Verilogo: Proactive Phishing Detection via Logo Recognition

Ge Wang, He Liu, Sebastian Becerra, Kai Wang,
Serge Belongie, Hovav Shacham, and Stefan Savage

Dept. of Computer Science and Engineering
University of California, San Diego

Abstract

Defending users against fraudulent Websites (i.e., phishing) is a task that is reactive in practice. Blacklists, spam filters, and takedowns all depend on first finding new sites and verifying that they are fraudulent. In this paper we explore an alternative approach that uses a combination of computer-vision techniques to proactively identify likely phishing pages as they are rendered, interactive queries to validate such pages with brand holders, and a single keyboard-entry filter to minimize false positives. We have developed a prototype version of this approach within the Firefox browser and we provide a preliminary evaluation of both the underlying technology (the accuracy and performance of logo recognition in Web pages) as well as its effectiveness in controlled small-scale user studies. While no such approach is perfect, our results demonstrate that this technique offers a significant new capability for minimizing response time in combating a wide range of phishing scams.

1 Introduction

Human behavior relies fundamentally on trust. In our daily lives we presume that, by and large, the information we are given is factual and benign. This assumption makes possible the tremendous scale of today's transactional commerce. But trust is easily exploited: scammers present themselves as a trusted party and thereby obtain otherwise confidential information. In on-line commerce this problem is exemplified by social engineering attacks that divert users, typically via an e-mail vector (phishing) or DNS poisoning (pharming), to fraudulent Websites purporting to represent a particular institution. Since the sites provide all the right visual trust cues (style, logos, buttons, etc.) to suggest a legitimate site from a familiar brand holder, users are fooled into providing their credentials, which are captured and monetized by the scammer.

Unfortunately, these attacks are successful precisely because it is difficult for users to distinguish between a genuine e-commerce or banking site and a fraudulent imitator. In a recent study, 90% of users were fooled by high-quality phishing sites [9]. Such sites may be visually indistinguishable even to experts. The mechanisms used to combat phishing sites typically involve external efforts that identify such sites and then distribute their IP addresses or URLs, which are used to populate blacklists (e.g., for anti-phishing toolbars [45, 25, 36] or spam filters) and to drive site takedown efforts [31, 32, 30]. Such efforts are by definition reactive, leaving users unprotected until sites are discovered and characterized — a process that has multiple stages of delay. Indeed, in their recent study of blacklist response time, Sheng et al. found that few blacklists had better than 20% coverage of new phishing sites found in spam e-mails, and it could take hours before more than 50% were identified [36].

In this paper, we propose a new *proactive* approach designed to improve this state of affairs: *on-line logo authorization.* The premise behind this fairly simple 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 and widely felt. Indeed, there is prior work demonstrating that users treat such visual cues as "trust cues" [9] and it seems likely that phishing yields would suffer significantly absent an appropriate logo for this reason.

We have exploited this observation within an extension to the Firefox browser called Verilogo. It uses computer-vision algorithms to automatically identify the presence of heavily-phished logos as each Web page is rendered, implements a protocol for online validation of such pages with the associated brandholders, and uses a simple keyboard-entry heuristic to filter out sites making non-fraudulent, but unauthorized use of a brand logo. The attraction of this approach is that it can be performed proactively on each user's computer, to offer protection against "zero-hour" phishing sites not yet discovered and categorized. We evaluate Verilogo both for its low-level accuracy and performance, as well as through small-scale user studies to understand how "human factors" impact its effectiveness. Our preliminary results suggest that this technique offers a useful new capability, one that can provide protection orthogonal to existing methods.

We believe that this and other topics in *adversarial computer vision* are important subjects for future research.

2 Background

The ease and anonymity provided by the Internet allow fraud to be trivially perpetrated through a wide range of vectors. The most popular fraud method is phishing, because the e-mail vector lends itself to tremendous scale.[1]

A phishing campaign may target millions of users, enticing them to visit a doppelganger site that may be visually and functionally identical to the site it purports to be. Many users have relatively simple fraud-detection strategies and are easily tricked by such subterfuge into providing their credentials [9, 10, 42]. While there is evidence that better user education can reduce suceptibility, even after extensive training many users still relinquish their information [22].

The stolen credentials are monetized by the scammers, directly or via third-parties who liquidate accounts for a commission [32, 39, 14]. Loss estimates vary considerably (a 2007 Gartner survey estimated losses of $3.2B, while Moore and Clayton provided a measurement-based estimate of $320M), but are clearly sizeable [31].

Consequently, fraudulent sites and, specifically, phishing has enjoyed a wide 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 defense.

2.1 Finding phish

Candidate phishing sites are typically detected via the content in either the e-mail vector or on the Website. To convince recipients to visit the embedded URL pointing to a phishing site, scammers

[1]For the remainder of this paper we use phishing to refer to fraudulent sites. Note that our approach does not depend on the vector, since it focuses on the content of the site itself.

must communicate some urgent imperative in the envelope ("Your account has been compromised" or "you have won a special lottery") as well as identify the institution. Thus, one approach is to use machine learning techniques to automatically classify e-mail text as being "phishy" or not [13, 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 [17, 26, 21].

Actual site content, combined with the URL used to reach it, can be mined for heuristic tokens common to phishing sites [7]. More sophisticated versions of this approach create content "signatures" of heavily phished sites and match suspicious pages against these feature sets [25, 28, 4, 43]. 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 search results hosting said textual content [46]. Finally, similar to our own work, there have been several efforts to use vision algorithms in identifying phishing pages [16, 6], although these differ in several key respects. First, they focus on matching whole pages to reference pages from the brand holder, while we focus exclusively on the logo (which we argue is a less fragile feature since it is common to virtually all phishing campaigns while the page makeup can change considerably). Second, these systems are focused on offline use (i.e., for sorting through suspicious pages), possibly due to the fact that vision algorithms are often computationally intensive, and thus difficult to apply to latency-critical applications. Part of our contribution is the embedding of vision-based techniques for phishing detection into an on-line browser context.

The effort closest to our own is Envisional's proprietary ImageFlare technology that matches logos against reference images. Its intent is to allow a brandholder to protect the integrity of its brand by scouring the Internet for its use. Adapting a centralized system like ImageFlare to phishing detection for all popular brands would pose scalability challenges, and even then would not provide online client-side protection. Verisign's Secure Internet Letterhead [18], which would embed logos in X.509 certificates for presentation to the user, is complementary to our approach: Internet Letterhead allows users to gain confidence in authorized logos; our proposal alerts the user to unathorized logos.

2.2 Defending against phish

Phishing defenses can be deployed on e-mail servers; in users' browsers; and indirectly through site takedown efforts.

