Judging a site by its content: learning the textual, structural, and visual features of malicious Web pages

Sushma Nagesh Bannur
Department of Computer Science and Engineering, UCSD
sbannur@cs.ucsd.edu

Lawrence K. Saul
Department of Computer Science and Engineering, UCSD
saul@cs.ucsd.edu

Stefan Savage
Department of Computer Science and Engineering, UCSD
savage@cs.ucsd.edu

ABSTRACT

The physical world is rife with cues that allow us to distinguish between safe and unsafe situations. By contrast, the Internet offers a much more ambiguous environment; hence many users are unable to distinguish a scam from a legitimate Web page. To help address this problem, we explore how to train classifiers that can automatically identify malicious Web pages based on clues from their textual content, structural tags, page links, visual appearance, and URLs. Using a contemporary labeled data feed from a large Web mail provider, we extract such features and demonstrate how they can be used to improve classification accuracy over previous, more constrained approaches. In particular, by analyzing the full content of individual Web pages, we more than halve the error rate obtained by a comparably trained classifier that only extracts features from URLs. By training classifiers on different sets of features, we are further able to assess the strength of clues provided by these different sources of information.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; I.5.1 [**Pattern Recognition**]: Models—*Statistical*

General Terms

Algorithms, Experimentation, Security

1. INTRODUCTION

The ubiquity of the Internet, the explosion in Web-based applications, the rise of consumer e-commerce—all of these have incentivized criminals to target Web users in a variety of online scams. Users are vulnerable to these scams precisely because they are so comfortable navigating Web pages and the URLs that name them. As these elements have emerged to form the standard platform for online services, so too have they become central to a wide variety of online criminal activity.

There are many categories of such attacks. For example, phishing attacks direct users to Web pages that closely duplicate those of popular legitimate services. These attacks aim to defraud users of personal information such as usernames and passwords, social security numbers, and credit card numbers, all of which are then monetized. Other scams involve the advertising of counterfeit products; these products are sold without license or regulatory oversight through Web sites that masquerade as legitimate vendors. A common example of this activity is the sale of counterfeit drugs by online pharmacies. Finally, some attackers use Web pages as a vector to infect visitors with malware. These attacks may target latent software vulnerabilities in Web browsers or media applications (e.g., Acrobat Reader), or they may entice users via social engineering to install a malicious application, browser plugin, or video codec. Such malware can be used to capture private user information or to support other scams (e.g., by sending e-mail, clicking on ad links, etc.); the potential for use can also be resold to others. These examples are are just the most common among a wide range of attacks that use the Web as a vector.

However, despite the broad range of goals in these scams, they all typically share a common element: the victim is only exposed to the attack if they visit the adversary's Web page (we call such a page as "malicious" page). Thus, attackers must advertise URLs that link to their pages and convince users to click on these links. These URLs can be advertised through a range of mediums, including email, social networking, search engine results, blog posts, sponsored advertising, and so on. Indeed, there is evidence that large numbers of Web pages returned in search results are malicious [13]. Thus, there is a clear need for a Web filtering system that can inspect a Web page and detect if it is malicious. By embedding such filters into browsers, search engines, or other Web services, users can be protected from attacks that they would otherwise encounter in their daily use of the Internet.

The most common approach to such filtering is *blacklisting*, a technique whereby particular properties of a Web page (typically its domain name, IP address, or full URL) are compared against a compiled list of previously identified malicious Web sites. However, a blacklist is only as effective as the quality and timeliness of the information upon which it is based.

How then are *new* malicious web sites identified? Some are detected by using "honeypot" systems as oracles—e.g., e-mail accounts that should have no legitimate traffic but still receive unsolicited spam. Others are detected by running browser instances in a Virtual Machine (VM) environment. In particular, using instrumentation techniques, a VM environment can simulate the impact of visiting a Web page [16, 13, 11]. Since all state changes to the virtual machine are recorded, a malicious Web page is revealed whenever a visit results in changes to an unauthorized file system or system configuration. However, the heavyweight operation of this approach does not lend itself to large-scale, real-time classifi-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AISec'11, October 21, 2011, Chicago, Illinois, USA.
Copyright 2011 ACM 978-1-4503-1003-1/11/10 ...$10.00.

cation. Moreover, this method is mainly designed to identify sites that inject malware; it does not address other fraud scenarios such as phishing and counterfeit sales.

In general, most techniques that drive production blacklists are based on such *post hoc* information (i.e., that is only available after a malicious site has been advertised for some time and is positively identified as such) and thus are not robust against new pages and fail in a range of scenarios as elaborated in [14]. An alternative approach, which is the focus of this paper, is based on applying methods from machine learning; these methods use such post hoc information to infer the prevalent patterns that distinguish malicious and legitimate Web pages more generally (including previously unseen pages). For example, [9, 10] show how to classify malicious Web sites from the information contained in the URL of the Web page, and [1, 3, 17] outline related approaches for specifically detecting phishing sites. Our work is driven by the belief that we can improve classification in this regime by combining many sources of potential information (including URL data and meta data, page contents, link structure and visual components).

With this motivation, we study how to identify malicious Web pages using methods in supervised learning. Our approach is not targeted to a particular variety of threat, but rather designed to detect any malicious Web page for which quality post-hoc training data can be obtained reliably (although efficacy will undoubtedly vary in different scenarios). We include information from both the URL and the content of the Web page as input to our classifiers; in particular, we extract features from the URL, textual content, structural tags, page links, and visual appearance of a Web page. We experiment with batch models of classification that are trained in an offline setting. The labeled examples are obtained from a live data feed (in part supplied by a large Webmail provider). We observe that by extracting features from each Web page's content in addition to (previously considered) features of its URL, we can reduce the classification error rate by over 50%. Together, using the full range of features across 10 days sample data, our approach identifies malicious Web pages with about 98% accuracy and only a marginal number of false positives.

