Detecting Malicious Domain Name Based on the Web Page Structure Similarity

Xiaoyan Liu[1,a], Yue Shi[2,b], Yanan Cheng[1,c], Haiyan Xu[1,d], Zhaoxin Zhang[1,e]

[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
[2]China Academy of Information and Communication Technology, Beijing, China
a. hit_lxy6@163.com, b. shiyue1@catr.cn, c. mrcheng0910@gmail.com
d. grace3666@163.com, e. heart@hit.edu.cn

Keywords: Document object model, web page structure, Simhash algorithm, hierarchical similarity.

Abstract: In order to detect the malicious domain name accurately, a method of detecting the malicious domain name based on the similarity of web page structure of Web document object model is proposed. The key is how to calculate the hierarchical similarities among web page structures quickly and effectively. This method first obtains the source code of the domain name and analyzes its DOM tree structure, constructs the DOM tree level tag attribute name sequence to describe the characteristics of the domain name's web page structure, and then defines the DOM tree distance based on the idea of Simhash algorithm to measure the similarity between the web page structures. Experiment shows that the method can detect the similarity of domain page structure effectively with high accuracy and recall rate.

1. Introduction

In recent years, the continuous growth of all kinds of malicious domain names has posed a huge threat to the personal privacy, property security, physical and mental health of users, even national security and social stability. The existence of malicious domain names has seriously hindered the healthy development of the Internet. Therefore, how to monitor and detect malicious domain names and control them in time effectively has become a hot issue in the field of network security.

Most of the existing researches based on the similarity of malicious domain name are based on the similarity of web content, using the URL or web page content as the feature for detection. Tf. idf algorithm is used to extract 5 keywords of the page, using keywords to see if the domain name of the webpage is within the top 30 results returned by Google search to detect phishing page[1].A fast filtering tool based on machine learning for malicious code detection of web pages by using static recognition technology, content of web pages hmtl, JavaScript code, and URL information is proposed in[2]. Reference[3] explored the possibility of utilizing confidence weighted classification combined with content based phishing URL detection to produce a dynamic and extensible system for detection of present and emerging types of phishing domains. Reference[4] extracted features

from the URL, textual content, structural tags, page links, and visual appearance of a Web page, they experimented with batch models of classification that are trained in an offline setting to identify malicious Web pages. However, in the actual situation, in order to achieve the purpose of low-cost, rapid and large-scale generation of malicious domain names, malicious domain name registrants will register a large number of domain names with different but similar web page structure. As a result, in recent years, there have been a lot of researches on malicious domain name detection methods based on DOM tree structure corresponding to HTML documents.The first method to measure tree similarity is the edit distance (ED) algorithm proposed by Tai[5]. Later, Joshi and others proposed a method to measure the similarity of DOM tree based on the path matching of DOM subtree[6]. Great progress has also been made in the application of DOM tree structure similarity to malicious domain name detection. For example, Rosiello et al. first used DOM tree layout similarity comparison to alert phishing web pages,they mainly compared the tags of two web pages and extracted the isomorphic tree subgraph of DOM tree[7]. Reference[8] presented an approach which calculates similarity of two web pages based on genetic algorithm and applied it to detecting phishing web pages using DOM-Tree structure. Cui, Qian and others proposed a PD algorithm based on Hamming distance to measure the similarity of two DOM trees by defining the tag name vector in advance[9].

In this paper, we propose a method to detect the malicious domain name based on the hierarchical similarity of the document object model corresponding to the domain name. Compared with the traditional technology of malicious domain name detection using page content, this method does not need to extract a large number of features, which simplifies the difficulty of feature analysis,avoids the problem of feature change and considers the hierarchy of web page structure. Also, compared with the other researches using DOM tree structure, we are the first who use the idea of Simhash algorithm to propose an algorithm (HS algorithm) to calcuate the distance of the web page structure among domain names, which can effectively reduce the running time.