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 a major browser) by preventing access to phishing sites as they are requested or rendered. Browser filtering may use local heuristics [7] or real-time blacklists [45, 25]. The former are inherently proactive, but are only effective for sites that match the heuristics (creating an "arms race" around heuristic evasion). Moreover, to achieve high catch rates, heuristic approaches can also produce very high false positive rates — e.g., in one recent study Spoofguard's heuristics found over 90% of zero day phishing sites, but misclassified over 40% benign sites as phish [45].

The blacklisting approach is reactive, requiring that the blacklist provider have received a sample of the phishing e-mail; visited the associated site; validated that it is indeed fraudulent; and made this information available to the client. This leads to low false positive rates but also to a considerable window of vulnerability between when a phishing site first goes up and when it is made available in a browser blacklist. Moreover, the blacklisting approach is sensitive to the choice of vector; e-mail is not the only method for defrauding users. As phishers deliver their URLs using social networking, blogs or search engine optimization so too must the blacklist provider track these data sources.

Finally, there are both volunteers and commercial organizations who work to disable fraudulent Websites or their domain names. This approach has the advantage of protecting all users, but typically has much higher latency than other reactive methods since it requires manual vetting of sites as well as 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. There is considerable variability in takedown time, with Moore and Clayton reporting average takedown times of 64 to 94 hours for typical sites [32], but with dramatically better response time for brands represented by commercial takedown companies [30].

we believe that ours provides a distinct contribution in this considerable body of work. Succinctly, we are focused on providing client-side detection of new phishing sites — thus offering the latency benefits of heuristics — but using a feature set (visual logo matching) that is both robust and produces far fewer false positives in practice.

3 Logo-based phishing detection

Our approach consists of three steps: first, we identify whether a Web page contains a given logo. Second, we determine if the brand holder has authorized the hosting IP address to use its logo. Finally, for sites that both contain logos and are unauthorized we issue a warning only if the user enters keyboard input into the page. We first introduce our threat model, then describe each of these steps in turn.

3.1 Threat model and assumptions

Our work focuses on combatting phishing Websites. We consider the case where a user has been led to a Web page posing as a legitimate brandholder page but whose actual intent is to steal user credentials. We aim to identify Websites using brand holder logos to solicit information, and to help users make informed decisions as they are browsing the Web.

3.2 Vision-based logo detection

A naïve way to look for logos is to compare images embedded with `<img>` tags for binary equivalence with a reference database of logos (e.g., using a hash). This approach is easily defeated: bit-for-bit equivalence is inherently sensitive to minor transformation of the logo images. Indeed, we see such minor transformation in practice. In a recent test of just such an approach, Afroz and Greenstadt find that only 54% of phish sites are detected in this matter and state that "When logo-detection fails, it is because some logos are resized or the design is slightly changed in a way that is unnoticeable to the naked eye" [4].

Beginning with the introduction of the Scale Invariant Feature Transform (SIFT) [24] in 2004, the problem of near-duplicate image retrieval has seen many advances and is now regarded as a solved problem. Roughly, SIFT takes an image and discovers keypoint locations that are invariant to scale and orientation. 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 problem 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.[2]

Potential candidate matches in the logo database are scored based on the proportion of keypoints that match, with point correspondence being scored based on the Euclidean distance ratio between the nearest and second-nearest neighbor. (In practice, two keypoints are said to match if this ratio is 0.8 or above — a value determined experimentally by Lowe and copied since.) We approximate the Nearest neighbors computation using a simple approximation due to Bies and Lowe [5] that greatly improves efficiency. To narrow down the number of potentially matching logos to compare, we take a similar coarse-to-fine approach as in [33].

This process produces a keypoint match percentage score for each logo that we turn into a binary feature based on a threshold. A higher threshold decreases false positives; a lower threshold permits "fuzzier" matching, reducing false negatives. In our implementation, we consider two threshold setting regimes: one in which the threshold is set to a single value for all logos, which is simple but can generate unnecessary false positives, and one in which poorly performing (insufficiently unique) logos are given distinct thresholds.

3.3 Brand authorization

While the SIFT algorithm provides the means for robust image matching, this is not, by itself, an anti-phishing defense. As we alluded earlier, the mere presence of a bank's logo on a page is insufficient to conclude that the site is fraudulent. Indeed, the page may be owned by the bank itself or may simply be a third-party site (e.g., a news site) that is displaying the logo incidentally and with no intent to defraud. We address these two cases independently.

To identify pages hosted by the brand holder (or its designates) we propose to *ask* them. We consider two approaches, inspired by the SPF [40] and DomainKeys [23] mechanisms used for similar purposes in binding IP addresses and the domains used in e-mail "From:" addresses. In both cases we assume that the browser is distributed with a logo database that may be further annotated with information such as the brand holder's registered domain and possibly a certificate. Given the relatively small number of brand holders who are actively phished, this assumption seems reasonable.

In the first case we define a new DNS record type, BAP (Brand Authorization Policy), that lists IP addresses or domains authorized to display the brand holder's marks using a syntax similar to SPF. Upon detecting that a particular logo is present on a page, the browser requests the BAP record from the associated brandholder's registered domain. If the Web page is being hosted by a site address contained in the BAP record then it is considered benign and all further processing stops. To bootstrap use, in the event that the BAP record is not found, the browser may default to use a local database of whitelist rules per brand.

The disadvantage of this approach is that it requires a DNS server update for each new site that the brandholder wishes to authorize. An alternative approach, modeled on DomainKeys, is to have brand holders embed a digital signature in their logos (e.g., in the GIF or PNG comment field) that covers both the image content and the hosting IP address. Upon finding a logo on a Web

[2] There have been many improvements since the SIFT algorithm was described. It should be possible to increase Verilogo's efficiency and precision by applying these.

page, the browser then determines which image object was rendered at that location and validates it using the associated brand holder's public key; the key may be included in the browser's brand database or distributed via a separate PKI such as DNSSEC.

Our prototype includes a simplified version of the BAP approach, but we believe implementing either technique is feasible; the logistical issues are similar to those involved in SPF and DomainKeys, which have seen wide adoption [11].

3.4 Input filtering

Unfortunately, popular logos are routinely used in news sites that report on the associated brands. Such sites are unlikely to be "authorized", and even if the brand holder wanted to authorize them, the challenge of tracking all sites making incidental use of a logo is likely insurmountable. Instead, we rely on a simple heuristic to filter out sites that might be "true phishing sites" from those that could not be: We track when a user provides keyboard input to an unauthorized page and use this event to trigger a warning.

We've chosen to explore this very simple heuristic for two reasons. First, we believe it is highly robust to evasion. Phishing sites need input to acquire user credentials and thus sites that are simply "viewed" do not place a user at risk. We have found few confirmed counter-examples to this rule, excepting a small number of phishing sites that request the user to download and run executable software (presumably a malware of some sort). In principle, one could easily add a "user attempted to download executable" heuristic to address these cases, but we do not evaluate that here.

