different classifiers over a corpus of phishing emails, using as features the frequency of the 43 most-popular words in the corpus [1].

Kolari et al. use URLs found within a blog page as features to determine whether the page is spam [13]. They use a "bag-of-words" representation of URL tokens, and we use a similar set of features.

6.3 Non-Machine Learning Approaches

Highly Predictive Blacklists use a Page Rank-style algorithm to estimate the likelihood that an IP address will conduct a network intrusion against particular networks [27].

Moshchuk et al. use VMs to analyze downloaded trojan executables, relying on third-party anti-spyware tools to detect whether VMs were infected by executables [18]. Wang et al. detect drive-by exploits by using behavioral detection (monitoring anomalous state changes in the VM) as well as detecting exploits of known vulnerabilities [25]. Provos et al. monitor VM state changes and use multiple anti-virus engines to detect VM infection [22]. Moshchuk et al. also construct a VM-based web proxy defense that uses behavior-based detection and adds only a few seconds of overhead to page rendering for the end-client [17].

Many commercial efforts exist to detect malicious URLs, including McAfee SiteAdvisor [15], IronPort Web Reputation [12], the WebSense ThreatSeeker Network [26], WOT Web of Trust [2], and Google Toolbar [10]. These approaches are based on blacklist construction, user feedback, and proprietary feature analysis.

7. CONCLUSION

We have described an approach for classifying URLs automatically as either malicious or benign based on supervised learning across both lexical and host-based features. We argue that this approach is complementary to both blacklisting — which cannot predict the status of previously unseen URLs — and systems based on evaluating site content and behavior — which require visiting potentially dangerous sites. Further, we show that with appropriate classifiers it is feasible to automatically sift through comprehensive feature sets (i.e., without requiring domain expertise) and identify the most predictive features for classification.

An open issue is how to scale our approach to handle millions of URLs whose features evolve over time. We address the issue in subsequent work by using online learning algorithms [14].

Acknowledgments

Thanks go to Alvin AuYoung, Kirill Levchenko, Patrick Verkaik and Michael Vrable for insightful comments on earlier drafts of this paper. Thanks to anonymous reviewers for their valuable feedback. This work was supported by National Science Foundation grants NSF-0238323, NSF-0433668 and NSF-0829469 and by generous research, operational and in-kind support from Cisco, Google, Microsoft, Yahoo and the UCSD Center for Networked Systems.

8. REFERENCES

[1] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair. A Comparison of Machine Learning Techniques for Phishing Detection. In *Proceedings of the Anti-Phishing Working Group eCrime Researchers Summit*, Pittsburgh, PA, Oct. 2007.

[2] Against Intuition. WOT Web of Trust. http://www.mywot.com.

[3] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *Proc. of the USENIX Security Symposium*, Boston, MA, Aug. 2007.

[4] A. Bergholz, J.-H. Chang, G. Paaß, F. Reichartz, and S. Strobel. Improved Phishing Detection using Model-Based Features. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, Aug. 2008.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Publishing Company, New York, NY, 2006.

[6] C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. http://www.csie.ntu.edu.tw/ cjlin/liblinear/.

[8] I. Fette, N. Sadeh, and A. Tomasic. Learning to Detect Phishing Emails. In *Proceedings of the International World Wide Web Conference (WWW)*, Banff, Alberta, Canada, May 2007.

[9] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A Framework for Detection and Measurement of Phishing Attacks. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, Alexandria, VA, Nov. 2007.

[10] Google. Google Toolbar. http://tools.google.com/firefox/toolbar/.

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Publishing Company, New York, NY, 2001.

[12] IronPort. IronPort Web Reputation: Protect and Defend Against URL-Based Threat. *IronPort White Paper*, 2008.

[13] P. Kolari, T. Finin, and A. Joshi. SVMs for the Blogosphere: Blog Identification and Splog Detection. In *Proceedings of the AAAI Spring Symposium on Computational Approaches to Analysing Weblogs*, Stanford, CA, Mar. 2006.

[14] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying Suspicious URLs: An Application of Large-Scale Online Learning. In *Proc. of the International Conference on Machine Learning (ICML)*, Montreal, Quebec, June 2009.

[15] McAfee. SiteAdvisor. http://www.siteadvisor.com.

[16] D. K. McGrath and M. Gupta. Behind Phishing: An Examination of Phisher Modi Operandi. In *Proc. of the USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, San Francisco, CA, Apr. 2008.

[17] A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy. SpyProxy: Execution-based Detection of Malicious Web Content. In *Proc. of the USENIX Security Symposium*, Boston, MA, Aug. 2007.

[18] A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy. A Crawler-Based Study of Spyware on the Web. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, San Diego, CA, Feb. 2006.

[19] Netscape. DMOZ Open Directory Project. http://www.dmoz.org.

[20] Y. Niu, Y.-M. Wang, H. Chen, M. Ma, and F. Hsu. A Quantitative Study of Forum Spamming Using Context-based Analysis. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, San Diego, CA, Mar. 2007.

[21] OpenDNS. PhishTank. http://www.phishtank.com.

[22] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All Your iFRAMEs Point to Us. In *Proc. of the USENIX Security Symposium*, San Jose, CA, July 2008.

[23] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

[24] F. Sha, A. Park, and L. K. Saul. Multiplicative Updates for L_1-Regularized Linear and Logistic Regression. In *Proceedings of the Symposium on Intelligent Data Analysis (IDA)*, Ljubljana, Slovenia, Sept. 2007.

[25] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, San Diego, CA, Feb. 2006.

[26] WebSense. ThreatSeeker Network. http://www.websense.com/content/Threatseeker.aspx.

[27] J. Zhang, P. Porras, and J. Ullrich. Highly Predictive Blacklisting. In *Proc. of the USENIX Security Symposium*, San Jose, CA, July 2008.

[28] Y. Zhang, J. Hong, and L. Cranor. CANTINA: A Content-Based Approach to Detecting Phishing Web Sites. In *Proceedings of the International World Wide Web Conference (WWW)*, Banff, Alberta, Canada, May 2007.