Catching Phish: Detecting Phishing Attacks From Rendered Website Images

Arel Cordero* arel@cs.berkeley.edu

Tamara Blain* eb-9d9@cs.berkeley.edu

December 12, 2006

Abstract

This paper proposes a computer vision based approach for defending against the pandemic threat of *phishing*, a social engineering attack on personal identity and property. Presently, many layers of defense exist, but there is no general solution. Our proposal is intended to complement existing strategies. We examine the problem of relying only on *images* of rendered web pages to identify phishing attacks. To our knowledge, we are the first to propose using only rendered website images to identify phishing sites. Our approach is amenable to both a browser setting, to protect end users, and a server setting, to help automate the identification of phishers. As we explore in this paper, a number of security benefits emerge from relying only on image data for classification. Our experimental results show that even at a first pass, reasonable classification rates can be achieved.

1 Introduction

Phishing attacks are increasing in rate and sophistication. The expected financial loss to U.S. consumers in 2006 increased to 2.8 billion dollars[1]. The average cost per incident also increased this year by an order of magnitude to $1,244. Phishing is proving very costly and it is getting worse.

Typically, an attack is mounted by the impersonation of a reputable website, to fool unsuspecting users into divulging sensitive information (e.g., credit card numbers or passwords). Unsolicited mail is often the vehicle for luring individuals to these fake, usually temporary, sites. It is therefore natural to consider measures to protect email and web browsing. While neither should be neglected, we initially focus on the latter.

Our goal is to get a machine to detect when it has been directed to a phishing site. This is important as often phishing emails slip through earlier lines of defenses (e.g., because of an adaptive adversary, or because a user's email environment changes).

Prior research, such as [7, 2] has investigated the use of visual or structural elements to classify phishing pages, but this work has primarily relied on information in the source (e.g., HTML) or in the document object model (DOM). We, however, argue that this luxury is unsustainable because, in effect, there is an arms race, and the discrepancy between the apparent structure in source code and the resulting visual impression may be arbitrary. For many of us, the artifacts of such an arms race appear daily in our inbox. Fortunately, phishing is not spam; it has some very appealing properties that lend optimism to its prevention.

Our paper makes the following research contributions:

- We propose and argue for the use of rendered images as a basis for phishing detection.
- We implement and discuss an initial prototype of this system.

2 Background

Why phishing is special. On the surface, phishing and spam may seem similar, but they have fundamental differences. First, phishing actively attempts to rob you. More interestingly, though, from a vision point of view, the phishing adversary is constrained, unlike the spammer, by the need to appear legitimate to visitors. This limits the extent the phisher can evade detection.

*University of California, Berkeley, CA, 94720

For instance, the CAPTCHA-like ads designed to throw off current spam detectors, would not generally work in the phishing domain. By training classifiers to detect the visual features of a website, this severely limits what a phisher can do about being detected. A robust defense against phishing attacks may be possible.

What is hard about this problem. The constraints on images of legitimate websites, inherent in the phishing domain, simplify the task of object recognition, in general a difficult problem. It does not however make this problem trivial. Several difficulties an algorithm must overcome are:

- The effect of window size on rendering and layout.
- Font and font size, can drastically change appearance.
- Advertisements change rapidly, there is the danger of dominating other features.
- Asynchronous network (visual layout may change as a page loads).
- Dynamic content prevalent (see our assumptions in Section 2).
- Popular web design templates may lead to higher false positive rate.

Threat model. We assume we have a secure channel for verifying the authenticity of a known, legitimate site. This is not unreasonable by the fact that there is often only a small set of targeted sites that we would attempt to protect. For instance, July 2006 saw 154 unique brands hijacked by phishing campaigns in the U.S., 15 of which comprised 80% of the volume[4]. Most legitimate sites already employ cryptographically strong methods of authentication such as with the use of digital signatures. A simpler, though probably effective, method would use the URL to determine authenticity.

A second important assumption is that the user is familiar with the appearance of the services she uses (such as PayPal, Washington Mutual, etc.) Phishing attacks are less likely to succeed when dealing with sites not familiar to the visitor.

