Title
Phishingpole: a logo recognition system to detect fraudulent websites

Permalink
https://escholarship.org/uc/item/4c02n8w4

Author
Becerra-Licha, Sebastian

Publication Date
2009

Peer reviewed|Thesis/dissertation
Phishingpole: A logo recognition system to detect fraudulent websites

A Thesis submitted in partial satisfaction of the requirements for the degree Master of Science in Computer Science by

Sebastian Becerra-Licha

Committee in charge:
Professor Stefan Savage, Chair
Professor Serge Belongie
Professor Geoff Voelker

2009
The Thesis of Sebastian Becerra-Licha is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2009
DEDICATION

This thesis is dedicated to my parents: to my mother, who tirelessly home-schooled me from kindergarten through high school, and to my father who ignited my passion for computers from an early age and showed me the wonders of the Internet. Thank you.
# TABLE OF CONTENTS

Signature Page ........................................ iii
Dedication .............................................. iv
Table of Contents ....................................... v
List of Figures .......................................... vii
List of Tables .......................................... ix
Acknowledgements ...................................... x
Abstract ................................................ xi

Chapter 1 Introduction ................................... 1

Chapter 2 Background .................................... 5
2.1. Finding phish ........................................ 6
2.2. Defending against phish ............................... 7

Chapter 3 Logo Matching ................................. 9
3.1. Overview .............................................. 9
   3.1.1. Feature extraction ............................... 9
   3.1.2. Logo retrieval .................................. 10
3.2. Implementation Details .............................. 12

Chapter 4 Controlled Evaluation ......................... 14
4.1. Hand labelled site data ............................. 14
4.2. Logo image data ..................................... 15
4.3. Calibration experiment ............................... 15
4.4. Adversarial experiment ............................. 16

Chapter 5 Live Evaluation ............................... 21
5.1. Spamsscatter feed classification .................... 21
5.2. Webmail feed classification ......................... 21
   5.2.1. Methodology .................................. 22
   5.2.2. Results ...................................... 27

Chapter 6 Error Analysis ............................... 30
6.1. Logos that lack detail .............................. 30
6.2. Logos with mostly text ............................. 31
6.3. Syndicated logos .................................... 32
LIST OF FIGURES

Figure 3.1: Given an input screenshot, we first run SIFT to discover keypoints in the image. These keypoints are invariant to scale and orientation and are used when matching against logos in our database. Next, we search our database of company logos and find the logo with the highest proportion of matching keypoints to those from the screenshot. If the proportion is above a threshold, the logo is returned as a match. 12

Figure 4.1: The distribution of logos in the Spamsscatter data set. Our method achieved a recall rate of 95.2%. 15

Figure 4.2: ROC curve for the calibration test. The equal error rate (EER) occurs at a threshold of 0.1968, denoted by the red X. In our experiments, we chose a threshold of 0.3 to distinguish false positives. 16

Figure 4.3: Results of logo matching after applying rotation transformations. 17
Figure 4.4: Results of logo matching after applying scaling transformations. 18
Figure 4.5: Results of logo matching after applying shearing along the x-axis transformations. 19
Figure 4.6: Three versions of the Wells Fargo logo: The original, sheared by 10 degrees, and sheared by 20 degrees. After inserting these transformed images back into the screenshot, the proportion of matching keypoints to the Wells Fargo logo in our reference data base are 0.86, 0.44 and 0.11, respectively. A shear of 20 degrees causes the match proportion to fall below our threshold of 0.3. 19

Figure 5.1: The screenshot generation pipeline: The URL database is populated by our data source in realtime. The Coordinator polls for new URLs, and instructs the Selenium Remote Controls to take screenshots. The screenshots are then stored on a central file server. 23

Figure 5.2: The screenshot processing pipeline: Screenshots are divided into slices and these slices are assigned to nodes. In the first stage, each node preprocesses the images. In the second stage, the logo matching algorithm is performed. The data is then gathered by the Coordinator and the URL database is updated. 24

Figure 5.3: Running time of the logo matching algorithm for a varying number of screenshot keypoints. The green mark at 1,246 features indicates the average number of screenshot keypoints and the red marks are one standard deviation away at 369 and 2,143. The running time at the mean of 1,246 features is 1.9 seconds. 26
Figure 5.4: Distribution of the most commonly-matched logos in the web-mail data set.

Figure 6.1: The Orange logo is not suitable for matching using our SIFT-based approach. The logo lacks enough detail to generate a reliable set of feature points for matching. Here it is shown in its original form on the left, and after preprocessing on the right.

Figure 6.2: An example of a false positive; the CNN logo is displayed in the green box and a non-CNN screenshot is shown below it. The matching feature points between the string portion of the logo and the text in the screenshot triggered a false positive.
TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 5.1</td>
<td>Average running time to process a single screenshot.</td>
<td>25</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>False negative counts from evaluating 500 negative classifications. Upon inspection, we discovered that we had a different version of the yahoo.com logo in our reference data set, leading to the high number of unclassified screenshots from that company. Ignoring the false negatives from yahoo.com, we measured a false positive rate of 1.2%.</td>
<td>27</td>
</tr>
<tr>
<td>Table 6.1</td>
<td>Breakdown of the logos that were matched in syndicated form. Of the 1,920 total logo matches, 83 of them (4%) were in syndicated form.</td>
<td>32</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor, Professor Stefan Savage, for his support and guidance. This project is his brainchild, and I hope that I have done it justice.

