Web Phishing Detection Based on Page Spatial Layout Similarity

Weifeng Zhang
School of Computer, Nanjing University of Posts and Telecommunications, China
E-mail: zhangwf@njupt.edu.cn

Hua Lu
Department of Computer Science, Aalborg University, Denmark
E-mail: luhua@cs.aau.dk

Baowen Xu
Department of Computer, Nanjing University, China
E-mail: bwxu@nju.edu.cn

Hongji Yang
Software Technology Research Laboratory, De Montfort University, England
E-mail: hyang@dmu.ac.uk

Keywords: web phishing, page spatial layout similarity

Received: July 8, 2012

Web phishing is becoming an increasingly severe security threat in the web domain. Effective and efficient phishing detection is very important for protecting web users from loss of sensitive private information and even personal properties. One of the keys of phishing detection is to efficiently search the legitimate web page library and to find those page that are the most similar to a suspicious phishing page. Most existing phishing detection methods are focused on text and/or image features and have paid very limited attention to spatial layout characteristics of web pages. In this paper, we propose a novel phishing detection method that makes use of the informative spatial layout characteristics of web pages. In particular, we develop two different options to extract the spatial layout features as rectangle blocks from a given web page. Given two web pages, with their respective spatial layout features, we propose a page similarity definition that takes into account their spatial layout characteristics. Furthermore, we build an R-tree to index all the spatial layout features of a legitimate page library. As a result, phishing detection based on the spatial layout feature similarity is facilitated by relevant spatial queries via the R-tree. A series of simulation experiments are conducted to evaluate our proposals. The results demonstrate that the proposed novel phishing detection method is effective and efficient.

Povzetek: Opisana je detekcija spletnega ribarjenja na osnovi podobnosti strani.

1 Introduction

Along with the wide use of the Internet, a rapidly growing number of people are using various online services such as e-banking, online shopping, etc. These services give users great convenience. Meanwhile, the number of phishing web sites also increases very quickly. According to the statistics of PhishTank, over 1.3 million phishing sites are verified and registered in its database merely in the first two years after its launch. There are about 5919 online phishing sites, an d the number of offline phishing sites are close to 1.3 millon. And everyday a large number of new phishing sites were found, in average 600 sites are submitted and verified daily. Phishing sites cheat users by simulating the interfaces of genuine web sites, and defraud users of their sensitive information like user ID, password, and credit card number. Illegal uses of such stolen information can cause significant loss to users. Plenty of phishing web sites are found every day and phishing fraud is an increasing crime on the Internet. Therefore, it is desirable that phishing sites are detected effectively and users can get alerts to avoid being trapped.

Phishing sites are often sent out in random spam emails. Such emails often target those users who have no experience on network security, and make them believe that the emails come from legitimate organizations. Typically, these emails fake some reasons to require users to update their account information. When a user tries to log in through the interface provided in the email, sensitive information like user name and password will be stolen by the phishing site. Although many users nowadays have more or less experience on security and many email gateways can filter out most spams, a considerable number of users still become victims of phishing sites. There exist various anti-phishing approaches. A detailed review can be found in Section 2. Blacklist based approaches can precisely filter

out any phishing web page included in the blacklist. However, it is very hard to maintain up-to-date blacklists because phishing pages often have short existence and new phishing pages come out at a good pace [1, 2]. Web feature based approaches analyze the feature differences between genuine web page and phishing web page, extract critical features, and construct classifiers to classify subsequent web pages in phishing detection. Such approaches work fast and are able to detect phishing pages that have not been identified before. The difficulty of such approaches lies in determining classification features as phishing pages are always created to be alike to their corresponding genuine pages, which lowers detection accuracy. The third-party tools based phishing detection approach use the third-party tools like search engines to detect phishing pages [3, 4]. We examined this method by experiments, and found lots of phishing pages can be found in the search results, which may be due to SEO(search engine optimization) methods used by creators of phishing sites. If the phishing pages can be searched in search engines, this approach fails basically.

Recently, whitelist approach is often used in phishing page detection. It constructs a feature library from those web pages that are likely to be imitated by phishing pages, and then identifies phishing pages by computing the similarities between a web page and the feature library. As the whitelist approach is based on similarity search rather than exact matching, its detection speed is greatly affected by the feature library size as well as the search strategies. Some filtering methods, e.g., measuring file size, counting image number, and hierarchical clustering, have been used to reduce the number of signature pages to be compared. These methods can also rule out relevant web pages and thus results in errors in phishing detection. Furthermore, hierarchical clustering based filtering usually produces poor clustering results because of the differences among individual web pages, which inevitably impairs the filtering accuracy of feature library.

In order to solve the above problems, we in this paper propose a novel phishing detection approach that exploits the relevant spatial layouts of web pages. Song etal use a vision-based page segmentation algorithm to partition a web page into semantic blocks with a hierarchical structure. Then spatial features (such as position and size) and content features (such as the number of images and links) are extracted to construct a feature vector for each block [5]. A phishing page and its corresponding genuine page are close to each other visually. Consequently, page elements (e.g., text fields, images, buttons, etc.) in the phishing page are probably placed at the same or similar positions and of the same or similar size to their counterparts in the genuine one. Such spatial layout similarities can be used to determine whether a suspect page is close enough to a genuine page to be a phishing page.

Our approach makes use of spatial index on web page spatial layouts to quickly filter out unqualified candidates from the feature library. Also, it takes into account the important spatial layout features in defining and computing the similarity between web pages. On one hand, we improve the filtering performance (filtering speed and filtering accuracy) on feature library by including spatial layout features in the library.On the other hand, we combine spatial layout features with existing popular features to improve the accuracy of phishing detection. By an R-tree indexing pages in the feature library based on spatial features, we are able to fast obtain candidate pages in the library that are visually close to a suspect page. The R-tree can also filter out a considerable number of pages in the library without computing the concrete similarities.

A crucial part in our approach lies in capturing the layout characteristics of a web page to protect, i.e., a page likely to be imitated by a phishing page. After displaying a web page by the rendering engine (also known as layout engine) in a web browser, we develop two segmentation methods to extract its layout features. One method works by analyzing the page's DOM tree. For each tree node, it produces the placement, width, and height of the corresponding page block. Nested blocks are allowed in this method. Based on image segmentation, the other method employs image edge detection techniques to divide the entire image of a web page into non- overlapping blocks.

All those blocks obtained in either way are represented in rectangles and constitute the feature library. We then build an R-tree to index the entire feature library to facilitate search in phishing detection. Subsequently, given a suspect phishing page, its layout features are extracted likewise and used to compare against those pages whose layouts have been captured in the feature library. Through the R-tree, most pages in the library are filtered out based on spatial layout dissimilarity. Further, the concrete similarity is computed between the suspect page and each of the few candidate pages passing the filter. The suspect page will be regarded as phishing page if the similarity is greater than a pre-specified threshold.

We make the following contributions in this paper:

- First, we define the spatial layout features for web pages, and develop two feature extraction methods that make use of relevant functionality of modern web browsers.
- Second, we present an effective web page similarity definition that takes into account the proposed page spatial layout features.
- Third, we design an R-tree index for legitimate web page library that is organized based on spatial layout features, and propose library search algorithms for phishing detection.
- Fourth, we conduct an extensive experimental study to evaluate our proposals. The results suggest the proposed approach is effective and efficient.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work on phishing page detection.