Second, selecting a minimal part of the design space makes our analysis straightforward and conservative. Clearly, false positives could be reduced even further using more advanced site heuristics or learning algorithms [25, 28, 4, 43, 46, 7, 17, 26, 21] but teasing apart their relative contribution would obscure the basic result we're trying to demonstrate. Moreover, each new heuristic can offer additional opportunities for evading detection that we are not prepared to evaluate here. For example, the impact of domain reputation systems has led phishers to host their pages on compromised sites (effectively bypassing any "low reputation hosting" heuristic). That said, we believe a widely-fielded system would inevitably combine our approach with other such measures to still further reduce its false positive rate.

3.5 Mashups

Mashups — sites that combine content from multiple legitimate sites — pose special problems. Willing brandholders could authorize specific mashups by means of policy files analogous to Flash's `crossdomain.xml`; see, e.g., [20, 34].

4 Verilogo browser

Our prototype, called Verilogo, consists of a Firefox browser addon and (for ease of implementation) a matching server spawned in a separate process. The basic workflow (illustrated in Figure 1) mirrors the description in Section 3. It uses a modified version of Robb Hess' SIFT code [19] and the OpenCV 2.0 computer vision library. The server pre-loads all key features from the logo database (which contains 352 instances in our experiments) and initializes a FIFO for communicating with the browser.

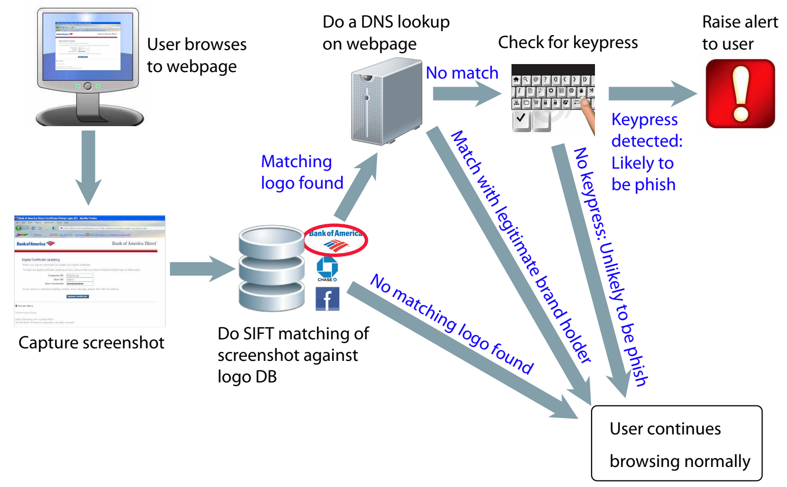

User browses to webpage
Capture screenshot
Do SIFT matching of screenshot against logo DB
Matching logo found
Do a DNS lookup on webpage
No match
Check for keypress
Raise alert to user
Keypress detected: Likely to be phish
Match with legitimate brand holder
No keypress: Unlikely to be phish
No matching logo found
User continues browsing normally

Figure 1: Verilogo prototype workflow. As sites are rendered, the viewport contents are matched against a logo database using SIFT. Upon a sufficiently strong match we contact the brand-holder for authorization. Absent such authorization, we defer any alert until the user provides keyboard input.

Our Firefox addon hooks event notifications to determine when a page is loaded and then grabs the entire "viewport" using the firefox "Screengrab" addon.[3] The resulting (PNG) bitmap is saved to disk and its location is communicated to the matching server over their shared FIFO. These operations are run in their own thread context, do not block the user, and generally are overlapped with browser I/O and user reaction time. The browser also records whether the page receives any keystrokes, for use in later filtering (matching latency is generally sufficiently short that it is not necessary to serialize this step with matching).

Logos used to indicate that a site represents a brand are typically found at the top of the Web page. To optimize this common case, the matching server evaluates the viewport in a series of overlapping strips (starting from the top). By overlapping the strips by N pixels we ensure that any logo of size N can be captured in its entirety in a single strip (in our experiments we use $N = 100$). As N increases overhead decreases, but the latency to find logos on the top of a page is increased. Moreover, logos larger than N may be matched imperfectly. It would also be easy to randomize the strip boundaries to defeat adversaries who aim to place very large logos along our default boundaries.

The server crops the widest possible section of each strip that contains a single color (common on width-limited pages and, by definition, not containing logos). It performs SIFT matching against the reference database. If one or more logos' match percentage are above a set threshold, they are reported back to the browser via a complementary FIFO channel. Our implementation uses a single threshold, but per-logo thresholds might increase accuracy.

Finally, upon receiving a signal that a logo has been found, the browser initiates a DNS request to the associated brand holder for an associated BAP record and validates that the hosting IP address is contained within. If it is not, and if a keystroke has been received, the browser displays an alert.

[3]A more sophisticated addon implementation could perform incremental screen captures to minimize latency, which our matching server is already designed to handle.

While most anti-phishing alerts are generic ("this page is bad"), Verilogo provide semantically meaningful context to allow the user to reason about their situation. A Verilogo warning communicates that "an alert is being raised because this site is diplaying the Bank of America logo (shown here), but is not authorized to do so by Bank of America. If you are being asked to do something risky, such as provide your Bank of America credentials, it is likely that this site is fraudlent and you should not continue using it." This allows the user to evaluate whether the site really is attempting to represent the Bank of America and act accordingly. Although security dialog "clickthrough" is a challenge facing any security warning, we believe that providing specific and meaningful information rather than general or "jargon-y" warnings is much more helpful in guiding informed user decisions.

5 Evaluation

5.1 Datasets

For our evaluation we consider several datasets. The first, Brand, contains screenshots from legitimate brandholders that feature their logo (e.g., www.citibank.com for Citibank). We selected twenty-three brands chosen arbitrarily from several lists of commonly phished sites [29, 1, 2] and for each brand gathered ten distinct brand-owned pages containing their logo (230 in total). The second dataset, Phish, used the same brands, but contained up to ten screenshots from known phishing pages for each brand taken from the PhishTank database (219 in total).[4] Both datasets represent examples that we desire our matching algorithm to detect (although we did not perform any prior matching or dataset filtering based on matching results).

We also gathered two datasets to represent "typical" Internet pages. One consists of 183 screenshots from Alexa's Hot URLs list on January 29th, 2010; the other of 254 screenshots from Yahoo's Random Link service. We combined these two datasets (for a total of 437 pages) to represent a combination of both popular pages (Alexa Hot) and a random sample of Websites (Yahoo Random) — roughly sampling the range of sites a user might typically encounter. For debugging purposes, we labeled each page by hand with the set of logos we identified on it. We observed that roughly 10% of these pages contained a logo in our database.