Many thanks are also due to Serge Belongie and Kai Wang for their help on the image recognition aspects of this project, in addition to the numerous miscellaneous details with which they helped me.

I would also like to thank Kaisen Lin, Jim Hong, and Tess Winlock for their moral support—I do not think that I could have made it to the finish line without them.

This thesis is based on a paper titled “A logo recognition system to detect fraudulent websites” that has been submitted for publication in ACM CCS, 2009, Becerra, Sebastian; Wang, Kai; Belongie, Serge; Savage, Stefan. The thesis author was the primary investigator and author of this paper.
ABSTRACT OF THE THESIS

Phishingpole: A logo recognition system to detect fraudulent websites

by

Sebastian Becerra-Licha

Master of Science in Computer Science

University of California, San Diego, 2009

Professor Stefan Savage, Chair

Fraudulent web sites are designed to fool users into believing they represent genuine brands, such as banks or e-commerce sites. In doing so, these sites inevitably make use of branded logos which are among the strongest visual marks for a brand and well-established as key “trust cues” to typical users. This thesis explores the efficacy of image-based logo recognition as a tool for detecting or classifying such sites. We show that modern vision algorithms can provide accuracy above 99%, offering a significant new capability for improving response time and reducing overhead in combating such scams.
Human behavior fundamentally relies on trust. In our daily lives we presume that, by and large, the information we are given is factual and generally benign. This assumption allows communications to be cheap and allows the tremendous scale of today’s transactional commerce. At the same time, this trust is easily exploited; scammers may present themselves as a trusted party and thereby obtain otherwise confidential information. In the world of online commerce, this problem is exemplified by social engineering attacks - typically via an email vector (phishing) or DNS poisoning (pharming) - to fraudulent websites purporting to represent a particular institution. Since these sites provide all the right visual trust cues (style, logos, buttons, etc) users are easily fooled into providing their credentials, which are captured and monetized by the scammer.

Unfortunately, these attacks are successful precisely because it is difficult for a lay user to distinguish between the genuine version of an e-commerce or banking site and a fraudulent imitator. In one recent study, 90% of users were fooled by high-quality phishing sites [8]. Indeed, the sites may in fact be visually indistinguishable even to an expert. Thus, the mechanisms used to combat fraudulent Web sites typically involve external efforts that identify such sites and then distribute their IP addresses. These in turn are used to populate blacklists (e.g., for anti-phishing toolbars [29] or spam filters) and to drive site takedown efforts [21, 23, 22].

In this effort, two types of information are necessary to identify a site as fraud-
ulent. First, it must be determined that the site contains content that would lead a user to believe that it represents a particular brand-holder (e.g., Washington Mutual). Second, it must be established that the site is in fact fraudulent (it is not being operated by the brand-holder or by a third-party authorized to represent the brand-holder). This latter constraint is typically inferred via the communication vector (e.g., since Washington Mutual does not send spam e-mails directing its users to login) or via the particular place the site is hosted (e.g., since Washington Mutual’s on-line banking sites are not typically hosted at Geocities). However, the first classification can be considerably more challenging since it requires interpreting the meaning of a Web page. Consequently, while anti-phishing blacklists are frequently populated by both automated sources (e.g., heuristics looking for common fraud patterns in e-mail text) and individual user reports, it is common for such submissions to be manually vetted before they are exported for use in a live defense. This is done to protect against accidental false-positives or explicit poisoning. Thus, the human effort to manage anti-phishing efforts inevitably scales with the number of sites being reported. Moreover, until a site is reported to a blacklist and this report is visually inspected by a trusted party, even users who subscribe to such blacklists are vulnerable to the site; which is to say nothing of the imperfect information sharing between those who gather intelligence on fraudulent sites [22].

In this thesis, we describe a new tool in the anti-phishing arsenal designed to improve this state of affairs: image-based logo matching.

The premise behind this idea is that a critical element in virtually all fraudulent sites is the brand mark, or logo, of the institution being imitated. Indeed, in our experience, all but the lowest quality phishing campaigns make use of such logos. This is not surprising, as logos are both designed for easy recognition and heavily marketed by brand-holders and thus their association with a given institution is deep. In one controlled study, Dhamija et al. found that content features, such as logos and other design elements, are treated as “trust cues” by virtually all users (moreover, for 25 percent of users these were the only trust cues validated). Indeed, absent a Washington Mutual logo, it seems unlikely that a fraudulent Washington Mutual banking site could achieve
Thus, were it possible to accurately and automatically determine when a Web page contains the logo of a bank or e-commerce site, this would provide strong circumstantial evidence that the site was representing the associated institution. This in turn suggests two possibilities for integrating such a capability into on-line defenses. First, within the existing anti-phishing regime, candidate fraudulent sites could be automatically compared against a corpus of popularly defrauded logos – thereby significantly winnowing the set of sites requiring manual vetting and thus shrinking response time. More radically, brand-holders might identify the set of IP addresses authorized to use a logo for commerce purposes (e.g., in much the same way that DomainKeys [17] and SPF [27] authorize the use of domain names for sending e-mail). Individual client browsers could then look for matching logos as site pages were rendered (perhaps popping up a window to the effect of “This site is using a Washington Mutual logo and is not authorized by Washington Mutual. Be careful not to provide any personal or confidential information on this site”).