Section 3 presents the layout features of web pages and our extraction methods. Section 4 details how to build R-tree to index the web page layout features in the feature library. Section 5 reports on an extensive experimental study. Finally, Section 6 concludes the paper and discusses several directions for future work.

2 Related work

The email based approach filters out phishing links in an email by anti-spam technologies. Fette et al. proposed machine learning based PILFER [6] that extracts 9 features from the links in an email and trains a random tree as a classifier for subsequent pages. Bergholz et al. [7] generates extra email features through a trained Markov chain and a new model of potential class topics. These extra email features, together with basic ones, are used to train a classifier that is able to reduce over two thirds false positives than PILFER. Moreover, Bergholz et al. [8] proposed a detection method that simulates hidden interference information by OCR technology. It however incurs long running time and shows poor classification performance because the used OCR technology is not good at recognizing texts in disturbed images.

The blacklist based method constructs a blacklist of collected phishing site URLs. When a user visits a URL, the URL will be checked up in the blacklist. If the URL is in the blacklist, the access to it will be blocked. Popular web browsers such as Microsoft Internet Explorer and Google Chrome have built-in anti-phishing blacklists [9]. Some other software, e.g., NetCraft [2], SiteAdvisor [10], can filter out phishing sites in the their blacklists by browser toolbars. However, it is a big challenge to maintain the blacklist up-to-date in the presence of the dynamics of phishing sites.

The third-party tools based anti-phishing mainly depends on the ranking of search engines. Such methods make use of the fact that new web sites and sites with few clicks are usually ranked lowest by search engines. Given a web page, the TF-IDF scores of each term in it are calculated, and the top 5 terms are selected to generate a lexical signature of the page. Subsequently, the lexical signature is sent to a search engine (e.g., Google) to evoke a search. The web page is regarded as a phishing page if its domain name is out of the top N in the search result [1, 9]. Orthogonally, Moore et al. [11] proposed to use search engines to find potentially vulnerable hosts. In particular, analyzing search logs discloses the search terms commonly used by attackers, which indicate vulnerable web sites.

There also exists similarity based anti-phishing. Such an approach decides a given web page to be a phishing page if its similarity to a genuine page is greater than a prespecified threshold. So far two kinds of similarities have been used: structural and visual. Liu et al. [12] proposed to detect phishing web pages by computing the structural similarity between corresponding DOM trees. Angelo et al. [13] proposed to compute the similarity by comparing the HTML tags of two web pages. In contrast, visual similarities are computed based on the image features of two web pages [1, 14, 15]. It is noteworthy that our approach in this paper not only extracts new visual features from web pages, i.e., spatial layout, but also combines visual similarity with structural similarity when comparing web pages. To the best of our knowledge, this is the first work that is characterized by such comprehensiveness in phishing detection.

3 Spatial layout features of web page

After downloading a requested web page, a web browser analyzes the html document, extracts the embedded web links, executes the scripts in the html document, and then decides whether to issue other URL requests. When the resources (e.g., texts, pictures, etc.) contained in the page are returned, the web browser will render these resources according to their properties. The user then can see in the browser the web page with both texts and visual resources. As phishing pages are always intended to make people believe they are the genuine pages, they look very similar or even identical to their target genuine pages. As a result, the rendering features of a phishing page and the counterparts in the target genuine page are very similar visually. Accordingly, the basic phishing detection process consists of the following three steps (see Figure 1).

Step 1: Access the web page (denoted as url) and its embedded resources. *Step 2*: Extract the features of the rendered web page. All the extracted features form a signature that is denoted as S (url). *Step 3*: Compute the similarity between S(url) and those signatures in the feature library. If there exists one signature S_i in the library such that similarity between S (url) and S_i is greater than a pre-specified threshold, the web page url is judged as a phishing page, and an alert is issued.

From the above steps it can be seen that two points are very important for good results of phishing detection. It is vital to extract critical and suitable features from rendered web pages. It is also very helpful to improve the accuracy of feature similarity computation. Conventional features of rendered web pages include text features, image features, overall image features, etc. They are covered in detail elsewhere [16]. In this paper, we focus on the spatial layout features of rendered web pages, which have not been considered in the literature, and utilize them in effective phishing detection.

3.1 Extraction of spatial layout features

After a web page is completely downloaded by a browser, it is resolved into a HTML DOM tree and all its embedded elements are rendered in the browser through a rendering engine. When a page is being rendered, it is easy to get the rectangle region that each page element occupies in the browser. Such rectangle regions can be obtained in

Figure 1: Feature library based phishing detection

an alternative way as follows. A rendered web page can be divided into image segments by some edge detection algorithm which can return the rectangle region each image segments occupies in the browser.

Definition 1. *(**Spatial Feature**) A spatial feature of a web page element is a rectangle denoted as* $rect = \langle left, top, width, height \rangle$*, where* $(top, left)$ *represents the top-left corner coordinate of the rectangle in the browser,* $width$ *represents the rectangle's width, and* $height$ *represents the rectangle's height.*

The spatial features of all elements in a web page form the spatial layout feature of that web page. As mentioned above, the spatial layout feature can be obtained by either combining a browser rendering engine with the page DOM tree or applying image detection algorithms after page rendering. Next, we present these two options in Sections 3.1.2 and 3.1.2 respectively.

3.1.1 DOM tree based spatial layout feature extraction

DOM tree based extraction of spatial layout features needs to call the browser layout engine and analyze a DOM tree using some tools. The browser layout engine integrates texts, images, css files, java scripts, and other related resources altogether to render a web page, by referring to the page's DOM tree. Assume the pixel in the upper-left corner of the web browser display region is origin (0, 0), the X-axis is from left to right, and Y-axis is from top to down, both measured in units of pixel. The browser engine positions each DOM tree node according to its four numerical attributes: X coordinate, Y coordinate, width and height. These four attributes form the layout feature of a page element corresponding to a DOM tree node.

The layout feature of a web page element is obtained as follows. We first call the parsing engine of a chosen web browser to parse the web page and get a corresponding DOM tree. After that, we traverse the DOM tree, and get the corresponding display region of each node. If a node's display region area is larger than a pre-specified threshold (we set it 50 in this paper), the layout of this node is generated. Here we ignore DOM tree nodes with small display regions because small regions are insignificant visually and thus unimportant in phishing detection.

Definition 1 defines the general spatial features of web page elements. Applying it to the DOM tree based feature extraction, the spatial layout feature of a web page is a set of relevant rectangles, as defined in the following.

Definition 2. *(**DOM Tree based Spatial Feature**) Given a web page* p*, and its DOM tree* $DT(p)$*, its DOM tree based spatial feature* $SFD(p)$ *is a set of sufficiently large rectangles, each corresponding to a node in the DOM tree. Formally,* $SFD(p) = \{rect_i \mid \text{Area}(rect_i) > T_{area}, \exists node_i \in DP(p) \text{ s.t. } node_i\text{'s extent is } rect_i\}$.