Unfortunately, we do not have access to a user activity trace to test our keyboard heuristic, so instead we hand categorized each of the pages in terms of a subjective assessment of the opportunity and likelihood for engaging keyboard input. We classified pages into four categories:

- **No Input**. Pages that have no opportunities for input.
- **Incidental Input**. Pages that allow input, but for features that are incidental to the normal use of the Web page (e.g., search bars on non-search pages).
- **Optional Input**. Pages that allow input and for which the input provides additional functionality that is part of the site's primary use, but for which it is still common to use the site without providing input (e.g., blog pages that contain a comment section).
- **Essential Input**. Pages for which user input is essential to accessing key functionality (e.g., login pages or survey forms).

Our logo database includes logos for 166 distinct brands (again taken from lists of commonly phished brands) with 176 distinct instances. There are more instances than brands because some

[4]Comerica and M&I Bank did not have ten such examples in the database so their phishes are under-represented.

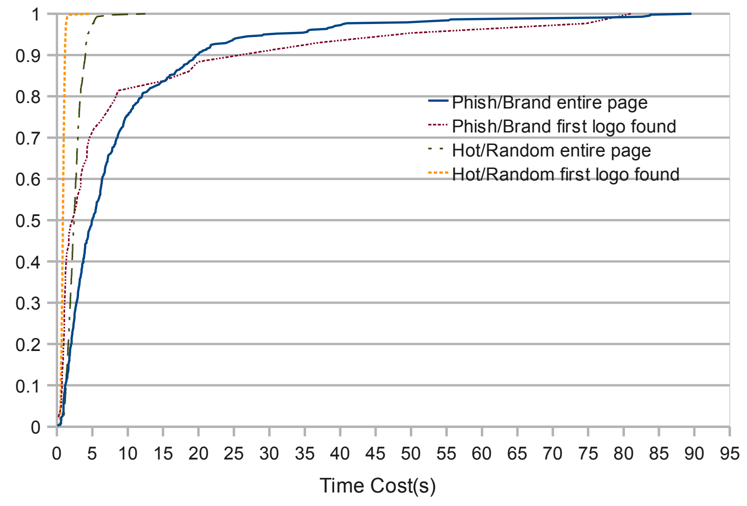


Figure 2: Cumulative Distribution Function (CDF) of the time (in seconds) to match the first logo found in a page, and the time (in seconds) to complete analyzing a page.

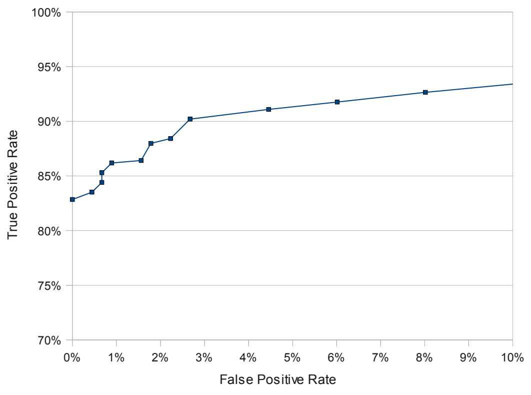
brand holders have multiple versions of logo (either used at the same time, or that have changed over time). Finally, we store two versions of each logo, a normal one and a color inverted one (for a total of 352 logos). The inverted version is included separately because it is a common stylistic practice to invert the color scheme (positive space to negative space) *and* because this transformation also inverts the direction of the keypoint gradients (and hence a normal logo will not match an inverted one well).

5.2 Performance

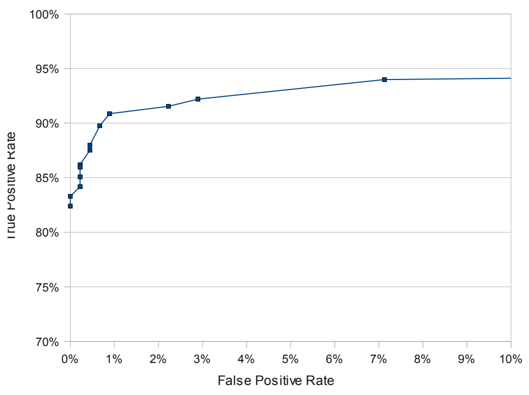
We first consider Verilogo's matching performance against our datasets, which reflects the *latency* before an alert is raised. The tests are performed on an HP Pavilion Elite desktop with a 2.5Ghz quad-core CPU (only one core was used for our matching task) with 6GB of DRAM (our memory footprint is modest however).

In Figure 2 we show the distribution of latencies to resolve the *first* logo on a page (if a page truly is attempting to represent a brand holder this first logo is almost always the one of interest) and to complete the analysis of the entire page (approximating a worst case scenario in which a logo at the bottom of a phishing page is a successful trust cue).

We consider two datasets separately. The Phish/Brand case captures the time to find a logo on a typical brand site (whether genuine or fraudulent) and captures the expected latency between rendering and notification. The first logo is identified within 1.5 seconds 99% of the time and virtually all such pages are scanned in their entirety within 5 seconds (bank pages in particular tend to be short). Note that the Phish/Brand datasets represent the type of pages where latency is most critical. Since the first logo identified is almost always the correct one (most brand holders place their logo at the top), we expect the measurement of 1.5 seconds to be the most representative



(a) Single threshold



(b) Multiple threshold

Figure 3: Receiver Operating Characteristic (ROC) curve of match performance on the Phish and Brand datasets. The graph on the left uses a single threshold value for all logos while the one on the right increases the threshold for a small number of indistinct logos.

of matching performance.

By contrast, the Hot/Random group of pages exhibits a far greater range. This is largely because many of these pages (particularly in the random set), are much longer (some well over 8000 pixels) or wider (some over 1200 pixels) and matching time is roughly proportional to pixel area. Still, the first logo is matched within roughly 6 seconds for 75% of the pages (for which a logo is found) and all matching within 10 seconds for 75% as well. The tail of the distribution is quite long however and 2–3% of these pages take over a minute to complete both the first match and complete scan. However, these latencies are non-critical since they proceed in parallel with page viewing and do not correspond to those cases used to represent banks and other heavily phished brand holders (it makes little sense for a phisher to try to garner trust by making a user scroll to the bottom of a 8000 pixel page to find the brand logo).

Finally, we remark that SIFT optimization has become a minor industry unto itself. Commercial implementations may offer performance several times greater than that of the Hess implementation used in our prototype. Moreover, recent research has shown that SIFT is easy to parallelize effectively, both across multicore CPUs and on GPUs [44, 12, 37]. For example, the open-source SiftGPU implementation can process an 800x600 image in 1/20th of a second [41].

