Mitigate Web Phishing Using Site Signatures*

Chun-Ying Huang[1], Shang-Pin Ma[1], Wei-Lin Yeh[1], Chia-Yi Lin[1], and Chien-Tsung Liu[2]
[1] Department of Computer Science and Engineering, National Taiwan Ocean University
[2] Networks and Multimedia Institute, Institute for Information Industry
Email: {chuang,albert}@ntou.edu.tw, {wlyeh,cylin}@snsl.cs.ntou.edu.tw, netsaga@nmi.iii.org.tw

***Abstract*—Phishing is now a serious threat to the security of Internet users' confidential information. Basically, an attacker (phisher) tricks people into divulging sensitive information by sending fake messages to a large number of users at random. Unsuspecting users who follow the instruction in the messages are directed to well-built spoofed web pages and asked to provide sensitive information, which the phisher then steals. Statistics published by the anti-phishing working group (APWG) show that, at the end of Q2 in 2008, the number of malicious web pages designed to steal users' confidential information had increased by 258% over the same period in 2007. Therefore, protecting users from phishing attacks is extremely important.**

Existing anti-phishing solutions detect mimicked phishing pages by either text-based features or visual similarities of web pages. The former one can be bypassed using image based phishing attacks while the latter one may suffer from great variants of phishing pages. In this paper, we propose a novel technique that identify the real domain name of a visiting web page based on signatures created for web sites. Site signatures, including distinctive texts and images, can be systematically generated by analyzing common parts from pages of a web site. On matching a signature, the domain name of the visiting URL is checked first and then redirected if the domain name is unmatched. The result shows the proposed method achieves a high accuracy and low error rates.

***Index Terms*—Anti-Phishing, Feature Selection, Image Extraction, Site Signature, URL Redirection**

I. Introduction

Phishing is a malicious activity whereby an attacker (phisher) tries to trick Internet users into providing confidential information [6]. It is a serious problem because phishers can steal sensitive information, such as users' bank account details, social security numbers, and credit card numbers. To achieve this goal, a phisher first sets up a fake website that looks almost the same as the legitimate target website. The URL of the fake website is then sent to a large number of users at random via e-mails or instant messages. Unsuspecting users who click on the link are directed to the fake website, where they are asked to input their personal information. Although the process of setting up a fake website sounds complicated, reports show that it is much easier than before as there are now "phishing kits" [14], [5] that can create a phishing site in a very short time. Users believe that responsible enterprises should protect them from phishing attacks; thus, in addition to the risk of personal information leakage, successful phishing attacks can seriously damage business enterprises, especially a company's brand reputation [13], [19].

As phishing is a serious threat to both users and enterprises, several anti-phishing techniques have been developed. In general, the techniques can be classified as either list-based or heuristic-based technologies. List-based techniques maintain a black list or a white list, or both. Many anti-phishing mechanisms use a black list to prevent users from accessing phishing sites. However, the effectiveness of black list filtering depends on the coverage, freshness, and accuracy of the list. The URLs are usually reported by Internet users or collected by web crawlers, and list maintainers are responsible for verifying whether or not the listed URLs are really phishing sites. Though a well maintained black list can filter most well-known phishing sites, it obviously can not filter unreported, uncollected, or unanalyzed URLs. No list can guarantee 100% coverage and up-to-date freshness; and list-based filtering techniques often generate false negatives.

Some anti-phishing mechanisms use white lists that contain the names of trusted domains. If a user visits an unlisted website, a white-list based filter may block the site immediately or require the user to make decisions on the fly. The drawback of this method is that the user may become annoyed if some sites are blocked or if the system constantly requests confirmation. Sometimes, it may even be difficult for the user to make a decision. In the end, the user may loose patience with having to validate unlisted sites and decide to disable the filter mechanism.

