Detecting Phishing Web Pages based on DOM-Tree Structure and Graph Matching Algorithm

Le Dang Nguyen  
Haiphong University  
Haiphong, Vietnam  
Nguyendl@hus.edu.vn

Dac-Nhuong Le  
Faculty of Information Technology  
Haiphong University, Vietnam  
Nhuongld@hus.edu.vn

Le Trong Vinh  
Hanoi University of Science  
Vietnam National University, Hanoi, Vietnam  
Vinhlt@vnu.edu.vn

ABSTRACT
Most modern day phishing attacks occur by luring users into visiting a malicious web page that looks and behaves like the original. Phishing is a web-based attack which end users are lured to visit fraudulent websites and give away personal information unconsciously. The key problem for checking phishing pages is timely and efficiently calculation. There is still remaining a large space for phishing detection methods. In this paper, we present 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. Our experimental evaluation demonstrates that our solution is feasible in practice.

Keywords
Phishing Detection, Graph Matching, Genetic Algorithm, DOM-Tree, Tree Matching.

1. Introduction
Phishing [1] is the act of convincing users to give up critical personal information, either through conversation or some form of content manipulation. Most modern day phishing attacks occur by luring users into visiting a malicious web page that looks and behaves like the original. Once the user, if convinced that the page is authentic, may give up private information including authentication credentials or banking information. This information is typically used to commit some form of identity theft or fraud. The phishing techniques are classified as follows: Spoofed e-mails are sent to a set of victims asking them (usually) to update their passwords, data account, etc. MSN, ICQ, AOL and other IM channels are used to reach the victims. Social engineering techniques are used to gain victim’s sensitive information. Calling the victims on the phone, classic social engineering techniques are used by phishers. Another kind of attack is based on the internet browser vulnerabilities. This approach is usually adopted to automatically install dialers. In a typical attack, the phisher sends a large number of spoofed (i.e. fake) e-mails to random Internet users that seem to be coming from a legitimate and well-known business organization (e.g. financial institutions, credit card companies, etc). The e-mail urges the victim to update his personal information as a condition to avoid losing access rights to specific services (e.g. access to online bank account). By clicking on the link provided, the victim is directed to a bogus web site implemented by the attacker. The phishing website is structured as a clone of the original website so that the victim is not able to distinguish it from that of the service he/she has access to. There are many different features from various websites. Table 1 shows the phishing factor indicator.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Layer</th>
<th>Layer</th>
<th>Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Component</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>URL &amp; Domain Identity</td>
<td>1</td>
<td>Using the IP Address</td>
<td>Layer One</td>
</tr>
<tr>
<td>(Weight = 0.3)</td>
<td>2</td>
<td>Abnormal Request URL</td>
<td>Sub weight = 0.3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Abnormal URL of Anchor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Abnormal DNS record</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Abnormal URL</td>
<td></td>
</tr>
<tr>
<td>Security &amp; Encryption</td>
<td>1</td>
<td>Using SSL certificate</td>
<td>Layer Two</td>
</tr>
<tr>
<td>(Weight = 0.2)</td>
<td>2</td>
<td>Certification/authority</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Abnormal Cookie</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Distinshed Names Certificate(DN)</td>
<td></td>
</tr>
<tr>
<td>Source Code &amp; Java script</td>
<td>1</td>
<td>Redirect pages</td>
<td>Sub weight = 0.4</td>
</tr>
<tr>
<td>(Weight = 0.2)</td>
<td>2</td>
<td>Straddling attack</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Phishing Attack</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>OnMouseOver to hide the Link</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Server Form Handler (SFH)</td>
<td></td>
</tr>
<tr>
<td>Page Style &amp; Contents</td>
<td>1</td>
<td>Spelling errors</td>
<td>Layer Three</td>
</tr>
<tr>
<td>(Weight = 0.1)</td>
<td>2</td>
<td>Copying website</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Using forms with Submit button</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Using Pop-Ups windows</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Disabling Right-Click</td>
<td></td>
</tr>
<tr>
<td>Web Address Bar</td>
<td>1</td>
<td>Long URL address</td>
<td>Sub weight = 0.3</td>
</tr>
<tr>
<td>(Weight = 0.1)</td>
<td>2</td>
<td>Replacing similar char for URL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Adding a prefix or suffix</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Using the @ Symbol to confuse</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Using hexadecimal char codes</td>
<td></td>
</tr>
<tr>
<td>Social Human Factor</td>
<td>1</td>
<td>Emphasis on security</td>
<td></td>
</tr>
<tr>
<td>(Weight = 0.1)</td>
<td>2</td>
<td>Public generic situation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Buying time to access accounts</td>
<td></td>
</tr>
</tbody>
</table>