2. Method

The detection of malicious domain names can be based on the similar structure of web pages. The DOM tree, which is the document object model corresponding to HTML, can reflect the hierarchical structure of web pages well. In addition, a large number of domain names with different domain names but similar web page structure exists in cyberspace. Based on these observations, first of all, we must consider the hierarchy of DOM tree when calculating the similarity of web page structure. Secondly, we should keep the text information of web page hierarchy to the greatest extent.Lastly, we simplify the process of comparison when calculating and comparing the similarity. Therefore, this paper proposes the extraction method based on the hierarchical structure features of DOM tree, defines the measurement method of DOM tree similarity, calculates the similarity among the domain name of the type to be detected and the web page structure of the domain name collection of the known type to achieve the purpose of malicious domain name detection. The main flow of the system is shown in Figure 1.

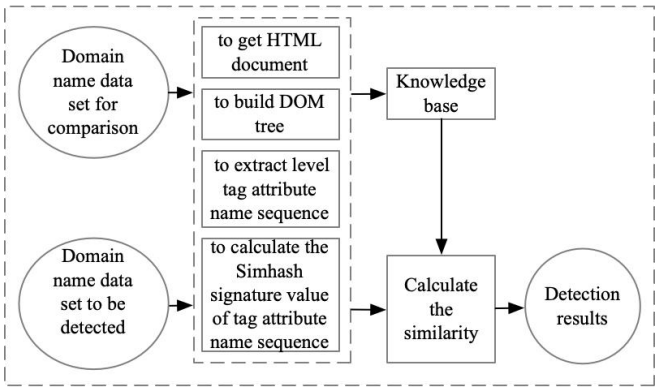

Figure 1: System flow.

2.1.Construct Tag Attribute Name Sequence

HTML document is a semi-structured text markup language, and DOM is the document object model, which defines the logical structure of HTML document and XML document, and gives a method to access and process HTML document and XML document. Therefore, the domain name web page structure can be represented by a hierarchical and orderly DOM tree. Each node in the DOM tree corresponds to an element in the document, and each edge corresponds to the parent-child relationship between two nodes. Because the DOM tree itself represents the web page structure, the information of each node in the DOM tree structure can be extracted and the complete node information of the DOM tree can be preserved. This paper does not only considering that the tag name of each node of the DOM tree but also includes all the attribute names of the node, we can construct a text feature sequence representing the DOM tree hierarchy and store it in the database. The specific construction steps are as follows:

- Step 1: Get the content of the HTML document of the domain name web page, and use the third-party Python library to parse the HTML document to construct the corresponding DOM tree.
- Step 2: Start from the current layer, the breadth first traversal algorithm is used to traverse each node of the DOM tree layer by layer from left to right. In the process of traversing each node of the DOM tree, the tag names and all attribute names of the nodes currently traversed by the current layer are extracted one by one and stored in the database table corresponding to the current layer number.
- Step3: Repeat Step 2 until the required number of layers m is reached, stop traversing, and obtain the text sequence table composed of tag name and attribute name of each layer node in the DOM tree structure of the domain name web page, including the layer where the body is located to the required layer.
- Step 4: The number of the layer where the body node is located as 0, and the text sequence of all nodes contained in each layer from the layer where the body is located to the layer whose number is m is extracted to form the text feature sequence representing the DOM tree structure.

Because the page information is mainly within the body node, this paper does not consider the information of each node in the head subtree, and extracts the information of the nodes including the body node and the nodes within it. In addition, through the analysis of the DOM tree structure, it is found that the shallow nodes of the tree have great influence on the structure of the web page: if the structure of the web page is similar, the information of the shallow nodes of the DOM tree will be similar, but with the increase of the level, the information difference of the nodes of the DOM tree will increase, so the number of layers to be studied in this paper is not all the layers of DOM

tree, but the number of layers that can achieve the purpose of similarity research, reducing the interference of the deep node information on the construction of text feature sequence. For example, Figure 2 is an HTML document and its corresponding DOM tag attribute name tree, and Table 1 is its corresponding hierarchical tag attribute name sequence.