Heuristic-based mechanisms employ several criteria to determine whether a website is a phishing site. Common criteria includes the domain name of a web page, the complete URL of a web page, visual similarities, input fields embedded on a web page, and keywords. Heuristic-based mechanisms may use only one criterion to assess web sites. For example, the basic CANTINA filter [24] only calculates the TF-IDF score. In contrast, the advanced CANTINA filter and the SpoofGuard filter [3] use a weighted score based on several criteria. Given a set of predefined weights for each criterion, the overall score used to evaluate a site is calculated by

$$s = \sum w_i P_i,$$

where w_i and P_i are, respectively, the weight and the probability of a given criterion i.

Our approach is also a heuristic based method. Compare with other heuristic based methods, our approach differs in three aspects. First, most anti-phishing techniques detect

*This work was supported in part by National Science Council under the grant number NSC 97-2218-E-019-004-MY2, NSC 99-2218-E-019-001-MY2 and by Taiwan Information Security Center at NTUST(TWISC@NTUST) under the grant number NSC 99-2219-E-011-004.

phishing pages only by text-based features. They do not use images as a feature because they are relatively more difficult to be extracted. There are researches that detect phishing pages by visual similarities [8], [2]. However, it is possible that there can be only limited parts of a phishing page looks similar to the official web page. While visual similarity based solutions create signatures using a whole page, if the ratio of similar parts to all parts is not high enough, a phishing page can pass the check. Second, the signature used to detect phishing pages are composed by common features extracted from an entire website instead of a single page. Thus, one signature can be used to detect different targeted pages or variants of a website. This reduces the required space to store signatures. Third, we force a detected phishing page to be redirected to the correct one instead of providing hints to users. Since users are not always aware of alerts displayed by anti-phishing toolbars [23]. It would be better if URL redirections are enforced when the accuracy of detection rates is good enough and the error rate is limited.

The remainder of this paper is organized as follows. In the next section, we review existing solutions to phishing attacks. Section III explained the proposed solution in detail, which includes how the features of a signature are chosen and how to match web pages against the signatures. In Section IV, we provide an evaluation of the proposed approach. A concluding remark is finally given in Section V.

II. Related Work

There are various methods for protecting users from phishing attacks as they surf the Web. Most web browsers have built-in anti-phishing solutions [17], [15] that block phishing sites based on well maintained black lists and white lists. For example, Firefox blocks malicious web sites based on lists from Google [9] and StopBadware.org [22]. Clearly, the effectiveness of phishing detection depends on the coverage, freshness, and the accuracy of the employed list. Many other solutions can be implemented as add-ons, plug-ins, or extensions for web browsers [3], [9], [18], [24].However, researchers [4] evaluate several popular anti-phishing toolbars and conclude that the phishing detection rate is not good enough. While the better implementations can detect more than 75% of phishing attacks, the least effective ones can detect less than half of such attacks. Furthermore, a toolbar with a high detection rate may also have a high false positive rate, i.e., it blocks non-phishing sites.

As mentioned in Section I, in addition to using black lists, some approaches employ heuristics to distinguish between phishing and non-phishing sites [12], [8], [3], [24], [21], [7]. In [24], the authors identify distinctive keywords by TF-IDF in the visiting web page and search these keywords via Google. The authors assume that the results returned by Google should include the visiting web page, or at least having the same domain name suffixes. Then, by matching the domain name of the visiting web page against those results returned by Google, a phishing page can be identified if nothing matched. In [3], several different heuristics including host names, URLs, password fields, hyperlinks, and image hashes, are combined and used to compute an overall phishing score. When a score exceeds, a pop-up window is shown to notify the user. Fu et al. [8] propose to use visual similarities to detect phishing pages. They treat a whole web page as an image and convert it to a low resolution image. The colors and coordinates of converted images are stored as signatures. On visiting a web page, the whole page is also converted to a low resolution image and then used to match against those signatures in a database. The similarity distance is computed by the Earth mover's distance (EMD) algorithm Chen et al. [2] propose another visual similarity based solution. Instead of applying EMD for low resolution images, they extract web page features based on contract context histogram [10] and use the feature to match two web pages.