To summarize, on today's processors SIFT-based scanning of Web pages is completely feasible and is poised to become dramatically cheaper still with respect to computational cost, due to increasing parallelism.

5.3 Accuracy

We first consider the accuracy of the SIFT algorithm itself. We focus here on only the Brand and Phish datasets, as these pages are the ones known to definitively contain logos and are the ones that are most essential for our matching algorithm to detect correctly.

Figure 3 shows the ROC curves for matching correctness. A false positive is when our algorithm detects a logo that was not present on the page; a false negative is when we fail to find the logo

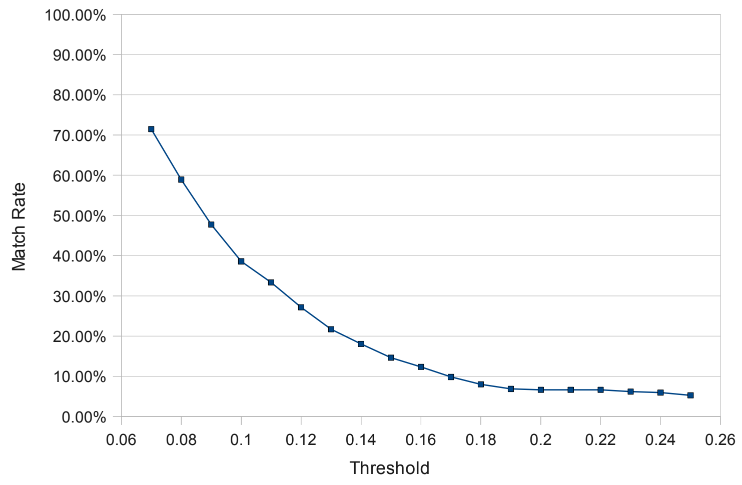


Figure 4: Match rate of our 352-entry logo database against Hot/Random dataset pages as a function of matching threshold — the same for all logos.

that does appear on the page. We consider the results using two different threshold rules.

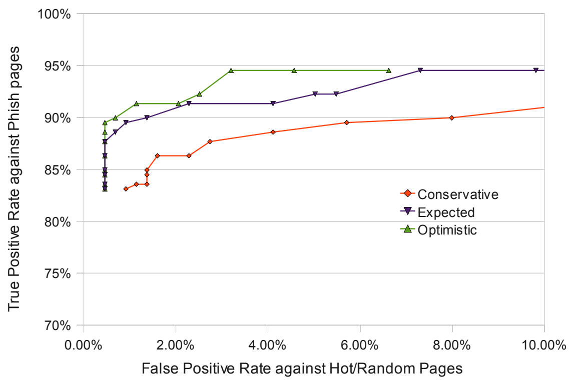
Using a single threshold (Figure 3 (a)), the performance is fairly good. With a threshold of 0.15, over 90% of pages are correctly matched and under 3% are matched to the wrong logo (note such mismatches can occur against any of the 352 instances in our logo database and not only for the 23 logos for which we have collected individual pages).

Many of the false positives are due to a small number of poorly performing logos with little discriminative power. For example, the Friendster logo and the inverted version of the Facebook logo are very difficult to distinguish as a result of similar words in a sans serif font (e.g., in headers or titles). Indeed, under a reasonable threshold setting, we find that four logos contribute more than 40% of the false positive cases. To mitigate this problem, we implemented a separate threshold for 13 such poorly performing instances (a multiplicative factor of between 1.3 and 2, selected by hand). Consequently, common pages with no logo present will be less likely to trigger a match (at the potential expense of missing some matches for these instances). The benefit of this optimization can be seen in the "Multiple threshold" version of the graph (Figure 3(b)) which shows that a 90% true positive rate is now achieved with 1% false positives.

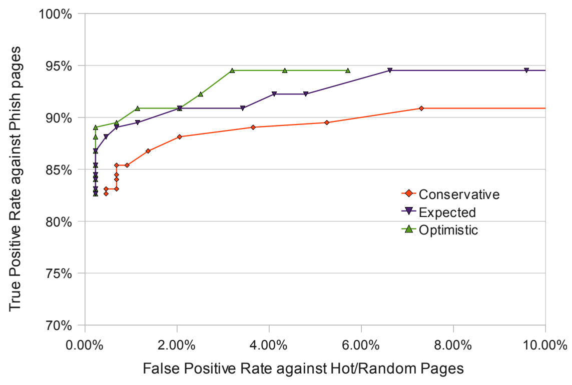
5.4 Keyboard heuristic

To get a sense for how many pages exist that make incidental use of logos, Figure 4 graphs the match rate as a function of the threshold value for the Hot and Random datasets (representing the typical pages that users encounter on the Internet). While for reasonable settings of the threshold most of these pages do not match, there is still a significant number that do. At a threshold of 0.17, 10% of the pages match at least one logo in our database (however, many of these matches are likely to come from the "poorly performing" logos we mentioned earlier).

However, our browser does not necessarily generate an alert for each of these pages, and will only notify the user if they provide keyboard input into the page. Thus Figure 4 is a conservative measure of our system's false positive rate and the true false positive rate is the intersection of this



(a) Single threshold



(b) Multiple threshold

Figure 5: Receiver Operating Characteristic (ROC) curve of overall accuracy on the Phish and Hot/Random dataset. False negatives are calculated against the Phish dataset while false positives are calculated against the Hot/Random dataset. The graph on the left uses a single threshold value for all logos while the one on the right increases the threshold for a small number of indistinct logos.

set with some function of the site content and user behavior. However, evaluating this heuristic is difficult to do directly. As we indicated earlier, lacking a live user trace, we have used qualitative classifications of user input likelihood to capture different ranges of potential false positive outcomes.

Figure 5 illustrates this impact while evaluating the accuracy of the complete system. Each graph provides a ROC curve in which a false positive indicates that an alert would be generated when it should not have been, and a false negative is a phishing site that was not detected. False negatives are calculated *only* against the Phish dataset (this is purposely pessimistic to test the worst case; in practice phishing pages are a small number of the pages visited) while false positives are calculated against the Hot/Random dataset (which more closely resembles the pages that users will typically visit and so stresses our keypress heuristic).

The lines labeled "Conservative" represent the ROC curves in which a keypress is deemed to have happened for any page not in the "No Input" class (i.e., if there was any way to provide input, then we assume you did so). The "Expected" curve is slightly more liberal, and assumes that keypresses *also* did not occur on pages in the "Incidental Input" category (input possible, but unlikely as with search bars). Finally, the "Optimistic" curve assumes that keypresses also did not occur in the "Optional Input" category (sites like blogs for which input is common, but not necessary) but that users always entered keystrokes in the "Essential Input" class.