The outline of the paper is as follows: Section 2 describes the infrastructure used for data collection. Section 3 provides a detailed illustration of the features used to represent a Web page. Sections 4 and 5 describe our strategies for offline classification; we discuss the batch learning algorithms used and the evaluations performed. Finally, we summarize our findings and conclude in Section 6.

2. DATA

Our study is driven by a variety of data sources that have been generously shared with us. In particular, our malicious examples come from "known bad" feeds of URL data provided by third-party Web mail, anti-spam and security providers (based on the feed sources described in [7]). Note that these feeds are, by their nature, biased towards URLs that are distributed via e-mail and may not capture diversity present in other Web vectors (e.g., Search-Engine Optimization, Blog spam, Social networks, etc.) For benign examples, we use contemporaneous random selections of URLs found on Twitter (retroactively using Twitter's own post-hoc filtering to remove URLs known to be abusive as per [15]). Due to Twitter's role as a social communication medium, we expect this collection to be highly diverse and to capture the kinds of benign URLs that users are likely to encounter.

For every URL (in any feed) we follow all shorteners or redirects to produce a canonical "final" URL for use in the study (i.e., we do not attempt to classify shortened URLs or free-hosting services used to redirect traffic, but rather we classify the ultimate destination URL). We then visit this final site and capture the raw HTML contents of the page being served there, a screenshot of the page (as rendered by Firefox) and additional network metadata concerning the naming and hosting of the URL and any embedded links.

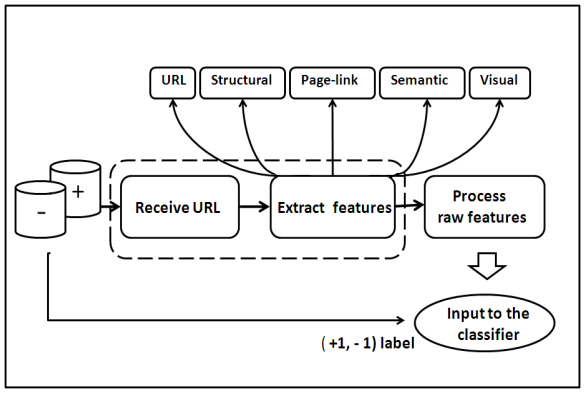

Figure 1: Infrastructure for data collection and feature extraction.

Since malicious Web sites are frequently short-lived, it is necessary to perform the feature extraction contemporaneously, when the data is still fresh, to guarantee that we have access to the original Web page that was marked malicious as well as the associated meta-data (e.g., hosting, name server, or WHOIS records). Figure 1 abstractly illustrates the data collection and feature extraction infrastructure used to support our study, which handles roughly 6000 URLs containing unique domains each day. Thus, each URL is processed as soon as it is received and all features are collected immediately thereafter (excepting purely synthetic features, such as those produced by computer vision techniques, that can be produced entirely post-hoc if the screenshot of the Web page is already captured). The ratio of malicious to benign URLs in the collected data is approximately 1:2 and we perform our evaluations using 10 days of data.

3. FEATURES

We extract features from both the URL and the content of the Web page, incorporating both the hidden as well as visible content. Visible information on a Web page includes the URL, textual information, non-textual information in the form of images, audio and video, page links to other sources and the overall visual appearance of the Web page. The hidden information consists of the HTML code of the Web page, the embedded scripts and the stylesheets used for formatting. Additionally, we gather a number of indirect features using the information available on the Web page. While gathering information, we make a trade-off between speed and classification performance. For example, when extracting the features from the page links of a Web page, we limit the feature extraction to a few links to ensure it completes within acceptable time limits. The gathered raw data is processed and the features are scaled such that each feature is within the range [0, 1].

We categorize the information gathered for a Web page as URL, structural, page link, semantic and visual features. Features corresponding to each of these categories are explained in detail in the following sections. In general, gathering information from the various attributes of a Web page results in a very large number of features; after 10 days of data collection, the dimensionality of the feature vector is about half a million. However, the representation is extremely sparse, with only a small number of features having

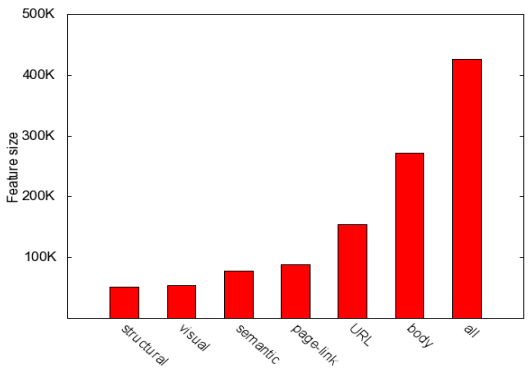

Figure 2: Number of features from different categories after 10 days of data collection.

non-zero values for any particular Web page. Figure 2 outlines the contribution to the total feature dimensionality from each of the feature categories.

3.1 URL

The work of Ma et al. [9, 10] explores how to detect malicious Web pages based only on information derived from their URLs. URL features are categorized into lexical and host-based features. We incorporate the described features and additionally propose some new URL features.

We include the information in the HTTP header as the header features. The header contains a number of useful details about the Web page such as the content-type, content-language, server, mime-version, cookie, and HTTP status code. For example, header fields *Server* and *X-Powered-By* indicate the software used by the server to handle the request. It is likely that a Web page injecting malware might be running outdated versions of software with known vulnerabilities. Another example is the information in the HTTP status code; it is unlikely to associate an error page as being malicious. We include most of the HTTP header details as features except for information that is related to the time of crawling.

We also observe that it is uncommon for a legitimate Web page to contain an explicit IP address as domain name. It is also uncommon to contain an explicit port id after the domain name in the URL. We therefore include features that indicate whether or not the URL contains such terms.

