Dynamic document object model formation technique for corporate website protection against automatic coping of information

Vasily Galushka[1,*], *Daniil* Marshakov[1], and *Vladimir* Fathi[1]

[1]Don State Technical University, Computer systems and information security department, 344000 Gagarina sq. 1, Rostov-on-Don, Russian Federation

Abstract. The article describes solution path of the problem of information automatic copying from web-sites in the Internet, which is implemented using parsing techniques based on regular expressions or function libraries. To protecting against this type of information security threat, it is proposed to dynamically generate and periodically change the object model of the HTML document when generated and sent to the browser. These changes should affect the values of the identifying tag attributes and the structure of the object model tree. As attribute values it is offered to use character sets of limited length obtained as a result of random numbers hashing; change of the structure of the object model should be done by adding of additional tags at the corresponding levels of the hierarchy of the tree representing it. The simultaneous application of these methods excludes the possibility of algorithm compile for the necessary information extraction in the overall structure of the web page.

1 Introduction

Currency, that is the correspondence to the current moment of time, is an inalienable property of any information, however, in some cases it acquires special significance. This is the case of information available on the websites on the Internet and has value for short time interval, for example, news or stock-exchange quotations short-term forecasts. In addition to standard security threats, such websites are subject to another kind of negative impact - automatic copying of information, carried out with the hardware and software help without human intervention, with minimal resources, often in real time. Such process is called parsing [1-5] and is realized on the basis of the regular expressions [5-6], or with specialized program functions libraries use [7].

Notwithstanding that the information resources above as an example are objects of intellectual property and should be protected by copyright law, unfortunately, the practice shows [8] the complexity of legal methods applying at Internet security providing, as well as epy problems of author rights attribution and confirming, which leads to the need to develop technical and software security tools.

2 Parsing

The methods of information automatic copying on web-sites are based on the syntactic and semantic analysis of the HTML- page source code. This is due to the fact that access to any other information can not be obtained from outside, at least when proper security tools used by the site, web server or hosting administrator. For technique of protection against automatic copying development it is necessary to establish the reasons and vulnerabilities allowing to make this breach of website's security, and for this purpose to analyze of HTML language features and to reveal those whichever allow to realize copying of the information contained on web pages with software and without human involvement.

It is obvious that it's impossible to do without the human involvement in this process, so the phrase "without the human involvement" means only the process of direct data transfer from the site-victim to the site of the attacker at the user request. In this case, human intervention at the stage of parsing system setting up is necessary. Human intervention functions includes manual analysis of the structure of the HTML code, definition of blocks with useful information and their separation from the rest of the document, as well as cleaning of the content from the tags that specify the markup, because on the website on which information is copied, the markup can, and, most likely, will be completely different. Then, based on the analysis, parsing rules will be compiled and written in the programming language, most often using any library that facilitates the work with the pages source code.

It is important that these actions, although they are rather difficult and require high qualification of the person performing them, but are executed only once — at a stage of website creation. At the same time, there is

[*] Corresponding author: galushkavv@yandex.ru

no method to hide the HTML code from such an attacker, since in compliance with the standards it must be transmitted as the clear text to the browser. It follows from the HTML language assignment — the markup of a document is its main function, that is, document dividing into parts or blocks, each of which must be logically separated from the rest and have all the necessary information about the final representation of its contents, as well as the content itself. The HTML language does not provide any built-in protection. It is assumed that if the information on the sites is intended for open access, then it makes no sense to complicate both the language itself, and the means of its interpretation by additional information security features.

Regardless of the implementation method, parsing uses the object model (DOM) of the HTML document to search for the information of interest to it and to separate the content from the design defined using HTML tags and their attributes [9-11]. Therefore, protection from this type of information security threats should be based on changing objects included in the DOM, in order to make difficult, and ideally impossible, the algorithmic analysis of the object model by software.

The object model is a universal and generalized document structure description and has the form of tree whose the tags are nodes of a tree, and the branches are the connections between them [12]. The document object model is built by the web browser on the analysis of the HTML page source code and serves as an internal intermediate representation formed prior to direct draw of the page

The simplest and, the most frequently parsing quite effective way is the search for the tag by its attributes: "id" and, more rarely, "class". These attributes allow the most exact identification of the tag containing the required information and are present at any HTML-design template, and their values are associated with the corresponding identifiers in the CSS file for use of the styles described in it. To eliminate the possibility of program creation capable to execute search of the tag by these attributes, it is necessary to assign random values to them at each page load.