Under the most conservative interpretation false positives can be held under 1% with a true positive rate of 83%. This is further reduced to under 0.5% under either of the more aggresive assumptions. A more significant difference arises when pushing the true positive rate to 90%. Here the conservative assumption produces to an 8% false positive rate, while the expected and optimistic models produce 1.4% and 0.7% respectively. However, we believe that these two curves are more likely to reflect real world use.

Our reported false-positive rate is intentionally conservative in two key ways. First, it represents the effectiveness of Verilogo entirely on its own. In a real deployment Verilogo would be combined with other mechanisms: page structure, link structure, hosting, etc., as we note in Section 3.4. We intentionally chose to focus on understanding Verilogo's contribution in isolation. Second, note also that the user is *alerted* only if he both navigates to a false-positive page and begins entering data, something that, as we discuss below, appears to be much rarer than the 1% rate above.

A user study. Finally, we conducted a small user study to explore the effectiveness of the keypress heuristic when applied to real user browsing behaviors. We asked six participants, all graduate students from a local computer science department, to install and use a modified version of the Verilogo browser plugin. The modified plugin does not display any alerts or perform any DNS lookups. However, it performs the image matching step and detects if a keypress has been incurred. Using the collected data, we could obtain the set of Websites that had a logo detected and incurred a keypress. When we remove all brand holder Web pages and phishing pages from this set, we would be left with the set of "false positive" pages (non-brand holder pages that are not phishing sites, but that would have generated an alert from our system).

We asked the participants to use the plugin for the span of a few hours to a few days. In total, 864 Websites were visited participants visited an average of 144 Websites in the course of the study. We examined the set of Websites that both had logos detected by the matching algorithm and had incurred keypresses, and removed all brand holder sites from this set. The remaining Websites should have been either phishing pages or false positives. However, there were no such Web pages, meaning that we had achieved a false positive rate of 0%. Although the human factors study was performed with only a small number of participants, we believe that these results are encouraging and indicate that our simple keypress heuristic may be effective for filtering out non-brandholder, non-phishing sites that contain a brand holder logo.

6 Discussion

6.1 Out-of-scope attacks

No single defense can completely secure the Web. Our tool is intended to prevent traditional Web Phishing for popular brands. Keyloggers and other malware installed on users' computers could steal their credentials from login sessions to legitimate sites; Verilogo cannot prevent these attacks. Similarly, if users are tricked into browsing to "browse-by" sites that exploit browser bugs to install malware, no logos need be present and no keyboard input is necessary for the attack to succeed. But such browse-by exploits are not phishing attacks. For Verilogo to be effective, we require that the user's computer be uncompromised and secure. This is admittedly a strong assumption given the state of today's systems security, but the same assumption is made by all in-browser security features that could be disabled by on-host malware.

6.2 Issues with logo matching

In the course of running our evaluation we observed issues with certain logos and brands that made them a challenge for Verilogo. We discuss these issues below. Fundamentally, our conclusion is that some logos more easily protected with Verilogo than others, because of features of the logos themselves or of the businesses they represent. Logos whose distribution is limited to brandholder pages; that do not frequently change; and that are highly distinct will receive better protection than logos lacking these qualities. While it would be better to have a defense that protected all logos equally, we believe that Verilogo can provide phishing defense for many of the most important

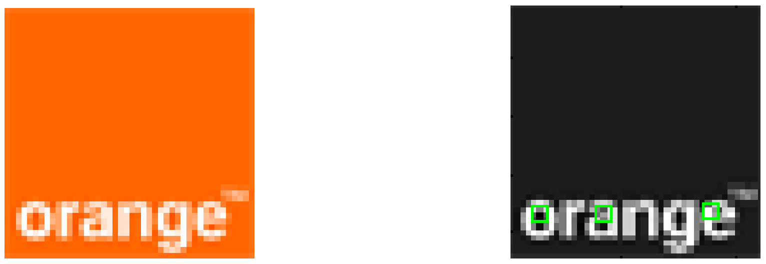


Figure 6: A logo not suitable for SIFT matching. The logo, as it appears online, lacks enough detail to generate a reliable set of feature points for matching. On the left, the logo; on the right, the logo as processed by our system, with SIFT keypoints in green.

e-commerce brands today. The varying amount of protection afforded to different logos is analogous to the “0x20” DNS spoofing defense [8], which is less effective for domains with short names like Amazon’s `a9.com`.

Scalability. The SIFT matching algorithm scales logarithmically with the number of logos in the database. Because the number of commonly phished sites is small, we expect that the logo database will also be small. For sites with smaller reach, such as regional banks, we envision a protocol by which they prompt their users to add the site logo to their Verilogo database. This would keep the total number of logos detected for any given user small.

Widely distributed logos. Some logos (e.g., Visa’s) are widely distributed on legitimate e-commerce sites, which of course require keyboard entry. This would likely lead to a false positive with our method as we have considered it, in isolation. It is unrealistic to expect that Visa will be able to individually track and authorize every page that uses its logo. It may be impossible to secure logos that, by their use, are extremely widely distributed, using techniques such as ours that are designed to enforce limited logo distribution.

Highly variable logos. We found that several of the brands in our study were represented by several variant logos; when our initial logo set was incomplete, we saw poor matching behavior. For example, Orkut is represented by several different marks, and PayPal has two versions of their logo, one with hollow text and one without. When several logos represent a brand, all must be included in the database. (This is especially important when a brand changes from an old logo to a new one, as phishing pages could take advantage of the brand equity associated with the old mark even if all legitimate sites have switched to the new one. Popular brandholders are unlikely to change logos frequently, as this would erode brand recognition.) A few logos (notably Google’s) appeared at widely varying scale. Although SIFT is intended to be scale-invariant, very small versions of logos become pixelated and lose information, leading either to more false positives or more false negatives. However, we believe that, using a relatively small number of logo database entries, we can provide coverage and protection for most of the commonly phished brands. We hope to explore scaling the logo database to include brands from smaller organizations and the frequency of required updates in future work.

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 been 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 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) [27], 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. An alternative option might simply be removing such logos from the database to minimize potential false positives (since they lack discriminatory power, they are not very useful at detecting phishing sites either).

Logos with mostly text. In many of the logos in our dataset, the image contains an insignia and the company name. For example, in the case of the CNN logo, 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 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 [15]. In that work, more statistically distinct features were discovered and given higher weight. We expect that feature points coming from the distinctive "CNN" region would provide greater discriminative power and be assigned a higher weight; it may be preferable to define a logo to exclude less-distinctive portions, such as CNN's ".com".

6.3 Possible evasion techniques

No defense is foolproof, and ours is no exception. Phishing sites can attempt to evade our system in one of two ways. Either they can set up their pages so that Verilogo cannot find logos on the page, or they can take advantage of our keyboard-entry heuristic to avoid having an alert raised.