| Total Weight | 1 |

Many researchers have been engaged in anti-phishing research and developed many solutions. Existing anti-phishing technology is divided into three categories: Server-based detection, Browser-based detection, Independent third party detection. Server-based detection refers to the server through authentication to prevent phishing, for example: E-certificate, Dynamic security skin [1], etc. Browser-based detection refers to prompt the user by plug-ins embedded in the browser. The main purpose of independent third-party detection methods is to discover and share relevant information on phishing sites including e-mail detection [2], network behavior detection [3], personal information prevention
In this paper, we propose a new algorithm for detecting phishing web pages using DOM-Tree structure and graph matching based on Genetic algorithm. The rest of this paper is organized as follows: Section 2 summarizes related works and Section 3 presents how to present a web site by DOM-Tree (Document Object Model) structure. Section 4 presents our new genetic algorithm for graph matching using DOM-Tree. Section 5 presents our simulation and analysis results, and finally, section 6 concludes the paper.

2. Related works
Current anti-phishing solutions are text-based page analysis, such as URL analysis, HTML parsing, and keyword extraction. However, phishers can easily avoid detection by using non-HTML components, such as images, flash, and activeX, etc. In [11], J.Y. Chen et al proposed an image-based anti-phishing scheme called by robust local feature-based scheme for phishing page detection and discrimination. Wenyin et al suggested a layout-similarity-based solution [12]. It classifies a web page as a phishing page if its “visual” similarity value is above a predefined threshold. The webpage is decomposed into salient blocks according to “visual clues”. The visual similarity between two web pages is measured. A web page is considered a phishing page if the similarity to the legitimate web page is higher than a threshold.

In [13], Rosiello et al proposed a solution based on DOM-AntiPhish to compute the similarity value extracting the DOM-Tree of the considered webpages. When a password associated with a certain domain is reused on another domain, the system compares the layout of the current page with the page where the sensitive information was originally entered. For the comparison the DOMTree of the original webpage and the new one are checked. If the system determines that these pages have a similar appearance, a phishing attack is assumed. The process of DOMAntiPhish is shown in Figure 1 belows.

![DOM-Tree Extraction](image1.png)
![Java Software Call](image2.png)
![Similarity Layout Calculation](image3.png)
![Phishing Report](image4.png)

**Figure 1. DOM-Anti-Phish process**

With the DOMAntiPhish approach, it could be possible for attackers to use a combination of images to create a spoofed web page that looks visually similar to a legitimate web page. Hence, the DOM of the spoofed web page could be different and detection would be evaded. One possibility of dealing with this limitation could be to take a conservative approach and tag web pages as being suspicious that contain a large number of images or that mainly consist of images. Another possible problem could be DOM obfuscation attempts that would make the visual look similar to the legitimate web page while at the same time evading detection. Furthermore, one can always take a more conservative approach by reducing the phishing alert threshold. Also, if phishers are forced to alter the look and feel of their phishing pages, these pages will become less convincing and more suspicious to the victims.

A variant of web credential is to use a database or list published by a trusted party, where known phishing web sites are blacklisted. For example, Netcraft antiphishing toolbar [19], prevents phishing attacks by utilising a centralised blacklist of current phishing URLs. Other Examples include Websense, MCAFee’s anti-phishing filter, Netcraft anti-phishing system, Cloudmark SafetyBar, and Microsoft Phishing Filter [20]. The weaknesses of this approach are its poor scalability and its timeliness. Note that phishing sites are cheap and easy to build and their average lifetime is only a few days. APWG provides a solution directory at (Anti-Phishing Working Group) [21] which contains most of the major antiphishing companies in the world. However, an automatic antiphishing method is seldom reported. The typical technologies of antiphishing from the user interface aspect are done by [22] and [23]. They proposed methods that need Web page creators to follow certain rules to create Web pages, either by adding dynamic skin to Web pages or adding sensitive information location attributes to HTML code. However, it is difficult to convince all Web page creators to follow the rules [24]. In [20, 24, 25, 26], the visual similarity of Web pages is oriented, and the concept of visual approach to phishing detection was first introduced.

3. PROBLEM DEFINITION

3.1 DOM-Tree representation
The DOM-Tree is an internal representation used by browsers to represent a webpage. An example of how a DOM-Tree is built over an HTML webpage is shown in Figure 2. For every website where a form is used to enter sensitive information successfully, its DOM-Tree data structure is associated within the “watch” list described in [12].

