Computer Science and Information Systems 16(3):815–830 https://doi.org/10.2298/CSIS180915028L

Algorithm of Web Page Similarity Comparison Based on Visual Block

Xingchen Li[1], Weizhe Zhang[1,2], Desheng Wang[1], Bin Zhang[2], and Hui He[1]

[1] School of Computer Science and Technology,
Harbin Institute of Technology, Harbin,China
16s003084@stu.hit.edu.cn, (wzzhang,wangdesheng0821,hehui)@hit.edu.cn
[2] Cyberspace Security Research Center,
Peng Cheng Laboratory, Shenzhen, China
bin.zhang@pcl.ac.cn

Abstract. Phishing often deceives users due to the relative similarity to the true pages on a layout and leads to considerable losses for the society. Consequently, detecting phishing sites has been an urgent activity. By researching phishing web pages using web page screenshots, we discover that this kind of web pages use numerous web page screenshots to achieve the close similarity to the true page and avoid the text and structure similarity detection. This study introduces a new similarity matching algorithm based on visual blocks. First, the RenderLayer tree of the web page is obtained to extract the visual block. Second, an algorithm that will settle the jumbled visual blocks, including the deletion of the small visual blocks and the emergence of the overlapping visual blocks, is designed. Finally, the similarity between the two web pages is assessed. The proposed algorithm sets different thresholds to achieve the optimal missing and false alarm rates.

Keywords: phishing, similarity comparison, visual block, web rendering.

1. Introduction

With the advent of the network information age, the Internet has significantly improved people's works and lives [16] and has accelerated the spread of information. However, prohibited tools for illegal profit-making have emerged simultaneously. Phishing websites is one of the killers that threaten network development in China. These websites confuse users when a domain name similar to the real website is used [18], thereby seriously damaging the vital interests of regular enterprises and numerous netizens and seriously disrupting China's network trading environment. At present, phishing websites mainly target financial and e-commerce websites [7] to defraud money. With the increase in the popularity of family digitalization and network broadband and the emergence of new consumption patterns (e.g., e-commerce, online bank payment, and online settlement), phishing attacks have become the most serious threat to the realistic safety of e-commerce [17].

Web page similarity comparison relates to network rendering technology in addition to active detection technology. Network rendering technology involves the rendering principles of a browser and a kernel and the generation process of the rendering tree.

The rendering principle of a browser is presented in Fig. 1. A browser parses a web page into three steps.

1) A document object model (DOM) tree is produced using HTML, XHTML, or SVG files..
2) The CSS files will generate a CSS rule tree.
3) JavaScript scripts, mainly through the DOM and CSSOM API, are used to operate the DOM and CSS rule trees.

After completing the parsing, the browser engine constructs the rendering tree through the DOM and CSS rule trees. The main steps are as follows:

1) The rendering tree is not equivalent to the DOM tree because of a header or a display; none is unnecessary in the rendering tree.
2) The CSS rule tree must complete the match and attach the CSS rule to each element (or frame) on the rendering Tree, which is the DOM node.
3) The position of each frame (i.e., each element) is calculated. This procedure is called the layout and reflow process.

Finally, the API of the operating system is drawn which is called native GUI.

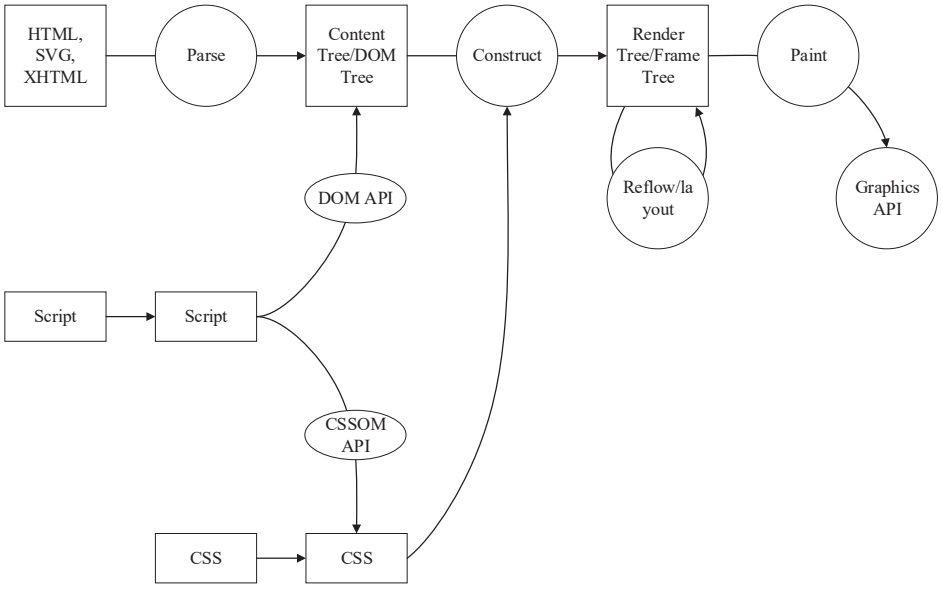


Fig. 1. Browser rendering principle

The construction process of the rendering tree is illustrated in Fig. 2. The browser's rendering page must first build the DOM and CSSOM trees. The DOM tree captures the attributes and relationships of the document markup, whereas the CSSOM tree presents the appearance of all elements after rendering. The CSSOM and the DOM trees are merged into the rendering tree and then used to calculate the layout of each visible element. The results are outputted into the rendering process to render the pixels to the