III. The Proposed Solution

The rationale behind the proposed solution is simple. A successful phishing attack exploits the phenomenon that users usually determine the correctness of a website by only visual similarities. They are not always aware of the URL of a visiting page. Hence, to mitigate such an attack, our solution tries to automatically do the check for users. We extract common stable features from a website as its signature and then create the map between the signature and the website domain name. On matching a signature, the corresponding domain name is compared with the domain name of the visiting URL. If they are unequal, a URL redirection is enforced to prevent the user from being phished.

In the proposed solution, we try to extract common *stable* features that must be kept when mimicking a web page. Features extracted from a web site are then grouped as a site signature and used to detect phishing pages that try to mimic the targeted website. The proposed solution contains two parts. One is to build site signatures for websites and another is to match web pages against site signatures. Features included in a site signature can be classified into two types, i.e., text-based and image-based features. The extraction of these two types of features are discussed separately in this section.

Readers may notice that the proposed solution cannot prevent users from pharming attacks, which are able to make the domain name of a phishing page indistinguishable from a valid website. The discussion of pharming attacks is out the scope of this paper. Interested readers can refer to researches focused on the prevention of pharming attacks [11].

A. Extraction of Text-Based Features

Text-based signatures can be extracted from different parts of a web page. The extraction of text-based signatures is much simpler then image-based signatures and it is also common is similar researches. Thus, here we only briefly introduces what and how we extract from a website. The featured texts are extracted from the following three parts of a website:

1) *Title keywords*: Based on our observations, we find that web pages of the same domain name usually has some common keywords in the title. For example, almost

every eBay web page has a keyword `eBay` placed at the beginning of the title and almost every Yahoo web page has a keyword `Yahoo` embedded in the title. Therefore, by counting the frequency of each word appeared in titles of web pages, we are able to obtain the keywords that used most often in title.

2) *URL keywords*: URL is a common feature to detect phishing sites because attackers often tries to confuse users by embedding strings similar to the domain name of the targeted website in phishing URLs. Hence, we extract URL keywords as parts of the signature from the domain name of a valid website.
   Not all words appeared in a domain name are extracted as keywords. To extract keywords from a domain name, we first split words in the domain name by the dot symbol. Then, we remove top level domains (TLDs) and country-code top level domains (ccTLDs). Common words used in domain, for example, `www` and `mail`, are removed.
3) *Content keywords*: We also extract content keywords from a website. We treat sampled pages of a website as a single document and extract all words as keyword candidates. To effectively identify the real keywords of the website, we also collect documents from on-line news websites like BBC and CNN as the base to compute the TF-IDF [21] score. The words with top N highest TF-IDF socre are chosen as the keywords of the website.

It is worth noticing that the signatures are extracted from the entire website, not a single web page. Thus, the extracted features can be used to identify most pages of a website.

B. Extraction of Image-Based Features

Image-based features is important to determine the real domain name of a visiting web page. To extract image-based features efficiently, we assume that the image-based features must be seen in the very first page of a website. For example, we can always see the website's logo, which is usually a common image feature, in the website's welcome page. Thus, we always start the image-based feature extraction from the welcome page of a website.

The complete algorithm to extract image-based features is depicted in Figure 1. The input of the image-based feature extraction process is the URL of the welcome page of a website and the maximum number identified common image blocks. First, we extract all images embedded in the welcome page as image feature candidates. Then, we sampled another w pages within the same domain name and also extract images embedded in those pages. Sampled pages are usually picked up randomly by following links embedded in the welcome page. Now we have two sets of images: S_c contains candidates collected from the welcome page and S_d contains images used to identify featured images in S_c.