The layout similarity of two websites is calculated considering their associated DOM-Tree representation [13]. They start from the assumption that if two websites have the same DOM-Tree, then they must produce an identical layout. It is still possible that two websites with two different DOM-Trees could render the same layout, but if this is the case then: the administrator of the website has done some structural modification and the website was “copied” to look like the original website, i.e., we are on a phishing web site. Only in the second case we advise the user about the possibility of a phishing attack, since in the first case the domain of the website is already in the trusted list.

![An example of a DOM-Tree representation](image5.png)

**Figure 2. An example of a DOM-Tree representation**

Given two DOM-Trees, we compare their similarity in two different ways:

1) Comparing the tags of the two web pages;
2) Extracting regular subgraphs from the trees.
Since the first approach is straight forward, in the following, we focus on the latter technique. If regularity exists in a graph, it must be possible to identify a relevant number of templates and their instances in the examined graph. Templates represent particular subgraphs of the original graph with at least two instances. The generic regularity extraction problem consists of identifying large existing templates with many instances and covering the graph by the identified templates [18, 19, 20].

4. OUR PROPOSED ALGORITHM

Our proposed system has 4 steps to detect phishing web pages as shown in Figure 4. In our work, the true and phishing pages are represented in the form of DOM tree.

4.1 Building DOM-Tree

The usual first step is to build a DOM tree (tag tree) of a HTML page. Most HTML tags work in pairs. Within each corresponding tag-pair, there can be other pairs of tags, resulting in a nested structure. Building a DOM tree from a page using its HTML code is thus natural. In the tree, each pair of tags is a node, and the nested tags within it are the children of the node. The algorithm build DOM-Tree has two steps:

- HTML code cleaning: Some tags do not require closing tags (e.g., `<li>`, `<hr>` and `<p>`) although they have closing tags. Additional closing tags need to be inserted to ensure all tags are balanced and formatted tags need to be fixed.
- Tree building: simply follow the nested blocks of the HTML tags in the page to build the DOM tree. It is straightforward.

In this paper, we use DOM API [11] to construct the tree for web pages. Each page contains zero or one doc type nodes, one root element node, and zero or more comments or processing instructions; the root element serves as the root of the tree for the page. Parser converts source document into syntactic token, from this token tree is generated.

4.2 Computing similarity of two graphs

In this section, we propose a new algorithm to compute the similarity of the true and the phishing website. In general, assume that, there are two graphs, the phishing graph $G_P$ and the true graph $G_T$ – the procedure of computing the similarity of them.

![Figure 5. Computing similarity of two graphs](image)

We can define the problem of graph matching as follows: Given two graphs $G_T = (V_T, E_T)$ and $G_P = (V_P, E_P)$, with $|V_T| > |V_P|$, the problem is to find a one-to-one mapping $f: V_T \rightarrow V_P$ such that $(u,v) \in E_T$ if and only if $(f(u), f(v)) \in E_P$ [14]. Genetic algorithms belong to a larger class of evolutionary algorithms, which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. Our genetic algorithm is described as follows:

![Figure 3. DOM Antiphish Flowchart](image)

3.2 Problem Formulation

The input for the regularity extraction problem are two DOM-Trees $T$ and $T'$ representing the true and the phishing web page. A tree is a graph $G(V, E)$ in which any two vertices are connected by exactly one path. Trees are used as a model for representing the structure of web sites, where the set of tags of a web site corresponds to the set of vertices $V$ of a tree, while the hierarchy among the tags is represented in the form of DOM tree. Our genetic algorithm is described as follows:

- **Definition 3.1 (Template Generation Problem):** Given two DOM-Trees $T(V, E)$ and $T'(V', E')$ find all the equivalent or weak equivalent pairs of subgraphs in $T$ and $T'$ that are not completely included in any other subgraph of $T$ and $T'$ respectively.

- **Definition 3.2 (Regularity Extraction Problem):** Given two DOM-Trees $T(V, E)$ and $T'(V', E')$, find the set of templates $\Omega$ to cover $T$ and $T'$, where the number of vertices in $T \in \Omega$ are maximized and the similarity penalty is minimized.

- **Definition 3.3 (Graph Covering Problem):** Given a graph $G(V, E)$ and a set of templates $\Omega$, find a cover of $G$ such that $\forall T \in \Omega$, the number of vertices $T_i$ is maximized and the total similarity penalty is minimized.
4.2.1 Represent and decode an individual

Each individual is a vector of \( n \) components \( I=(i_1,i_2,...,i_n) \) which means that we map vertex \( j \in V_r \) to vertex \( i_j \in V_r \) or vertex \( i_j \in V_p \) is the image of the vertex \( j \in V_r \). In this way, each individual \( I \) has \( n \) genes with \( m \) vertices in \( G_r \).