In the definition T_{area} denotes a pre-specified threshold that helps filter out small rectangles.

Figure 2: DOM tree based spatial layout features

Figure 2 shows an example of DOM tree based spatial layout features. Each block represents the position and size of a rendered page element, which corresponds to a DOM tree node. Note that we filter out small rectangles with area smaller than 50 measured in pixels. It can also be seen from the figure that different topology relationships exist among these rectangles. A rectangle recti contains another one rectj if the former's corresponding DOM tree node ni is an ancestor of the latter's node nj. The adjacent relationship between two rectangles also reflects their corresponding nodes are adjacent in the DOM tree. In contrast, the overlap relationship between rectangles results from the relayout of corresponding DOM tree nodes, which is done by the rendering engine according to relevant CSS rules and/or java script codes.

DOM tree based layout feature extraction needs to call web browser APIs to get the layout information. However, web browsers (e.g., the Microsoft Internet Explorer) only allow us to get the width/height values of every block and the Top/Left values relative to its parent block. In order to get the X and Y coordinates of each block, we need to recursively add the Top values and the Left values of the related blocks. Figure 3 gives an example of calculation of the block coordinates.

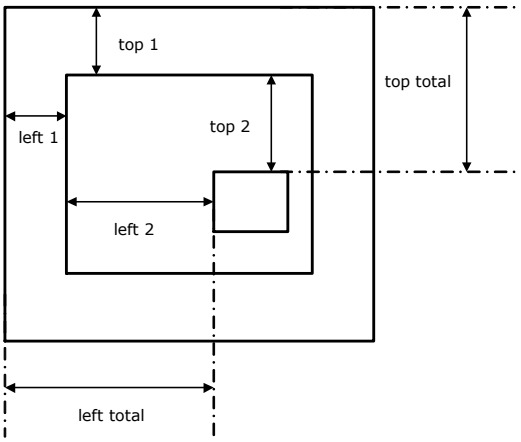

Figure 3: Calculation of block coordinates

In this example, we need to calculate the X and Y coordinates of the innermost block. As the Microsoft Internet Explorer does not provide methods to directly get these two parameters, we calculate them through the parameters of other blocks. First, we get the Top value and the Left value of block A relative to its parent block. Second, we obtain the Top value and Left value of the parent block relative to the upper block, and so on. Finally, if we find that the upper block has the label 'Body', the iteration stops. As a result, the sum of all the Top values is the coordinate Y, and the sum of all the Left values is the coordinate X. In Figure 3, we compute the left total and top total as follows:

left total = left 1 + left 2, and top total = top 1 + top 2.

Here, the left total is the value of the block relative to the left value of the browser window, i.e., the X coordinate of the block's top-left corner. Likewise, top total is the Y coordinate of its top-left corner.

3.1.2 Image segmentation based spatial layout feature extraction

The image segmentation based extraction produces image blocks in a rectangle. Each rectangle is also represented by four numerical attributes: the X coordinate, the Y coordinate, the width, and the height. Such rectangles are extracted as follows. We first parse the web page, render it through the browser's rendering engine, and get the entire page in an image. After that, we divide that image into a collection of smaller image blocks based on the gaps between them, using some image edge detection algorithm. Finally, we calculate the layout feature, i.e., the top-left corner coordinates and size, of each image block thus obtained.

Definition 3. ***(Image Segmentation based Spatial Feature)*** *Given a web page p, its image segmentation based spatial feature $SFI(p)$ is a set of sufficiently large rectangles, each corresponding to an image block in the rendered page. Formally, $SFI(p) = \{rect_i \mid \text{Area}(recti) > T_{area}, rect_i \in Blocks_{IS}(p)$[1]$\}$.*

Rectangles in Figure 4 are obtained from a rendered web page through image segmentation, where each rectangle reflects the position and size of a displayed element. Note less blocks are generated by image segmentation than by DOM tree based extraction. Therefore, we set the area threshold T_{area} to a smaller value 20 in this example. As a result, those blocks whose areas are smaller than 20 are excluded. Compared to the DOM tree based layout features in Figure 2, the image segmentation based blocks in Figure 4 are more visible to human visual discrimination.

Figure 4: Image segmentation based spatial layout features

After the definitions and extractions of web page spatial layout features, we are ready to introduce the concept of corresponding blocks between two web pages.

[1]We use $Blocks_{IS}(p)$ to denote all the blocks that are obtained by an image segmentation method applied to page p.

3.2 Corresponding blocks matching

The spatial similarity between web page blocks can be calculated based on a number of ways, including distance, shape, size, and topological relations. Zhang gives a fuzzy topological surface matching method [17]. Tong put forward a theory of probability matching model [18]. And Masuyama calculate the possibility of matching based on overlap area ratio of two blocks [19].

Our web page similarity definition is based on a concept called corresponding blocks. A pair of corresponding blocks A_i and B_i come from pages A and B respectively, and they are visually close to each other. Intuitively, if each A_i in a suspicious page A has a corresponding block B_i in a genuine page B, A is a phishing page that imitates B. We propose two rigorous definitions for corresponding blocks.

Definition 4. ***(OA based Corresponding Blocks)*** *Given blocks A_i and B_i that are extracted by a same method from web pages A and B respectively, if the size difference (both width and height) between A_i and B_i is less than a threshold T_{size}, and the ratio between the their overlap area and max(Area(A_i), Area(B_i)) is larger than a threshold $T_{overlap}$, A_i and B_i are considered as corresponding blocks.*

The idea behind OA based corresponding blocks is that if the overlap occupies a very high portion of either block, these two blocks are regard as a pair matching in phishing detection. An example is shown in Figure 5. The size difference between the left two blocks does not exceed a pre-specified threshold, and the ratio between their overlap area and the larger block area exceeds a pre-specified threshold. Therefore, the two blocks are determined to be corresponding blocks. In contrast, the two blocks in the right have a small portion as overlap, and they have a large difference in height. As a result, they are not corresponding blocks.

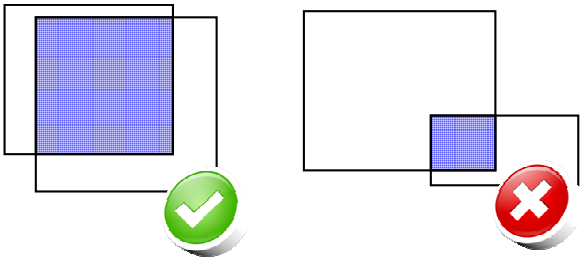

Figure 5: OA based block matching

Definition 5. ***(CDA based Corresponding Blocks)*** *Given blocks A_i and B_i that are extracted by a same method from web pages A and B respectively, if the Euclidean distance between their centers is less than a threshold T_{dist}, and their size difference (both width and height) is less than a threshold T_{size}, A_i and B_i are considered as Center Distance and Area (CDA) based corresponding blocks.*

On the other hand, the CDA method is based on the center distance and area. If the center distance between the query block and a block in the feature library is less than the predetermined threshold and their size difference is sufficiently small, they are matched as corresponding blocks. For example, the left two blocks are corresponding blocks in Figure 6. In contrast, although the center distance of the right two blocks does not exceed the threshold, their height difference is too large, and therefore they are not corresponding blocks.

