基于页面布局相似性的钓鱼网页发现方法

张鹏[1]，邹学强[2]，黄彩云[1]，陈志鹏[1]，孙永[1]，刘庆云[1]

(1. 中国科学院 信息工程研究所，北京 100093；2. 国家计算机网络应急技术处理协调中心，北京 100029)

摘　要：随着互联网金融的快速发展，如何快速有效地发现钓鱼网页是保障用户金融安全的前提条件，针对钓鱼网页与真实网页布局结构相似的特点，本文提出了基于页面布局相似性的钓鱼网页发现方法 ，该方法首先抽取出网页中带链接属性的标签作为特征，然后基于该特征提取网页标签序列分支来标识网页；接着通过网页标签序列树对齐算法将网页标签序列树的对齐转换成网页标签序列分支的对齐，使二维的树结构转换成一维的字符串结构，最后通过生物信息学 BLOSUM62 编码的替换矩阵快速计算对齐分值，从而提高钓鱼网页的检测效果，一系列的仿真实验表明文中方法可行，并具有较高的准确率和召回率。

关键词： 页面布局, 钓鱼网页, 标签序列树

中图法分类号：TP319

Phishing Attacks Discovery based on HTML Layout Similarity

ZHANG Peng[1], ZOU Xueqiang[2], HUANG Caiyun[1], CHEN Zhipeng[1], SUN Yong[1], LIU Qingyun[1]

(1. Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China;

2. National Computer Network Emergency Response and Coordination Center, Beijing 100029, China)

Abstract: With the rapid development of Internet finance, how to discover phishing sites quickly and efficiently is the precondition to protect the financial security of user. Based on the similarity of the layout structure between the phishing sites and real sites, in this paper, we present an approach to discover phishing sites. First, we extract the tag with link attribute as a feature, and then based on the feature, we extract the page tag sequence branch to identify website, followed by the page layout similarity-HTMLTagAntiPhish, we convert the alignment of page tag sequence tree into the alignment of page tag sequence branches, this converts two-dimention tree structure into one-dimention string structure, and finally through the substitution matrix of bioinformatics BLOSUM62 coding, we compute alignment score quickly to improve the phishing sites detection efficiency. A series of simulation experiments show that this approach is feasible and has higher precision and recall rates.

Key words: layout similarity, phishing attacks, tag sequence tree

1　引言

随着互联网的蓬勃发展，以社交网站、在线支付为代表的在线服务取得了长足进步，并给人们的日常生活带来了巨大便利。然而，在线服务同时也存在着安全隐患，例如，一些不法分子通过伪造合法身份窃取个人信息，对人们的财物和隐私安全构成严重威胁。其中，钓鱼网页是窃取个人隐私信息的主要途径之一。因此，准确识别钓鱼网页对保护网络安全具有重要价值和现实意义。

为此，安全研究者提出了钓鱼网页的多种识别方法。按照是否检测页面内容来分，这些方法可以分为两类：内容无关识别方法[1]和基于页面内容的识别方法[2]。内容无关识别方法是对一个给定的网页，提取其主机信息、网址信息及域名注册信息（不包括页面内容），以此为依据建模并判断该网页是

收稿日期：

基金项目：本课题得到国家自然科学基金项目(No. 61402464, 61502478)资助

Foundation Items：National Natural Science Foundation of China (No. 61402464, 61502478)

否是钓鱼网页。这类方法的优点是检测一个网页所需要的计算资源比较少，但准确率较低。

为了克服这一缺点，基于内容的检测方法使用一些已有的分析技术[3]分析网页的页面内容，提取更多的特征以辅助检测。这种方法在一定程度上提升了识别准确率，但需要付出更长的检测时间并占用更大的网络带宽。

此外，基于内容的检测方法需要人工确认页面中的敏感信息，这意味着用户需要将特定敏感信息（例如在线支付密码）和特定网站域名（例如在线支付网站）进行关联，这使得当同一个在线支付密码在多个网站合法使用的时候，会被认为是钓鱼行为而产生错误的警告。

一般而言，钓鱼网页看起来和真实网页相似，否则用户无法被诱骗输入他们的敏感信息，为此本文提出基于网页布局相似性的钓鱼网页发现方法，当关联某个网页的在线支付密码在另一个网页使用时，该方法不会立刻发出警告，而是比较当前网页和基准网页的布局相似度，当这两个网页的布局相似度大于指定阈值时，则被认为是钓鱼网页。其中的主要创新点如下：