3 Object model automatic generation with dynamic attributes use

In accordance with the CSS standard [13], the first characters of identifiers can not be the numbers or special characters such as commas, colons and the like, and they are usually written as text values. However, the random number generator in PHP, as well as in any other programming language, is focused on the numbers formation only. The MD5 hashing use to text formation from numbers is proposed. Although this algorithm already some time is not considered reliable and strong [14-15], its application for this problem is justified, since the hashing algorithm functions, in this case, reduce only to the transformation of values randomly provided by the random number generator [16]. Since the algorithm MD5 generates an alphabetic and numerical sequence of 32 characters in length, 2 problems arise: to shorten the sequence length and to exclude the digits appearance in the beginning of sequence.

To solve these problems, the names formation will be made in accordance with the following rule:

$name := prefix + substring_{0,5}(md5(rand(0, PHP\_INT\_MAX)))$, where

- *prefix* — text constant, defined for the site, in whole;
- $substring_{k,n}$ — is the function of part of a string selection starting with the k-th position with the length of n symbols;
- *md5* — is the function of the hash sum computing;
- $rand(l,m)$ — is the function generating a random number in the range from l to m;
- *PHP_INT_MAX* — the predefined constant, meaning the maximum integer value supported by this PHP version.

The prefix presence allows to prevent the use of the value starting with the digit resulting from the hashing as the identifier, and the assignment of a 5-character substring (each of which can be one of 26 letters of the English alphabet or one of 10 digits) will provide $(26 + 10)^5$ = 60 466 176 different versions of identifiers, which is enough for most sites, and, this amount can be increased, if necessary.

The software implementation of this technique can be divided into several stages.

1. Separation of the CSS file into static and dynamic parts
2. Random values generation for properties of tags that meet the requirements for CSS-identifiers.
3. The received values writing to a file with the dynamic part of the CSS.
4. The received values assignment to tags with information to be protected.

In this form the algorithm is applicable at relatively small load on the site, since, with a large number of access to the site, CSS file overwriting will be performed too often and can lead to the fact that the page with one set of identifiers of tags will be sent to the user, at this time another request will come, new random values not corresponding to those sent will be written to the file. In this regard, the presented algorithm should be applied separately for each user, the names of CSS-files should be stored in the database, and the files themselves should be dynamically generated on request.

4 Dynamic object model automatic formation based on a change in document tree structure

The above described method does not provide protection against tag search by its position respectively to other tags. So for DOM tree, shown in Figure 1 as an example, the conditional path to the contents of the first paragraph would be written without any attributes use in the form of the following parsing rule:

$\text{HTML} \rightarrow \text{BODY} \rightarrow \text{P}_1$, where

P_1 — is the first <P> tag from the set of all tag data.

In general, the rule can be written as:

$\text{Root} \rightarrow ... \rightarrow \text{Parent} \rightarrow \text{Child} \rightarrow ... \rightarrow \text{Element}_i$

This record realizes the passage through the tree in depth and is the fastest way to access to the tree finite elements (leaves), which, as applied to the structure of the HTML document, usually contain all the useful information [17-18].

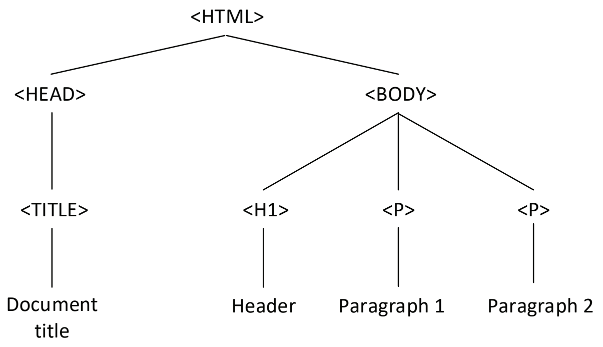

Fig. 1. Document object model tree.

The possibility of such calls to the content for its automatic copying can be prevented by structure of the DOM changing by adding of some random number of additional <P> tags before the information in which it is necessary to be protected. Thus, in the object model tree, a new "extra" element will be added to the same hierarchy level as the protected one. We will designate such approach as "increase in width of tree" (Figure 2).

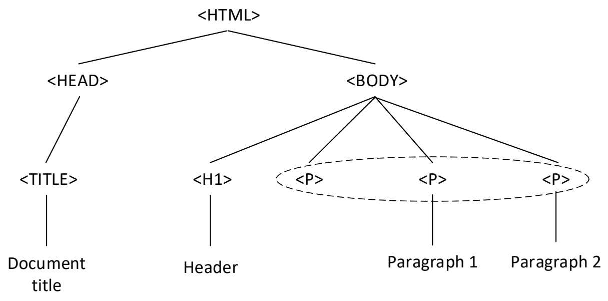

Fig. 2. Increase in width of tree.