screen. The rendering tree, which contains only the nodes required to render the web pages, traverses the nodes in each DOM tree, thereby finding the style of the current node in the CSSOM rule tree to generate the rendering tree. In addition, the rendering tree calculates the exact location and size of each object in the layout of the web page. Every visible node is traversed from the root node of the DOM tree during the traversal process. The invisible nodes that are not reflected in the rendering output (e.g., script and meta tags) will be ignored. Similarly, some nodes that are hidden by CSS will also be ignored in the rendering tree. The suitable CSSOM rules for each node are then determined and applied. The matching begins from the right side of the selector to the left, thereby indicating that the matching begins from the child node of the CSSOM tree to the parent node. In addition, the time required to execute the rendering tree construction, layout, and drawing will depend on the size and application style of the document and on the device that runs the document. That is, a large document indicates additional work that the browser must complete. In addition, the drawing is time-consuming when the style is increasingly complex.

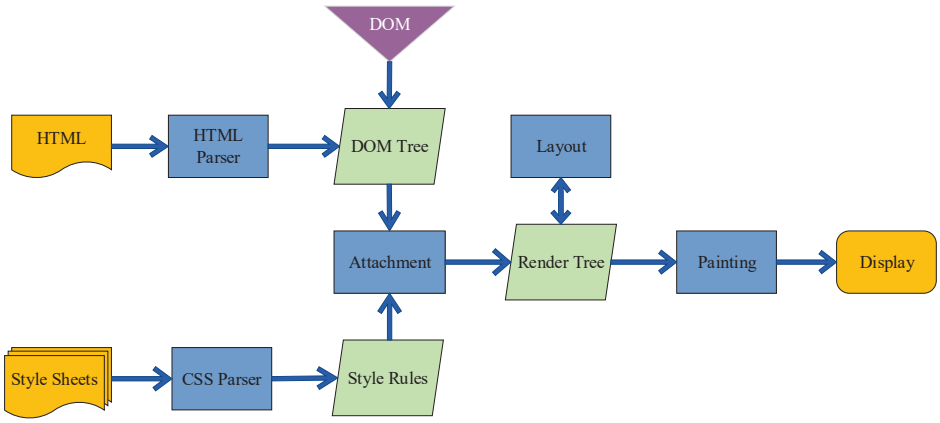


Fig. 2. The construction process of render tree

On the basis of some visual nodes of the DOM tree, the browser will build the corresponding nodes in accordance with the node properties. These nodes will also form a rendering tree. The browser will create new nodes on the basis of this rendering tree to form a RenderLayer tree. The structural relationship between the three types of trees is demonstrated in Fig. 3.

On the one hand, the rendering tree is a new tree that is built in accordance with the DOM tree. The nodes of the rendering tree do not directly correspond to those of the DOM tree. When encountering the non-visual nodes in the DOM tree, the rendering tree will not create new nodes. After building the rendering tree, the layout operation will calculate the related attributes, including location, size, and floating. This information will allow the rendering engine to know the location and the manner of drawing the elements.

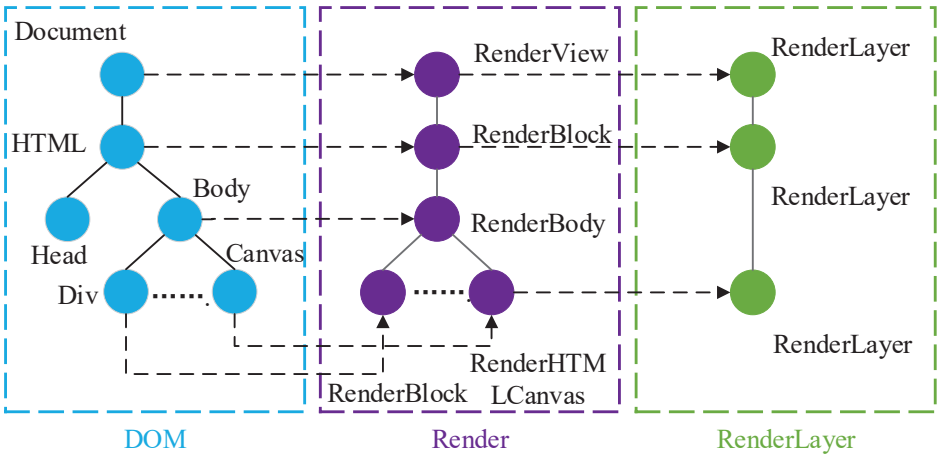


Fig. 3. The relationship between the three trees

On the other hand, the RenderLayer tree is a new tree that is built on the basis of the rendering tree. In addition, the nodes of this tree do not directly correspond to those of the rendering tree. Alternatively, a one-to-many relationship occurs among the nodes. In some cases, the tree must create new RenderLayer points. In the present study, the visual information of the web pages is obtained by analyzing the RenderLayer trees of the web pages and then extracting the information.