2.2. Detect Malicious Domain Name

First of all, the main idea of this paper is to use DOM tree hierarchy to construct the text feature sequence of node information, and use the similarity of text feature sequence to measure the similarity of DOM tree structure. Secondly, based on the main idea of the Simhash algorithm proposed by Charikar[10], the similarity algorithm between DOM tree structures is proposed based on hierarchical distance. The key of the algorithm is to calculate the hierarchical similarity according to the attribute information of the node tag in the hierarchy of DOM tree. Using the Simhash algorithm, the content of the text feature sequence is transformed into n-bit signature, and the similarity of the original feature sequence is calculated by comparing the similarity of the signature. Finally, the distance between the two DOM trees is obtained by multi-layer accumulation. Using the idea of Simhash algorithm to calculate the signature value corresponding to the sequence of text features mainly means that by grouping each text data to calculate the hash value of each group of features respectively, and the n-bit signature value of each text is finally determined.

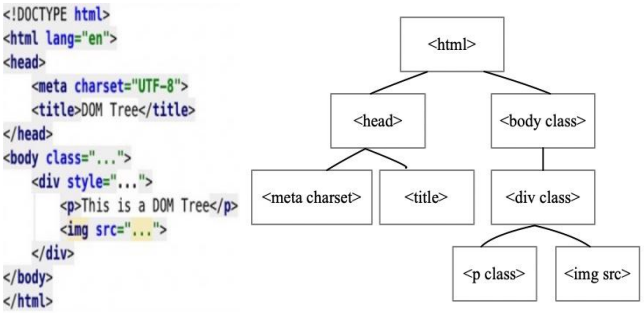

Figure 2: HTML document and DOM tree example.

Table 1: Example of tag attribute name sequence.

The Layer number	The tag attribute name sequence
1	bodyclass
2	divsclass
3	pclass imgsrc

The related variables and definitions of the algorithm are given below:

D_1, D_2 represent two DOM trees, and $D\_F_1=\{F^1_1, F^1_2, \ldots, F^1_i, \ldots, F^1_m\}$, $D\_F_2=\{F^2_1, F^2_2, \ldots, F^2_i, \ldots, F^2_m\}$ represent the set of text feature sequences at the level of two DOM trees, in which F^1_i and F^2_i are the i-th level text feature sequences of trees D_1 and D_2 respectively. $D\_S_1=\{S^1_1, S^1_2, \ldots, S^1_i, \ldots, S^1_m\}$, $D\_S_2=\{S^2_1, S^2_2, \ldots, S^2_i, \ldots, S^2_m\}$ are sets of signature values corresponding.

- The difference value of the i-th layer of the two DOM trees D_1 and D_2 is defined as the Hamming distance between S^1_i and S^2_i. First, the Hamming distance of the x-th position of S^1_i and S^2_i is defined as follows:

$$\mathrm{Ham\_Dis}(S^1_{i\_x},S^2_{i\_x}) = \begin{cases} 1, S^1_{i\_x} = S^2_{i\_x} \\ 0, S^1_{i\_x} \neq S^2_{i\_x} \end{cases} \quad (1)$$

Then, the Hamming distance between S^1_i and S^2_i is:

$$\mathrm{Ham\_Dis}(S^1_i,S^2_i) = \sum_{x=1}^{n} \mathrm{Dis}(S^1_{i\_x},S^2_{i\_x}) \quad (2)$$

- The total difference value of the two DOM trees D_1 and D_2 is defined as the sum of the difference values of all m layers:

$$\mathrm{Ham\_Dis}(D\_S_1,D\_S_2) = \sum_{i=1}^{m} \mathrm{Ham\_Dis}(S^1_i,S^2_i) \quad (3)$$

- The distance value of the two DOM trees D_1 and D_2 is defined as follows:

$$d(D_1,D_2) = \frac{\mathrm{Ham\_Dis}(D\_S_1,D\_S_2)}{m*n} \quad (4)$$

The pseudo code description of HS algorithm how to calculate tree distance is as follows:

Input: D_S_1, D_S_2

Output: $d(D_1,D_2)$

```
Begin:
01.    def HS(D_S1, D_S2):
02.    while i≤ m:
03.    while x≤ n:
04.    calculate Ham_Dis(S^1_i_x,S^2_i_x)
05.    calculate
     Ham_Dis(D_S1,D_S2)
06.    calculate d(D1,D2)
07.    return d(D1,D2)
End
```

In the above description, n represents to serialize the attribute names of each layer of the DOM tree of the domain name into 64 bit values by using the idea of the simhash algorithm, and m represents the number of layers selected in the research from the body label layer. It can be seen that the time complexity of the algorithm is O (m*n).