We will also assume that at least some salient features of each site will be unique, and persistent, that is, there will only be periodic changes in the general appearance of a site.

We assume when a page changes or updates (i.e., when there is dynamic content) we get periodic resamples of the site. For example, for video, AJAX, Flash or other dynamic content, we can imagine snapshots taken every few seconds[1]. A stronger assumption, motivated by this one, is that we treat all images as static content.

For our prototype, we use the image of the whole page for classification. It may, however, be better to assume the classifier also "sees" what the user sees. Otherwise there is the possibility for an adversary to include a phishing website as a small part of a long page, jumping to the location when the page opens.

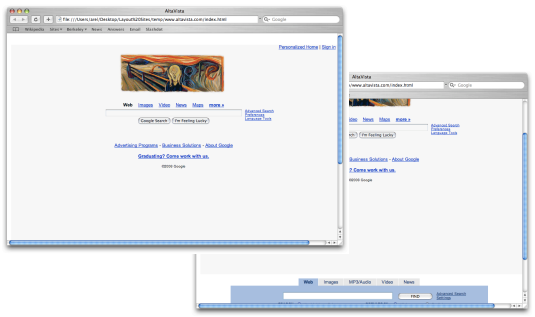
Attacks on existing phishing classifiers. A victim of a phishing attack need not be aware of the underlying DOM of a website. Their awareness most likely extended only to the final product, the result of rendering the web page. Therefore, it is often advantageous to the adversary to obfuscate the source code underneath: classifiers trained to detect features at that level will fail.

To illustrate an instance of an attack on such systems, imagine the page an adversary wants to replicate has three columns, something that in the genuine source code is represented by objects types (such as `<DIV>` objects). The adversary, wishing to present the same site, but with different underlying code, chooses to have only one column, manually creating the illusion of three columns (say by inserting whitespace every line). To an unsuspecting target of a phishing scheme, both sites may look identical, but to machine that analyzes internal document structure, the sites will have different structure. The problem is that the user interprets the semantics of the rendering differently than the document structure implies.

Classifiers that train on the text content of a site are also vulnerable. For instance, instead of whole words, which can be easily parsed by machine and turned into features, a clever adversary may obfuscate the English text by individually placing characters throughout the page so their superposition once the site is rendered spells out English words. To a similar effect, an adversary could replace text with an image of text.

Many varieties of attacks are possible that cleverly de-correlate the source from the result. For instance, in Figure 1, the alteration of only one

[1]Note if the website is coming from a trusted source it is not necessary to sample. This would only be relevant for unknown sites, until it is added to the set of trusted sites.



```
<img src="http://us.i1.yimg.com/us.yimg.com/i/us/av/logo.gif"
      width=199 height=77>

<img src="logo.gif"
```

Figure 1: The source code for what first appears to be Google, exactly mimics AltaVista except for the single modified line above.

line of source code from the front page of AltaVista, can (crudely) transform the website appearance to an exact replica of Google's. Note, code like this could live on any server, a phishing detector would normally have no reason to suspect the page looks like Google. And, in this case, prediction based on the source code would almost certainly have recognized the page as AltaVista.

We suggest that by training only on the final rendering of pages, attacks such as the ones above are rendered useless.

Security goals. We want a robust method of detecting phishing, that cannot be indefinitely misled by a malicious adversary while also misleading visitors to the site. We want to abstract away the adversary's ability to obfuscating his attack from a phishing detector.

3 Our approach

The architecture. In order to classify new websites as phishing attacks, we want a classifier that answers the question: *"Does this new site look like one I already know?"*

If the answer is *no*, then we would like to rest assured that it is either not a phishing site, or if it is, it is sufficiently distorted from the original to raise visitor's suspicions. If the answer is *yes*, then we ask a follow up question: *"Is this site that* looks *like x, actually x?"*. We assume we can securely verify the authenticity of positive cases by, for example, matching the URL, or by verifying a digital signature (Figure 4).

A system that can answer the above two questions accurately, and efficiently, could be very effective at detecting phishing attacks on a web browser. It could also be useful for recognizing suspicious sites crawled by a server. In that case, its output could be a blacklist, or graylist, similar or complementary to ones already in use on the web[3].