In this study, we devise a method to detect phishing sites based on visual blocks. The major contributions of this study are presented as follows:

1) Obtain the RenderLayer tree of the web page where the visual block will be extracted.
2) Design an algorithm that will settle jumbled visual blocks. This process involves deleting the small visual blocks and merging the overlapping visual blocks.
3) Define the similarity of the two web pages, and set different thresholds for the algorithm to achieve the optimal miss and error rates.

The remainder of this paper is organized as follows: Section 2 presents the related work. Section 3 discusses the similarity matching algorithm. Section 4 introduces the evaluation results. Section 5 provides the conclusions drawn from this study.

2. Related Work

Phishing detection methods are mainly divided into two categories, namely, detecting URL and identifying the content of the web pages. The former identifies a phishing website in accordance with the URL through mathematical modeling and is suitable for detecting large quantities of web pages.

The research on web page similarity can be divided into the following categories: traditional text, structural, and visual similarity studies.

The text-based similarity detection extracts the text contents of the web page and compares them with the text template of the phishing web page to determine if the latter is a phishing website. The commonly used algorithms are Simhash and Imatch algorithms. Takama et al. [3] proposed a fast HTML web page detection approach, which provided direct access to node information by hashing the web page. The detection of the changes in the two versions of a page was accomplished by performing similarity computations after transforming the web page into an XML-like structure. It showed rapid improvements when compared to the results of a previous approach. Zhuang et al. [21] extracted 10 different types of features such as title, keyword and link text information to represent the website. Roopak et al. [14] proposed a novel method for detecting phishing pages by searching similar web pages through web mining and then comparing the web pages by matching the HTML source codes and computing the cosine similarity of the textual contents. The experiment results indicated that the detection rate is higher in the proposed mechanism than in other existing methods. Nichele et al. [12] presented a domain taxonomy-based clustering approach, which integrated page similarities to compute the session similarity. In the paper [15], authors used the Jaccard coefficient formula to determine the similarity among the web pages. This approach can be applied for usage and navigation clustering purposes. Huo et al. [13] proposed an N-gram algorithm to realize the comparison of web page similarities. Anari et al. [2] determined the similarity of web pages based on learning automata and probabilistic grammar.

DOMs are first used to express the internal structure of HTML files, thus contributing significantly to HTML parsing. Typical algorithms for structural similarity research are web page similarity measurement methods based on edit distance and statistical features. Edit distance, which is also known as Levenshtein distance, refers to the minimum number of edit operations required between two strings, from one to another. Licensed editing operations involve replacing one character with another, inserting one character, and deleting one character. In general, a small edit distance indicates a considerable similarity between two strings. Similar to the string, the number of edit operations is defined by inserting, deleting, and modifying the nodes of the tree. Similarly, a small edit distance of the two trees implies a huge similarity between the two trees. Therefore, the similarity of the web page can be calculated by comparing the edit distance of the web page's DOM tree. Web page similarity measurement methods based on statistical features include a method based on link and node features. The link feature focuses on the order of the nodes, while the node feature focuses on the content of the nodes. In the paper [9], authors constructed the links from the root node to the leaf node in the DOM tree of the independent web page and regarded the link frequency as a feature of the link to judge the similar web pages based on link frequency similarity. This algorithm is difficult but simple because it only considers tag information. Alpuente et al. [1] proposed a functional technique that transforms each web page into a compressed, normalized tree to represent a visual structure. An optimization of this technique, which was developed on the basis of memorization, achieved remarkable improvements in efficiency in time and space.

Visual similarity detection uses the similarity of the image to match the similarity of the web pages [8,19,11]. By cutting the web page image and extracting the size, color, pixel, and other features of each sub-graph, the distance between the two features and the similarity with the feature template are calculated. The visual similarity detection system of the DOM structures of web pages proposed by Liu et al. [20] focused on the similarity

of web page typesetting. Liu et al. [10] detected phishing web pages in accordance with the similarity among the visual blocks after rendering the screening page, which involves two processes. The first process ran on local email servers and monitored emails for keywords and suspicious URLs, while the second process compared the potential phishing pages against actual pages and assessed the visual similarities among them in terms of key regions, page layouts, and overall styles. Fu et al. [6] proposed a phishing detection scheme that calculated the visual similarity of web pages based on Earth mover distance. In the paper [5], authors developed an automated process for crawling web pages. The Porter stemmer algorithm was used to convert the keywords into basic forms, and the term frequency–inverse document frequency method was utilized to obtain the importance of a keyword. In the paper [4], authors detected phishing using website components, such as source code, CSS, and logo. Furthermore, this method adopted the Jaccard coefficient formula to determine the similarity among web pages.