4.2.2 Initialization

We use fully random initialization in order to initialize a set of individuals \( P \). Each individual is initialized by a randomly generated integer \( m \) in \([1..n]\).

4.2.3 Fitness Function

We define a function to check for each \((u,v) \in E_r\), such that

\[
\text{check}(u,v) = \begin{cases} 
1 & \text{if } (u,v) \in E_r \text{ and } (f(u), f(v)) \in E_r \\
0 & \text{otherwise} 
\end{cases}
\]

(1)

A condition is said satisfied by function \( f \) if given edge \((u,v)\) is an edge of graph \( G_r \), then \((f(u), f(v))\) is an edge of graph \( G_p \), where \( f(u), f(v) \) are images of \( u, v \) over function \( f \) and vertices of the two graphs are labeled the same.

The cost function \( \text{cost}(I) \), calculates the total of all edges \((u,v)\) satisfying the mapping \( f \), is given by:

\[
\text{cost}(I) = \sum_{(u,v) \in E_r} \text{check}(u,v)
\]

(2)

The fitness function is given by:

\[
\text{fit}(I) = \frac{1}{\text{cost}(I)}
\]

(3)

If the \( \text{cost}(I) \) be the greater then \( \text{fit}(I) \) is as close to 1 as possible so the solution \( I \) is as close to the optimal solution.

4.2.4 The crossover operator

This operator mimics the mating process in the nature. The edges \((u,v) \in E_r\) mapped to \((f(u), f(v)) \in E_r\) cannot carry consecutive genes. Therefore, we use a multi-crossover type to derive the good parts from two individuals to obtain greater hybrid offspring. To ensure the randomness of the crossover operator, we choose any two individuals randomly in the population with the crossover probability smaller than crossover probability \( p_c \) (\( p_c \) is a parameter of the algorithm).

4.2.5 The mutation operator

The mutation operation is a kind of random change in the individuals. In our algorithm, pointwise mutation is adopted, in which one gene in an individual is changed with a certain probability, referred to as the mutation probability. This operator allows the algorithm to search for new and more feasible individuals in new corners of the solution space. To do mutation, an individual is randomly selected from the individuals.

First, we randomly pick an integer \( k \in [1..n] \) (\( k \) is called the mutation point); then, randomly select vertex \( j \) to replace vertex \( k \) in individual \( I \), so that vertex \( j \) does not coincide with any vertex of individual \( I \). The mutation operator is a random process so to ensure that the algorithm will randomly select individuals in any population \( I \) (individual \( I \) represents a subgraph of \( G_r \)). \( I \) is randomly generated with a mutation probability of individuals. It is smaller than mutation probability \( p_m \) (\( p_m \) is a parameter of the algorithm).

4.2.6 Selection of individuals for the next generation

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process. In each generation, we select \( P \) individuals with the highest fitness value among the existing population.

4.2.7 Termination condition

The algorithm will stop after \( G \) generations (\( G \) is a parameter) or when the average value of the individuals is unchanged or we can find an individual \( I \) satisfying \( \text{cost}(I) = |E_r| \).

4.2.8 Our algorithm

Pseudocode of our genetic algorithm for inexact graph matching problem [16] is described as follows:

---

**Algorithm 1: GA for Computing similarity**

Input: \( G_r=(V_r, E_r) \) and \( G_t=(V_t, E_t) \), with \( |V_r| > |V_t| \)

Output: \( S = \text{Similarity of } G_r \text{ and } G_t \)

Begin

\( t=0; \)

\( P(0) = \text{Initial Population();} \)

\( P(0).fit = \text{getFit(P(t));} \)

While (After \( G \) generations) or \( \text{cost}(I) = |E_r| \) is unchanged

\( r1 = \text{rand()} \times 10/10.000; \) // Crossover probability

\( \text{if} \ (r1 < p_c) \)

\( \text{child}_c = \text{crossover}(P(t)); \)

\( \text{child}_c.fit = \text{getFit(child}_c); \)

\( r2 = \text{rand()} \times 10/10.000; \) // Mutation probability

\( \text{if} \ (r2 < p_m) \)

\( \text{child}_m = \text{mutation}(P(t)); \)

\( \text{child}_m.fit = \text{getFit(child}_m); \)

\( P(t+1) = \text{bestFit}(P(t), \text{child}_c, \text{child}_m); t++; \)

Return \( S = (1/\text{getFit(P(t+)/m}); \)

End.