Finally, Web pages supporting the SSL protocol (i.e., https) provide a digital certificate to the client. The certificate authority (CA) issuing the certificate attests that the Web page is offered by a known party (and hence is more likely to be legitimate and trusted). To incorporate the credibility of this attestation we use details of the CA issuer and the start/end dates of the certificate as features.

3.2 Structural

The HTML code contains the entire description of the Web page. Only part of this information is visible to the user on the Web page; the rest remains hidden. We extract structural features from the hidden information which describes the structure of each Web page.

To obtain a high level summary of the structure, we gather information from the HTML tags that appear in the HTML code. Specifically we count the number of times each HTML tag appears. These simple HTML tag counts often indicate whether the Web page has content which requires attention, such as if the page contains any forms, embedded objects, scripts, etc. We expect such information to be useful for detecting certain types of malicious Web pages. For example, Web pages for phishing attacks contain forms where victims are expected to enter their credentials. Other tag counts can indicate the heavy use of scripting languages, particularly Javascript, that are commonly used to frame malicious attacks. Javascript code can be used to discover vulnerabilities on the user's computer for subsequent targeting or can directly implement such attacks (e.g., heap spraying).

The HTML tag counts contain information about the general layout of the page — if the page contains a title, sub-headings, tables, etc. The HTML tag counts may also reveal certain aspects of display formatting – if the HTML contains stylesheets, specific fonts, frames etc. In general, the layout of the Web page reflects how much professional attention was paid to its design. Thus layout information may be useful for classification because malicious pages are purposefully designed to attract users.

We also include the HTML attributes associated with HTML tags as features. These features capture more detailed structural properties of the Web page. Some HTML tag attributes are assigned values; for certain value types, we incorporate the HTML tag attribute and value pair as a feature. For example, the value of *type* attribute for the *input* tag within a *form* conveys useful information. An input type of *radio* or *checkbox* seems potentially less suspicious than an input type *text*. The *language* attribute for a script specifies the scripting language used; this attribute also contains useful clues for classification. In particular, some scripting languages are more commonly used for framing malicious attacks than others.

3.3 Page link

The links on a Web page can also provide information to help determine whether it is malicious. We include features that count the number of internal and external links. A page link is an internal link if its domain name matches that of the base URL; otherwise it is an external link. A Web page containing only internal links seems less likely to be malicious than one containing external links. We also include a feature that counts the number of unique hostnames on each page. This feature counts the number of different domains to which external links on the page are directed.

Web pages often have embedded objects such as images, audio, video, plugins and binaries. For each object, the Web page will contain a link to its source. To capture information about these objects, we incorporate features that indicate the file types associated with the page links that appear on the Web page. Specifically, based on the associated file type we categorize the page link as image, video, audio, data, text, binary, Web, game, system file, compressed file, code or display related file. We include features that count the number of page links associated with specific file categories. We also include features that count the number of times a particular file extension appears on the Web page. We expect that certain object types are more likely to be associated with malicious Web pages. For example, it is known that many attacks are framed using image, audio and video files. We also expect the size of these files to be correlated with the presence of an attack. For this reason, we include features corresponding to the minimum, maximum and median file size for every file extension type that appears on the Web page.

It can be very expensive to download large files merely to compute their size; hence we retrieve this information from the *Content-Length* attribute in the HTTP header of the file. But it can also be expensive to extract this information from the header for every link, and hence we compute the file size only for image, audio and video files. We summarize this information into features by computing the mean and variance of the URL lengths over the links on the page. We also count the number of links that specify hostnames

as explicit IP addresses and that have a particular URL protocol. Since it is expensive to extract the header attributes for every link, we only perform this analysis for links with image or video extensions. We summarize this information into features by counting the number of instances of each header attribute.

It is known that the host-based information contained in URLs provide important features for classification of malicious web sites. We include WHOIS and geographical features for the page links; these are the same WHOIS and geographical computed in [10] for the URL of the page itself. To keep the computation bounded, we limit the number of hostname lookups to 20. We also perform blacklist lookups for these page links subject to this limit. We do not analyze lexical features of the URLs from page links due to the very high dimensionality that results from semantic feature representations.

3.4 Semantic

Our semantic features capture textual information visible on the Web page. We include these features because certain text seems more likely to indicate a malicious Web page. For example, we are less suspicious of Web sites that sell clothes than male enhancement drugs. Also, we expect certain words to appear on particular types of malicious Web pages (e.g., the words "password" and "credit-card" on phishing sites).

The simplest way to extract semantic features is to model the text as a "bag of words." In this representation, the semantic features simply count the number of times each word appears in the text. We also include the total word count as a feature.

Another way to extract semantic features is to use a term frequency-inverse document frequency (tf-idf) representation; this approach is common in text-mining. In a tf-idf model, each word in a document is assigned a weight that represents its importance relative to the entire corpus. The term frequency is the frequency of the word within the document, and the inverse document frequency is the reciprocal of the number of documents containing that word. The weight assigned to a word in a document is the product of the term-frequency and the inverse document frequency. The tf-idf representation highlights words that occur rarely in the corpus but frequently in one document (as opposed to very common words that appear in most documents). We evaluate both tf-idf representations and bag-of-words representations of semantic features.

Our approach is different from that of Zhang et al. [17], who also use textual information for phishing detection. For each Web page, they generate a lexical signature using only the 5 words with largest tf-idf weights. These words are then used in a Google search engine query. Our approach differs by considering the frequencies (weighted or unweighted) of all the words on each Web page.

3.5 Topic modeling

We also explore an alternative, more compact representation of textual information based on topic modeling. The basic idea behind this approach is to model each document as containing information about a particular distribution of topics. Intuitively, these topics capture higher-order features of each Web page's content and context.