3. Similarity Matching Algorithm

This section introduces a web page similarity matching algorithm based on visual blocks. The algorithm consists of three parts. The first part is the extraction of the web page's visual information. In this part, the RenderLayer tree of the web page must be analyzed, and the visual information, including the location and size information of the visual nodes, must be extracted. The second part is the merging of visual nodes. Given the scattered numerous visual nodes of the web, the small ones can be ignored directly, while the overlapping ones require collision analysis and merge processing to maintain the consistency of the visual blocks with the original web page and improve the calculation efficiency. The last part is the web page similarity comparison, which utilizes the optimal free matching algorithm. For each visual block in the target web page, given that the visual blocks with coordinates, length, and width within a certain threshold range is found in the template web page, the blocks are considered similar. When the number of similar visual blocks is greater than a certain threshold, the two web pages are considered similar

3.1. Extracting Visual Blocks

Most web visual information are stored in the CSS file. Thus, obtaining the coordinates, lengths, and widths of the visual nodes directly from the HTML source code is difficult. The RenderLayer tree is more consistent with the visual structure of the original web page than the rendering tree. The non-visual nodes in the original web page (e.g., HEAD, META, and SCRIRT nodes) have been deleted.

The structure of the RenderLayer tree is shown in Fig 4. The first line represents the total size of the layer point, and the remaining lines signify the visual nodes of the layer point. The coordinates of the visual nodes in the layer are all relative coordinates. Thus, each visual node must convert the coordinates into the absolute coordinates of the web page.

To find the father node of each visual node, the structure of the RenderLayer tree must be analyzed. This study introduces an algorithm that converts the text form of the RenderLayer tree into an HTML source code to facilitate parsing. The conversion process is presented in Algorithm 1.

```
layer at (0,0) size 1010x6635
  RenderBlock {HTML} at (0,0) size 1010x6635 [bgcolor=#FFFFFF]
    RenderBody {BODY} at (0,0) size 1010x6635
      RenderBlock {DIV} at (0,0) size 1010x6635
        RenderBlock {DIV} at (0,0) size 1010x30 [color=#555555] [bgcolor=#FFFFFF] [border: none (
        1px solid #DDDDDD) none]
        RenderBlock {DIV} at (0,30) size 1010x149 [bgcolor=#F6F6F6]
          RenderBlock {DIV} at (10,109) size 990x40
            RenderBlock (floating) {UL} at (180,0) size 810x40 [bgcolor=#FF7300]
        RenderBlock {DIV} at (0,838) size 1010x266 [bgcolor=#FAFAFA]
          RenderBlock {DIV} at (10,20) size 990x263
```

Fig. 4. Structure of the renderlayer tree

Algorithm 1 Converting algorithm

Input: D_c: Depth of the current line, D_p: Depth of the previous line, T_c: Tag of the current line, E_c: End tag of the current line

Output: F: Format of HTML source code

```
1:  function CONVERTING
2:      for line do
3:          F.wirte(T_c)
4:          if D_c > D_p then
5:              S.push(E_c)
6:          end if
7:          if D_c < D_p and and D_p is the last line of the layer node then
8:              while ! S.empty() do
9:                  F.write(S.pop())
10:             end while
11:         end if
12:         if D_c < D_p and and D_p is not the last line of the layer node then
13:             depth = D_p − D_c
14:             while depth != 0 do
15:                 F.write(S.pop())
16:                 depth = depth - 1
17:             end while
18:         end if
19:         if D_c == D_p then
20:             F.write(S.pop())
21:             S.push(E_c)
22:         end if
23:     end for
24: end function
```

The tags of each line in the RenderLayer tree is written in the output result. The tags include the name, coordinate position, and the length-width attributes of the tag. Then, through the comparison of the depths, the relationship between the current and the previous lines can be determined. On the one hand, if the depth is larger in the current row than in the previous line, then the node is the child node of the previous ones. Thus, the tail tag is added to the stack. For example, if the tag is <HTML>, then the tail tag is also <HTML>. The stack is used to save the sequence of the nodes. On the other

hand, if the depth is smaller in the current line than in the previous line and the previous line is the last line of the RenderLayer point, then the line is the starting line of the new layer point (i.e., the previous layer point has ended). The stack stores the tag order of the previous layer point. Thus, all elements in the stack must be discarded. The tail tags are then written in the output results. However, if the depth is smaller in the current line than in the previous line, but the previous line is not the last line of the layer point, the depth difference between the current and the previous lines must be determined to select the number of element output of the stack. If the depth of the current line is the same as that of the previous line, then the current node is the brother node of the last node. The top element output of the stack is written in the output results, and the current node tag is attached to the stack. The top end tag in the stack is popped and written in the output, and then the end tag of this node is attached to the stack. The whole algorithmic process is equivalent to restoring the visual attributes to the HTML source code, thereby unifying all tag formats in the code. The output results are shown in Fig. 5.

```
<html x=0 y=0 length=1010 width=6635>
<body x=0 y=0 length=1010 width=6635>
<div x=0 y=0 length=1010 width=6635>
<div x=0 y=0 length=1010 width=30>
</div>
<div x=0 y=30 length=1010 width=149>
<div x=10 y=109 length=990 width=40>
<ul x=180 y=0 length=810 width=40>
</ul>
</div>
</div>
<div x=0 y=838 length=1010 width=266>
<div x=10 y=20 length=990 width=263>
</div>
</div>
</div>
</body>
</html>
```