From equation (4), it can be seen that the larger the distance value $d(D_1,D_2)$ between two trees is, the more dissimilar their web page structure is.

- For the minimum value of the tree distance between a domain D to be detected and all malicious domain names in the knowledge base:

$$\mathrm{Min\_d(D)} = \min\{d(D,D_1),\ldots,d(D,D_i),\ldots,d(D,D_N)\} \quad (5)$$

The determination of the type of domain name to be detected is mainly based on formula (5). Specifically, when the min $Min\_d(D)$ of a domain name to be detected is less than the threshold V, the domain name can be determined as a malicious domain name.

3. Experiment

In order to verify the feasibility and effectiveness of the proposed hierarchical similarity detection method based on domain page structure, this paper mainly implements the detection method through Python language and evaluates the detection method in this paper through the results of the malicious detection of domain name in experiments.

3.1.Experimental Datasets

All the web page datasets used in the experiment come from the real network environment. The legal domain name data set is from Alexa. Alexa is a website that specially publishes the world rankings of websites maintained by Amazon. In this experiment, we collected the top ten thousand domain names from Alexa website that can be accessed normally and are legal websites.

Malicious domain name datasets are mainly gambling and pornographic domain names. 12000 malicious domain names that can be accessed normally are collected from the third-party authoritative detection website.

3.2.Indicators for Evaluation

In this paper, TPR, FPR and $Precision$ are selected as evaluation indexes. Their calculation formulas are as follows:

$$TPR = \frac{TP}{TP+FN} \tag{6}$$

$$FPR = \frac{FP}{FP+TN} \tag{7}$$

$$Precision = \frac{TP}{TP+FP} \tag{8}$$

In the formula, TPR represents recall rate, characterizes the ratio of positive instances of malicious domain names identified by the detector to positive instances of all malicious domain names, FPR represents the rate of misjudgment, characterizes the ratio of negative instances of legitimate domain names mistakenly identified by the discriminator to negative instances of all legal domain names, Precision represents the accuracy rate, which describes the proportion of positive instances of malicious domain names recognized by the discriminator to all positive instances of malicious domain names recognized by the discriminator.

3.3.Experiment I

The purpose of this experiment is to verify the feasibility of the proposed method. Use the system flow shown in Figure 1 to process the domain name data set in the collected black-and-white list. Based on the following two criteria:

- the page information is mainly within the < body > tag, and the < head > tag does not contain the main information of page structure.
- if the structure of the web page is similar, the information of the nodes in the shallow layer of the DOM tree is similar, but with the increase of the level, the information difference of the nodes in the DOM tree will increase.

Therefore, this experiment selects eight layers of node information from the body to construct the text feature sequence of each layer. At the same time, when using the Simhash algorithm to

calculate the n = 64 bit signature value of the text feature sequence, the text feature sequence is grouped according to the interval of p = 3. In this experiment, 2000 domain names are randomly selected from the domain name data set as the target domain name set to be detected for maliciousness.

It can be seen from Table 2 and Figure 3 that when the threshold value is 0.1, TPR and FPR reach a good compromise of 88.65% and 1.34%, respectively. The higher the threshold value, the lower the precision is mainly caused by the increase of misjudged web pages.

Table 3 shows the evaluation indexes and running time results of extracting DOM tree information of different level under the optimal threshold of 0.1. From Table 3, it can be seen that too few layers will lead to higher FPR and lower precision; too many layers will reduce TPR and increase system operation time.

Table 2: Experimental results under different threshold.