We use non-negative matrix factorization(NMF) [5, 6] to encode the textual content of each Web page's HTML as a distribution over topics. NMF transforms the original d–dimensional bag-of-words representation to k–dimensional topic distribution with $k \ll d$. Specifically, NMF learns a part-based representation of the data by factorizing the term-document matrix $\boldsymbol{X}$ into non-negative matrices $\boldsymbol{V}$ and $\boldsymbol{H}$ with dimensions $d \times k$ and $k \times n$ respectively, such that $\boldsymbol{X} \approx \boldsymbol{VH}$. The "parts" in this representation correspond to semantically meaningful topics in the Web page text. In particular, the textual content of each Web page is modeled as arising from an additive combination of words from different topics. (The non-negativity constraints enforce the purely additive nature of this combination.)

To perform NMF, we minimize the reconstruction error between $\boldsymbol{X}$ and $\boldsymbol{VH}$ using multiplicative update rules. These update rules are given by:

$$\boldsymbol{H_{an}} \leftarrow \boldsymbol{H_{an}} \frac{\sum_i \boldsymbol{V_{ia}} \boldsymbol{X_{in}} / (\boldsymbol{VH})_{in}}{\sum_j \boldsymbol{V_{ja}}}, \quad (1)$$

$$\boldsymbol{V_{ia}} \leftarrow \boldsymbol{V_{ia}} \frac{\sum_n \boldsymbol{H_{an}} \boldsymbol{X_{in}} / (\boldsymbol{VH})_{in}}{\sum_p \boldsymbol{H_{ap}}}. \quad (2)$$

The low dimensional features $\boldsymbol{H}$ then serve as the input for subsequent models of classification.

For test Web pages $\boldsymbol{X'}$ whose output label must be predicted, the low dimensional representation $\boldsymbol{H'}$ is computed using the same iterative update as in eq. (1). For these Web pages, however, the basis vectors $\boldsymbol{V}$ are not updated throughout the iteration; they are fixed to the values that were obtained during the training phase.

3.6 Visual

Visual features encode the information related to the appearance of the Web page. We limit our analysis to the holistic screenshot of the Web page and do not consider the images on the page as candidates for individual visual feature extraction. It would be interesting to investigate the visual features of individual images within the Web page. However, it is very expensive to retrieve each image and computing individual visual features; this does not seem practical for large-scale applications. We include features from the color histogram, the holistic image summary, and recognizable local patterns within the Web page.

The color histogram of an image gives the distribution of the various colors in the image. We use color histogram features to analyze if there exists any similarity between the pattern of colors used for each class of Web pages. As the number of color shades in RGB colorspace can be as large as $256 \times 256 \times 256$, we limit the color space to a pre-computed fixed 3-3-2 bit color palette.

We capture the holistic visual appearance of the Web page using a low dimensional feature vector to obtain the high-level summary. For this task, we use *gist* [12], which estimates the holistic visual structure using a few perceptual dimensions which describe the spatial properties of the image. Spatial properties measure each image's *degrees of naturalness*, *openness*, *roughness*, *expansion* and *ruggedness*. If we consider just the gist features, we expect Web pages that look alike to be projected close to each other in the multi-dimensional gist feature space.

We use *SIFT* (Scale-Invariant-Feature-Transform) [8] on each Web page to identify local visual objects and obtain a global summary. SIFT computes feature descriptors at particular points of interest in an image; these descriptors are invariant to scale, orientation, and affine distortion (and robust to certain types of noise). These properties make SIFT ideal for matching images based on the similarity of their local visual content. The SIFT algorithm detects keypoints from high contrast regions of the image such as local object edges. Gradient directions and magnitudes are then computed in a local region around each keypoint; these orientations become the features of each keypoint. Each keypoint feature descriptor has 132 dimensions: 128 dimensions for these orientations, and 4 for its location, scale and rotation.

For Web page classification, we include visual features that capture basic statistics of the SIFT keypoint descriptors. In particular,

we compute the number of keypoints as well as the mean and variance of the SIFT keypoint orientations. These operations yield 257 visual features, which we refer to as *sift-stats*.

After using SIFT to identify the local visual objects in each Web page, we also match the local images found in the Web page against a repository of common Web logos. The repository contains the logos of popular banks, e-commerce brands, and social-networking sites. The SIFT matching algorithm recognizes logos in Web pages by identifying the nearest neighbors in its repository; more precisely, it computes the nearest neighbors in Euclidean distance between logo and image keypoint descriptors. The images in the repository are then assigned a score based on their distance to each local object in the Web page.

We include visual features that indicate whether a particular logo was identified in the Web page and, if so, the associated matching score. When an exact match is not found, there may still be a number of candidate matches with roughly the same score. Hence, we also incorporate information for the logos which are candidate matches but do not have the highest matching score. We refer to this set of visual features as *sift-matching*.

4. CLASSIFICATION

We study the problem of detecting malicious Web pages as an instance of binary classification. For simplicity we consider offline (or batch) algorithms for learning the classifier from labeled examples; in these algorithms, the model for classification is estimated on a finite set of training data, then deployed for making future predictions on (so-called) test data. The training data consists of n feature vectors $\boldsymbol{x_i} \in \Re^d$ and their corresponding binary labels $y_i \in \{-1, +1\}$. A positive label indicates a malicious Web page, and a negative label indicates a benign one. We use $\boldsymbol{X}$ to denote the input matrix of stacked feature vectors and $\boldsymbol{Y}$ to denote the target labels for classification.

Not all models of binary classification scale well to data sets with large numbers of examples and very high dimensional feature spaces. However, our problem lies in this regime, and therefore we limit our experiments to linear classifiers. These classifiers are parameterized by a weight vector $\boldsymbol{w} \in \Re^d$ and a scalar bias b; these parameters define the orientation and offset of the hyperplane decision boundary between different classes. We experiment with two types of linear classifiers — logistic regression (LR) and linear support vector machines (SVMs) — which we describe next.