Fig. 5. The output of algorithm 1

3.2. Merging Visual Blocks

Most web pages contain thousands of visual nodes. These visual nodes are disorganized and discrete, and some visual blocks are overlapped. To make the extracted visual blocks consistent with the original web pages, the visual blocks must be processed by deleting the small visual blocks (i.e., overlooking) and merging the overlapped ones. The reduction in the number of visual blocks can facilitate the improvement of computational efficiency.

Each visual block is a rectangle. The merging process of the visual blocks may be horizontal and vertical. The coordinates represent the position of a visual block in a web page. The visual blocks with proximal abscissas can be merged after the abscissa ordering. Similarly, the visual blocks with proximal ordinates can be merged after the ordinate ordering.

The current study presents a merging algorithm for visual blocks. Assuming that a list stores the visual blocks after ordering, in Algorithm 2, Pointer *i* points at the first element in the list of the visual blocks before merging, whereas Pointer *j* points at the second one. First, the visual blocks pointed by Pointers *i* and *j* must be assessed for suitability in merging. If the points are unsuitable, then both pointers are moved backward for one bit. Otherwise, the merging of two visual blocks is considered. Second, multiple overlapping similar visual blocks must be merged. Therefore, Pointer *j* must move backward for one bit until the visual blocks pointed by both pointers cannot be merged. At this time, the visual block pointed by Pointer *i* and the previous visual block pointed by Pointer *j* are merged. Pointer *i* points at Pointer *j*, which will move backward for one bit. When Pointer *j* points at the end of the list, one merge process ends. Finally, when the length of the list of two attached merging results did not change, or if no visual blocks were merged, then the algorithm ends.

Algorithm 2 merging algorithm

Input: L_{before}: A list saving visual blocks before merging, i: Pointer i, j: Pointer j
Output: L_{after}: A list saving visual blocks after merging

```
1:  function MERGING
2:      while L_after.size() not change do
3:          i ← L_before[0]
4:          j ← L_before[0]
5:          while j != L_before.size() do
6:              if !merge(i,j) then i ← i + 1 j ← j + 1
7:              end if
8:              if merge(i,j) then
9:                  while merge(i,j) do j ← j + 1
10:                 end while
11:                 L_after.add(merge(i,j-1))
12:                 i ← j
13:                 j ← j + 1
14:             end if
15:         end while
16:     end while
17: end function
```

Visual block merging mainly depends on the blocks' coordinates. Two blocks can be merged if their coordinates are adjacent. However, the merging rules vary for different cases.

Fig. 6 presents some possible cases of the horizontal merging of visual blocks. The rectangle with a thick line is called the reference block, and the one with a thin line is called the fitting block. The coordinates, lengths, and widths of the reference and fitting

blocks are $(X1, Y1)$, $a1$, and $b1$, and $(X2, Y2)$, $a2$, and $b2$, correspondingly. The conditions for horizontal merging are $Y1 = Y2$ and $b1 = b2$. If the abscissa is larger in the target block than in the reference block, then three cases are possible. The first case is where the reference and target blocks are overlapped, which is the most common case (Fig. 6(a)). The length after merging is $X2 - X1 + a2$. Fig. 6(b) illustrates the second case, wherein the target block is within the reference block because the length of the former is small. This case is generally the search box on the web page, and the length after merging is $a1$. The third case is depicted in Fig. 6(c), wherein both blocks are not overlapped. The length after merging is the same as that in the first case. In summary, the coordinates, length, and width of the visual block after merging are $(X1, Y1)$, max $(a1, X2 - X1 + a2)$, and b, respectively.

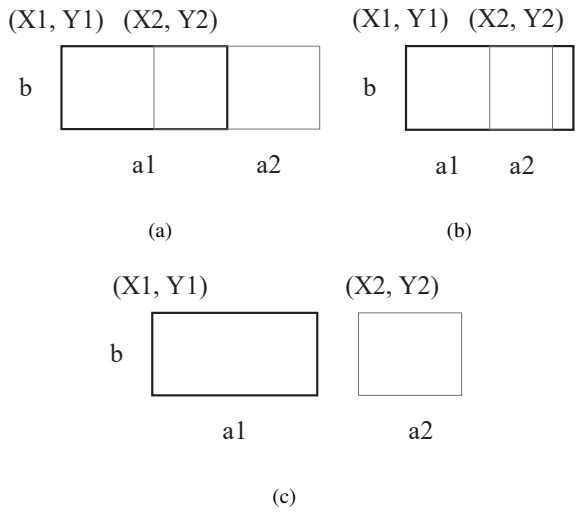


Fig. 6. Visual block merging rules

3.3. Similarity Comparison

The traditional ordered matching principle possesses many limitations. For example, the similarity of the two web pages cannot be objectively measured, and the deviation of the visual block in the web page causes errors. In addition, the similarity matching principle must have reflexivity and symmetry. Therefore, this study proposes an optimal free matching principle. For each visual block in the target web page, given a visual block with coordinates, length, and width within a certain threshold in the template web page, the blocks are considered similar.