Suppose the cardinalities of S_c and S_d are n and m, respectively. For each candidate $C_i \in S_c$ $(1 \leq i \leq n)$, we run our image extraction algorithm (explained later) to identify

```
Input: u - The welcome page URL of a target website;
       N - The maximum number of identified image
       blocks.
Output: S - The set containing image blocks for the
        website.
1  $S_c$ = find_image($u$) ;
2  $H$ = sample_hyperlink($u$);
3  $S_d$ = $\{\emptyset\}$ ;
4  foreach $h \in H$ do
5      $S_d$ = $S_d \bigcup$ find_image($h$);
6  $n$ = $|S_c|$ ;      /* number of images in $S_c$ */
7  $m$ = $|S_d|$ ;      /* number of images in $S_d$ */
8  $B$ = a map that maps an image block to a counter;
9  foreach $C_i \in S_c$, $1 \leq i \leq n$ do
10     foreach $D_j \in S_d$, $1 \leq j \leq m$ do
11         $b$ = cib_extract($C_i$, $D_j$);
12         if $b \neq \emptyset$ then
13             if $b \notin B$ then $B[b]$ = 1;
14             else $B[b]$ = $B[b]$ + 1;
15 $S$ = first $N$ common image blocks in $B$ (sort by counter
   values in a descendent order);
16 return $S$
```

Fig. 1. The algorithm to extract image-based feature.

common parts by matching C_i against each image $D_j \in S_d$ $(1 \leq j \leq m)$. The output of the image extraction algorithm are common image blocks that can be found in images of various pages belong to the same website. With the image extraction algorithm, we are able to avoid the misuse of a whole image as an image feature, as shown in Figure 2. When common image blocks are all identified, we then pick the most frequently used one as the representative of the site's image feature.

The most important component in the above procedure is the image extraction algorithm, i.e., the `cib_extract` function used in Figure 1. The input of this algorithm is two images and the output of the algorithm is the area of the common image block. In the algorithm, we need four steps to identify common image blocks. An overview of the four steps is depicted in Figure 2. Readers can find the complete algorithm description in Figure 3. First, we have to identify *key points* from each of the two input images. Key points are robust features that can be kept even if an image is resized, rotated, and distorted. Therefore, these features can be used to recognize image patterns. There are several well known algorithms [16] to identify key points of images. In this work, we choose SIFT to identify key points because its availability and performance, as discussed in [16]. The output of SIFT is a set of key points, each key point is represented as a multi-dimension vector. Suppose two sets, S_1 and S_2, contain key points identified from the two input images respectively. Pairwise key points from the two sets can be matched by measuring minimum Euclidean distance between any unmatched key points in the sets.

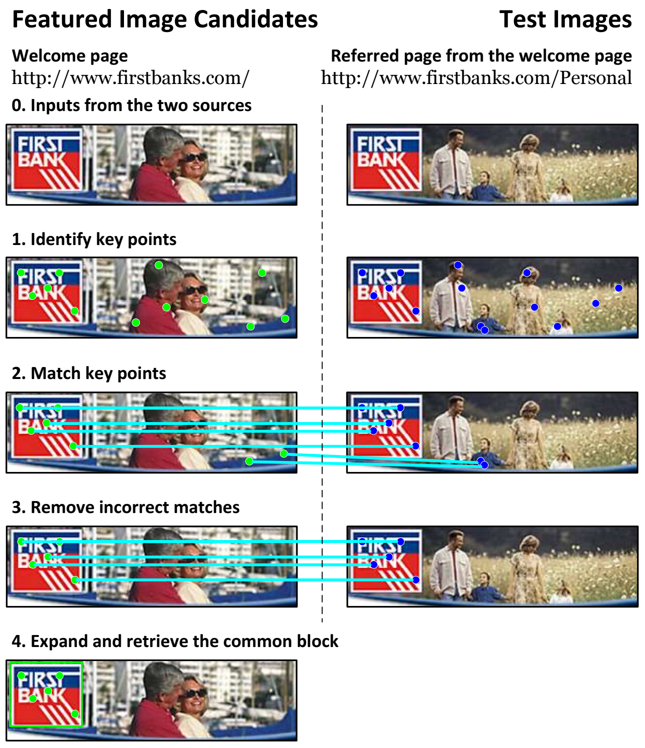
Featured Image Candidates
Test Images
Welcome page
http://www.firstbanks.com/
Referred page from the welcome page
http://www.firstbanks.com/Personal
0. Inputs from the two sources
1. Identify key points
2. Match key points
3. Remove incorrect matches
4. Expand and retrieve the common block

Fig. 2. An overview of the image extraction algorithm.