4.1 Logistic regression

LR is a probabilistic model of binary classification. It uses the logistic function to compute the probability that an input $\boldsymbol{x}$ should be assigned a positive label ($y = 1$):

$$P(y = 1|\boldsymbol{x}; \boldsymbol{w}) = \frac{1}{1 + e^{-(\boldsymbol{w}.\boldsymbol{x}+b)}} \tag{3}$$

In practice, a label is assigned by thresholding the conditional probability $P(y = 1|\boldsymbol{x}; \boldsymbol{w})$ in eq. (3).

The parameters $\boldsymbol{w}$ and b can be estimated by maximizing the conditional probability that the labeled examples in the training data are correctly classified. To guard against overfitting, though, it often helps to regularize the estimation procedure by imposing a penalty on large parameter values. The simplest form of regularization balances the conditional log-likelihood of correct classification against an ℓ_2-norm penalty on the weight vector $\boldsymbol{w}$. This model (known as ℓ_2-regularized logistic regression) is trained by maximizing the objective function:

$$\mathcal{L}_2(\boldsymbol{w}) = \sum_{i=1}^{n} \ln P(y_i|\boldsymbol{x_i}) - \mu\|\boldsymbol{w}\|_2^2 \tag{4}$$

The parameter μ is tuned on held-out data to determine the appropriate amount of regularization. For our experiments in ℓ_2-regularized logistic regression, we used the large-scale implementation in [2].

Another useful form of regularization balances the conditional log-likelihood of correct classification against an ℓ_1-norm penalty on the weight vector $\boldsymbol{w}$. This model (known as ℓ_1-regularized logistic regression) is trained by maximizing the objective function:

$$\mathcal{L}_1(\boldsymbol{w}) = \sum_{i=1}^{n} \ln P(y_i|\boldsymbol{x_i}) - \mu\|\boldsymbol{w}\|_1 \tag{5}$$

Again the parameter μ is tuned on held-out data to determine the appropriate amount of regularization. For our experiments in ℓ_1-regularized logistic regression, we used the large-scale implementation in [4]. An important difference between ℓ_2-regularized and ℓ_1-regularized logistic regression is that the latter favors *sparse* solutions in which many elements of the estimated weight vector are exactly zero. We use this property of ℓ_1-regularized logistic regression for feature validation, as explained in section 4.3.

4.2 Linear support vector machines

SVMs provide another solution to the problem of binary classification. An SVM computes the hyperplane decision boundary that maximizes the margin of correct classification between positively and negatively labeled examples. The maximum margin hyperplane is computed by minimizing an objective function with two terms: one that computes the hinge loss of misclassified (or nearly misclassified) examples, and another that penalizes the magnitude of the weight vector $\boldsymbol{w}$. The objective function can be written as:

$$\mathcal{L}(\boldsymbol{w}) = C\sum_{i=1}^{n} \max\{0, 1 - y_i(\boldsymbol{w}.\boldsymbol{x}_i + b)\} + \|\boldsymbol{w}\|_2^2, \tag{6}$$

where the parameter C (determined by cross-validation) balances the two terms on the right hand side. The objective function in eq. (6) is convex, and hence it has a single (global) optimum. Taking the dual yields the objective function:

$$\tilde{\mathcal{L}}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j (\boldsymbol{x_i}.\boldsymbol{x_j}), \tag{7}$$

which must be maximized subject to the constraints $0 \leq \alpha_i \leq C$ and $\sum_i \alpha_i y_i = 0$. This optimization is an instance of quadratic programming, for which there are many efficient solvers. Finally, the weight vector $\boldsymbol{w}$ is computed from the coefficients $\boldsymbol{\alpha}$ as:

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x_i}. \tag{8}$$

Note that the solution for the weight vector $\boldsymbol{w}$ is expressed as a linear combination of inputs from the training data. The labeled examples in this sum with nonzero coefficients are known as support vectors. Test inputs $\boldsymbol{x}$ are labeled by computing the dot product $\boldsymbol{w}.\boldsymbol{x}$ and thresholding against the bias b (which can be adjusted to trade off false positives versus false negatives). For our experiments with SVMs, we used the large-scale implementation in [2].

4.3 Feature validation

With so many features used for classification, it behooves us to understand their contribution to the decision-making process. We

must especially guard against spurious features that would compromise the integrity of our evaluations. For example, in our training data, the malicious and benign Web pages are collected from different feeds; thus we need to validate that our classifiers are not cuing on artificial characteristics of the feeds that would not generalize to newly collected Web pages. More generally, we must analyze our results to verify that (i) the prediction is not dominated by a handful of possibly spurious features, (ii) a reasonable number of *relevant* features are contributing to the linear decision boundary, and (iii) these contributing features appear to be ones that will generalize to newly collected Web pages.

We used the properties of l_1-regularized logistic regression to address these issues. Recall that in eq. (5), the magnitude of the parameter μ determines the number of non-zero elements in the weight vector $\boldsymbol{w}$ (and thus the number of relevant features that contribute to the model's decision boundary). To investigate the possibility of features that reflected spurious (but distinguishing) properties of the feeds, we trained ℓ_1-regularized models of logistic regression with increasingly high values of μ. If a model in this regime classified very accurately from only a handful of relevant features, then these features were manually investigated. If we discovered features that were correlated with spurious properties of the feeds, then we removed them from subsequent experiments. The results in the next section were obtained after this process of feature verification.

5. EVALUATION

In this section we first give an overview of our experiments and then present our results on classification.

5.1 Overview

We evaluate classifiers on data collected over a period of 10 days. The data consists of roughly 60,000 Web pages with twice as many benign Web pages as malicious ones. We use 70% of the data for training and 30% for testing. We train each classifier over 10 random splits of the data and report the average error rate from these experiments.

We begin by performing a preliminary evaluation of the semantic features to compare tf-idf, bag-of-words, and topic-based representations of each Web page's text. We perform a similar evaluation to compare the visual features from gist, sift-matching, sift-stats, and color-histograms.