Threshold	TPR%	FPR%	Precision%
0.05	81.79	0.41	98.77
0.1	88.65	1.34	97.82
0.15	89.84	10.26	89.28
0.2	93.26	35.81	74.52
0.25	95.37	38.69	70.13

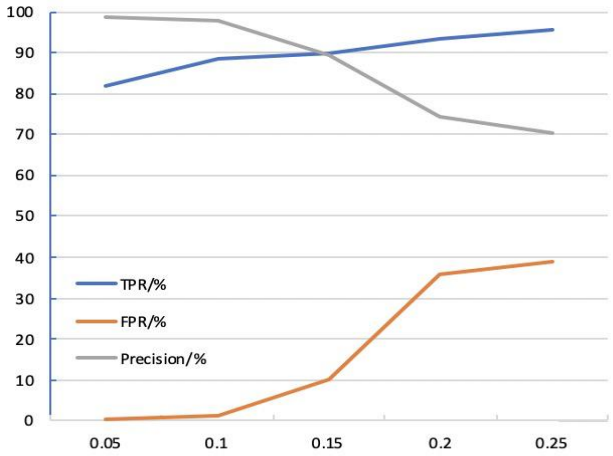

Figure 3: Experimental results under different threshold.

Table 3: Experimental results under different levels.

Level	TPR/%	FPR/%	Precision%	Time/s
6	88.97	10.58	86.24	1242
7	89.53	8.73	89.82	1458
8	88.65	1.34	97.82	1569
9	86.72	2.47	97.98	1698
10	84.25	1.28	96.13	1756

3.4.Experiment II

In order to test the detection performance and efficiency of this method, this section of experiment compares the HS algorithm proposed in this paper with other DOM tree similarity algorithms, including: ①edit distance algorithm (ED algorithm); ②tag attributes algorithm (PD algorithm). Table 4 shows the detection results of the optimal threshold value corresponding to each algorithm after using different DOM tree similarity algorithms in the system.

Histogram is used to compare the TPR and FPR values of different algorithms. It can be seen from Table 4 and Figure 4 that HS algorithm has the optimal detection effect. HS algorithm has higher TPR. The reason is that HS algorithm not only compares tag sequences, but also considers the attribute names and hierarchy of tags which is better than ED algorithm in this respect; PD algorithm only evaluates the occurrence rate of tags, so TPR is not too high; ED algorithm uses one-to-one comparison of tree nodes, which contains noise nodes, so TPR is low.

The runtime of different algorithms is shown in Figure 5. HD algorithm has the shortest running time. The runtime of HS algorithm is lower than that of ED algorithm. This is because the ED algorithm considers all levels of tree information. With the increase of tree depth, the complexity of label comparison is increasing. At the same time, the operations such as inserting, deleting and replacing of ED algorithm are also need a lot of time, which is not suitable for the comparison of a large number of datasets.

Table 4: Detection results of different algorithms.

Algorithm	Threshold	TPR/%	FPR/%	Time/s
HS	0.1	88.65	1.34	1569
ED	0.15	86.57	2.87	6986
PD	0.1	86.33	1.76	3305

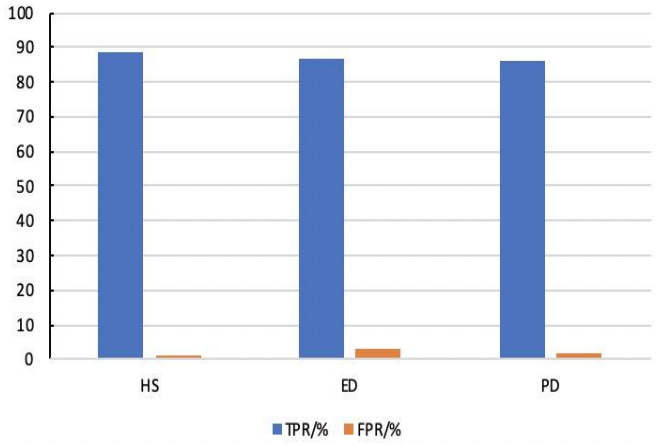

Figure 4: Comparison of TPP and FPP by different algorithms.