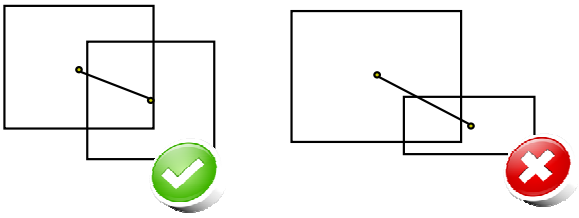

Figure 6: CDA based block matching

Given a query page and a page in the feature library, we can find all pairs of corresponding blocks between them according to either Definition 4 or Definition 5. In Section 4 we will present how to obtain all such corresponding blocks by making use of R-tree on the feature library. After obtaining all corresponding block between two pages, we are ready to compute the similarity between them.

3.3 Computing similarity based on spatial layout features

In this section, we use an example shown in Figure 7 to illustrate the similarity computation. Figure 7 shows the layouts of web pages A and B. Here A has 5 blocks and B has 6 blocks. We normalize both pages into the same coordinate system. With appropriate thresholds, we can find that A-1 and B-1 are corresponding blocks, A-2 and B-2 are corresponding blocks, A-5 and B-3 are corresponding blocks, and so are A-4 and B-4. In total, there are four pairs of corresponding blocks between pages A and B.

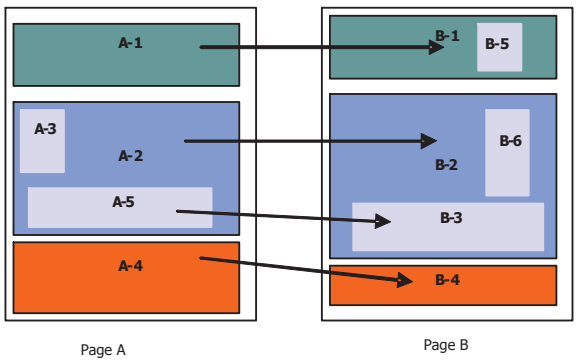

Figure 7: Corresponding blocks between two web pages

With the concept of corresponding blocks, we define the layout similarity between two web pages as follows in For-

mula 1. Here, n_A is the number of blocks in web page A, n_B is the number of blocks in web page B, while n_{cor} is the number of corresponding blocks between A and B.

$$Sim(A,B) = (1 - \frac{|n_A - n_B|}{max(n_A, n_B)}) \cdot \frac{n_{cor}^2}{n_A \cdot n_B} \quad (1)$$

Formula 1 calculates the ratio of the corresponding blocks in either page, normalizes the product of the two ratios with respect to n_A as well as n_B, and uses the result as the layout similarity between two pages A and B.

Alternatively, we can directly use the number of corresponding blocks n_{cor} as the similarity. We call this page similarity *Common block Number* (CN). Or, we can use the ratio of the corresponding blocks n_{cor}/n_A as the similarity when we decide whether page A is a phishing page. We call this page similarity *Common block Number Ratio* (CNR). In Section 5, both CN and CNR page similarity definitions will be implemented and compared with the one defined in Formula 1.

On the basis of extracting the page rendering features, the similarity of page signatures can be computed as described above. Further, page signatures can mix up text signatures, image signatures, overall picture signatures, layout signatures, and so on. The structure and search of the feature library have direct impact on the phishing detection speed. To speed up layout based similarity search in the detection, we create an R-tree to index all web page spatial layout features in the library.

4 Spatial layout feature based phishing detection

4.1 Motivation

Given the signatures of two web pages, their similarities can be calculated by simple matching [13], N largest match [20], EMD method [3, 21], Bipartite graph based matching method [16, 22], etc. However, for web phishing detection, a suspicious web page should be compared to the feature library whose size has a significant impact on the phishing detection speed. In practice, the feature library is often large and it needs to be frequently updated. This demands an efficient filtering method that quickly filters out irrelevant web sites from the feature library and enables to compare the suspicious page with a limited number of candidates from the feature library.

The bipartite graph based matching method uses some features to pruning web pages in the feature library, e.g., the number of DOM tree nodes, and the number of image nodes in web pages. Although this method improves the speed of phishing detection to some extent, it needs to traverse the entire feature library with time complexity. Therefore, this method does not apply to the case of large feature libraries.

Motivated as such, we propose to build a spatial index for the layout features of web pages in order to speed up the search of the feature library.

4.2 Indexing spatial layout features of web pages

R-tree is a tree based indexing data structure similar to B tree. It is widely used in spatial databases for indexing spatial data, which supports efficient processing of various spatial queries. For example, we can easily use R-tree to search for all gas stations within two kilometers to the current location. In R-trees, each spatial object is approximated by a minimum bounding rectangle (MBR), and MBRs are grouped to larger but fewer MBRs according to specific rules. The grouping of MBR is conducted recursively until a final single MBR is reached which is corresponds to the root of the R-tree.

Each node of an R tree has a certain number of entries. Each non-leaf node entry stores two fields: the address (pointer) of the corresponding child node, and the MBR of the corresponding child node. Each leaf-node entry stores the address (identifier) of the corresponding object and the object's MBR. It is noteworthy that each non-leaf node's MBR, stored in its corresponding entry in its parent node, is the MBR of all its child nodes' MBR. Based on the data partitioning described above as well as the consequent MBR containment property, R-tree can prune unpromising nodes in query processing and thus shorten the query time significantly.

In this paper, we use an R-tree to index the spatial features of all the web pages in the feature library. In particular, we create an R-tree and insert to it each spatial feature captured in an MBR for all web pages in the feature library. Each block is represented as $\langle pageID, blockID, MBR, pointer\rangle$, where $pageID$ is the identifier of the page that contains the block, $block$ is the identifier of the block within its page, MBR is the block's MBR, and $pointer$ points to other relevant information of that block. The overall indexing scheme is shown in Figure 8.

In particular, the Page Hash Table maps a given page identifier to the address of a Block Hash Table for that page. There are multiple Block Hash Tables, each for a specific page. Each such a Block Hash Table maps a block identifier to the R-tree leaf node where the block's index entry is contained.

4.3 Searching spatial layout feature library

Given a suspicious web page s that may be a phishing page, we need to find those legitimate pages that s is similar to. Here we employ the space layout based similarity to measure how similar two web pages are. All legitimate web pages in the feature library are organized based on their spatial layout features and indexed as described in Section 4.2. In other words, the phishing detection is reduced to a similarity search of the feature library using the spatial layout based page similarity metric.