Next we train classifiers on the combined features from all the different categories (URL, structural, page link, semantic, visual) under consideration. In follow-up experiments, we then estimate the contributions from individual categories of features. The regularization parameters for LR and SVMs are selected by 10-fold cross validation over the training data.

Each of the URL, structural, page link, semantic and visual features provide different information about the Web page. It is interesting to understand how much the features overlap in terms of the information they provide for classification. We explore this question by computing the agreement between classifiers trained on features from different categories. A low agreement between different classifiers indicates that the features in different categories are providing different types of information for classification.

We validate the contribution of features to classification accuracy using ℓ_1-regularized LR. We count the number of relevant features with non-zero weights and examine the differences between the most significant features with positive versus negative weights.

We generally train and test classifiers with a 1:2 ratio of malicious and benign Web pages. However, we also inspect how the performance depends on this ratio. This is done to show that the classification accuracy does not catastrophically degrade for skewed distributions of positively and negatively labeled examples.

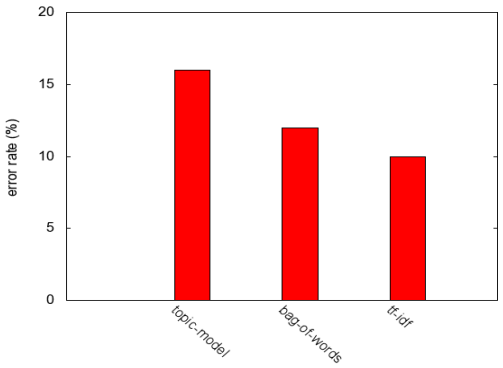

Figure 3: Classification with tf-idf, bag-of-words, and topic-based representation of semantic features.

5.2 Results

We first experiment with NMF to discover topics (i.e., persistent groupings of words) from the visible text on Web pages. To choose a particular number of topics, we examine the approximation error in NMF as the number of topics is increased from 100 to 1500. We choose the number of topics based on the point at which larger matrix factorizations yield diminishing returns in this approximation error. This procedure yields about 500 topics, or equivalently, 500 NMF-derived semantic features per Web page (based on the weights assigned to each topic). Table 1 lists the top weighted words for some of the most recognizable topics (e.g., gambling, home financing, pornography) identified by NMF.

Next, focusing on just semantic features, we compare the results from classification using bag-of-words, tf-idf, and topic-based representations of the visible text on Web pages. Figure 3 shows the results from SVMs on these different representations of semantic features. Tf-idf performs the best, and hence we use td-idf to derive the semantic features in all subsequent experiments.

We perform a similar analysis to determine the useful visual features among the gist, sift-matching, sift-stats and color-histogram features. Table 2 shows the number of visual features from each of these different categories. For sift-matching features we consider two representations, *sift-matching-best* and *sift-matching-all*. Sift-matching-best includes information only for the logos with the highest matching score whereas sift-matching-all includes information for all logos that are candidate matches. The other visual features are dense and potentially expensive to compute for large-scale classification. Given the extra overhead required to extract visual features, it is important to assess the benefits of including them.

We expect the visual features to contribute more when combined with the URL and other content-based features. However, in general we expect that truly useful features will lead to gains in classification even when they are used independently of others. Figure 4 shows the results from SVMs on different types of visual features. We observe that there is no gain from the gist and color-histogram features. Both the sift-matching and sift-stats features are useful, with sift-matching-all performing better than sift-matching-best. Therefore in subsequent experiments, we discard the gist and sift-matching-best features and consider only the sift-matching-all, sift-stats and color-histogram features as visual features. We continue to include color-histogram features because we

hormone	sign	shipping	lesbian	mortgage	error	craps	slim
sildenafil	form	shopping	slut	financing	timed	blackjack	cambogia
medication	case	code	horny	debt	temporarily	slots	polynicotinate
cure	password	discount	busty	wealth	unavailable	roulette	powerslim
sexual	lock	cart	blowjob	bank	server	poker	extract
impotence	caps	refund	tits	mortgages	problem	casino	garcinia
levitra	sensitive	special	pussy	applicants	moments	promotions	excercising
erection	username	checkout	cock	homeowner	busy	games	berry
cialis	log	prices	anal	freedom	proxy	video	antioxidant
viagra	forgot	clearance	ebony	interest	promotions	seven	dieting

Table 1: Top weighted words for some of the topics identified by NMF.

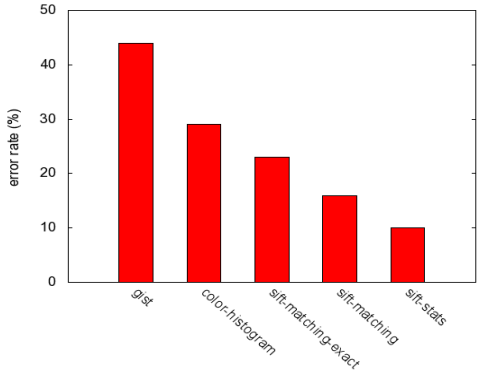

Figure 4: Classification with different visual features.

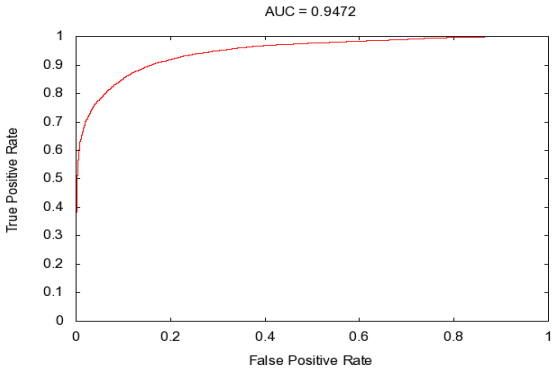

Figure 5: ROC curve for SVM classification on the best combination of features.

Feature	Size
Gist	960
Sift-matching-best	339
Sift-matching-all	53077
Sift-stats	257
Color-histogram	256

Table 2: Numbers of visual features of different types.