- 提出了基于页面布局的网页标识方法，该方法抽取出带链接属性的标签作为特征，然后基于该特征提取网页标签序列分支来标识网页。
- 提出了网页标签序列树对齐算法，该算法将网页标签序列树的对齐转换成网页标签序列分支的对齐，使二维的树结构转换成一维的字符串结构，然后通过生物信息学 BLOSUM62 编码的替换矩阵快速计算对齐分值

文章的组织结构如下：第 2 节介绍相关工作；第 3 节介绍 HTMLTagAntiPhish 方法的基本原理和具体实现；第 4 节是实验；最后总结全文。

2 相关工作

目前，钓鱼网页识别的方法主要包括三类：基于黑名单技术的识别方法，基于启发式规则的识别方法，基于机器学习的识别方法。

2.1 基于黑名单技术的识别方法

黑名单是一份包含钓鱼网页 URL，IP 地址或者关键词信息的列表。通过使用黑名单技术，人们可以准确识别已被确认的钓鱼网页，从而降低误报率 FPR。黑名单技术实现简单，使用方便，然而，黑名单仅能识别已经发现的钓鱼网页，不能正确识别之前未出现的钓鱼网页，从而容易引起漏判。为了改善漏判情况，Prakash. Pawan 等人[4]针对黑名单技术提出了一种改进方法 PhishNet。但它的识别能力依赖于原有黑名单集合的规模，并存在时间开销随黑名单规模扩大而线性增长的缺点。

除了上述漏判和时间开销大的问题，黑名单还存在更新时效性低的缺点。根据 S. Sheng 等人[5]的研究，约有 63%的网络钓鱼行为在最初的 2 小时内就结束了，而 47%~83%的钓鱼网页在发现 12 小时后才能录入黑名单。

2.2 基于启发式规则的识别方法

为了克服黑名单机制存在的漏判等缺点，研究人员设计并实现了基于启发式规则的钓鱼网页识别方法。这类方法的工作原理是依据钓鱼网页之间存在的相似性设计和实现启发式规则，进而发现和识别钓鱼网页。但是，对于大规模网页分类而言，简单的特征统计和启发式规则方法已经无法满足需求，主要体现在以下两个方面：

（1）误报率高。由于采用启发式规则的模糊匹配技术，这类方法将大大提升良性网页的误判概率。因此，相较于黑名单方法，启发式规则的识别方法误报率较高。

（2）规则更新难，依赖于领域知识。由于启发式规则是通过对已有恶意网页的特征统计或人工总结得到的，因此这些规则依赖于对应的领域知识，因此更新困难。

2.3 基于机器学习的识别方法

针对基于启发式规则识别方法存在的误报率高和规则更新难的问题，研究人员进一步提出了更加系统的基于机器学习的识别方法。

这类方法首先将钓鱼网页识别看作是一个文本分类或聚类的问题，然后运用相应的机器学习算法[6]（例如 k-means，DBSCAN，knn，C4.5，SVM 等）进行识别。目前，用于钓鱼网页识别的机器学习方法包括无监督方法和有监督方法。

本文针对钓鱼网页和基准网页的页面布局相似的特点，利用有监督的机器学习方法获得标识网页的网页标签序列分支，然后通过生物信息学 BLOSUM62 编码的替换矩阵快速计算网页标签序列分支对齐分值来发现钓鱼网页。

3 HTMLTagAntiPhish 工作原理

在本节，我们将详细介绍 HTMLTagAntiPhish 方法的原理。首先，HTMLTagAntiPhish 将网页源码结构化为标签序列树；然后，通过选取的标签序列分支标识网页；最后，计算网页标签序列分支对齐分值。对齐分值越高表示网页布局的相似度越高，当网页的布局相似度大于指定阈值时，则判定该网页为钓鱼网页。

3.1 网页结构化

从网络中捕获的数据包经过协议识别、流还原等步骤得到网页源码。然而在高速网络流环境下，捕获的数据包在还原的过程中或多或少存在缺失，因此需要提取出一些页面布局的特征标签，用这些特征标签来代表其页面布局特点。通过对网页源码样本集进行分析发现，网页源码中带链接属性标签的分布情况与网页的大小变化趋势一致。因此，本文将带链接属性的标签作为页面布局的特征标签，即 a、link、img 这三类标签，a 的链接属性是 href，link 的链接属性是 href，img 的链接属性是 src。

