Phishing Detection based on Web Page Similarity

[1]Radha Damodaram, [2] Dr. M.L. Valarmathi
[1]Dept. of BCA, SS & IT, CMS College of Science & Commerce, Coimbatore.
[2]Dept. of Computer Science & Engg Government College of Technology, Coimbatore.

Abstract
Phishing is a current social engineering attack that results in online identity theft. Phishing Web pages generally use similar page layouts, styles (font families, sizes, and so on), key regions, and blocks to mimic genuine pages in an effort to convince Internet users to divulge personal information, such as bank account numbers and passwords. A novel technique to visually compare an assumed phishing page with the legitimate one is presented. Five important features such as signature extraction, text pieces and their style, images, URL keywords and the overall appearance of the page as rendered by the browser are identified and considered. An experimental evaluation using a dataset collected of 150 real world phishing pages, along with their equivalent legitimate targets has been performed. The investigational results are satisfactory in terms of false positives and false negatives and an efficiency rate of about 98.11% for false positive pages and 92.95% for false negative pages has been obtained.

Keywords
Phishing, DOM Antiphish

I. Introduction

The underlying assumption of this system is that a phishing page aims to mimic the appearance of the targeted, legitimate page. Thus, when two pages are similar, and the user is about to enter information associated with the first page on the suspicious, second page, an alert should be raised. When the two pages are different, it is unlikely that the second page tried to spoof the legitimate site, and thus, the information can be transmitted without a warning. Whenever a suspected phishing email is found, the potential phishing URL is extracted from the email. Then, the corresponding legitimate page is obtained, using a search engine or, based on keywords, selecting among a predefined set of registered pages. Finally, a comparison is initiated and, if the outcome is positive, the email is blocked. To compare a target page (i.e., suspected page) with a legitimate page, four steps [1] are required:

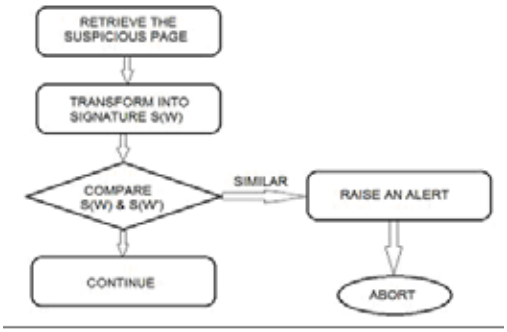

Fig. 1 : Block diagram of the process

1. Retrieve the suspicious web page w.
2. Transform the web page into a signature S (w).
3. Compare S (w) with the stored signature S (w´) of the supposed legitimate page w´ (i.e., the page targeted by the phishing page).
4. If the signatures are "too" similar, raise an alert.

Steps 2 and 3 represent the main aspect of this paper. We discuss these steps in detail in the next two subsections. The actual implementation of Step 4 depends on the specific application scenario in which the approach is used. For example, in Antiphish, raising an alert implies that the submission of sensitive data is canceled and a warning is displayed to the user.

II. Existing Approach

A. Antiphish

AntiPhish [5] is a browser plug-in that keeps track of sensitive information. Whenever a user attempts to enter sensitive information on one site, and this information has previously been associated with a different, trusted site, a warning is generated. This is effective when a user inadvertently enters bank login information on a phishing site. However, AntiPhish suffers from the problem that legitimate reuse of credentials is also flagged as suspicious.

B. DOM Antifphish

To address the usability problem of Antiphish, DOM AntiPhish [10] was proposed. For that approach, the authors compared the Document Object Models (DOMs) of the pages under analysis to determine whether the two pages are similar. When information is reused on a page that is similar to the original page (that is associated with the sensitive data), a phishing attempt is suspected. When the information is entered on a site that is completely different, the system assumes legitimate data reuse. Although DOM AntiPhish is able to identify phishing pages effectively, its major limitation is that the DOM tree is not necessarily a reliable feature to establish similarity between pages. In some cases, it is possible for the attacker to use different DOM elements to create a similar look-and-feel and appearance of a page. Furthermore, a phishing site that only consists of images cannot be detected.