The specific matching process is as follows:

1) Assume that (X, Y, L, W) represents the structure of each visual block, and X and Y are the abscissa and ordinate of the visual block, correspondingly, and L and W are the

respective length and width. The structure of the visual block of the target web page is (GX, GY, GL, GW), while that of the template web page is (TX, TY, TL, TW). Considering that two nodes are similar in position and the lengths and widths are only slightly different, the nodes are considered similar. In particular, when $|GX - TX|$, $|GY - TY|, |GL - TL|, |GW - TW|$ are within a certain threshold, the visual blocks are similar.

2) The nodes $\{GN1, GN2, GN3...\}$ in the target web page are compared with the nodes $\{TN1, TN2, TN3...\}$ in the template web page. When one node in the template web page is similar to that in the target page, the similar nodes can be added, and the comparison of the nodes is continued. The final number of similar nodes is denoted as SIM.
3) Assume that the total nodes of the target and template web page are G and T, respectively. In most cases, the total nodes of the target page are different from that of the template page. Therefore, the maximum values of G and T are taken. The similarity of the two web pages is denoted as $SIM/MAX(G, T)$.

4. Evaluation Result

The evaluation process consists of three parts. The first part is the evaluation of the extraction effect, which assesses if the extracted and merged visual blocks are visually consistent with the original web page. The second part evaluates the threshold of similar visual blocks. The similarity of two similar web pages (i.e., experimental objects) is compared under different thresholds. The threshold of the highest similarity is selected as the threshold of similar visual blocks. The last part examines the threshold of similar web pages, in which several similar and dissimilar web pages are regarded as datasets. The threshold with the lowest missing report and false alarm rates is taken as the threshold of similar web pages.

4.1. Extraction Effect

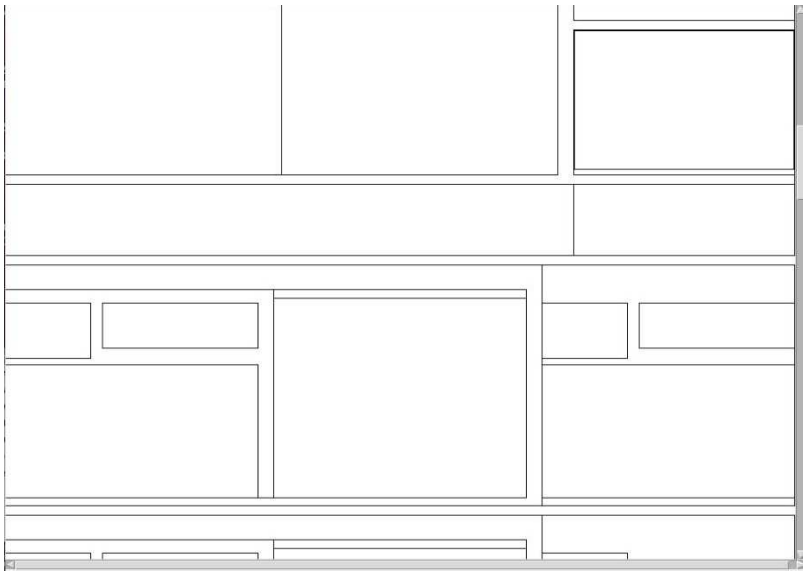
To verify if the extracted and merged visual blocks of the web page are visually consistent with the original web page, the drawing module TkInter is used to draw the visual block. This module is the interface of the standard Tk GUI toolkit and the built-in module of Python. The size of the web page is set as the size of the canvas. Then, the blocks in the visual block list are drawn on the canvas. The visual effect after drawing is compared with the original page, and the features of the web page determined by the algorithm are tested.

Fig. 7(a) presents a screenshot of Tencent. The web page is divided into blocks from the visual angle. The red line divides the web page into various visual blocks. Fig. 7(b) illustrates the extracted visual blocks and the distribution diagram. In this figure, the block division results of the web page from the visual angle and the results obtained by the algorithm are nearly the same. Therefore, the algorithm is validated.

4.2. Threshold of Similar Node

The visual block similarity threshold evaluation mainly assesses if two visual blocks are similar, and the threshold value when the web page similarity is the largest is considered the optimal result.