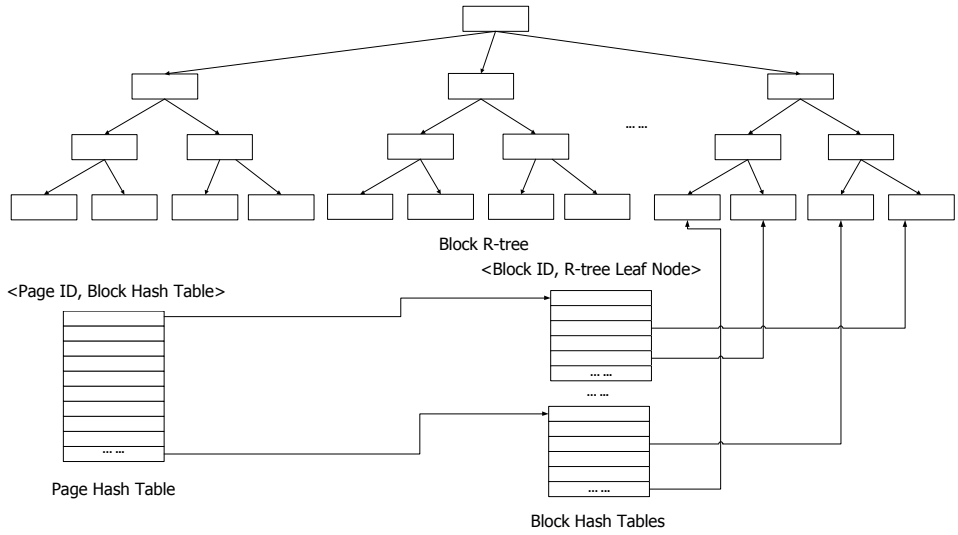

Figure 8: Index of the Library of Page Spatial Layout Features

The overall page similarity search is encapsulated in a function **pageSearch**. Its pseudo code is shown in Algorithm 1. It takes as input a suspicious web page s, the legitimate page feature library FL, and two integer numbers, N and K, for controlling the sizes of corresponding block matches and similar page matches respectively.

Algorithm 1 pageSearch(Suspicious web page s, legitimate web page feature library FL, number of blocks to match N, number of similar pages to return K)

1: $bs \leftarrow$ obtain all blocks from s ▷ Either DOM tree or image segmentation is employed here, depending on how the feature library is constituted.
2: $mbs \leftarrow$ initialize an array of $|bs|$ elements
3: $pages \leftarrow$ initialize a hash table that maps a page ID to a list of block IDs
4: **for** each block b_i in bs **do**
5: $mbs[i] \leftarrow$ topMatch(FL's R-tree, b, N)
6: **for** each $\langle PageID, BlockID\rangle$ in $mbs[i]$ **do**
7: add $\langle PageID, BlockID\rangle$ to $pages$
8: $H \leftarrow$ initialize a max-heap
9: **for** each page p that appears in $pages$ **do**
10: $score \leftarrow$ SIM(p, s) ▷ According to Equation 1
11: push $\langle p, score\rangle$ to H
12: **return** top-K pages in H

In particular, the algorithm first obtains the blocks of a given suspicious page s, and put them in bs (line 1). Here, either s's DOM tree or an image segmentation method is applied to generate all the blocks, depending on how the spatial layout features in the legitimate library are generated. Subsequently, for each block b in bs, it calls function topMatch that searches through the block R-tree to get the top-N corresponding block matches for b (line 5). Note OA based corresponding block matching is shown in Algorithm 2, and CDA based matching is shown in Algorithm 4.

Referring to Algorithm 2, OA based matching employs a max-heap H for matching blocks and a recursive search function **queryOA** via the library's block R-tree (line 3). The recursive function is shown in Algorithm 3. It conducts a depth-first traversal on the R-tree, and only considers nodes that overlap with the current block b (lines 3, 14, and 18). The overlapping blocks are kept in the max-heap H. Finally, the top-N ones are returned by Algorithm 2.

Algorithm 2 topMatchOA(legitimate web page feature library RF's R-tree R_{RF}, block b, integer value N)

1: $H \leftarrow$ initialize a max-heap
2: $OA_N \leftarrow 0$
3: queryOA(R_{RF}, b, OA_N, H)
4: **return** the top-N elements of H

The CDA based block match works as shown in Algorithm 4. Taking the mindist [23] metric between an R-tree node and the current block b as the key, it visits all block R-tree nodes in a best-first manner [24]. During the process, only those blocks with similar size are considered (line 6). Once the first N closest and most similar blocks are found, indicated by the result size, the algorithms returns the top-N matches (lines 10–11).

Refer to the page search (Algorithm 1) again. After the top-N corresponding block matches are obtained, it adds each result record $\langle PageID, BlockID\rangle$ to a list $pages$ (lines 6–7). After all blocks in the suspicious page s are processed likewise, the page search algorithm calculates the similarity between s and each page that appears in $pages$, and returns the top-K pages with the highest similarity scores (lines 8–12). The returned K pages will be used for further check to decide whether s is a phishing page or not. For that purpose, other page features like texts, images can be used to take a closer comparison between each returned page and the suspicious page s.

As a remark, we use the page spatial layout based simi-

Algorithm 3 queryOA(R-tree node n, block b, the current Nth overlap area OA_N, the current matched block heap H)

```
1: if n is a leaf node then
2:     for each block nb indexed by n do
3:         if nb does not overlap with b then continue
4:         if |H| < N then
5:             push ⟨nb.pageID, nb.blockID, OA(nb, b)⟩ to H
6:             if |H| = N then
7:                 update OA_N is necessary
8:         else if OA(nb, b) > OA_N then
9:             push ⟨nb.pageID, nb.blockID, OA(nb, b)⟩ to H
10:            update OA_N is necessary
11: else
12:    if |H| < N then
13:        for each child node cn of n do
14:            if cn overlaps with b then
15:                queryOA(cn, b, OA_N, H)
16:    else
17:        for each child node cn of n do
18:            if cn overlaps with b and OA(cn, b) > OA_N then
19:                queryOA(cn, b, OA_N, H)
```

Algorithm 4 topMatchCDA(legitimate web page feature library RF's R-tree R_{RF}, block b, integer value N)

```
1: result ← initialize an empty set
2: H ← initialize a min-heap
3: push ⟨R_RF.root, 0⟩ to H
4: while H is not empty do
5:     pop H's top element to ⟨n, dist⟩
6:     if node n is a block and n has comparable size length as b then add ⟨n.pageID, nb.blockID⟩ to result
7:     else
8:         for each child node cn of n do
9:             push ⟨cn, mindist(cn.center, b.center)⟩ to H
10:    if |result| = N then break
11: return result
```

larity SIM (line 10) in Algorithm 1. This similarity definition can be replaced by the other two page similarities CN and CNR that are defined in Section 3.3.

4.4 Overall phishing detection implementation

We use the above algorithms presented in Section 4.3 to compute the similarities between a suspicious web page and relevant pages in the legitimate library. Note that we do not compute such a similarity with respect to every page in the library. Instead, we prune irrelevant pages in the library through the R-tree.

The overall phishing detection procedure is shown in Figure 9. A specifically-designed browser plug-in records the protected web URLs that require username and password, extracts the spatial layout features (and other signatures like text and/or image) of that page, and stores the information in the feature library. The spatial layout feature R-tree is updated accordingly when the new piece of information is inserted to the feature library.

When a user accesses a current web page that also requires username and password but has a different URL from the stored one, the phishing detection will be invoked. The current page is used as the suspicious one.