However, while the phishing kits used by many fraudulent sites copy logos directly from the genuine site, an approach that required bit-for-bit equivalency would be fragile to minor transformation of the logo images. Instead, we apply image matching techniques from the computer vision field that achieve high matching accuracy that is invariant across a wide range of transformations (scale, rotation, stretching, color, etc.).

In the remainder of this thesis, we describe this technique and our experience with it in the following structure: in Chapter 2, we provide more background on fraudulent sites and describe related work focused on combating it, after which we describe the SIFT approach to image matching and our adaptation for the logo problem in Chapter 3. In Chapter 4 we demonstrate this technique in a controlled setting — showing both logo separation and the ability to match transformed images from a single reference logo. In Chapter 5, we show results on two data sets of spam and phishing-attempt sites: one from the Spamscatter [4] project, and the other from a major webmail provider. We demonstrate recall rates of 95.2% and 99.5% respectively, while keeping false positive
rates below 2% in these experiments. In Chapter 6, we analyze the characteristics of the logos that are challenging for our approach. We then conclude in Chapter 7 by recap-ping the opportunities and practical challenges in using an image-based logo matching approach to detect phishing sites.

This chapter, in full, is based on material from a paper titled “A logo recognition system to detect fraudulent websites” that has been submitted for publication in ACM CCS, 2009, Becerra, Sebastian; Wang, Kai; Belongie, Serge; Savage, Stefan. The thesis author was the primary investigator and author of this paper.
Chapter 2

Background

Fraud is as old as recorded history. However, the ease and anonymity afforded by the Internet allow fraud to be trivially perpetrated through a wide range of vectors including e-mail (phishing), DNS poisoning (pharming), typo-squatting (e.g., registering citybank.com to capture customers who mistype citibank.com) and search engine “optimization” (vectoring users via a search engine results for a particular institution’s site). The most popular method by far is phishing, because the e-mail vector lends itself to tremendous scale. ¹

A single phishing campaign may target millions of users, enticing them (either with fear – an urgent issue that must be dealt with immediately – or opportunity – a special deal or reward) to visit some doppelganger site that may be visually and functionally identical to the purported site. Many users have relatively simple fraud-detection strategies and thus are easily tricked by such subterfuge into providing their site-specific credentials [8, 9, 28]. These credentials, in turn, can be monetized by the scammers either directly or via third-parties who liquidate accounts for a commission [23, 26, 2]. While loss estimates vary considerably (a 2007 Gartner survey estimated losses of $3.2B, while Moore and Clayton provided a measurement-based estimate of $320M) there is broad agreement that they are sizeable [21].

Consequently, fraudulent sites and phishing in particular has inspired a wide

¹For the remainder of this thesis we use phishing to refer to all such fraudulent sites, although we note here that our approach does not depend on the particular vector used since it focuses on the content of the site itself.
range of defensive efforts. We can think of these in two dimensions: how phishing sites are identified and how this information is used to deploy a meaningful defense. We review each in turn.

2.1 Finding phish

Candidate phishing sites are typically detected via the content in either the e-mail vector and/or on the Web site. To convince recipients to visit the embedded URL pointing to a phishing site, scammers must communicate some urgent imperative in the envelope (“Your account has been compromised, please change your password” or “Login now to enter a special lottery”) as well as identifying the institution. Thus, one approach is to use machine learning techniques to automatically classify e-mail text as being “phishy” or not [10, 3]. Similarly, the embedded URL may itself have properties indicative of a phishing site (e.g., the presence of a bank name to the right of the domain name) and several efforts have focused on automatically classifying URLs in a similar manner [13].

Similarly, actual site content, combined with the URL used to reach it, can be mined for heuristic tokens common to phishing sites [7]. Another innovative approach creates textual fingerprints of a site and then uses popular search engines to validate that the site is one of the top 10 hosting said textual content [1]. Similar to our own work, there have been several efforts to use vision algorithms in identifying phishing pages [12, 6], although these focus on matching whole pages to reference pages from the brand-holder, while we focus exclusively on the logo (which we argue is intrinsically essential to virtually all phishing campaigns). The most similar effort to our own is a proprietary technology called ImageFlare, licensed by Envisional, that matches logos against reference images. However since it is commercial offering there is little information about how it works and we were unable to provide further comparison or evaluation against it.
2.2 Defending against phish

These techniques are in turn deployed in three rough classes of efforts: server-based e-mail techniques that filter inbound e-mail (the principal vector for fraudulent sites); browser-based client techniques that filter outbound site requests; and takedown efforts that shutdown the servers hosting individual sites and the domain names that are used as vectors. Each of these techniques has its pros and cons.

E-mail filtering can protect modest numbers of users (i.e., all using the same mail server) and prevents users from even receiving the vector. However, it also suffers from the same kinds of false positive problems and evasion issues that define the arms race of all spam filtering.

Browser-based filtering can protect a large number of users (e.g., virtually all users of Internet Explorer or Firefox) and acts by preventing access to phishing sites as they are requested or rendered. These solutions fall into two categories: those that use local heuristics [7] and those that use real-time blacklists [29]. The former are inherently proactive, but are only effective for sites that match the particular heuristics. In general this approach underperforms blacklist-based anti-phishing solutions. The latter approach is reactive – requiring the blacklist provider to first have received a sample of the phishing e-mail, visited the associated site, validated that the site is indeed fraudulent, and make this information available to the client. Thus, there can be a considerable window of vulnerability between when a phishing site first goes up and when it is found, processed, and made available in a browser blacklist. Moreover, depending on how blacklist information is communicated, this approach may add latency to all Web page accesses (as the site is looked up via a blacklist server).