In the third step, we have to remove incorrectly matched key points so that the common image block can be identified correctly. If a common image block can be found in two different images, we assume that the positions of key points matched in the block should be relatively the same within the block. Based on the assumption, an incorrectly matched key point pair ($P_{1,a}$,$P_{2,a'}$) in K can be identified by the following procedure.

1) Reset a counter CTR to zero.
2) Obtain the x- and y-coordinates of the two key points $P_{1,a}$ and $P_{2,a'}$ as $Pt_{1,a}$ and $Pt_{2,a'}$, respectively.
3) Randomly choose m matched key point pairs from the set K.
4) Iteratively choose one from the m matched key point pairs. Suppose a matched key point pair ($P_{1,b}$, $P_{2,b'}$) is chosen, the x- and y-coordinates of the two key points are also obtained as $Pt_{1,b}$ and $Pt_{2,b'}$. Then, compute the two vectors $\overrightarrow{v_1}$=($Pt_{1,b} - Pt_{1,a}$) and $\overrightarrow{v_2}$=($Pt_{2,b'} - Pt_{2,a'}$) and check whether $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ are parallel or not. This can be done by calculating the cosine value of the included angle for the two vectors by $\cos\theta = (\overrightarrow{v_1} \cdot \overrightarrow{v_2})/(|\overrightarrow{v_1}||\overrightarrow{v_2}|)$. If $\cos\theta$ is greater than a given threshold, the counter CTR is increased by 1.
5) After the m key point pairs are all processed, if the counter CTR is less than $0.3 \times m$, the key point pair ($P_{1,a}$,$P_{2,a'}$) are removed from K.

An example of the above procedure can be found in Figure 4. In the figure, the key point pairs (F, F') and (G, G') should be removed. By the key point removing algorithm, we can see

```
Input: $I_1$ and $I_2$ - The two images used to extract the
       common image block.
Output: $b$ - The common image block or $\emptyset$ if none is
        found.
1  $P_1$ = identify_keypoints($I_1$);
2  $P_2$ = identify_keypoints($I_2$);
3  $K$ = a map that maps a key point to another key point;
4  foreach $P_{1,i} \in P_1,\ 1 \le i \le |P_1|$ do
5      $min$ = $+\inf$;
6      foreach $P_{2,j} \in P_2,\ 1 \le j \le |P_2|$ do
7          $d$ = Euclidean_distance($P_{1,i}$, $P_{2,j}$);
8          $P_m = \emptyset$;
9          if $d < min$ then
10             $min = d$;
11             $P_m = P_{2,j}$;
12     if $min <$ {pre-defined threshold} then
13         $K[P_{1,i}] = P_m$;
14         Remove $P_{1,i}$ from $P_1$;
15         Remove $P_m$ from $P_2$;
16 foreach $P_{1,a} \in indexOf(K)$ do
17     $ctr = 0$;
18     repeat
19         Randomly select $P_{1,b}$ from $indexOf(K)$;
20         $P_{2,a'} = K[P_{1,a}]$;
21         $P_{2,b'} = K[P_{1,b}]$;
22         $\overrightarrow{v_1} = positioinOf(P_{1,b}) - positioinOf(P_{1,a})$;
23         $\overrightarrow{v_2} = positioinOf(P_{2,b'}) - positioinOf(P_{2,a'})$;
24         $\cos\theta = (\overrightarrow{v_1} \cdot \overrightarrow{v_2})/(|\overrightarrow{v_1}||\overrightarrow{v_2}|)$;
25         if $\cos\theta \ge$ {threshold} then $ctr = ctr + 1$;
26     until $m$ times ;
27     if $ctr < 0.3 \times m$ then
28         Remove $P_{1,a}$ and $K[P_{1,a}]$ from $indexOf(K)$ and
           $K$, respectively;
29 if $|K| <$ {threshold} then return $\emptyset$ $A = \{\emptyset\}$;
30 foreach $P_{1,a} \in indexOf(K)$ do
31     $a$ = flooding_fill_expand($positionOf(I_1, P_{1,a})$);
32     $A = A \bigcup a$;
33 return the area that covers all blocks in $A$
```