The spatial layout features (and signatures) of the current web page are then extracted by the plug-in. Next, a set of K candidate web pages are returned by the page search algorithm (Algorithm 1 in Section 4.3). Further, the similarity of signatures between the suspicious page and the feature library is computed. When the similarity is greater than the given threshold, an alarm is raised to alert the user that she/he may be accessing a phishing web site.

To ease the phishing detection we can apply the threshold strategy directly to the spatial layout feature based similarity in deciding whether a suspicious page is a phishing page or not. In particular, if one of the computed similarities (in Algorithm 1) is greater than a pre-specified threshold, the suspicious web page will be considered as a phishing web page directly without involving further signatures.

Note that the feature library can be stored either locally or on a remote server. In the latter scenario, the browser plug-in will send the current URL to the server when a user visits a web page that requires a password. The server will conduct the phishing detection for the URL and return the detection result to the client browser.

5 Experimental study

In this section, we evaluate the spatial layout feature based phishing detection through a series of experiments. We investigate the filtering performance of the spatial layout feature based library filtering, and the overall effectiveness of phishing detection based on spatial layout similarity.

5.1 Performance metrics

Throughout our experiments, we have the following variables relevant to the numbers of different web pages involved in phishing detection.

- A is the number of *phishing* web pages that are detected as *phishing* web pages.
- B is the number of *normal* web pages that are detected as *phishing* web pages.
- C is the number of *phishing* web pages that are detected as *normal* web pages.
- D is the number of *normal* web pages that are detected as *normal* web pages.

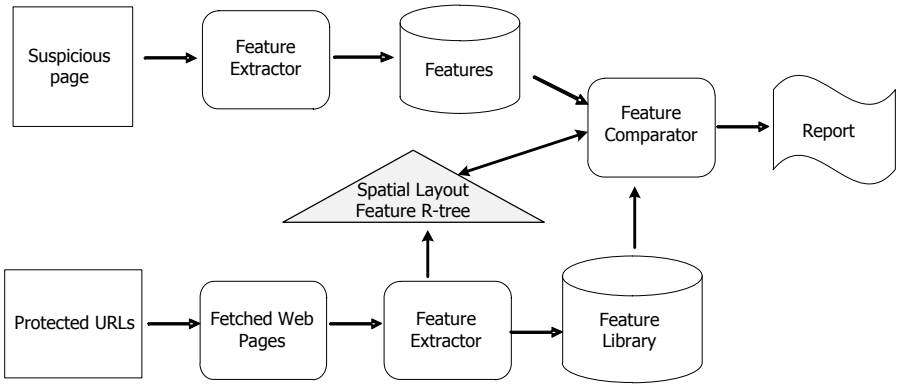

Figure 9: Spatial layout similarity based phishing detection

Accordingly, we consider the following performance metrics in our experimental study.

- Precision = A/(A+B)
- Recall = A/(A+C)
- True positive rate (TPR) = A/(A+C)
- False positive rate (FPR) = B/(B+D)

Precision is the ratio of correct reports in all phishing page reports. Recall describes the detected proportion of all phishing pages. These two evaluations are mutually exclusive. A high precision means a low recall, while a low precision means a high recall. For phishing detection, recall is more important and small numbers of incorrect reports of phishing pages are acceptable because the security is the major concern.

In the experiments, phishing web pages are referred as positive instances, and the normal web pages are referred as negative instances. TPR represents the probability that true phishing web pages are reported ass phishing web pages; FPR represents the probability that normal web pages are incorrectly reported as phishing web pages. Different thresholds can be used in obtaining the corresponding TPR and FPR, as well as the corresponding ROC curve. Accordingly, the corresponding AUC (area under the curve) can be calculated [4].

If the legitimate web page targeted by a phishing page is reserved after the feature library filtering, i.e., it is not pruned out, we call it a hit. Accordingly, we consider another performance metric *hit rate*. It is the number of hits divided by the total number of phishing web pages.

5.2 Experimental settings

We compare the design options listed in Table 1. All experiments are implemented in javascript and Java. They are run on a Pentium 5 desktop PC with double 2.6GHz CPUs, 2GB main memory. The PC runs on Windows XP SP3, and has a Mozilla Firefox 3.0 web browser.

We makes use of the data collected from web site PhishTank [25]. As an open and free anti-phishing web site, it allows users to easily submit the suspicioused web sites which will be confirmed by experts. We do not directly retrieve phishing web sites because most phishing web pages do not exist for a long period of time.

Specifically, we retrieve 100 positive pairs of English phishing pages and their corresponding real-world legitimate web pages. We also collect 100 negative legitimate English web pages from banks, credit unions and online services according to Yahoo! Directory [26]. In the experiments, the 100 target web pages of phishing web pages are stored in the feature library; the 100 corresponding phishing web pages (positive examples) and the 100 general web pages (negative examples) are used as suspicious web pages.

For each legitimate or suspicious web page, its DOM tree based spatial layout features are obtained by calling a Firefox plug-in. Whereas the image segmentation based spatial layout features are obtained using the nested Earth Mover's Distance based image segmentation [21].

5.3 Results

5.3.1 Hit rates

We first investigate the hit rates of different design options. Specifically, we compare 12 different methods that are obtained by selecting one option for each of the three phases in Table 1. The results are shown in Figures 10 to 13. As K increases, all methods get higher hit rates.

Figure 10 reports the results of those methods that use OA block matching following DOM tree based block generation. For small (less than 5) and large (larger than 18) K values, OA-SIM achieves the highest hit rates. Whereas all three methods perform very closely. This shows that the spatial layout feature based similarity (SIM) is able to improve the hit rates.

Figure 11 reports the results of those methods that use CDA block matching following DOM tree based block generation. For four particular K values (8, 9, 10, 11), all three methods have the same hit rate. Apart from that, CDA-SIM clearly outperforms the other two methods for

Table 1: Design Options

Phase	Options
Block generation	DOM tree based (DOM), Image segmentation based (IMG)
Corresponding Blocks Match	Overlap Area (OA), Center Distance and Area (CDA)
Page similarity	CN, CNR, SIM (Equ. 1)

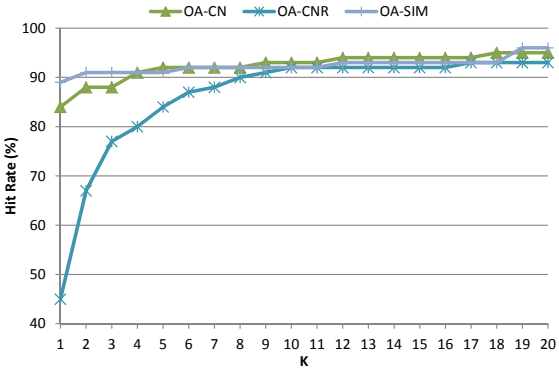

Figure 10: OA block matching for DOM tree segmentation

all K values. This suggests that SIM is a very good similarity definition that is able to get high hit rates.