Another approach — increase in depth of tree, involves the new nesting levels adding to the HTML document [19-20] block structure i.e., new hierarchy levels to the DOM tree adding (Figure 3).

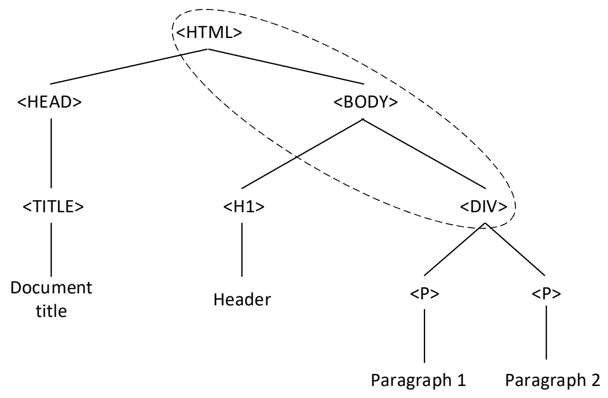

Fig. 3. Increase in depth of a tree.

It is important to correctly determine the branch of the tree to which the elements will be added, as well as their number and set of properties, and the properties of the added elements should not allow them to be distinguished from the real ones in the process of parsing. Let's consider the listed conditions in order.

When determining the tree branch, it should be guided by the following considerations:

– it should be on the same tree level as the node that contains protected information

– branch can be one level above the node with protected information, provided that it includes one of the ancestor node

The next parameter is the number of added elements, which should be a random value. This will allow to complicate parsing having excluded a possibility of path instructions on the object model tree to required element The best option is to add from 1 to 3 additional elements what on the one hand ensures that the tree will be changed at least by 1 element, and not too complicate the structure of the document and will not lead to its distortion by the browser, on the other hand.

In the process of implementation of these methods we must not forget that changes in the structure of the document must not affect its display in the browser and the user's perception, which imposes certain restrictions on the set of admissible changes. So, for example, already considered the <P> tag adding may cause too much padding for the following paragraph, and adding of the <DIV> tag can lead to a complete change in the page appearance. To avoid such situations add tags CSS-property "visible" with a value of "false" or property "display" with value "none" should be set.

5 Conclusion

The combination of the described methods of the document dynamic object model development one of which is based on use of accidental values of identification attributes of tags, and the second on change of the by increase in its width and depth, deprives of the malefactor of an opportunity to formulate and write an algorithm of website pages parsing for separation in them significant units of up-to-date information that in turn makes impossible its automatic copying.

References

1. J. Tidal. The Code4Lib Journal, no.18 (2012)
2. P. Patil, P. Chawan, P. Chauhan. International Journal of Advanced Research in Computer Engineering & Technology, no.1 (2012)
3. H. Kim, K. Doh, D. Schmidt. *Lecture Notes in Computer Science: 20th International Symposium*, 7935 (2013)
4. Z. Zhao, M. Bebenita, D. Herman, J. Sun, X. Shen. ACM Transactions on Architecture and Code Optimization, no.4 (2013)

5. Y. Han, S. Oh. Journal of the Korean Society for information Management, no.2 (2010)
6. S. Levithan, J. Goyvaerts *Regular Expressions Cookbook, 2nd Edition* (O'Reilly Media, Sebastopol, 2012)
7. Z. Chun-lin, S. Fang-hao. Computer Programming Skills & Maintenance, no.12 (2014)
8. G. Kunanbaeva. Bulletin of the Kazakh-American free university, no.4 (2009)
9. C. Lindley *DOM Enlightenment: Exploring JavaScript and the Modern DOM*, (O'Reilly Media, Sebastopol, 2013)
10. R. Győrödi, C. Győrödi, G. Pecherle, G. Cornea. WSEAS Transactions on Computers, no. 9 (2010)
11. S. Jensen, M. Madsen, A. Møller. *In proc. 19th ACM SIGSOFT symposium* (2011)
12. Document Object Model (DOM) Technical Reports, available at: https://www.w3.org/DOM/DOMTR
13. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, available at: https://www.w3.org/TR/2011/REC-CSS2-20110607/.
14. S. Kumar, E. Gupta. International Journal of Computer Science and Information Technologies, no. 5 (2014)
15. X. Zheng, J. Jin. *9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD) (*2012*)*
16. J. Trampert, A. Fichtner, J. Ritsema. Geophysical Journal International, no. 2, (2013).
17. L. Sang-Un. Journal of the Korea Society of Computer and Information, no. 7 (2014)
18. Z. Tao. Journal of Chinese Computer Systems, no.1 (2014)
19. PHP Documentation, available at: http://php.net/manual/en/function.rand.php
20. HTML 4.01 Specification, available at: https://www.w3.org/TR/1999/REC-html401-19991224/