Feature	Actual	Relevant	Positive	Negative
URL	153,918	7,608	5,713	1,895
Structural	51,973	4,742	2,568	2,174
Page-link	88,251	9,657	6,556	3,101
Semantic	77,689	8,170	3,634	2,083
Visual	257	257	106	151
Body	272,387	11,368	6,949	4,419
All	426,305	15,470	9,875	5,595

Table 4: Actual and relevant numbers of features from different categories.

suspect that they might be more useful in combination with the sift features.

Next we experiment with SVMs on different combinations of features. We begin by classifying Web page from just their URL features, then examine the gain by adding other types of features. Table 3 summarizes the results from these experiments. In Table 3, feature category URL-MSSV corresponds to the features from the studies of Ma et al. [9, 10] and feature category URL combines URL-MSSV with the additional features proposed in Section 3.1. We find that classification improves significantly when content-based features are included as well as URL features: in particular, we obtain accuracies up to 98.1%, reducing the error rate from URL features alone by more than 50%. The ratio of false positives to false negatives remains about the same for the different feature combinations. We do not observe any noticeable gains in classification accuracy by incorporating sift-matching and the color-histogram features. Thus in subsequent experiments with visual features, we only make use of the sift-stats features. Except for the gist and sift-matching visual features, we note that it is beneficial to combine features from all the other categories. The highest accuracy is obtained by combining URL, structural, page-link, semantic tf-idf and sift-stats features. We consider this combination of features as the optimal feature configuration. Figure 5 presents the ROC curve for classification with this feature configuration.

Next we measure the individual gain in classification accuracy from features in each category—URL, structural, page-link, semantic, and visual. For visual features, we limit the evaluation to only sift-stats features. Figure 6 compares the results from LR versus SVMs using various feature categories. In this figure, LR-1 and LR-2 correspond respectively to LR with l_1-norm and l_2-norm regularization. We observe that LR and SVM yield similar results, with LR-2 performing the best somewhat more often than LR-1 or SVMs.

Table 4 shows the number of features from different feature categories, as well as the number of relevant features from experiments in LR-1 classification. We validate that the classification is not dominated by a handful of features; also the numbers of features with positive and negative weights seem to be almost balanced. The

Features	Error rate(%)	FP	FN
URL-MSSV	4.3	378	405
URL	4.2	372	376
URL + structural	2.6	223	254
URL + structural + page-link	2.5	175	274
URL + structural + page-link + semantic	2.4	187	253
URL + structural + page-link + semantic + sift-matching	2.4	177	260
URL + structural + page-link + semantic + sift-stats	1.9	141	204
URL + structural + page-link + semantic + sift-matching + sift-stats	2.0	149	220
URL + structural + page-link + semantic + sift-stats + color-histogram	1.9	141	205

Table 3: Classification error rates and numbers of false positives and negatives from SVMs trained on different combinations of features.

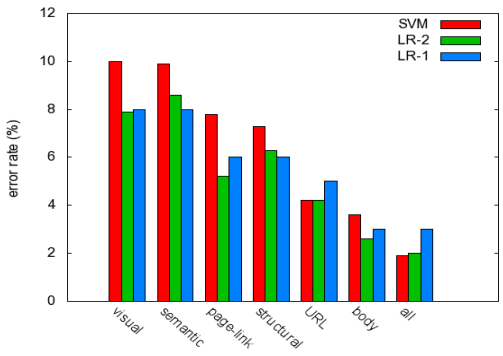

Figure 6: Error rates from different classifiers trained on different categories of features.

	Structural	Page-link	Semantic	Visual
URL	14.1	21.9	14.5	5.0
Structural	-	22.3	19.6	8.9
Page-link	-	-	20.2	9.7
Semantic	-	-	-	8.9

Table 5: Misclassification overlap (%) between classifiers trained on different categories of features.

Ratio	Precision(%)	Recall(%)
1:2	97.6	96.6
1:5	95.4	92.9
1:10	94.5	86.7
1:15	94.9	85.0
1:20	94.2	81.6

Table 6: Precision and recall of SVMs trained on different data sets with varying ratios of positively and negatively labeled examples.

relevant features were identified by regularizing LR-1 as much as possible without compromising classification accuracy. The results for LR-1 in Figure 6 were obtained at this level of regularization.

Table 5 shows how much overlap occurs between the errors of classifiers trained on different categories of features. The reported percentage value is computed with respect to the feature category with the higher number of misclassifications. Low values of overlap indicate pairs of feature categories that seem to provide orthogonal information to the classifier. They also hint at the potential gains from combining features from different categories.

Finally we train linear SVMs on data sets with varying ratios of malicious and benign Web pages (from 1:2 to 1:20). For these experiments we use the combination of feature categories that worked best at the ratio 1:2. We use 70% of the labeled Web pages for training and 30% for testing. The ratio of malicious to benign Web pages is the same in testing as training. For unbalanced data, raw accuracy is not the most informative measure of classification performance; instead we evaluate the classification performance in terms of precision and recall. Table 6 presents the results for precision and recall on data sets with different ratios of malicious and benign Web pages.

6. CONCLUSION

In this paper we presented the design and evaluation of a classifier to detect malicious Web pages. A key contribution of our work was to explore a fuller set of features that can be used for such classification. Our experiments show that a classifier can mine information contained in the URL, structural, page-link, semantic and visual features of Web pages. On the other hand, we found that the color-histogram, sift-matching and gist features did not help to improve classification. Beyond URL features, the most significant drops in error rates were obtained from the structural features and sift-stats visual features. The structural features are easy and fast to compute; thus they are natural candidates to incorporate into large-scale implementations. The visual features are more expensive to compute, though the sift-stats features provide a way to include visual information without a huge computational cost.