III. Implemented System

A typical phishing attack may be based on several techniques, including exploiting browser vulnerabilities or performing man-in-the middle attacks using a proxy. However, the most straightforward and widespread method consists of deploying a web page that looks and behaves like the one the user is familiar with. In this paper, an effective approach to detect phishing attempts by comparing the visual similarity between a suspected phishing page and the legitimate site that is spoofed is presented. When the two pages are "too" similar, a phishing warning is raised. In this system, three features to determine page similarity: text pieces (including their style-related features), images embedded in the page, and the overall visual appearance of the page as seen by the user (after the browser has rendered it) are considered. The similarity between the target and the legitimate page is quantified by comparing these features, computing a single similarity score. A comparison based on page features that are visually perceived is performed. This is because phishing pages mimic the look-and-feel of a legitimate site and aim to convince the victims that the

site they are visiting is the one they are familiar with. Once trust is established based on visual similarity, there is a higher chance that the victim will provide her confidential information. Typically, a victim's visual attention focuses both on the global appearance of the page and on salient details such as logos, buttons, and labels. In fact, these observations are supported by both common sense and literature. For example, Dhamija et al. [5] show that about one in four users base their trust only on page content (i.e., images, page design, colors) to decide whether the page is 12 legitimate or not. Obviously, these users are the ones that are most prone to fall victim to a phishing attack. This paper implements the phishing detection technique that is based on the similarity between the legitimate page and the phishing page based on its characteristics such as text components, image elements and overall appearance [13].

A. Description of the implemented System

One possible application scenario for the implemented system is to integrate the visual similarity detection scheme into the open source tool AntiPhish. AntiPhish tracks the sensitive information of a user and generates warnings whenever the user attempts provide this information on a web site that is considered to be un-trusted. It works in a fashion similar to a form-filler application. However, it not only remembers what information (i.e., a username, password pair) a user enters on a page, but it also stores where this information is sent to whenever a tracked piece of information is sent to a site that is not in the list of permitted web sites, AntiPhish intercepts the operation and raises an alert. Although simple, the approach is effective in preventing phishing attacks. Unfortunately, when a user decides to reuse the same username, password pair for accessing different online services, too many undesired warnings (i.e., false positives) are raised. By integrating the comparison technique into the existing AntiPhish solution, AntiPhish can be prevented from raising warnings for sites that are visually different. The underlying assumption is that a phishing page aims to mimic the appearance of the targeted, legitimate page. Thus, when two pages are similar, and the user is about to enter information associated with the first page on the suspicious, second page, an alert should be raised. When the two pages are different, it is unlikely that the second page tried to spoof the legitimate site, and thus, the information can be transmitted without a warning. 14 Of course, this technique can also be used in other application scenarios, as long as a baseline for the suspicious page is available. That is, it needs to know what the legitimate page looks like so that comparison can be made against it. For example, the approach could be part of a security solution that works at the mail server level. Whenever a suspected phishing email is found, the potential phishing URL is extracted from the email. Then, the corresponding legitimate page is obtained, using a search engine or, based on keywords, selecting among a predefined set of registered pages. Finally, a comparison is initiated and, if the outcome is positive, the email is b locked. To compare a target page (i.e., suspected page) with a legitimate page, four steps are required:

1. Retrieve the suspicious web page w.
2. Transform the web page into a signature S (w).
3. Compare S (w) with the stored signature S (w´) of the supposed legitimate page w´ (i.e., the page targeted by the phishing page).
4. If the signatures are "too" similar, raise an alert. Steps 2 and 3 represent the core techniques. These steps are discussed in detail in next two subsections. The actual implementation of step 4 depends on the specific application scenario in which the approach is used. For example in Antiphish, rising an alert implies that the submission of sensitive data is canceled and a warning is displayed to the user.

1. Signature Extraction

A signature S (w) of a web page w is a quantitative way of capturing the information about the text and images that compose this web page. More precisely, it is a set of features that describe various aspects of a page. These features cover,

(i) Each text section with its attributes,
(ii) Each visible image,
(iii) The overall visual look-and feel (i.e., the larger composed image) of the web page visible in the viewport1. The following paragraphs describe in more detail the features that this system extracts from a web page.