Finally, there are both volunteers and commercial organizations who work to disable fraudulent Web sites or the domain names they use in phishing e-mails. This approach has the advantage of protecting all users, but typically has much higher latency than other reactive methods, since it requires cooperation between security organizations, ISPs, hosting providers, brand-holders and registrars. Takedown efforts typically
use the same kinds of information used by third-party blacklist providers and also include the requirement for manual vetting. There is considerable variability in takedown time, with Moore and Clayton reporting average takedown times of 64 to 94 hours for typical sites [23], but with dramatically better response time for brands represented by commercial takedown companies [22].

We believe that our work will both provide a useful tool for reducing the overhead (and hence latency) of existing classification and vetting efforts, and present the opportunity for a proactive approach based on explicit authorization of brand marks.

This chapter, in full, is based on material from a paper titled “A logo recognition system to detect fraudulent websites” that has been submitted for publication in ACM CCS, 2009, Becerra, Sebastian; Wang, Kai; Belongie, Serge; Savage, Stefan. The thesis author was the primary investigator and author of this paper.
Chapter 3

Logo Matching

3.1 Overview

As previously stated, our overall approach consists of detecting phishing websites by taking screenshot images, extracting their features using SIFT and finding the best matching company logo in our database. Here, we describe each step in more detail.

3.1.1 Feature extraction

The process by which SIFT extracts and represents features from an image provides invariances to scale and rotation and is resilient to photometric variation. Keypoint discovery is performed, which produces keypoint descriptors.

Keypoint discovery  This is done by performing a search for extrema in an image’s scale-space. The purpose of this is to identify locations of the image that can be repeatedly discovered under different views of an object. This is done by applying a Difference-of-Gaussians (DoG) filter to the image and searching for local extrema in the result. Convolving a Gaussian filter with the input image is effective in reproducing the input at a different scale, determined by the standard deviation of the Gaussian. A DoG filter is the result of subtracting the result of convolving a Gaussian filter with an image at two adjacent scales. After the DoG has been calculated for an image, extrema
are found by comparing every point in the output and checking if it is either above or below its 8 neighbors in the point’s original scale, and its 9 neighbors in the scales directly above and below. If a point passes this test, it is considered an extremum in scale-space. Points are further pruned by rejecting those that have low contrast, as they are considered to be unstable. Points that pass these tests are assigned descriptors based on their local pixel neighborhoods.

**Keypoint descriptors** Keypoints are assigned descriptors whose statistics are measured relative to the keypoint’s orientation to preserve rotation invariance. The orientation of a keypoint is calculated by creating a histogram of the gradients in a region around the keypoint. The bins of the histogram represent gradient directions and the orientation of the largest bin is assigned to the keypoint. This histogram contains 36 bins to cover 360 degrees of orientations.

To calculate the descriptor for a keypoint, the image gradients in a local window around the point – using its original scale – are calculated, rotated relative to the keypoint orientation, and placed into histograms. These histograms contain 8 bins representing equally spaced gradient orientations. A 16×16 pixel window around a keypoint is considered. This window is divided into 4×4 subregions and the gradients in each subregion are added to the same histogram. Thus 16 histograms of 8 bins each are calculated. Concatenating these together gives us a 128 element descriptor vector for a keypoint. To be robust to illumination changes, the vectors are normalized to unit length, making the feature vector invariant to affine changes in illumination.

### 3.1.2 Logo retrieval

After features are extracted for an input screenshot image and a database of reference logos, we next need to retrieve the best matching logo. In this case, the best matching logo contains the highest proportion of keypoints matching the screenshot. The distance between keypoints is defined as the Euclidean distance between the two feature vectors. A keypoint match is one where the distance between its nearest neighbor
keypoint in the logo and its second nearest neighbor are above a certain threshold. In practice, this threshold is 0.8. Distinguishing matches using the top two matches has shown to be an effective way of separating object matches and background clutter.

Logo retrieval becomes a problem of finding the nearest two neighbors to all the keypoints in a screenshot; it is done using an efficient approximate presented in [5]. Given the features of screenshot and logo, a modification of a k-d tree, called Best-Bin-First (BBF) [5], is applied to find approximate nearest neighbors, which has been shown to improve performance by about 2 orders of magnitude while remaining more than 95% accurate in discovering nearest neighbors. For all logos, we would like to know the proportion of their feature points that match to the screenshot and return the one with highest proportion. However, when the number of logos is high – in our case 168 – comparing every pair of logos becomes an issue for performance, especially when might we want to run our matching near real-time within a browser rather than process the screenshots offline on a cluster of machines. To improve performance, we take a similar coarse-to-fine approach as [24] to narrow down the number of potential logos to perform comparisons over.