Fig. 3. The algorithm of extracting the common image block, i.e., the cib_extract function.

that pairwise vectors starting from F and F', e.g., $\overrightarrow{FA}/\overrightarrow{F'A'}$, $\overrightarrow{FC}/\overrightarrow{F'C'}$, and $\overrightarrow{FG}/\overrightarrow{F'G'}$, are not parallel. Therefore, both the key points F and F' should be removed. After finishing the key point removing process, all the key point pairs that are preserved in the set K are then used to identify the common image block in the final step.

In the final step, the area of the common image block is identified based on the selected key points in the set K. We iteratively select a key point P_k from K, read the color value of P_k, and make a "virtual flooding fill" starting from P_k's position. Instead of filling the obtained color in the image, we

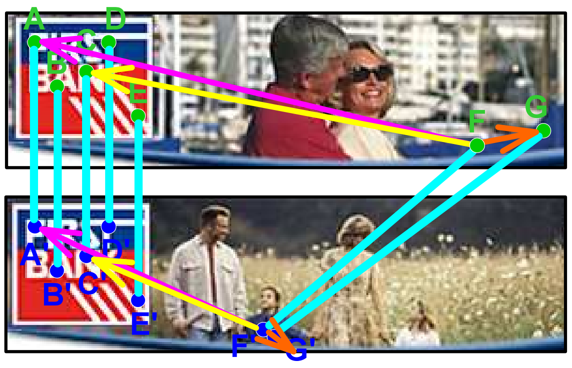

Fig. 4. Removing incorrectly matched key points from the images.

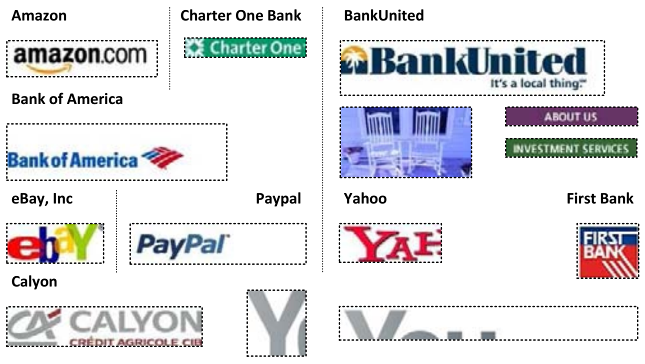

Fig. 5. Part results of common image block extraction.

use the flooding fill algorithm to visit all points that can be reached by the algorithm and keep the maximum/minimum x- and y-coordinates of visited points. A rectangle, which can be defined by the top-left corner (min-x, min-y) and the bottom-right corner (max-x, max-y), can be marked on finishing running a flooding fill. When all rectangles are marked, the common image block can be further identified by a rectangle that covers all marked rectangles. After analyzing a website, a number of representative image blocks can be extracted and these image blocks is then used to detect mimicked websites.

C. Matching Against Phishing Pages

On visiting a web page, the entire HTML document is analyzed and the matching process starts only if the HTML page contains at least a form field, a Java applet, or a Flash object. As phishing pages target on personal information, they must provide interfaces for users to provide their secrets. Therefore, a web page without form fields, Java applets, or Flash objects are not checked. Each feature of a site signature is used to match against the visiting web page. The result of matching a feature is a score ranging from 0 to 10 points. After matching all features, a weighted overall score is then obtained by $s = \sum w_i F_i$, using a set of predefined weights w_i for each feature F_i. A site signature is said to be matched if the overall score is greater than a predefined threshold.