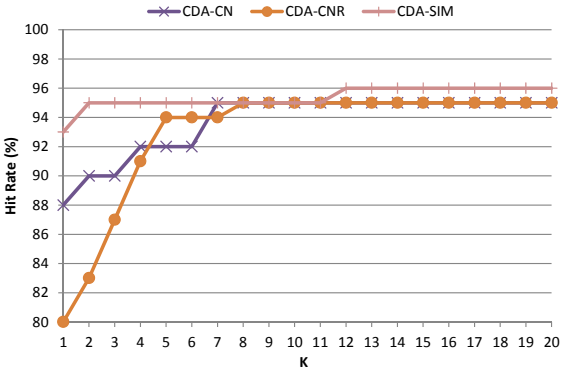

Figure 11: CDA block matching for DOM tree segmentation

In addition, comparing Figure 10 and 11 indicates that CDA is a better corresponding block matching option as it leads to higher hit rates that OA. Given to blocks, CDA takes into account not only their size difference but also the distance between their centers. Therefore, CDA captures corresponding blocks better than OA that considers overlapping areas only. A large overlapping area does not necessary mean two blocks are visually similar and close to each other.

Figures 12 and 13 report the results of those methods that use image segmentation based block generation, followed by OA and CDA matching respectively.

According to the results shown in Figure 12, SIM works very well as a novel similarity definition for methods using OA block matching. The method OA-SIM obtains the highest hit rates for all K values except 19 and 20 where OA-CN gets better with very slight differences.

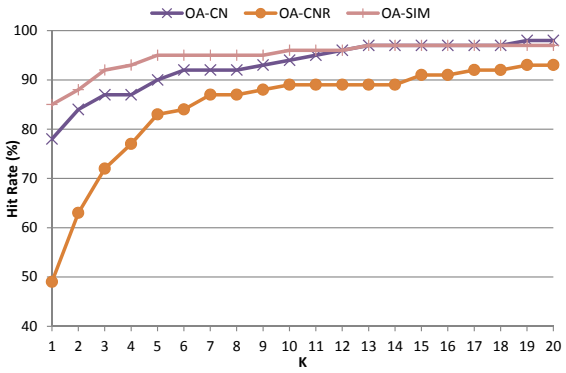

Figure 12: OA block matching for image segmentation

SIM still performs well for methods using CDA block matching, according to the results reported in Figure 13. All methods here are less steady, which indicates that CDA is more sensitive to the image segmentation based block generation.

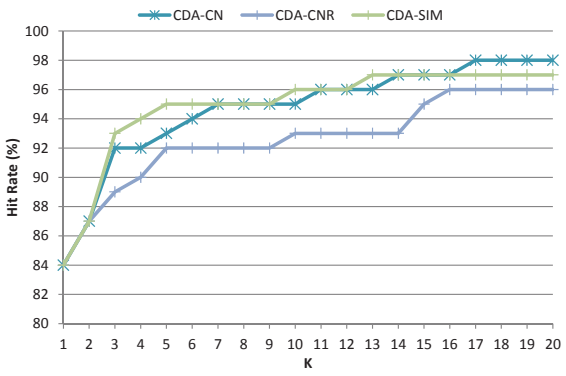

Figure 13: CDA block matching for image segmentation

To summarize, image segmentation based block generation yields better performance than DOM tree based block generation in terms of hit rates. This is because the segmentation captures the visual effects and spatial layout features better for web pages rendered in a browser. Also, CDA captures corresponding blocks better than does OA.

5.3.2 TPR and FPR

We investigate both TPR and FPR for different methods. They are two important indicators of the phishing detection accuracy. In particular, we study the receiver operating characteristic by plotting both indicators in ROC curves.

The results are reported in Figures 14 and 15, for DOM tree based block generation and image segmentation based block generation respectively.

Both ROC curves are convex compared to their corresponding main diagonals. This shows all three methods in comparison (OA-CNR, OA-SIM and CDA-SIM) are effective in phishing detection. Moreover, the curves of CDA-SIM have larger AUC (area under the curve) in both figures. This indicates that CDA-SIM is the best among all the three in consideration.

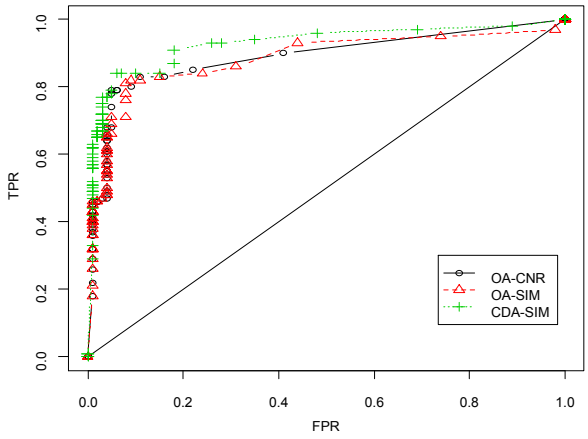

Figure 14: ROC curve of methods using DOM tree segmentation

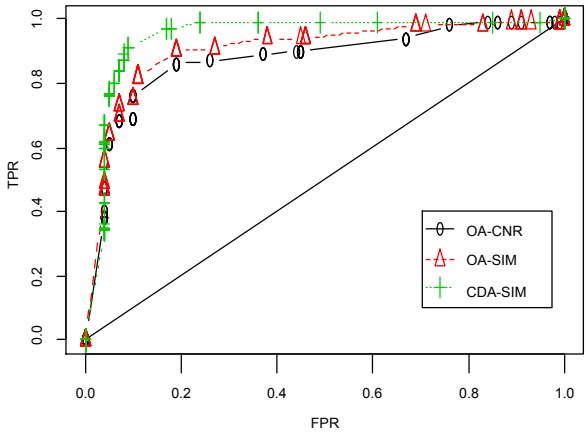

Figure 15: ROC curve of methods using image segmentation

5.3.3 Precision and recall

Next, we study another two performance metrics, namely precision and recall, for methods that use SIM as the page similarity. We omit CN and CNR because the results from previous experiments demonstrate that SIM has the best overall performance. We compare DOM tree based block generation and image based block generation, followed by OA or CDA block matching. The results are listed in Table 2.

The highest precision (0.933) is achieved by DOM-CDA-SIM. While the highest recall (0.919) is achieved by IMG-CDA-SIM. This demonstrates that CDA-SIM is a very good combination for phishing detection because it is able to achieve both high precision and high recall.

We also calculate the F1 score for each method using the following formula:

$$\text{F1 score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The results are also reported in Table 2. Among all the four methods in comparison, IMG-CDA-SIM is overall the best method for phishing detection as it achieves the highest F1 score.

5.3.4 Library search time

Finally, we study the execution time of the proposed spatial layout based phishing detection. In particular, we measure the execution time that is spent on search the legitimate web page library. We compare a sequential scan method with our R-tree facilitated library search. A sequential scan compares a given suspicious web page with each legitimate page in the legitimate library. We vary the R-tree fanout from 5 to 15 in the index for spatial layout features. We vary the legitimate library size from 1 to 100 to see its effect on the library search efficiency. The results on library search time are reported in Figure 16.