---

4.2.9 The complexity of our algorithm

The complexity of our algorithm is \( O(P.G.m^2.n^2) \) where, \( P \) is the population size, \( G \) is the number of iterations, \( m \) and \( n \) are the number of vertices of \( G_r \) and \( G_t \) respectively.

4.3 Phishing report

In general, we assume that given a graph \( G(V,E) \) and a set of templates \( \Omega \), find a cover of \( G \) such that \( \forall G_i \in \Omega \) the total similarity is greater than threshold \( \delta \).

---

**Algorithm 2: Phishing Detect**

Input: \( G_r=(V_r, E_r) \) and set of templates \( \Omega=\{G_1,...,G_N\} \)

\( \Delta = \text{Threshold \( \delta \)} \)

Output: \( R=\{G_i \mid S_{G_i} \geq \delta \} \)

Begin

\( R = \{\} ; \)

for \((i=1; i<=N; i++)\)

\( \text{if} \ (\text{Computing similarity}(G_i) > \delta) \)

\( R = R \cup \{G_i\}; \)

Return \( R \)

End.
5. EXPERIMENT AND RESULTS

Our algorithm is coded in C. We tested the program on a set of five sample pages (p1, p2, p3, p4, and p5) with 3 different sizes (less than 10 vertices, 10 to 20 vertices, and greater than 20 vertices) and 10 templates pages (T1, T2, ..., T10) with 3 different sizes (less than 20 vertices, 20 to 50 vertices, and greater than 50 vertices) to compute similarity between sample pages and templates pages. We compare our algorithm with Simple Tree Matching (STM) [17] algorithm.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GA</td>
<td>STM</td>
<td>GA</td>
<td>STM</td>
<td>GA</td>
</tr>
<tr>
<td>T1</td>
<td>72</td>
<td>64</td>
<td>61</td>
<td>87</td>
<td>83</td>
</tr>
<tr>
<td>T2</td>
<td>32</td>
<td>12</td>
<td>23</td>
<td>41</td>
<td>34</td>
</tr>
<tr>
<td>T3</td>
<td>52</td>
<td>43</td>
<td>28</td>
<td>13</td>
<td>76</td>
</tr>
<tr>
<td>T4</td>
<td>65</td>
<td>58</td>
<td>55</td>
<td>40</td>
<td>90</td>
</tr>
<tr>
<td>T5</td>
<td>99</td>
<td>90</td>
<td>29</td>
<td>22</td>
<td>63</td>
</tr>
<tr>
<td>T6</td>
<td>79</td>
<td>68</td>
<td>83</td>
<td>70</td>
<td>76</td>
</tr>
<tr>
<td>T7</td>
<td>71</td>
<td>61</td>
<td>87</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>T8</td>
<td>49</td>
<td>41</td>
<td>91</td>
<td>86</td>
<td>61</td>
</tr>
<tr>
<td>T9</td>
<td>80</td>
<td>71</td>
<td>61</td>
<td>51</td>
<td>78</td>
</tr>
<tr>
<td>T10</td>
<td>88</td>
<td>83</td>
<td>37</td>
<td>29</td>
<td>71</td>
</tr>
</tbody>
</table>

We tested our algorithm on a set of phishing web sites on phishtank.com [18]. This well-known site collects real phishing pages. As shown in Table 3, the higher similarity threshold means the more danger since the attacker could probably be able to imitate the legitimate web site by changing the DOM-Tree representation of the spoofed web site. In the experiment, we set threshold \( \delta \) from 0.1 to 0.9, in which the value of false position and false negative can be controlled in a reasonable range. We detected 100 website in phishtank dataset include: valid phishing, no valid phishing and unknow.

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP(%)</td>
<td>75</td>
<td>63</td>
<td>57</td>
<td>35</td>
<td>22</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FN(%)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>48</td>
<td>92</td>
</tr>
</tbody>
</table>

We can see from Table 3 that the results are strongly correlated to detecting phishing web pages with threshold \( \delta = 0.6 \). In fact, if the legitimate web page contains many particular elements in the DOM-Tree, it is easier to distinguish it from another web page.

6. CONCLUSION

Phishing is an important problem that results in identity theft. In this paper, we 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. Our experimental evaluation demonstrates that our solution is feasible in practice. Our algorithm proposed can detect the exact percentage higher than STM algorithm and faster execution time.

7. ACKNOWLEDGMENTS

This research is partly supported by the QG.12.21 project of Vietnam National University, Hanoi.

8. REFERENCES