Prototype implementation. Our prototype relies on the following technologies:

- `Cocoa / Safari engine` (website rendering)
- `GNU Octave` (data preprocessing)
- `Image Magick` (data preprocessing)
- `Python` (glue to hold everything together)
- `R statistical platform` (classification and testing)

Data and features. To collect data, we wrote a small web browser to capture images it renders with the Mac OS-X Safari engine, outputting TIFF images of the entire rendered page. We collected 119 images total, with some multiple views of the same sites by varying window size and advertising displays (Figure 3).

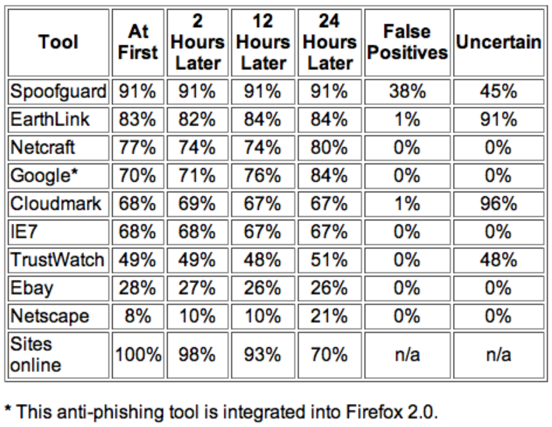
To turn this data into more manageable feature vectors, we compute a joint histogram[6] of each image. We accomplish this by discretizing the color space coarsely (into 64 bins). We then compute an edge density map of the image and discretize it into 4 bins. We proceed to create a joint histogram with the two features resulting in 256 features per image.

Especially because we have so few samples in our data set, it is important to reduce the dimensionality of our data. We applied Principal Components Analysis (PCA) and projected our dataset into a four dimensional space[2].

4 Experimental Results

Before applying PCA, our first attempt at classification yielded abysmal results because, although we performed feature selection on the

[2]We determined this and all the other parameters by searching through the parameter space and evaluating on a validation set distinct from the final test set.



Tool	At First	2 Hours Later	12 Hours Later	24 Hours Later	False Positives	Uncertain
Spoofguard	91%	91%	91%	91%	38%	45%
EarthLink	83%	82%	84%	84%	1%	91%
Netcraft	77%	74%	74%	80%	0%	0%
Google*	70%	71%	76%	84%	0%	0%
Cloudmark	68%	69%	67%	67%	1%	96%
IE7	68%	68%	67%	67%	0%	0%
TrustWatch	49%	49%	48%	51%	0%	48%
Ebay	28%	27%	26%	26%	0%	0%
Netscape	8%	10%	10%	21%	0%	0%
Sites online	100%	98%	93%	70%	n/a	n/a

* This anti-phishing tool is integrated into Firefox 2.0.

Figure 2: Recent study on phishing detection rates for leading systems[3].

Rendered Samples	Class
13	PayPal
86	Other
20	Phishing (PayPal)

Figure 3: Our data set consisted of 119 rendered images.

data, we did not have enough samples given the dimensionality of our features. After applying PCA, the classifiers were significantly more successful generalizing for the test cases (Figure 6). We tested three classifiers Support Vector Machines (SVMs), Naive Bayes, K-Nearest Neighbors (KNN). SVMs and K-NNs outperformed Naive Bayes, and achieved accuracy of around 85% This compares favorably to current approaches such as the Google / FireFox toolbar.[3].

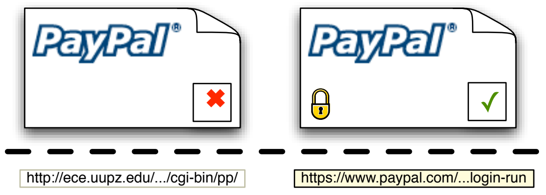

PayPal
PayPal
http://ece.uupz.edu/.../cgi-bin/pp/
https://www.paypal.com/...login-run

Figure 4: When a site appears similar to PayPal, or any other *target* site, its authenticity must be established (i.e., by URL, or SSL certificate).