2. Text Elements

A text element is a visible piece of text on the web page that corresponds to a
leaf text node in the HTML DOM tree. For each piece of text, extract:

i. Its textual content,
ii. Its foreground color,
iii. Its background color,
iv. Its font size,
v. The name of the corresponding font family, and
vi. Its position in the page (measured in pixel starting from the upper left corner).

Thus, for each text element, obtain a 6-tuple t that is called text tuple. The signature of the page contains a vector t0. . . tn of k tuples, where each of the k tuples represents one visible piece of text on the web page. Using JavaScript the Background color can be extracted using, this.

```
getActualBackgroundColor = function(node)
{var bgcolor; if (node.nodeName==”#document”)
{bgcolor = node.bgColor;}
else{if(node.ownerDocument.defaultView.get
ComputedStyle(node, null))
bgcolor=node.ownerDocument.defaultView.getComputedStyle
(node, null).getPropertyValue
(“background-color”);
if (bgcolor == “transparent”) {bgcolor = this.get Actual
BackgroundColor(node.parentNode);}
else {bgcolor = this.getActualBackgroundColor(node.
parentNode);}
{return bgcolor;}}
```

Fig. 2 : Extracting Code

3. Image Elements

For each visible image of the web page, the following attributes are extracted:

i. The value of the corresponding src attribute (i.e., the source address of the image),
ii. Its area as the product of width and height, in pixel,
iii. Its color histograms,
iv. Its 2D wavelet transformation, and
v. Its position in the page.

Thus, for each image element on the page, 5-tuple I, image tuple is obtained. The signature S (w) contains a vector i0. . . im of

tuples, where each tuple represents one visible image on the web page. To extract a color histogram from an image proceeds as follows: First, obtaining a square image of l × l pixels, the image is resized. For the image elements, set l = 128.

In case the image width w and height h are both lower than l, adjust the size to obtain a square image of 2k×2k, where k is the greatest value for which 2k ≤ w and 2k ≤ h. The resize operation leverages a built-in function in Firefox that is able to adjust images to fit into rectangles of arbitrary dimensions. It is done for performance reasons, so that the following computations can be executed faster. Using the scaled image, the second step is to compute the histogram of the RGB components, using n histogram cells. For the image elements, n = 5. This is done for all three colors in the RGB color-space. For example, consider the red component and 5 histogram cells. Count the number of pixels whose red component value is in the range [0, 50] (recall that each color channel has 8 bits) to obtain the first red histogram bar, the number of pixels whose red component value is in the range [51, 101] for the second bar, and so on; finally, normalizing the bar values such that their sum is equal to 1.The 2D wavelet transformation [12] is an efficient and popular image analysis technique that, essentially, provides low-resolution information about the original image. It can be calculated in O (n) time, where n is the size of a square, grayscale image. The wavelet transformation operates on the scaled, square image, which is obtained as described previously. In the first step, this image is converted into grayscale version. Then, compute the 2D wavelet transformation and take the first m×m wavelet coefficients at the low-resolution end of the matrix. For image elements, a value of m = 8 is used.

4. URL keywords

Detecting phishing page is to be redirected to the correct one instead of providing hints to users. Since users are not always aware of alerts that displayed by anti- phishing toolbars. It would be better if URL redirections are enforced when the accuracy of detection rates is good enough and the error rate is limited

5. Overall appearance

Finally, the overall image corresponding to the viewport of the web page as rendered by the user agent (i.e., the upper left portion of the rendered web page that fits a browser window maximized on a typical display is considered. In this case, a screen resolution of 1280 × 800 pixels) is used. For this image, color histograms and its 2D wavelet transformation is extracted. For the overall appearance, a single pair o that is called overall image tuple is obtained, which represents the overall visual image of the web page. The color histogram and the wavelet transformation computation are performed in the same way as for the individual image elements. To capture the overall appearance image with higher precision, the features on a larger image are computed. That is, l = 256 is used, meaning that the size of the scaled-down image is 256 × 256 pixels. Also, the number of cells for the color histogram is increased to n = 8, and m = 16 wavelet coefficients is selected.

IV. COMPARISION

A. Page signature

Once the features that capture the overall appearance of a page and each of its text and image elements are extracted, this page's signature S(w) is stored. The signature is simply the set of all text tuples, image tuples, and the overall image tuple:

S(w) = t0, . . . , tn, i0, . . . , im, oi.