In our system, we use the features from the screenshot image to vote for which logos to do exhaustive comparisons on, then compare those logos to the input one-by-one. First, we load the features of all the logos into a k-d tree with BBF search – as in the logo to screenshot comparison – while maintaining a pointer to the original logo from which the feature was extracted. Next, we take the features from the screenshot and find their matches among the logo features. We consider every feature match a vote for the logo from which it came and return the $K$ logos with the most votes. We perform matching with those $K$ logos and return the logo with the highest proportion of keypoint matches. In practice, we select $K = 30$. Next, we test the suitability of this approach to recognizing logos in a controlled setting.
Figure 3.1: Given an input screenshot, we first run SIFT to discover keypoints in the image. These keypoints are invariant to scale and orientation and are used when matching against logos in our database. Next, we search our database of company logos and find the logo with the highest proportion of matching keypoints to those from the screenshot. If the proportion is above a threshold, the logo is returned as a match.

3.2 Implementation Details

While exact image matching is straightforward and efficient, allowing for approximate matches is not as obvious a task, even to heavily rotated or transformed versions of an image. However, with the introduction of the Scale Invariant Feature Transform (SIFT) [18] in 2004, the problem of near duplicate image retrieval has seen many advances and is regarded today as a solved problem within the vision community. Broadly speaking, SIFT takes an image and discovers keypoint locations that are invariant to scale and orientation; keypoints that remain discoverable under image resizing and rotations. These locations in the image are assigned descriptors that capture statistics calculated in a local pixel window around each location. Image retrieval is then a matter of matching the features in an input image to a set of reference images. This method of determining keypoints and their representation has proven to be a robust method of image matching, and makes it an ideal algorithm for detecting fraudulent pages that contain logos of legitimate companies intending to deceive web users. Next, we outline our application of this algorithm.

In our system, we have a database of reference logos with their features previously extracted. To detect whether a website is potentially fraudulent, we take a screenshot of the page, extract its features using SIFT, and find the best logo match in our database. A visual depiction of our pipeline is shown in Figure 3.1. When the best
matching logo has a proportion of its keypoints matching above a parameterized threshold, we consider that a positive match.

This chapter, in full, is based on material from a paper titled “A logo recognition system to detect fraudulent websites” that has been submitted for publication in ACM CCS, 2009, Becerra, Sebastian; Wang, Kai; Belongie, Serge; Savage, Stefan. The thesis author was the primary investigator and author of this paper.
Chapter 4

Controlled Evaluation

To validate our approach, we first perform two experiments in a controlled setting. First, we perform a calibration experiment that measures the ability of the logo matcher to discriminate between different logos. The second experiment is based on the assumption that if the techniques we describe are widely used, phishers will attempt to circumvent logo matching by applying various affine transforms to logos. We call this the adversarial experiment.

4.1 Hand labelled site data

We acquired a current data set of screenshots of Web sites advertized in spam, graciously provided by Anderson et al. from their Spamscatter project [4]. In this data set, 125 screenshots were hand-categorized as containing company logos while the other 5,629 did not. The distribution of company logos (as determined manually) in this data set is shown in Figure 4.1. We evaluate our logo matching approach on this data set in live evaluation tests in Chapter 5. We use the screenshots that contained no logos in both the controlled and live evaluations.
Figure 4.1: The distribution of logos in the Spamscatter data set. Our method achieved a recall rate of 95.2%.

### 4.2 Logo image data

We obtained a list of over 150 commonly phished domains from the Spam URI Realtime Block Lists website\(^1\). After discarding domains with nonresponsive websites, we extracted the logo of each domain for a total of 158 logos. We then added the logos that we found in the Spamscatter dataset for a total of 166 logos in our reference database.

### 4.3 Calibration experiment

First, we test the discriminative power of our approach by evaluating it on a data set of 166 screenshots where every logo appears once, and 5339 screenshots where none appear. Since our approach is focused on detecting near-duplicate logos, a data set with one positive instance of each logo is sensible and tests the method’s ability to distinguish between the 166 logos. Additionally, testing on screenshots without logos

---

\(^1\)http://www.surbl.org/
allows us to evaluate how accurate the method is in distinguishing false negatives.

In this test, we achieved a recall of 92.17%: of the 166 screenshots that contained logos, we correctly categorized 153 of them. Figure 4.2 shows an ROC curve for distinguishing true positives and false positives. For this experiment we used a *global* threshold of 0.3, meaning that if the best logo match covered under thirty percent of keypoints then the screenshot was classified as having no logo. We achieved a false positive rate of 1.8%, where 100 out of the 5339 negative screenshots were classified as containing a logo in our database. The next experiment tests the resiliency of our approach to different image transformations.

![ROC curve for calibration data](image)

**Figure 4.2**: ROC curve for the calibration test. The equal error rate (EER) occurs at a threshold of 0.1968, denoted by the red X. In our experiments, we chose a threshold of 0.3 to distinguish false positives.

### 4.4 Adversarial experiment

This experiment tests the performance of our approach in the face of three affine transformations an adversary might apply to a logo to circumvent our system: rotation, scale, and shear. For each of 5 chosen logos, we took a screenshot of its
corresponding homepage and manually placed keypoints around the logo so that it could be replaced with a transformed version of the logo. We then ran the matching algorithm for this modified screenshot and the original logo, and measured how strong a match resulted. In Chapter 6 we take a closer look at common failure points in our approach.

![Varying rotation](image)

**Figure 4.3:** Results of logo matching after applying rotation transformations.

Figure 4.3 shows that our method is tolerant to logo rotation–even when a logo is rotated 50 degrees, a very noticeable amount to even the least attentive prospective phishing victim, the logos match well above the 0.3 threshold.
Figure 4.4: Results of logo matching after applying scaling transformations.