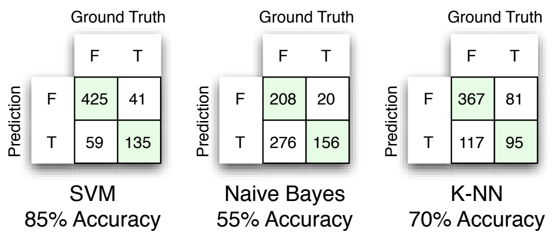


Figure 5: Average classification confusion matrices on validation test set, during training.

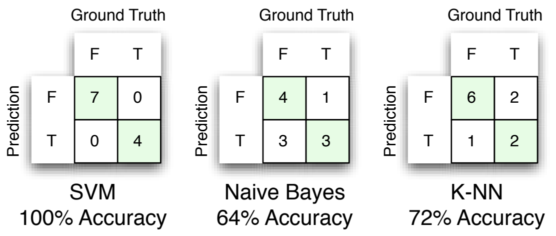


Figure 6: Final results of separate test set.

5 Related Work

Detecting phishing before it gets to the web browser, [2] analyses email structure to predict phishing attacks. Empirical evidence confirms phishing email has distinct linguistic and visual cues, for instance in the use of popular keywords. Features about emails are extracted and SVMs are trained to do the classification.

On the browser side, a lot of prior work has relied on blacklists to control phishing[3]. This approach is not only prevalent, but relatively effective. However, blacklists are less useful for protecting early visitors as it takes time to properly identify the threats.

and their performance usually increases as time goes on (until the site is removed).

The closest work we found to ours is [7]. In this paper, the authors analyze visual structure to compare websites to one another for phishing detection. However, unlike us, they rely on the DOM and other structural elements of the source code to to compare sites. Image processing does get used, however, to compare images in similar layouts. This leaves them vulnerable to attacks mentioned in Section 2.

6 Conclusion

The idea of using images as a basis for phishing detection has very appealing security consequences. The prototype we created was successful as a first attempt. Open questions remain. For instance, we only classified using one site, how feasible is it to classify using one-hundred sites or one-thousand sites? Could a system like this be extended to learn profiles of any new web site, to detect phishing or impersonation on any previously visited website?

We ran this experiment on a small data set, it would be beneficial to expand the data set, and include other rendering engines.

Other avenues for image comparison may also be worth pursuing. For instance, it is the job of the MPG4 standard to compare the amount one frame resembles the next. This could be very useful for detecting website similarity (thanks to Adrian), though the caveat is that something like MPG4 was not designed with an adversary in mind.

Another important area for improvement is in the representation of websites as feature vectors. We used a very simple color + edge joint histogram, which is insensitive to the flexible layouts of a website, but inflexible to possible changes of color schemes, for example. A better approach might be to identify higher-level features [5] or perhaps logo detection.

References

[1] Gartner says number of phishing e-mails sent to u.s. adults nearly doubles in just two years. Press Release, November 2006. `http://www.gartner.com/it/page.jsp?id=498245`.

[2] M. Chandrasekaran, K. Narayanan, and S. Upadhyaya. Phishing e-mail detection based on structural properties, 2006. `http://www.albany.edu/iasymposium/chandrasekaran.pdf`.

[3] L. Cranor, S. Egelman, J. Hong, and Y. Zhang. Phinding phish: An evaluation of anti-phishing toolbars, 11 2006. CMU-CyLab-06-018.

[4] A.-P. W. Group. Phishing activity trends report, 7 2006. `http://antiphishing.com/reports/apwg_report_july_2006.pdf`.

[5] X. Gu, J. Chen, W. Ma, and G. Chen. Visual based content understanding towards web adaptation, 2002.

[6] G. Pass and R. Zabih. Comparing images using joint histograms. *Multimedia Systems*, 7(3):234–240, 1999.

[7] L. Wenyin, G. Huang, L. Xiaoyue, X. Deng, and Z. Min. Phishing webpage detection. In *ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pages 560–564, Washington, DC, USA, 2005. IEEE Computer Society.