B. Signature Comparision

Once two signatures S(w) and S(w′) are available, the similarity score between the corresponding web pages w and w′ is computed. To this end, start by comparing pairs of elements from each page. Of course, elements are only compared with matching types (e.g., text elements are only compared with other text elements).

That is, all pairs of text elements are compared to obtain a similarity score st. Then, all image pairs are compared to obtain a similarity score si. Finally, the overall appearances of the two pages are used to derive a similarity score so. Using these three scores, a single similarity score s [0, 1] is derived that captures the similarity between the pages w and w′.

1. Text elements

Concerning the text elements, the comparison is done as follows. For each pair of text tuples ti of S(w) and tj′ of S(w′), the following computation is performed:

- The similarity between the two textual contents T and T′ is given as: 1 − dl (T, T′) (3.1)
  Max (length(T)), length(T′)
  where dl (T, T′) is the Levenshtein distance [10] between T and T'
- The similarity between the two foreground colors C and C′ is given as:
  1 − 1 (3.2) (3·255) L1(C, C′)
  where L1 is the 1-norm distance (also known as Taxicab metric or Manhattan distance
  [9]) between the colors expressed as 8-bit RGB points
  (i.e., L1(C, C′) = |r − r′| + |g − g′| + |b −b′|) (3.3)
- The similarity between the two background colors are computed, in the same way as above
- The similarity between the two font sizes F and F′, is expressed in pixel, as
  1 − |F− F′| (3.4) max(F, F′)
- The similarity between the names of the two font families are computed, setting 0 if they are equal, 1 otherwise
- The similarity between the two positions in the pages is given as 1 − d (3.5)

where d is the Euclidean distance between the two points and Md is the maximum Euclidean distance between two points in a viewport of 1280 × 600 resolution. Note that all the obtained similarities are in the range [0, 1], where 0 means no similarity and 1 means total match. The sum the 6 individual similarities scores is found using the weights (4/15), (4/15), (2/15), (2/15), (2/15), (1/15) (Whose sum is equal to 1), and obtain the similarity st i,j between ti and tj′ The weights were manually chosen based on the domain knowledge and the assessment of the importance of the corresponding features to the visual similarity between two text blocks. These weights only serve as a rough estimate for the different impact of features. Clearly, the content of the text and the text color are more important than the used font family. This intuition is reflected by the weights. The actual values have not been optimized, and it might be possible to further improve this system by tuning the weights. Once similarity score between all pairs of text elements are obtained, it is stored in a similarity matrix St. This matrix stores, for each pair of elements ti and tj′, the similarity between these two elements. The dimension of the matrix is n × m, where n is the number of text elements on page w and m is the number of text elements on w′.

```
<html> <h1 style=”color:rgb(255,0,0);”>Home
banking</
h1>
<p>Welcome!</p>
<p>Copyright 2007</p>
</html>
and:
<html>
<center>
<h1 style=”color:red;”>Your banking</h1>
<p style=”color:gray;”>Welcome!</p>
</center> </html>
```

Fig. 3 : HTML Coding for background extraction

As an example, suppose the textual parts of two pages whose corresponding HTML documents compared are shown in fig. 3.

The corresponding text tuples are, respectively:
t1 = {Home banking, (255, 0, 0), (255, 255, 255), 32, Serif, (8, 8)}
t2 = {Welcome!, (0, 0, 0), (255, 255, 255), 16, Serif, (8, 66)}
t3 = {Copyright 2007, (0, 0, 0), (255, 255, 255), 16, Serif, (8, 102)}
and:
t1′ = {Your banking, (255, 0, 0), (255, 255, 255), 32, Serif, (8, 21)}
t2′ = {Welcome!, (128, 128, 128), (255, 255, 255), 16, Serif, (8, 80)}

The system first compares each pair of text elements, computing a similarity score. For the tuples t1 and t1′, this yields:
St 1, 1 = (4/15)0.75+ (4/15)1+ (2/15)1+ (2/15)1+ (2/15)1+ (1/15)0.98375= 0.93225 When the computation is performed for each pair, the similarity matrix St can be determined. In this example, this is the following 3 × 2 matrix:

$$St = \begin{matrix} 0.9322500 & 0.5493813 \\ 0.5740278 & 0.8649771 \\ 0.6062897 & 0.5948105 \end{matrix} \quad (3.6)$$