Figure 4.4 shows that 3 of the 5 logos are tolerant to logo scaling between 70% and 130%, although the Wells Fargo and USAA logos are not matched when they are scaled down below 100%. This is due to the small size of their respective logos; as the logo is scaled down, detail is lost and the SIFT algorithm recognizes fewer keypoints.
Figure 4.5: Results of logo matching after applying shearing along the x-axis transformations.

Figure 4.6: Three versions of the Wells Fargo logo: The original, sheared by 10 degrees, and sheared by 20 degrees. After inserting these transformed images back into the screenshot, the proportion of matching keypoints to the Wells Fargo logo in our reference database are 0.86, 0.44 and 0.11, respectively. A shear of 20 degrees causes the match proportion to fall below our threshold of 0.3.

Figure 4.5 shows that our method is tolerant to 10 degrees of shear along the x-axis, but its performance suffers at 20 degrees and beyond. Figure 4.6 shows what the Wells Fargo logo looks like after being sheared 10 and 20 degrees along the x-axis; this is encouraging, as it means that phishers need to drastically alter a logo in order to avoid
detection via our method.

This chapter, in full, is based on material from a paper titled “A logo recognition system to detect fraudulent websites” that has been submitted for publication in ACM CCS, 2009, Becerra, Sebastian; Wang, Kai; Belongie, Serge; Savage, Stefan. The thesis author was the primary investigator and author of this paper.
Chapter 5

Live Evaluation

We performed evaluation on two data sets containing screenshots of sites marked as potentially spam or phishing sites: one acquired from Spamscatter and the other from a webmail provider.

5.1 Spamscatter feed classification

Recall that this data set consisted of 5,629 images, with 125 of them containing company logos with the distribution shown in Figure 4.1. The set of negative screenshots without logos was the same set used in the earlier calibration experiment. We achieved a recall of 95.2% while still using a threshold of 0.3 for distinguishing false positives. Since we used the same negative screenshot set, the false positive rate was again 1.8% among the 5339 screenshots without logos.

5.2 Webmail feed classification

We were provided access to an live feed of URLs that were suspected to be phishing or spam by a large webmail provider. This feed was labeled in an automated manner by the provider and, as a result, is very noisy both in terms of false positives and unreachable URLs. The URLs in the data feed have a very short shelf-life; in general phishing sites have median lifetimes of only a few days, and well under a day for sites...
monitored by “takedown” companies[21, 22]. These sites are often hosted on hacked servers or on botnets, and they commonly employ techniques such as fast-flux DNS in order to make it more difficult for the anti-phishing community to identify phishing attempts [20]. Therefore, it is crucial to visit the URLs as soon as possible in order to generate screenshots before they become unavailable. We gathered 11 days worth of data and processed a total of 1,226,953 URLs.

### 5.2.1 Methodology

#### Data collection framework

Our main implementation concern was to ensure that the framework scales well with respect to URL volume so that screenshots could be captured within hours of a new URL being received.

#### Selenium framework

In order to keep up with the URL volume (on average there are 108,000 new URLs per day), we decided to use the Selenium framework [14] to generate screenshots for each reachable URL. Selenium is designed with the automation of website unit tests in mind. It is especially useful for testing complicated AJAX functionality, as it has the ability to programatically control several web browsers, including Firefox and Internet Explorer. A single Selenium Hub can be used to control several Remote Control (RC) nodes, in order to take screenshots of URLs. This distributed framework allows us to deal with the high URL volume coming from our data source.

#### Screenshot generation pipeline

Our data collection framework pipeline is depicted visually in Figure 5.1: a Coordinator polls the URL database and issues requests to the Selenium Hub, which then offloads the requests to the RC processes. A computer can run multiple RC instances. Each RC can drive one web browser, performing actions such as visiting a
Figure 5.1: The screenshot generation pipeline: The URL database is populated by our data source in realtime. The Coordinator polls for new URLs, and instructs the Selenium Remote Controls to take screenshots. The screenshots are then stored on a central file server.
Figure 5.2: The screenshot processing pipeline: Screenshots are divided into slices and these slices are assigned to nodes. In the first stage, each node preprocesses the images. In the second stage, the logo matching algorithm is performed. The data is then gathered by the Coordinator and the URL database is updated.

URL, waiting for a page to load, and taking a screenshot. When a screenshot has been taken by an RC, it is stored on a file server.

The RCs run a customized version of Firefox that has certain aspects of javascript disabled; in particular, the `window.alert()`, `window.prompt()` and `window.confirm()` functionality is disabled, as the RC instances are not configured to handle these dialogs. Flash objects and Java applets are supported.

We used a total of 7 Amazon EC2 m1.small (1.7 GB memory, 1 virtual core) instances, each running 10 RC instances and processing an average of 4504 URLs per hour.
Table 5.1: Average running time to process a single screenshot.

<table>
<thead>
<tr>
<th>Preprocessing (grayscale &amp; cropping)</th>
<th>0.72s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing (feature extraction)</td>
<td>2.85s</td>
</tr>
<tr>
<td>Matching algorithm</td>
<td>1.90s</td>
</tr>
<tr>
<td>Total</td>
<td>5.47s</td>
</tr>
</tbody>
</table>

**Screenshot processing pipeline**

Once the screenshot of a webpage is captured, it is stored on a file server. The screenshots are then divided into slices to be processed in parallel among several nodes. The screenshot processing pipeline is shown in Figure 5.2. The screenshots are first preprocessed: they are cropped to a maximum size of 800x600 and converted to grayscale. Afterwards we extract their keypoints using the SIFT feature extraction algorithm described in Section 3.1. After preprocessing, we run our logo matching procedure to find the logo that most closely matches the screenshot.