There are two strategies to match texts. One is exact matching and another is similarity distance measurement. Note that both strategies are case insensitive. In the proposed solution, we match content keywords by exact matching. The score of matching content keywords depends on the weight of a matched keyword and the total number of matched keywords.

On the contrast, title keywords and URL keywords are both matched by similarity distance measurement. The distance of two input texts are measured by the Levenshtein algorithm [12]. A distance of zero indicates that two texts are exactly the same. To measure the similarity distance between a given input string s of length $|s|$ and a keyword k of length $|k|$, we follow the two principles below:

1) If $|s| \leq |k|$, we measure the similarity distance directly.
2) Otherwise, for each $n \geq |k|$, we split s into $(|s|-n+1)$ n-gram pieces and then measure the similarity distance between each n-gram text and k. The measured minimum similarity distance is then used as the final distance between the string s and the keyword k.

For example, matching a string `www.eday.com` against a keyword `ebay` would give us a similarity distance of 1 since the minimum distance can be evaluated by the 4-gram `eday` and the keyword; matching another string `www.e-bay.net` against a keyword `ebay` also give us a similarity distance of 1 since the minimum distance can be obtained by the 5-gram `e-bay` and the keyword. The score of matching title keywords and URL keywords depends on final similarity distance. A similarity distance of 0 get a full score of 10 points. Otherwise, it is linearly decreased to zero if the similarity distance is greater than or equal to $|k|$.

Matching image-based features is straight forward. Each image embedded in the visiting web page is extracted and then used to match against the common image block of a site signature. The score of image matching is evaluated by the ratio of matched key points to the total number of key points identified in the common image block. If n out of m key points from the common image block can be matched with key points of the input image, the image matching score is $(n/m) \times 10$. If there are multiple common image blocks available for a signature, the maximum score is used as the overall image matching score.

IV. Evaluation

A. Extraction of Common Image Blocks

We evaluate the common image block extraction algorithm for 50 well known web sites. In addition to popular phishing targets, most of these benchmark sites are selected from bank websites. For each website, we crawl 16 pages within the website and then use the proposed algorithm to identify common image blocks.

Figure 5 shows parts of the results of common image block extraction. We use a dashed rectangle to indicate the actual area extracted by the algorithm. The area can be much larger than the actual logo. This is due to the expansion made by the flooding fill algorithm. Since the expanded area does not affect the result of image matching, we can just ignore those additional blank areas. In the figure, we can see that the logo of a website can be always extracted correctly. The result also show that that the proposed algorithm is reliable even if the common image blocks are embedded in complicated background images. Readers may notice that some non-logo image blocks are extracted as well. Take the example of "BankUnited", besides the logo itself, the algorithm also extracts other common image blocks that are used among all other pages within the same domain. Such a result depends on the design template used by the website and hence it is unavoidable. Readers may also notice that some extract common image blocks from "Yahoo" and "Calyon" websites are fragmented. For the former one, it is because the key point identification algorithm we chosen only identify key points on the left half parts of the image. Although it only extracts parts of a common image block, it does not affect the effectiveness of feature matching. For the latter one, we find the original images downloaded from the "Calyon" website are fragmented. The fragmentation is not caused by the proposed algorithm.

B. Detection of Phishing Web Pages

For the detection rate of phishing web pages, the evaluation is done for both phishing sites and benign websites. For phishing sites, we collect 200 phsihing samples from Phish-Tank [20]. The samples contain top 10 well known phishing pages including Amazon, Bank of America, eBay, Paypal, and Yahoo. We also collect another 270 benign websites from Alexa [1] that are not included in site signatures to evaluate the false positive rates of the proposed solution. The 186 out of the 200 phishing samples can be detected and then redirected to the correct websites accurately. The error rates are 1% and 6% in terms of false positives and false negatives, respectively.