2. Image elements

Concerning the image part, the comparison is done as follows. For each image tuple ii of S(w) and for each image tuple ij′ of S(w′): • The similarity between the two src attributes using the Levenshtein distance is computed, as above; 23 • The similarity between the two image areas A and A′, expressed in pixels, is given as

$$1 - |A - A'| \quad (3.7) \quad \max(A, A')$$

- The similarity between the two matrices representing the color histograms C and C′ as 1 − L1(C, C′) is computed, using the 1-norm distance
- The similarity between the two matrices representing the 2D wavelet transformations is computed, using the 1-norm distance as above
- The similarity between the two positions in the pages is computed, as described previously for text tuples. Again, all similarities are in the range [0, 1]. Then sum these similarity values, using the weights (4/11), (2/11), (2/11), (2/11), (1/11). The result is the distance si i,j between ii and ij′ . Based on these distance values, a similarity matrix Si that captures the similarity between the image elements of S(w) and S(w′) is derived. Again, the weights are set according to the assessment of the visual impact of each feature.

3. 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 [14].

4. Overall appearance

Finally, concerning the appearance of the overall image, similarity in terms of color histograms and of 2D wavelet transformations is computed, in the same way as described previously for images. The similarity index so as the average of the two values is obtained.

5. Individual similarity scores

To obtain a similarity score s from a similarity matrix S (st for the text matrix St and si for the image matrix Si), average the largest n elements of the similarity matrix. The largest n elements of the matrix are selected using the following iterative, greedy 24 algorithm:

(i) select the largest element of the matrix
(ii) discard the column and the row of the selected element. These steps are repeated until a number n of elements are selected or the remaining matrix is composed of either no rows or no columns. Then, n = 10 is set for the text similarity matrix and n = 5 for the image similarity matrix. In other words, the n most matching items (either among text blocks or among images) are extracted between the two web pages under comparison, avoiding considering an item more than once.

Consider the average of the greatest n values in the matrix instead of considering the whole matrix in order to avoid the case in which the comparison outcome is influenced mainly by many non-similar elements rather than by few, very similar elements (which are typically the ones that can visually lure the user). For example, consider a phishing page in which there are very few images (e.g., the logo and a couple of buttons) that are very similar to the ones in the legitimate page. Also, imagine that there are a large number of graphical elements, possibly small and actually rendered outside of the viewport, which are not present in the original page; if the average over the entire matrix elements are taken, the outcome would be biased by the low similarity among the many dissimilar elements. However, the user would be tricked by the few elements that are very similar.

6. Final similarity score

The final outcome of a comparison between two signatures is the similarity scores. This score is obtained based on the individual scores for text and image elements, as well as the overall

appearance: s = at st + ai si + ao so. When s is large, the two pages are similar. A threshold t is used in order to discriminate between the two cases: w and w′ are considered similar if and only if s ≥ t, not similar otherwise. In this paper text, image and overall appearance are considered for comparison.

V. Implementation and Result Analysis

A. Experimental Evaluation

A dataset that consists of negative and positive pairs of web pages. For the positive pairs, pairs of real-world legitimate pages and corresponding phishing pages are selected. The phishing pages are obtained from the PhishTank public archive (http://www.phishtank.com). For each phishing page, the corresponding legitimate page is retrieved by visiting the web site of the spoofed organization immediately after the attack appeared on Phish Tank. To build the negative part of the dataset, a number of common web pages are collected, unrelated to the legitimate ones. Then the set of positive pairs are partitioned into three subsets, based on their visual similarity. That is, three levels of dissimilarity are defined as perceived by a human viewer who manually looks and compares a legitimate web page and the corresponding phishing page. Each subset is denoted with a dissimilarity level label: Level 0 identifies pairs with a perfect or almost perfect visual match. Level 1 identifies pairs with some different element or with some minor difference in the layout. Level 2 identifies pairs with noticeable differences.