**Running time**

Table 5.1 lists the average of each step running time per screenshot. Times were measured on Amazon EC2 m1.small instances.

The average screenshot had 1,246 features, with a standard deviation of 877, while outliers were observed to have as many as 11,000 features. These outliers were special cases of the background containing patterns of high detail. One example of a screenshot generating more than 11,000 features was a website containing a crosshatch textured CAPTCHA. As shown in Figure 5.3, our logo matching algorithm scales exponentially with the number of screenshot feature points, which made it necessary to examine a maximum of 5000 features per screenshot, ignoring the rest of the features. However, in our experiments, we rarely observed this behavior.
Figure 5.3: Running time of the logo matching algorithm for a varying number of screenshot keypoints. The green mark at 1,246 features indicates the average number of screenshot keypoints and the red marks are one standard deviation away at 369 and 2,143. The running time at the mean of 1,246 features is 1.9 seconds.

Filtering false positives

To restrict our logo recognition to sites that were potentially phishing attempts, we filtered out logo matches in URLs from the same domain. For example, when the best logo match for a URL of the form http://google.com/group/[./]* is the Google.com logo, we did not consider that a phishing attempt. URLs in these domains are known to contain their own company’s logo, and in these cases are spam rather than phishing. If, however, the best match was the Ebay.com logo in the Google.com domain, we did consider it a phishing attempt. In general, we filtered out matches where the URL is of the form http[s]://[./]*X.com/[./]* where the best matched logo was from company X.

Unreachable URLs

We found that a very high fraction (83%) of the URLs were not reachable, either due to DNS errors, 404 errors, or server misconfigurations. This left us with
Table 5.2: False negative counts from evaluating 500 negative classifications. Upon inspection, we discovered that we had a different version of the yahoo.com logo in our reference data set, leading to the high number of unclassified screenshots from that company. Ignoring the false negatives from yahoo.com, we measured a false positive rate of 1.2%.

<table>
<thead>
<tr>
<th>Logo</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>yahoo.com</td>
<td>48</td>
</tr>
<tr>
<td>google.com</td>
<td>2</td>
</tr>
<tr>
<td>bankofamerica.com</td>
<td>1</td>
</tr>
<tr>
<td>paypal.com</td>
<td>1</td>
</tr>
<tr>
<td>mastercard.com</td>
<td>1</td>
</tr>
<tr>
<td>networksolutions.com</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total (%)</strong></td>
<td>54 (10%)</td>
</tr>
<tr>
<td><strong>Total w/o yahoo.com (%)</strong></td>
<td>6 (1.2%)</td>
</tr>
</tbody>
</table>

208,582 reachable URLs to perform logo matching.

5.2.2 Results

We validated all the screenshots that resulted in a match above our threshold of 0.3, and validated 500 randomly selected screenshots that fell below the threshold. Our match set contained 1920 screenshots. Of these 1920 screenshots, we correctly classified 1910 of them with a logo appearing on the page for a recall of 99.5%. We analyze the failure cases in Chapter 6.

Among the 500 selected screenshots below our threshold, 54 of them contained logos in our reference logo set. The details are shown in Table 5.2. However, after inspection of all 54 cases, we found that the screenshots with the most unmatched logos were those corresponding to a version of the Yahoo logo that was different from the one that we had in our reference set. Accordingly, we consider a more accurate false positive count to be closer to 6, excluding those that contained the Yahoo logo. This results in a false positive rate of 1.2%. 
Figure 5.4: Distribution of the most commonly-matched logos in the webmail data set.

Most matched logos

Figure 5.4 displays the distribution of matched logos in the webmail data set. The top two, Paypal and Ebay, are consistent with the results found in [21].

Hidden iframes

It is interesting to note that even after filtering out false positives by domain name, Google remained high on the logo match list. We examined a subset of these sites more closely and discovered that some loaded a hidden iframe with the actual Google site – and appeared to be overlaying advertisements via the frame. In other cases, these sites just issued a location redirect header to the real Google website. The hidden iframes are likely being used as an infection vector for drive-by downloads as discussed in [25]. Currently we do not have the capability to keep track of redirects, as everything is stored under the original URL, but it would be interesting to analyze this behavior further.
Social Networks

There were 11 phishing attempts for Facebook and 1 for Friendster in the 11 days worth of data that we analyzed. This merits further inspection, as it indicates that at least some phishers are beginning to look towards social networks as potential targets for phishing attempts. Jagatic et al discuss the phenomenon of phishers using social networking sites such as Facebook and Myspace to harvest data and deliver targeted ‘spear-phishing’ attacks, although they do not discuss the possibility of phishing for account credentials on social networking sites [15].

Logo syndication

There are several logos, such as the Amazon, Facebook, and Visa logos, that appear often in our webmail data set in syndicated form. In these cases, the logo is not being used in a phishing attempt, but rather as a payment option on spam websites. We analyze this further in Section 6.3.