网页源码结构化为标签序列树的数据结构包含<标签名，标签链接属性值，标签之间的包含关系与兄弟关系，用来辅助标记标签的标识信息>。如图 1 所示，标识信息为 0 表示该标签为起始标签，标识信息为 1 表示该标签为结束标签，标识信息为 2 表示该标签为单标签。根据 HTML 文档定义，单标签不能嵌套其他标签，单标签有 13 个：meta, br, hr, area, input, link, basefont, param, keygen, source, col, frame, embed。

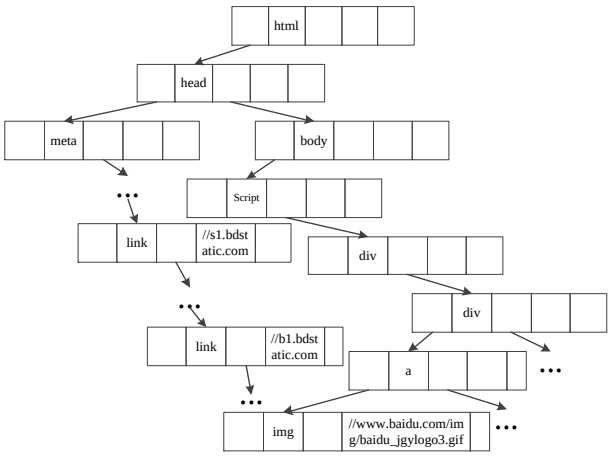

图 1 网页标签序列树结构

逐行读取网页源码，对源码按照标签的分割标识进行切分，得到网页的所有标签及其链接属性值，将所有的标签按照标签与标签之间的嵌套关系进行组织，得到的逻辑结构如图 1 所示。为方便处理网页标签序列树中不包含的特殊标签，例如<!document-->和<!if>等网页结构控制说明符，图中显示的含义为，html 标签没有兄弟标签和链接属性，只有子标签。而 head 标签和 body 标签均为 html 的子标签，且 head 标签和 body 标签相互之间不是包含关系，head 标签出现在 body 标签的前面，所以将 head 标签定义为 html 的子标签，body 标签定义为 head 标签的兄弟标签，依次扫描网页源码构造网页标签序列树。

3.2 网页标识

虽然网页标签序列树能够唯一标识网页，但网页标签序列树是二维结构，直接对其进行处理的时间复杂度高，故需要进行降维处理。另外，网页标签序列树中有些标签对网页结构的贡献度不大，所以需要从网页标签序列树中提取能有效标识网页的标签序列分支，具体实现如算法 1 所示。

Algorithm 1: ExtractPathes

Input: *TagTreeofWebsiteFile*
Output: the all tag path of the *TagTreeofWebsiteFile*
1: Initialize *Stack*←ø
2: Traverse the *TagTreeofWebsiteFile*
3: **if** (*Stack.empty*()) **then**
4: **if** (!isEndTag(*tempTag*)) **then**
5: *Stack*.push(*tempTag*);
6: **else**
7: skip the *tempTag*;
8: **end if**
9: **else**
10: **if** (*curTag.flag*=2 & *tempTag.flag*=1) **then**
11: path=printStack(Stack);
12: printPathToFile(path, tagPathFileName);
13: Pop;
14: **else**
15: **if** (*curTag.flag* =0 & *tempTag.flag* =1) **then**
16: *curTag.flag*=1;
17: **if** (*curTag.flag*=1 & *tempTag.flag*=1) **then**
18: path=printStack(Stack);
19: printPathToFile(path, tagPathFileName);
20: Pop;
21: **else**
22: Stack.push(tempTag);
23: **end if**
24: **end if**
25: **end if**

算法 1 将网页标签分为三类：起始标签，标记为 0，结束标签，标记为 1，单标签，标记为 2，当起始标签遇到其对应的结束标签之后，标记信息修改为 1，表示该标签不再嵌套其他标签。算法顺序扫描结构化之后的网页源码。当栈为空时，若当前

标签不是结束标签，则直接缓存到栈中，否则跳过结束标签，继续扫描下一行。当栈不为空时，当扫描的标签是起始标签或单标签时，直接入栈；当扫描的标签是结束标签时，需要扫描栈，寻找该结束标签对应的起始标签，并将起始标签的标记信息修改为 1，标识该起始标签不能再嵌套其他标签。当扫描的标签遇到这两种情况时：1)若当前标签为单标签，扫描的标签为结束标签时，则需要输出当前的标签序列分支，原因是单标签不能嵌套其他的标签；2)若当前标签为结束标签，扫描的标签为结束标签时，则需要输出当前的标签序列分支，原因是结束标签之后不能再嵌套结束标签。每一次输出都是从栈顶位置输出至栈底位置，输出之后要将相应的当前标签从栈顶移除，并将扫描的标签压入栈顶，保证下一次扫描标签正确处理。输出结果中包含两类信息：标签名和标签的链接属性值，单元数据格式为“标签名{链接属性值}”。