The partition positive pairs are chosen into different subsets for the following reason: The majority of phishing pages exactly mimic the appearance of the legitimate page. This is not surprising, as the miscreants do not wish to raise suspicion. However, there are also cases where visual differences do exist. These differences may be simply due to the poor skills of the attacker (e.g., mistakes in a text translated to a foreign language). However, some differences may be voluntarily inserted, both at the source level and at the rendering level. This could be done to evade anti-phishing systems, while, at the same time, keeping the look-and-feel as close to the original web page as possible. Similar 26 evasion techniques are sometimes used by spammers for image-based spam [2].

That is, although some randomized alterations are applied to the original image, from the user's point of view, the image remains identical. The negative pairs set are partitioned into two subsets. One subset consists of web pages with a login form. The second one has no such forms. The pages in the first subset mainly selected from Internet banking web sites. The second subset was chosen by performing a manual selection of pages, aimed at obtaining a heterogeneous and random sample set varying in size, layout, and content.

A substantial portion of pages are chosen with a login form to make the experiments more realistic and challenging. This is because pages that maintain a login form are more likely to be compared against legitimate pages when trying to detect phishing pages. The results obtained are illustrated in the Fig.s shown below.

Fig. 4: Text Comparison

Fig. 5 : Image Comparison

Fig. 6 : Overall Appearance

Fig. 7 : Output page for phishing website

Fig. 8 : Output page for legitimate website

B. Result analysis

For this experiment, the test set was composed of 180 positive pairs (85 of Level 0, 47 of Level 1, and 48 of Level 2).

Table 1 : Result analysis

Levels	False Positives	Fake Positive Rate	False Negatives	False Negative Rate
All (0,1 and 2) 48	2	1.1	1	2.08%
Only 0 and 1 47	1	0.1	1	2.08%
Only 0 85	1	0.1	1	2.08%

Table 1 summarizes the results of the tests. It can be seen that this approach detects all phishing pages classified as Level 0 and 1, while it fails to detect 2 out of 180 positive pairs of Level 2. Hence, an overall false positive rate (FPR) equal to 1.11% and an overall false negative rate (FNR) equal to 2.08%. Here the verification is done, by visual inspection that those pairs were indeed difficult to detect by this visual similarity-based approach.

In general, the choice of t depends on the desired tradeoff between possible false positives and possible false negatives. Hence, it depends on the context in which the proposed approach is supposed to be deployed. For example, if the comparison approach is implemented as part of the AntiPhish tool, it may be preferable to select lower values for t. The reason is that when using the visual comparison component with AntiPhish, a large number of possible false positives is already filtered out, since the comparison is invoked only when a user's known credentials are about to be transmitted to an untrusted web site. Therefore, the comparison may be relaxed towards accepting more false positives (warnings) in favor of avoiding missed detections.

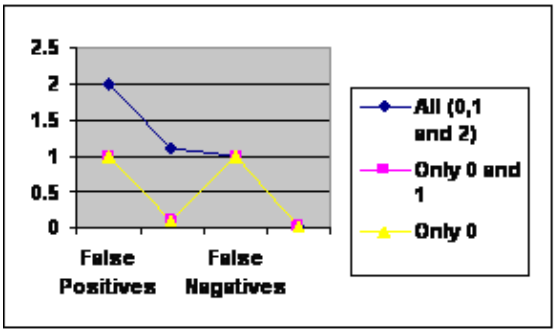

Fig. 9 : Comparison of three levels

Finally, the average computation time for comparing two pages is measured as shown in the Fig. 4.6. Note that such an operation involves both the signature extraction and the signature comparison phase. In this experimental analysis comparison phase is focused. The extraction phase consists mainly of retrieving information which, in practice, is already available to the browser that has rendered the page. Moreover, the legitimate page signature is typically extracted once at a previous point in time.

In this experiment, it took about 3.8 seconds for positive pairs and about 11.2 seconds for negative pairs to be compared. The key idea to improve performance is to execute the comparison operations in an order such that the most expensive operations are executed last.