This chapter, in full, is based on material from a paper titled “A logo recognition system to detect fraudulent websites” that has been submitted for publication in ACM CCS, 2009, Becerra, Sebastian; Wang, Kai; Belongie, Serge; Savage, Stefan. The thesis author was the primary investigator and author of this paper.
Chapter 6

Error Analysis

We describe characteristics of logos that are challenging for our approach and discuss potential extensions to overcome them.

Figure 6.1: The Orange logo is not suitable for matching using our SIFT-based approach. The logo lacks enough detail to generate a reliable set of feature points for matching. Here it is shown in its original form on the left, and after preprocessing on the right.

6.1 Logos that lack detail

The first step in our matching algorithm is keypoint discovery, which finds extrema in an image’s scale-space. This has shown to be highly effective at recovering the same locations of an object when viewed from different viewpoints. However, in the cases where a company logo is extrema-impoverished, the matching step fails. Figure
6.1 shows an example of a logo without much detail. In this case, the Orange logo – as it appears on the web – is 44×44 pixels in size, with most of the logo being a solid colored background. The text portion of the logo is 10 pixels high. As a result, only three keypoints are discovered in the logo. Logos in this category are not amenable to SIFT-style matching using scale-space extrema and require a different approach. One possibility is to perform matching using maximally stable extremal regions (MSERs) [19], which discover regions of an image that remain relatively unchanged across different thresholding values. In the case of Orange, the solid colored background would be considered a stable region and could be a useful feature to use for logo recognition.

6.2 Logos with mostly text

Figure 6.2: An example of a false positive; the CNN logo is displayed in the green box and a non-CNN screenshot is shown below it. The matching feature points between the string portion of the logo and the text in the screenshot triggered a false positive.

In many of the logos in our dataset, the image contains an insignia and the company name. Figure 6.2 shows an example of this in the logo for CNN. In this
case, there is a distinctive rendering of ‘CNN’ and a common rendering of ‘com’. The challenge here is that the feature points discovered in the portion of the logo that looks like standard text tend to match regions of text on the website.

One way to deal with this problem could be to use optical character recognition (OCR) to detect character regions in the logo and place less weight on keypoint matches found in those regions. A more principled solution would be to learn feature point weightings on the logos automatically as proposed by [11]. In that work, more statistically distinct features were discovered and given higher weight. We would expect feature points coming from the distinctive ‘CNN’ region would provide greater discriminative power and be assigned a higher weight.

6.3 Syndicated logos

Table 6.1: Breakdown of the logos that were matched in syndicated form. Of the 1,920 total logo matches, 83 of them (4%) were in syndicated form.

<table>
<thead>
<tr>
<th>Logo</th>
<th>Num Syndicated</th>
<th>Num Total</th>
<th>Percent Syndicated</th>
</tr>
</thead>
<tbody>
<tr>
<td>barclays.com</td>
<td>6</td>
<td>6</td>
<td>100%</td>
</tr>
<tr>
<td>amazon.com</td>
<td>8</td>
<td>10</td>
<td>80%</td>
</tr>
<tr>
<td>facebook.com</td>
<td>7</td>
<td>11</td>
<td>64%</td>
</tr>
<tr>
<td>networksolutions.com</td>
<td>16</td>
<td>34</td>
<td>47%</td>
</tr>
<tr>
<td>visa.com</td>
<td>12</td>
<td>51</td>
<td>24%</td>
</tr>
<tr>
<td>google.com</td>
<td>12</td>
<td>166</td>
<td>7%</td>
</tr>
<tr>
<td>ebay.com</td>
<td>13</td>
<td>248</td>
<td>5%</td>
</tr>
<tr>
<td>paypal.com</td>
<td>9</td>
<td>611</td>
<td>1%</td>
</tr>
</tbody>
</table>

Table 6.1 shows that 4% of the total matches were false positives corresponding to syndicated logos. The high number of Network Solutions entries is due to their parking service for expired or new domains.

This chapter, in full, is based on material from a paper titled “A logo recognition system to detect fraudulent websites” that has been submitted for publication in ACM CCS, 2009, Becerra, Sebastian; Wang, Kai; Belongie, Serge; Savage, Stefan. The
thesis author was the primary investigator and author of this paper.
Chapter 7

Conclusion

In this thesis, we have explored the potential of an image-based system for automatically recognizing branded logos on Web sites for the purposes of fraud detection. In a sense, this capability gets at the very essence of how fraud detection works – first a site is evaluated as representing some brand and then it is checked to make sure the representation is unauthorized. In practice, this approach is inherently reactive and inevitably time-consuming. We suggest that automated logo matching is now a robust technology (achieving over 99% accuracy on real data) and can minimally play a valuable role in reducing the overhead and improving the accuracy of these operations. However, we believe the larger opportunity will arise from using this technology proactively within the browser. With integration into the page rendering engine and some more effort at optimization, we believe logo matching could be performed in real-time on every page viewed. With this capability, it now becomes feasible to integrate information about brand-based trust cues into client-side defense mechanisms. For example, a system like AntiPhish [16] could easily be modified to prevent the entry of sensitive information when at a site displaying an unauthorized bank brand logo.

This chapter, in full, is based on material from a paper titled “A logo recognition system to detect fraudulent websites” that has been submitted for publication in ACM CCS, 2009, Becerra, Sebastian; Wang, Kai; Belongie, Serge; Savage, Stefan. The
thesis author was the primary investigator and author of this paper.
Bibliography