We are concerned whether phishers can transform a logo to the point where the matching algorithm will no longer be able to detect the logo, or omit it completely, but still be able to fool a user into accepting the site as a legitimate brand holder site. Our intuition, supported by our user study discussed below, is that SIFT is quite effective at matching images across a range of transformations; transformations that defeat it should alter the logo drastically enough to be noticeable to people. Omitting the brand logo would likely significantly reduce the effectiveness of a phishing site.

A user study. To test our claim that SIFT is tolerant of logo transformations, we collected screenshots of sixteen brands whose logos are in our logo database, and applied simple transformations to the logo portion of the screenshots. Table 1 summarizes the transformations we applied.

We performed logo matching on the modified screenshots using a threshold of 0.15, and found that six of the screenshots were not correctly matched. Examining the correctly matched logos, we noted that SIFT appears to be tolerant of rotation, shear up to 10 degrees in x and 30 degrees in y, changes in perspective, changes in size, and removal of parts of the logo in certain cases. However, SIFT can be defeated by more drastic shear transformations, removing key distinctive portions of the logo, or changing the logo font.

Next, we wanted to determine if modifications to the logos in the brandholder Web pages would

Brand	Transformation	Match?
Alliance	30 degree shear x	no
Amazon	removed part of logo and changed font	no
Capital One	6 degree rotate	yes
Chase	3 degree rotate	yes
Citi	removed part of logo and changed perspective	yes
Comerica	changed perspective	yes
eBay	30 degree shear y	yes
Facebook	changed font	no
Google	resized logo	yes
HSBC	removed part of logo	yes
IRS	removed part of logo	no
PayPal	20 degree shear x	no
US Bank	changed perspective	yes
Wachovia	removed part of logo	yes
Wells Fargo	changed font	no
Western Union	10 degree shear x	yes

Table 1: Transformations applied to 16 brandholder logos. Using a threshold of 0.15, SIFT failed to correctly detect the logo in six of the screenshots.

arouse user suspicion. We conducted a human factors study to find the answer. For the user study, we used the sixteen modified screenshots, and also collected an additional 49 screenshots belonging to popular brand holders. The study consisted of 11 participants — graduate students, undergraduate students, and staff from a local computer science department. We asked participants in the study to rate their familiarity with each of the 65 Web pages (the modified Web pages were interspersed with the unmodified ones), and whether the Web page appeared to be "suspicious-looking" or "phishy." Participants were also asked to explain why a page looked suspicious. The 49 unmodified Web pages were included in the study as a control set, in order to examine the effect of priming participants by asking them to look for suspicious Web pages.

On average, 19.7% of participants reported unmodified Web pages as suspicious, while 51.7% reported modified Web pages as suspicious. Interestingly, these numbers changed only marginally (by 1 or 2%) when considering only the set of data where participants reported being moderately to very familiar with the given Web page. This is likely due to a combination of brand recognition (participants may notice or be familiar with a brand's logo but may not visit the Web page) and aesthetics (even if a user is not familiar with a brand, certain transformations, such as rotation or a drastic shear, may seem aesthetically "wrong"). Another interesting observation is that certain unmodified Web pages appeared to participants to be particularly suspicious-looking. For example, 6 out of 11 participants reported Microsoft's `live.com` Website as being suspicious. Furthermore, a number of participant responses indicated that poor aesthetics, or what appeared to be an unprofessional design, often contributed to a belief that a page was "phishy."

It appears that a number of users would notice a phishing attack if the brand logo was modified as described. Furthermore, many of these transformed logos would still be recognized by the image matching algorithm. Thus, the efficacy of an attack that attempted to subvert Verilogo by using simple transformations to modify a brandholder logo would likely be quite limited. We expect that modifying a logo in a way that would both subvert SIFT and almost completely escape user detection is a non-trivial effort.

Other evasion techniques. Attackers may try to evade logo detection by mounting an attack on less popular brands whose logos are not in our database, or on Web pages that rely on other trust cues (such as favicons or trust seals) rather than logos. We plan to investigate the scalability of our approach with respect to handling larger logo databases and including alternative trust cues, as well as the ease of updating the database to include new trust cues.

We have encountered sparse examples of phishing sites which make use of virtual keyboards or pinpads for login. These techniques would bypass our current mechanism. However, we believe that, first, forcing phishers to rely on virtual keyboards when their legitimate targets do not will reduce the effectiveness of phishing; and, second, that Verilogo can be refined with additional heuristics in anticipation of such attacks. We stress that our keyboard-input heuristic is intentionally a simple and conservative one to allow better evaluation of the underlying scheme.

Phishers might also attempt to separate the "landing" page featuring the logo from the page in which the user is prompted to enter his private information. Since this second page would have to be free of any logos, this should again reduce the effectiveness of the phishing site. It should also be possible to modify Verilogo to track logo-taint across page navigation by click, so that a landing page featuring the Bank of America logo raises a warning when, on the next page, the user initiates keyboard input.

Finally, attackers create pages that, over the time the user views them, alternate between showing and hiding the logo, in the hopes of having the logo not present when Verilogo captures a page-image PNG for analysis. We believe that having such "Cheshire cat" logos on phishing pages will reduce their believability to users (since logos on legitimate brandholder sites do not disappear and reappear), but we have not performed a user study to verify this belief.

Note that capturing a PNG of the entire page as rendered means that attackers have no advantage from embedding logos using cropped iframes from legitimate sites — the relevant domain is the one shown in the address bar.

It is possible that users will click through or ignore Verilogo's warnings, a problem with any system that alerts the user to potentially unsafe actions [35, 38]. When Verilogo identifies a suspicious site, it may be able to alert a phishing clearinghouse or the brandholder in addition to warning the user, leading to a takedown or blacklisting that would protect all phished users, including those who click through Verilogo's warnings. In this way Verilogo acts as a distributed brandholder-oriented phishing detection system as well as a user-oriented real-time phishing detection system. (For the privacy implications of such phishing alerts, see below.)

6.4 User privacy

One concern with our proposed DNS-based system for logo authorization is the fact that the DNS queries it uses to determine if the logo appears on a page owned by the brand holder divulge, first, the fact that the particular user has viewed a page at the site being queried about and, second, that this site includes the brandholder's logo. (Arguably the second concern has a positive side, since such reports can facilitate site takedowns.) One solution is to use a DomainKeys-style approach, in which users do not communicate directly with brandholder servers. Alternatively, we believe that cryptographic techniques can be used to perform BAP-record lookups in a privacy-preserving way.