(a)



(b)

Fig. 7. Extraction effect result

Table 1. Visual block threshold testing table

Coordinate threshold	Length-width threshold	Similarity command
50	50	0.6216
50	60	0.6293
50	70	0.6293
...	...	...
190	180	0.9421

Table 1 shows a part of the data in the visual block similarity threshold test. When different coordinate and length/width thresholds are selected, the similarity degrees of the web pages vary. The similarity is low when the coordinate and length/width thresholds are small. The similarity degree also increases with the coordinate and the length-width thresholds. For example, when the coordinate threshold is 190 and the length/width threshold is 180, the web page similarity is the largest, that is, nearly reaching 95%. Therefore, 190 and 180 are taken as the corresponding coordinate and the length/width thresholds of the similar visual blocks.

4.3. Threshold of Similar Web Pages

The web page similarity threshold is determined as follows. Some similar and dissimilar web pages are selected as the target datasets. When the missing report and false alarm rates of similar web pages are at the minimum, the web page similarity threshold is considered the optimal value. The web pages of some post bars (e.g., Baidu post bar) are selected as the test object because of the similar typesetting. A total of 3000 URLs in the Baidu post bar are used as template web pages. These URLs are divided into 3 groups with 1000 URLs each. The final results are averaged to reduce the error. The test set includes some web pages of post bars and other web pages. If the web page is that of a post bar but is not considered similar to the template web page, then this web page is considered missing. If the web page is not a post bar web page but is regarded as similar to the template web page, then the web page is considered an error alarm.

During the comparison, the threshold of the similarity judgment starts from 0.35 and ends at 0.8, with an interval of 0.05, thereby yielding 10 similarity thresholds (Fig. 8). With the increase in the threshold, the false alarm and missing report rates decrease initially and then increase. When the threshold is 0.45, the error and miss rates obtain minimal results. Thus, 0.45 is selected as the threshold of the web page similarity. The minimal error and miss rates are 0.26 and 0.04, respectively.

Some popular phishing website detection methods, such as Kaspersky and IPDCM [21], are available; from which, we cannot publicly access codes. These methods may not achieve improved performance in this experiment because they use numerous webpage screenshots to obtain similarity with the true page and avoid text and structure similarity detections. In summary, the proposed algorithm is a more effective tool than other traditional tools in assessing and identifying phishing websites given its natural way of dealing with quality factors rather than exact values.

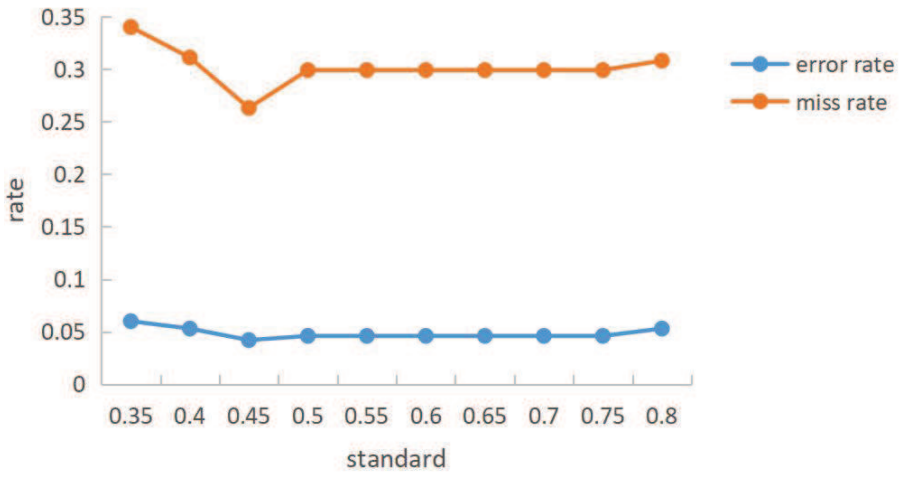


Fig. 8. Web pages similarity threshold testing result

5. Conclusion

This study investigates the visual block similarity of the web pages and proposes a new algorithm for visual block extraction and similarity comparison. The proposed algorithm analyzes the RenderLayer tree of the web page, extracts visual information by making the matching rules, determines the feature sets of the visual blocks corresponding to a web page by processing the visual nodes, and obtains similarity matching in accordance with the optimal free matching principle. After testing, the final number of the acquired visual nodes is approximately 5% of that of the original web page. In addition, to determine the threshold of the algorithm, numerous similar web pages are selected and tested under different threshold settings to achieve the optimal error and miss rates.

Acknowledgment This work was supported in part by the Key Research and Development Program for Guangdong Province (2019B010136001) and the National Key Research and Development Plan under Grant 2017YFB0801801, in part by the National Natural Science Foundation of China (NSFC) under Grant 61672186 and Grant 61872110. Professor Zhang is the corresponding author.