Experiments on an up-to-date URL feed illustrate that the proposed approach can identify malicious Web pages at about 98% accuracy. In total, the features we consider from Web page content more than halve the error rate that is obtained from URL features alone. The level of performance worsens but does not significantly deteriorate on unbalanced data sets with different ratios of malicious and benign Web pages.

Finally we discuss some limitations of the present study. We have evaluated our results mainly in terms of classification error rates, or in the case of very unbalanced data sets, precision and recall. But in any deployed system, the utility of a filter for malicious Web pages will depend very much on the user experience. This overall experience will depend on a variety of factors that we have not addressed. Certainly, a classifier with a low error rate may still have too many false positives to be useful in a deployed system. It is clear that the classifier's threshold will need to be

tuned to the relevant part of the ROC curve, and that this tuning can only be done in the context of an actual deployment. To adapt to the non-stationary environment of Web-based attacks, a deployed system would also need to be continually retrained in an online framework [10]. Despite these limitations of our work, we believe that the overall trends of our results remain valuable: they suggest (at least qualitatively) the potential gains from incorporating content-based features into an automatic filter to detect malicious Web pages.

Any deployed system would also operate in an adversarial environment. Therefore it is worth discussing the robustness of our classifiers against an adversary who is attempting to exploit their weaknesses. We have already mentioned how our classifiers utilize many disparate features to detect malicious Web pages: with so many relevant features, the solutions they discover are not easily interpretable. This lack of transparency has its advantages and disadvantages. On one hand, we believe it complicates adversarial strategies to evade detection by simply altering one or a few aspects of a malicious Web page (e.g., removing particular keywords or logos). On the other hand, the very lack of transparency limits the guarantees that we can place on the robustness of these classifiers. Two final points are worth making. First, as mentioned previously, any deployed system would have to be continually retrained in an online setting [10] so that it could adapt to adversaries on the same time scale that adversaries are reacting to its decisions. Second, even if an adversary can reverse-engineer a deployed system to some degree, it remains true that the very presence of a classifier has increased the amount of resources that must be committed to (profitably) operate a malicious Web site.

A final limitation of our study is that we have grouped together all sorts of malicious Web pages when in fact they may rely on entirely different strategies to attract and/or dupe users. For example, some Web pages may have been created for altogether benign purposes, then corrupted by the silent injection of malicious content from an outside party. Different types of malicious footprints are likely to be signaled by different types and combinations of features. Thus further gains seem possible by specializing the methods for feature extraction and model estimation in this work to the detection of particular types of attacks.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback and He "Lonnie" Liu for the use of his SIFT extraction and matching code. This work was supported in part by by National Science Foundation grants NSF-0433668 and NSF-0831138, by the Office of Naval Research MURI grant N000140911081, and by generous research, operational and in-kind support from Yahoo, Microsoft, Google, and the UCSD Center for Networked Systems (CNS).

7. REFERENCES

[1] A. Bergholz, G. Paa, F. Reichartz, S. Strobel, and J.H. Chang (2008). Improved Phishing Detection using Model-Based Features. In *Proceedings of the Fifth Annual Conference on Email and Anti-Spam (CEAS-08)*.

[2] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 9:1871-1874.

[3] I. Fette, N. Sadeh, and A. Tomasic (2007). Learning to Detect Phishing Emails. In *Proceedings of the Sixteenth International World Wide Web Conference (WWW-07)*, pages 649–656. Banff, Alberta, Canada.

[4] K. Koh, S.J. Kim, and S. Boyd (2007). An interior-point solver for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research* 8:1519–1555.

[5] D.D. Lee and H.S. Seung (2001). Algorithms for Non-negative Matrix Factorization. *Advances in Neural Information Processing Systems 14*, pages 556-562. MIT Press: Cambridge, MA.

[6] D.D. Lee and H.S. Seung (1999). Learning the parts of objects with nonnegative matrix factorization. *Nature* 401: 788-791.

[7] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Felegyhazi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, D. McCoy, N. Weaver, V. Paxson, G.M. Voelker, and S. Savage (2011). Click Trajectories: End-to-End Analysis of the Spam Value Chain. In *Proceedings of the IEEE Symposium on Security and Privacy*. Oakland, USA.

[8] D. Lowe (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[9] J. Ma, L.K. Saul, S. Savage, and G.M. Voelker (2009). Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In *Proceedings of the Fifthteenth ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD-09)*, pages 1245-1253. Paris, France.

[10] J. Ma, L.K. Saul, S. Savage, and G.M. Voelker (2009). Identifying Suspicious URLs: An Application of Large-Scale Online Learning. In *Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML-09)*, pages 681-688. Montreal, Canada.

[11] A. Moshchuk, T. Bragin, S.D. Gribble, and H.M. Levy (2006). A Crawler-Based Study of Spyware on the Web. In *Proceedings of the Thirteenth Annual Symposium on Network and Distributed System Security (NDSS-06)*. San Diego, CA.

[12] A. Oliva and A. Torralba (2001). Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision* 42(3): 145–175.

[13] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu (2007). The Ghost in the Browser Analysis of Web-based Malware. In *Proceedings of the USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)*. Cambridge, MA.

[14] S. Sinha, M. Bailey, and F. Jahanian (2008). Shades of Grey: On the effectiveness of reputation-based blacklists. In *Proceedings of the International Conference on Malicious and Unwanted Software (Malware)*, pages 57–64. Alexandria, VA.

[15] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song (2011). Design and Evaluation of a Real-Time URL Spam Filtering Service. In *Proceedings of the IEEE Symposium on Security and Privacy*. Oakland, CA.

[16] Y.M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King (2006). Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In *Proceedings of the Thirteenth Annual Symposium on Network and Distributed System Security (NDSS-06)*. San Diego, CA.

[17] Y. Zhang, J. Hong, and L. Cranor (2007). CANTINA: A Content-Based Approach to Detecting Phishing Web Sites. In *Proceedings of the Sixteenth International World Wide Web Conference (WWW-07)*, pages 639–648. Banff, Alberta, Canada.