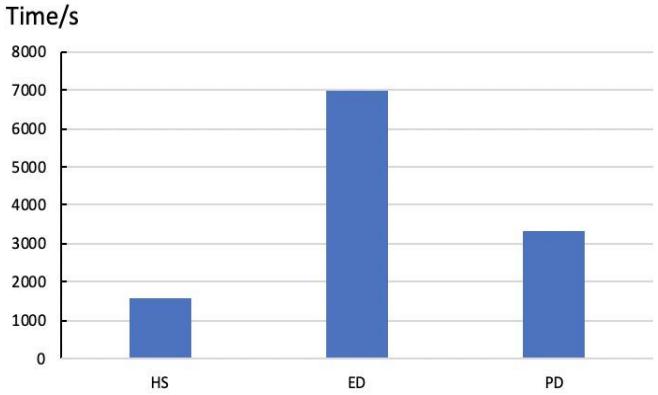

Figure 5: Runtime of different algorithms.

4. Conclusions

In this paper, we use the characteristics of the similar web page structure of the malicious domain name, and construct the text features that can represent the DOM tree hierarchy by using the tag attribute name sequence of the DOM tree, we calculate the similarity of the DOM tree tag attribute name sequence between the domain name to be detected and the name of the known malicious type. Our experimental evaluation demonstrates that our solution is feasible. The experimental results show that this method is feasible with high recall and accuracy. The HS algorithm proposed in this paper can effectively solve the problem of DOM tree similarity measurement.

Acknowledgments

This work is supported by the National Key Research & Development Program of China under Grant No.2018YFB1800202, No.2017YFB0803001.

References

[1] Yue Zhang, Jason I. Hong, and Lorrie F. Cranor, "Cantina: A content-based approach to detecting phishing web sites," In Proceedings of the 16th international conference on World Wide Web, Association for Computing Machinery, New York, 2007, pp. 639-648.

[2] Davide Canali, Marco Cova, Giovanni Vigna, and Christopher Kruegel, "Prophiler: A fast filter for the large-scale detection of malicious web pages," In Proceedings of the 20th international conference on World wide web, Association for Computing Machinery, New York, 2011, pp. 197–206.

[3] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner, "Lexical feature based phishing URL detection using online learning,", In Proceedings of the 3rd ACM workshop on Artificial intelligence and security (AISec '10), Association for Computing Machinery, New York, 2010, pp. 54-60.

[4] Sushma Nagesh Bannur, Lawrence K. Saul, and Stefan Savage, "Judging a site by its content: learning the textual, structural, and visual features of malicious Web pages," n Proceedings of the 4th ACM workshop on Security and artificial intelligence (AISec '11), Association for Computing Machinery, New York, 2011, pp. 1-10.

[5] Kuo-Chung Tai, "The Tree-to-Tree Correction Problem," Association for Computing Machinery, vol. 26, 1979, pp. 422–433.

[6] Sachindra Joshi, Neeraj Agrawal, Raghu Krishnapuram, and Sumit Negi, "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 (KDD '03), Association for Computing Machinery, New York, 2003, pp. 577-582.

[7] A. P. E. Rosiello, E. Kirda, 2. Kruegel and F. Ferrandi, "A layout-similarity-based approach for detecting phishing pages," 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007, Nice, France, 2007, pp. 454-463.

[8] Le Dang Nguyen, Dac-Nhuong Le, and Le Trong Vinh, "Detecting phishing web pages based on DOM-tree structure and graph matching algorithm," In Proceedings of the Fifth Symposium on Information and Communication Technology (SoICT '14), Association for Computing Machinery, New York, 2014, pp. 280-285.

[9] Qian Cui, Guy-Vincent Jourdan, Gregor V. Bochmann, Russell Couturier, and Iosif-Viorel Onut, "Tracking Phishing Attacks Over Time," In Proceedings of the 26th International Conference on World Wide Web, Association for Computing Machinery, New York, 2017, pp. 667-676.

[10] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma, "Detecting near-duplicates for web crawling," In Proceedings of the 16th international conference on World Wide Web, Association for Computing Machinery, New York, 2007, pp. 141-150.