In particular, during the similarity index computation (either st, for the text section, or si, for the images), the n (with n = 10 for the text part and n = 5 for the image part) similarity values are kept for the most similar pairs of tuples found so far. When evaluating a new pair of tuples, the evaluation can be stopped once it determine that this pair cannot exceed the similarity values of one of the top n pairs, even if all remaining features comparisons yield perfect similarity. For example, suppose that: (i) a pair of images is considered, (ii) the distance is computed in terms of positions on the page and image sizes, and (iii) the distances are such that with the given weights, the corresponding matrix element will be lower than the 5th retest element of the matrix. In this case, no need to compute, or extract, the two Haar transformations or the Levenshtein distances. Once it is determine that, after looking at the score for the overall appearance, two pages cannot exceed the similarity threshold t, then no need to compute any of the two similarity matrices for the text and image elements. This results in impressive speeds-ups. A negative comparison between two pages is produced in a few milliseconds.

VI. Conclusion

Phishing has becoming a serious network security problem, causing financial loss of billions of dollars to both consumers and e-commerce companies. Phishing has made e-commerce distrusted and less attractive to normal consumers. The characteristics of the web similarity that are exploited in phishing pages are studied. In this paper, an effective and novel approach to detect phishing attempts by comparing the visual similarity between a suspicious page and legitimate target page has been presented.

When checking for visual similarity, five page features: signature extraction, text pieces and their style, images embedded in the page, URL keywords and the overall visual appearance of the web page as rendered by the browser are considered. Features that are visually perceived by users are considered because, as reported in literature, victims are typically convinced that they are visiting a legitimate page by judging the look-and-feel of a web site.

A dataset consisting of 150 real phishing pages with their corresponding target legitimate pages are considered. The results, in terms of false alarms and missed detection, are satisfactory. 1% false positives were raised and only 2.08% false negatives were calculated.

References

[1] Eric Medvet, Engin Kirda, Christopher Kruegel, "Visual similarity-based phishing detection", Computational intellligence in cyber security 2009. CICS '09. IEEE

Symposium pages: 30-36
[2] Masanori Hara, Akira Yamada, Yutaka Miyake D. "Visual similarity-based phishing detection without victim site information", IEEE International Conference on Security and Privacy in Communication Networks (Secure Comm), 2008.
[3] APWG. "Phishing Activity Trends - Report for the Month of December, 2007". Technical report, Anti Phishing Working Group, Jan. 2008. [Online] Available : http://www.antiphishing.org/reports/ apwg_report_dec_2007.pdf.
[4] H. Aradhye, G. Myers, J. Herson. "Image analysis for efficient categorization of image-based spam e-mail". Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on, 2:914–918, 29 Aug.-1 Sept. 2005.
[5] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, J. Mitchell. "Client-side defense against web-based identity theft". In 11th Annual Network and Distributed System Security Symposium (NDSS '04), San Diego, 2005.
[6] R. Dhamija, J. D. Tygar. "The battle against phishing: Dynamic security skins". In Proceedings of the 2005 symposium on Usable privacy and security, New York, NY, pages 77–88. ACM Press, 2005.
[7] P. Mutton. "Italian Bank's XSS Opportunity Seized by Fraudsters". Technical report, Netcraft, Jan. 2008. [Online] Available :http://news.netcraft.com/archives/2008/01/08/italian_banks_xss_opportunity_seized_by_fraudsters.html. 33
[8] E. F. Krause. "Taxicab geometry". 1987.
[9] V. I. Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". Soviet Physics Doklady, 10:707–710, 1966.
[10] A. Rosiello, E. Kirda, C. Kruegel, F. Ferrandi. "A Layout-Similarity-Based Approach for Detecting Phishing Pages". In IEEE International Conference on Security and Privacy in Communication Networks (SecureComm), 2007.
[11] L.Wenyin, G. Huang, L. Xiaoyue, Z.Min, X. Deng. "Detection of phishing webpages based on visual similarity". In 14th International Conference on World Wide Web (WWW): Special Interest Tracks and Posters, 2005.
[12] R. Stankovic, B. Falkowski. "The Haar wavelet transform: its status and achievements". Computers and Electrical Engineering, 29:25–44, 2003.
[13] Microsoft. "Sender ID Home Page". [Online] Available : http://www.microsoft.com/mscorp/safety/technologies/senderid/default.ms%px, 2007.
[14] Chun-Ying Huang, Shang-Pin Ma, Wei-Lin Yeh, Chia-Yi Lin1, Chien-Tsung Li, "Mitigate Web Phishing Using Site Signatures". Department of Computer Science and Engineering, National Taiwan Ocean University.