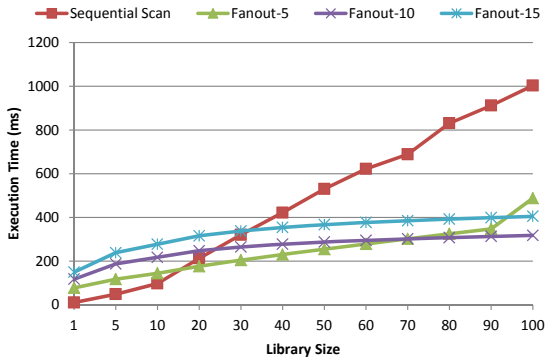

Figure 16: Legitimate library search time comparison

The sequential scan works the best for very a small legitimate library with no more than 10 pages. As the library size increases, the propose spatial layout feature based filtering catches up and clearly outperforms the sequential scan when the library contains more than 30 pages. A legitimate library usually is large with tens or even hundreds of legitimate pages, it is therefore beneficial to employ the proposed spatial layout feature based search to filter out irrelevant legitimate pages quickly. This can speed up the overall phishing detection.

Table 2: Results of Precision and Recall

Method	Precision	Recall	F1 Score
DOM-OA-SIM	0.911	0.837	0.872
DOM-CDA-SIM	0.933	0.857	0.894
IMG-OA-SIM	0.826	0.909	0.865
IMG-CDA-SIM	0.910	0.919	0.915

6 Conclusion and future work directions

In this paper, we propose a spatial layout similarity based approach for phishing web detection. This novel approach takes into account important spatial layouts of web pages. For this approach, we first invent two meaningful methods that extract the spatial layout features from web pages. After obtaining such spatial layout features, we define a similarity function to quantize how visually similar two web pages are. Such a similarity measurement indicates how a suspicious page is a phishing one in relation to a legitimate web page. In order to speed up searching the legitimate feature library, we further design an R-tree index for all spatial layout features in the library and develop search algorithms accordingly. Finally, we evaluate the proposed approach through a series of experiments. The results demonstrate the effectiveness and efficiency of our proposal.

Several directions exist for future work. First, it is of interest to combine the spatial layout features proposed in this paper with other types of features available for web pages. For example, text features of web pages can be extracted together with spatial layout features. As a result, phishing detection can make use of a mix of different types of features.

Second, it is relevant to integrate DOM tree and image segmentation in web page block generation. The former is easy to implement with accessible web browser APIs; while the latter captures the visual and spatial layouts of web pages more closely to the way humans do. Combining these two methods may generate blocks that are more decisive in phishing detection.

Third, the library search algorithms (Section 4) in this paper can be further optimized by processing relevant spatial queries in a collective way. Such optimization is more relevant when the legitimate page library is large.

Acknowledgments

Weifeng Zhang's work was supported by the National Natural Science Foundation of China under Grant No.61272080 and No.61100135, Opening Foundation of Guangxi Key Laboratory of Trustworthy Software, and Opening Foundation of Jiangsu Key Laboratory of Computer Information Processing Technology in Soochow University (Grant No.KJS0714).

References

[1] Yue Zhang, Jason Hong, and Lorrie Cranor (2007). Cantina: a content-based approach to detecting phishing web sites. In *WWW*.

[2] (2011). Netcraft Anti-phishing Toolbar. http://toolbar.netcraft.com.

[3] Anthony Y. Fu, WenYin Liu, and XiaoTie Deng (2006). Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD). *IEEE Trans. Dependable Sec. Comput.*, 3(4):301–311.

[4] Yue Jiang, Bojan Cukic, and Yan Ma (2008). Techniques for evaluating fault prediction models. *Empirical Software Engineering*, 13(5):561–595.

[5] R. Song, H. Liu, J.R. Wen, and W.Y. Ma (2004). Learning blockimportance models for webpages. *In Proceedings of the WWW*.

[6] I. N. Fette, Sadeh, and A. Tomasic (2006). Learning to detect phishing emails. ISRI Technical report, CMU-ISRI-06-112.

[7] André Bergholz, Jeong Ho Chang, Gerhard Paass, Frank Reichartz, and Siehyun Strobel (2008). Improved phishing detection using model-based features. In *CEAS*.

[8] André Bergholz, Gerhard Paass, Frank Reichartz, Siehyun Strobel, Marie-Francine Moens, and Brian Witten (2008). Detecting known and new salting tricks in unwanted emails. In *CEAS*.

[9] Ziv Bar-Yossef and Maxim Gurevich (2009). Estimating the impressionrank of web pages. In *WWW*, pages 41–50.

[10] (2011). McAfee SiteAdvisor. http://www.siteadvisor.com.

[11] Tyler Moore and Richard Clayton (2009). Evil searching: Compromise and recompromise of internet hosts for phishing. In *Financial Cryptography*, pages 256–272.

[12] WenYin Liu, GuangLin Huang, XiaoYue Liu, Min Zhang, and XiaoTie Deng (2005). Detection of phishing webpages based on visual similarity. In *WWW*

(Special interest tracks and posters), pages 1060–1061.

[13] Angelo P. E. Rosiello, Engin Kirda, Christopher Kruegel, Fabrizio Ferr, and Politecnico Di Milano (2007). A layout-similarity-based approach for detecting phishing pages. In *SecureComm*, pages 454–463.

[14] Ponnurangam Kumaraguru, Ro Acquisti, Lorrie Faith Cranor, and Jason Hong (2010). Teaching johnny not to fall for phish. *ACM Trans. Internet Techn.*, 10(2).

[15] WenYin Liu, XiaoTie Deng, GuangLin Huang, and Anthony Y. Fu (2006). An antiphishing strategy based on visual similarity assessment. *IEEE Internet Computing*, 10(2):58–65.

[16] WeiFeng Zhang, YuMing Zhou, Lei Xu, and BaoWen Xu (2010). A method of detecting phishing web pages based on hungarian matching algorithm. *Chinese Journal of Computers*, 33(10):1963–1975.

[17] QiaoPing Zhang, DeRen Li, and JianYa Gong (2004). Surface entity matching techniques of city map database. *International Journal of Remote Sensing*, 8(2):107–112.

[18] XiaoHua Tong, SuSu Deng, and WenZhong Shi (2007). Probability based map entity matching method. *International Journal of Surveying and Mapping*, 36(2):210–217.

[19] A Masuyama (2006). Methods for detecting apparent differences between spatial tessellations at different time points. *International Journal of Geographical Information Science*, 20(6):633–648.

[20] E. Medvet, E. Kirda, and C. Kruegel (2008). Visual-similarity-based phishing detection. In *SecureComm*, pages 1–6.

[21] JiuXin Cao, Bo Mao, JunZhou Luo, and Bo Liu (2009). A phishing web pages detection algorithm based on nested structure of earth mover's distance (Nested-EMD). *Chinese Journal of Computers*, 32(5):922–929.

[22] Harold W. Kuhn (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

[23] Nick Roussopoulos, Stephen Kelley, and Frédéic Vincent (1995). Nearest neighbor queries. In *SIGMOD Conference*, pages 71–79.

[24] Gísli R. Hjaltason and Hanan Samet (1999). Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2):265–318.

[25] (2011). PhishTank. http://www.phishtank.com.

[26] (2011). Yahoo! Directory. http://dir.yahoo.com/.