References

1. Alpuente, M., Romero, D.: A tool for computing the visual similarity of web pages. In: Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on. pp. 45–51. IEEE (2010)
2. Anari, Z., Anari, B.: Determining the similarity of web pages based on learning automata and probabilistic grammar. Advances in Computer Science: an International Journal 4(3), 44–53 (2015)

3. Artail, H., Fawaz, K.: A fast html web page change detection approach based on hashing and reducing the number of similarity computations. Data & Knowledge Engineering 66(2), 326–337 (2008)
4. Chiew, K.L., Chang, E.H., Tiong, W.K., et al.: Utilisation of website logo for phishing detection. Computers & Security 54, 16–26 (2015)
5. Cruz, I.F., Borisov, S., Marks, M.A., Webb, T.R.: Measuring structural similarity among web documents: preliminary results. In: Electronic Publishing, Artistic Imaging, and Digital Typography, pp. 513–524. Springer (1998)
6. Fu, A.Y., Wenyin, L., Deng, X.: Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd). IEEE transactions on dependable and secure computing 3(4), 301–311 (2006)
7. Goel, D., Jain, A.K.: Mobile phishing attacks and defence mechanisms: State of art and open research challenges. Computers & Security 73, 519–544 (2018)
8. Jain, A.K., Gupta, B.B.: Phishing detection: analysis of visual similarity based approaches. Security and Communication Networks 2017 (2017)
9. Joshi, S., Agrawal, N., Krishnapuram, R., Negi, S.: A bag of paths model for measuring structural similarity in web documents. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 577–582. ACM (2003)
10. Liu, W., Deng, X., Huang, G., Fu, A.Y.: An antiphishing strategy based on visual similarity assessment. IEEE Internet Computing 10(2), 58–65 (2006)
11. Marchal, S., Armano, G., Gröndahl, T., Saari, K., Singh, N., Asokan, N.: Off-the-hook: an efficient and usable client-side phishing prevention application. IEEE Transactions on Computers 66(10), 1717–1733 (2017)
12. Nichele, C.M., Becker, K.: Clustering web sessions by levels of page similarity. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 346–350. Springer (2006)
13. Pengyu, L., Lijun, J., Bin, J.: The research of the maximum length n-grams priority chinese word segmentation method based on corpus type frequency information. In: Proceedings Of The National Conference On Information Technology And Computer Science. pp. 71–74 (2012)
14. Roopak, S., Thomas, T.: A novel phishing page detection mechanism using html source code comparison and cosine similarity. In: Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on. pp. 167–170. IEEE (2014)
15. Satiabudhi, G., Andjarwirawan, J., Setiadi, R.S.: Web Page Similarity Searching Based on Web Content. Ph.D. thesis, Petra Christian University (2012)
16. Sonowal, G., Kuppusamy, K.: Phidma–a phishing detection model with multi-filter approach. Journal of King Saud University-Computer and Information Sciences (2017)
17. Srinivasa Rao, R., Pais, A.R.: Detecting phishing websites using automation of human behavior. In: Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security. pp. 33–42. ACM (2017)
18. Varshney, G., Misra, M., Atrey, P.K.: A phish detector using lightweight search features. Computers & Security 62, 213–228 (2016)
19. Varshney, G., Misra, M., Atrey, P.K.: A survey and classification of web phishing detection schemes. Security and Communication Networks 9(18), 6266–6284 (2016)
20. Wenyin, L., Huang, G., Xiaoyue, L., Min, Z., Deng, X.: Detection of phishing webpages based on visual similarity. In: Special interest tracks and posters of the 14th international conference on World Wide Web. pp. 1060–1061. ACM (2005)
21. Zhuang, W., Jiang, Q., Xiong, T.: An intelligent anti-phishing strategy model for phishing website detection. In: International Conference on Distributed Computing Systems Workshops (2012)

Xingchen Li received the BS degree in information security from Harbin Institute of Technology, Weihai, China, in 2016 and received Master degree in Computer Science and Technology from Harbin Institute of Technology, China, in 2018. His research interests include virtualization techniques for cloud computing and information security, etc.

Weizhe Zhang is corresponding author of this paper. He is currently a professor in the School of Computer Science and Technology at Harbin Institute of Technology, China, and director in the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China. His research interests are primarily in parallel computing, distributed computing, cloud and grid computing, and computer network. He has published more than 100 academic papers in journals, books, and conference proceedings. He is a senior member of the IEEE. Contact him at wzzhang@hit.edu.cn.

Desheng Wang received the BS degree in computer science and engineering from Harbin Engineering University, China, in 2015. He is currently working toward the PhD degree in the School of Computer Science and Technology, Harbin Institute of Technology. His research interests include virtualization techniques for cloud computing and machine learning, etc.

Bin Zhang received his Ph.D. degree in Department of Computer Science and Technology, Tsinghua University, China in 2012. He worked as a post doctor in Nanjing Telecommunication Technology Institute from 2014 to 2017. He is now a researcher in the Cyberspace Security Research Center of Peng Cheng Laboratory. He has published more than 30 papers in refereed international conferences and journals. His current research interests focus on network anomaly detection, Internet architecture and its protocols, network traffic measurement, information privacy security, etc.

Hui He is currently a full professor of network security center in the Department of Computer Science, China. She received the Ph.D. in department of computer science from the Harbin Institute of Technology, China. Her research interests are mainly focused on distributed computing, IoT and big data analysis. She is a member of the IEEE.

Received: September 15, 2018; Accepted: July 11, 2019.