V. Conclusion

The goal of web phishing is to steal users' personal secrets, such as account names, passwords, and credit card numbers. Although there are a number of methods for detecting phishing behavior and protecting users from attacks, most of them are based on text features or visual similarities of whole web pages. In this paper, we propose a solution that tries to reduce the number of password phishing attacks by redirecting users to a correct web pages. A site signature, including text- and image-based features, are extracted accurately and used to identify the real domain name of the visiting web page. While only common stable features are extracted as site signatures, a single signature can be used for variants of a web site. This reduces the required space to store signatures without killing the accuracy rate for detecting phishing pages. Experiments show that the proposed solution is able to detect 94% phishing attacks and has low error rates.

VI. Acknowledgement

This work was conducted under the "Next Generation Security Technology Deployment and Enablement Project" of Institute for Information Industry which is subsidized by the Ministry of Economy Affairs of the Republic of China. We also thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] Alexa. Alexa: The web information company. [online] http://www.alexa.com/.

[2] K.-T. Chen, J.-Y. Chen, C.-R. Huang, and C.-S. Chen. Fighting phishing with discriminative keypoint features of webpages. *IEEE Internet Computing*, pages 30–37, May 2009.

[3] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell. Client-side defense against web-based identity theft. In *NDSS '04: Proceedings of the 11th Annual Network and Distributed System Security Symposium*, 2004.

[4] L. Cranor, S. Egelman, J. Hong, and Y. Zhang. Phinding phish: An evaluation of anti-phishing toolbars. In *NDSS '07: Proceedings of the 14th Annual Network and Distributed System Security Symposium*, 2007.

[5] D. Danchev. DIY phishing kits introducing new features. ZDNet, May 2008. [online] http://blogs.zdnet.com/security/?p=1104.

[6] R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590, New York, NY, USA, 2006. ACM.

[7] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 649–656, New York, NY, USA, 2007. ACM.

[8] A. Y. Fu, W. Liu, and X. Deng. Detecting phishing web pages with visual similarity assessment based on earth mover's distance. *IEEE Transactions on Dependable and Secure Computing*, 3(4):301–311, 2006.

[9] Google, Inc. Google safe browsing for Firefox. [online] http://www.google.com/tools/firefox/safebrowsing/.

[10] C.-R. Huang, C.-S. Chen, and P.-C. Chung. Contrast context histogram - an efficient discriminating local descriptor for object recognition and image matching. *Pattern Recognition*, 41(10):3071–3077, October 2008.

[11] C. Karlof, U. Shankar, J. D. Tygar, and D. Wagner. Dynamic pharming attacks and locked same-origin policies for web browsers. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 58–71, New York, NY, USA, 2007. ACM.

[12] V. I. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1(1):8–17, 1965.

[13] S. McDonnell. Bank of Ireland - Statement. Bank of Ireland, September 2006. [online] http://www.bankofireland.com/press_room/latest_releases/2006/press_releases_news_154758_11.html.

[14] R. McMillan. 'Rock Phish' blamed for surge in phishing. InfoWorld, December 2006. [online] http://www.infoworld.com/article/06/12/12/HNrockphish_1.html.

[15] Microsoft Corporation. PhishingFilter: Help protect yourself from online scams. [online] http://www.microsoft.com/protect/products/yourself/phishingfilter.mspx.

[16] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.

[17] Mozilla Project. Firefox phishing and malware protection. [online] http://www.mozilla.com/en-US/firefox/phishing-protection/.

[18] NetCraft, Ltd. Netcraft anti-phishing toolbar. [online] http://toolbar.netcraft.com/.

[19] C. O'Brien. Bank of Ireland to refund phishing victims. ZD-Net, September 2006. [online] http://news.zdnet.co.uk/security/0,1000000189,39283133,00.htm.

[20] OpenDNS. PhishTank: Join the fight against phishing. [online] http://www.phishtank.com/.

[21] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[22] stopbadware.org. Badware website clearinghouse. [online] http://stopbadware.org/home/clearinghouse.

[23] M. Wu, R. C. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610, New York, NY, USA, 2006. ACM.

[24] Y. Zhang, J. I. Hong, and L. F. Cranor. Cantina: a content-based approach to detecting phishing web sites. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 639–648, New York, NY, USA, 2007. ACM.