7 Conclusions

In this paper, we have described an image-based system for automatically recognizing branded logos on Websites for the purposes of fraud detection. This capability goes to the essence of how fraud detection works: first a site is evaluated as representing some brand and then it is checked whether the representation is unauthorized. In practice, this approach is inherently reactive and inevitably time consuming. We suggest that automated logo matching and authorization is an appealing complement (proactively achieving over 90% accuracy). We've demonstrated a working implementation and shown that it is feasible to integrate near real-time analysis into browsers on today's computer systems (and will only become cheaper since Web pages are growing longer more slowly than processors are getting faster). We demonstrated that a simple and robust heuristic, deferring alerts until the user has entered keyboard input, is sufficient to limit false positives to between 1 and 2% under reasonable models of when users interact with pages. Our user studies suggest that users may notice images sufficiently distorted to defeat our SIFT matching, but we believe future research is called for on the feasibility of logo-specific adversarial modifications, and on adversarial computer vision generally.

References

[1] Top targets for the identified threats. `http://www.ghacks.net/wp-content/uploads/2009/12/toptargets.png.jpg`.

[2] Statistics about phishing activity and phishtank usage, Dec. 2009. `http://www.phishtank.com/stats/2009/12/ ?y=2009&=12`, 2009.

[3] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair. A comparison of machine learning techniques for phishing detection. In *Proc. eCrime '07*.

[4] S. Afroz and R. Greenstadt. Phishzoo: An auomated web phishing detection approach based on profiling and fuzzy matching. Technical report, Drexel University, 2009.

[5] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proc. CVPR '97*.

[6] J.-Y. Chen and K.-T. Chen. A robust local feature-based scheme for phishing page detection and discrimination. In *Proc. Web 2.0 Trust Workshop*, 2008.

[7] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell. Client-side defense against web-based identity theft. In *Proc. NDSS '04*.

[8] D. Dagon, M. Antonakakis, P. Vixie, T. Jinmei, and W. Lee. Increased DNS forgery resistance through 0x20-bit encoding: SecURItY viA LeET QueRieS. In *Proc. CCS '08*.

[9] R. Dhamija, D. Tygar, and M. Hearst. Why phishing works. In *Proc. CHI '06*.

[10] J. S. Downs, M. Holbrook, and L. F. Cranor. Behavioral response to phishing risk. In *Proc. eCrime '07*.

[11] L. Eggert. Global SPF depoyment. Online: `https://fit.nokia.com/lars/meter/spf.html`, 2010.

[12] H. Feng, E. Li, Y. Chen, and Y. Zhang. Parallelization and characterization of sift on multi-core systems. In *Proc. IISWC '08*.

[13] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In *Proc. WWW '07*.

[14] J. Franklin, V. Paxson, A. Perrig, and S. Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *Proc. CCS '07*.

[15] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Proc. ICCV '07*.

[16] A. Y. Fu. Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd). *IEEE Trans. Dependable Secur. Comput.*, 3(4), 2006. Senior Member-Wenyin,, Liu and Senior Member-Deng,, Xiaotie.

[17] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *Proc. WORM '07*.

[18] P. Hallam-Baker. Secure internet letterhead. In *Proc. W3C Workshop on Transparency and Usability of Web Authentication*, Mar. 2006.

[19] R. Hess. SIFT feature detector. `http://web.engr.oregonstate.edu/~hess/`, 2009.

[20] C. Jackson and A. Barth. ForceHTTPS: Protecting high-security web sites from network attacks. In *Proc. WWW '08*.

[21] M. Kumar, P. Prakash, M. Gupta, and R. R. Kompella. Phishnet: Predictive blacklisting to detect phishing attacks. In *Proc. Infocom '10*.

[22] P. Kumaraguru, J. Cranshaw, A. Acquisti, L. Cranor, J. Hong, M. A. Blair, and T. Pham. School of phish: A real-world evaluation of anti-phishing training. In *Proc. SOUPS '09*.

[23] B. Leiba and J. Fenton. DomainKeys Identified Mail (DKIM): Using Digital Signatures for Domain Verification. In *Proc. CEAS '07*.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2), 2004.

[25] C. Ludl, S. Mcallister, E. Kirda, and C. Kruegel. On the effectiveness of techniques to detect phishing sites. In *Proc. DIMVA '07*.

[26] J. Ma, L. K. Saul, S. Savage, and G. M. V. lker. Identifying Suspicious URLs: An Application of Large-Scale Online Le arning. In *Proc. ICML '09*.

[27] J. Matas and K. Zimmermann. A new class of learnable detectors for categorisation. In *Proc. SCIA '05*.

[28] E. Medvet, E. Kirda, and C. Kruegel. Visual-similarity-based phishing detection. In *Proc. SecureComm '08*.

[29] T. Moore. Personal communication, 2010.

[30] T. Moore and R. Clayton. The consequence of non-cooperation in the fight against phishing. In *Proc. eCrime '08*.

[31] T. Moore and R. Clayton. Examining the impact of website take-down on phishing. In *Proc. eCrime '07*.

[32] T. Moore and R. Clayton. The impact of incentives on notice and take-down. In *Proc. WEIS '08*.

[33] P. Moreels and P. Perona. A probabilistic cascade of detectors for individual object recognition. In *Proc. ECCV '08*.

[34] T. Oda, G. Wurster, P. van Oorschot, and A. Somayaji. Soma: Mutual approval for included content in Web pages. In *Proc. CCS '08*.

[35] S. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The emperor's new security indicators. In *Proc. IEEE Oakland '07*.

[36] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang. An empirical analysis of phshing blacklists. In *Proc. CEAS '09*.

[37] S. N. Sinha, J. michael Frahm, M. Pollefeys, and Y. Genc. Gpu-based video feature tracking and matching. Technical report, In Workshop on Edge Computing Using New Commodity Architectures, 2006.

[38] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. Cranor. Crying wolf: An empirical study of SSL warning effectiveness. In *Proc. USENIX Sec. '09*.

[39] R. thomas and J. Martin. The underground economy: priceless. *;login*, 31(6), 2006.

[40] M. W. Wong. Sender Authentication: What To Do, 2004.

[41] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `cs.unc.edu/~ccwu/siftgpu`.

[42] M. Wu, R. C. Miller, and S. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proc. CHI '06*.

[43] C. Yue and H. Wang. Anti-phishing in offense and defense. In *Proc. ACSAC '08*.

[44] Q. Zhang, Y. Chen, Y. Zhang, and Y. Xu. Sift implementation and optimization for multi-core systems. In *Proc. IPDPS '08*.

[45] Y. Zhang, S. Egelman, L. Cranor, and J. Hong. Phinding phish: Evaluating anti-phishing tools. In *Proc. NDSS '07*.

[46] Y. Zhang, J. I. Hong, and L. F. Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proc. WWW '07*.