表 1 网页标签编码

标签	a	div	input	script	link	span	style	li	meta	i
编码	A	C	D	E	F	G	H	I	K	L
标签	img	p	body	form	head	html	noscript	pre	title	ul
编码	M	N	P	Q	R	S	T	V	W	Y

提取网页源码标签序列树中所有的分支路径，得到如图 2 所示的标签序列分支集合，标签序列分支有 29 条。分析这 29 条路径，并不是所有的路径都包含带链接属性标签，路径中包含带链接属性的标签数占路径中总标签数的比例不同，包含带链接属性标签的序列分支中，带链接属性的标签有重复。为了使用尽可能少的信息来有效地标识网页，从中筛选满足以下两个条件的标签序列分支：1、包含尽可能多的带链接属性标签和尽可能少的普通标签，2、序列分支包含的带链接属性标签集合不是其他序列分支的子集。扫描所有的标签序列分支，选择其中 5 条标签序列分支，如图 3 所示，其中的存储形式为：

"html->head->meta->meta->link{//s1.bdstatic.com}->link{//t1.baidu.com}->link{//t2.baidu.com}->link{//t3.baidu.com}->link{//t10.baidu.com}->link{//t11.baidu.com}->link{//t12.baidu.com}->link{//b1.bdstatic.com}->title->style->style->style->style->style->script->noscript->meta"。

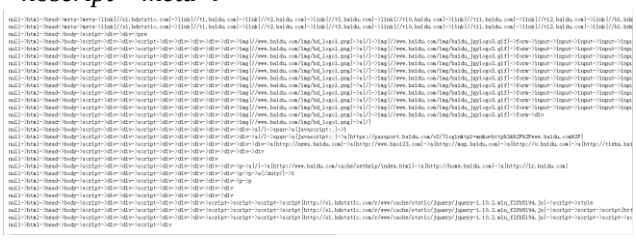

图 2 网页标签序列所有分支

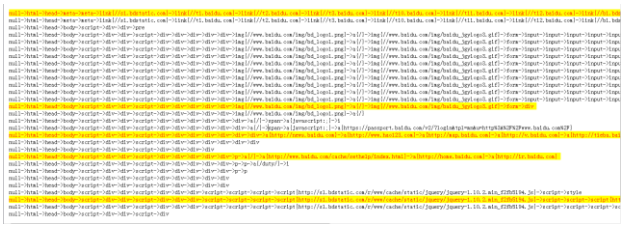

图 3 网页标签序列有效分支

得到的原始标签序列分支从字符串的角度考虑仍然是二维的，对二维字符串处理的时间复杂度较高，还需要进一步做降维处理。为此本文基于有监督学习，选取其中对页面布局区分度最大的 20 个标签，并对标签按照字母序和出现的次数进行排序，然后按生物信息学中 BLOSUM62 替换矩阵中的代码进行编码[7]，如表 1 所示，按字母序进行排列，前面生成的那条网页标签序列分支对应的编码结果为：“SRKKFFFFFFFFHHHHH ETK”。

3.3 标签序列对齐算法

对于相似性网页，它们之间相似度越高，由网页解析得到的网页标签序列树越对齐，从网页标签序列树中提取的相应网页标签序列分支的对齐分值越高。此处，将网页标签序列树的对齐转换成网页标签序列分支之间的对齐，使二维的树结构转换成一维的字符串结构，降低了计算的复杂度。

对于一维编码后的两个标签序列分支，根据它们的对齐分值来判断相似度，其中的字符串对齐算法，如算法 2 所示，具体的流程图如图 4 所示，下

面将详细描述对齐算法。

Algorithm 2: TagsAlignment

Input: $TagsMatchingeq_1$, $TagsMatchingeq_2$
Output: the *score* of alignment
1: Initialize *score*←0
2: $TagsMatchingeq_1$.*length*>$TagsMatchingeq_2$.*length* ?
(str_1=$TagsMatchingeq_1$; str_2=$TagsMatchingeq_2$;) :
(str_1=$TagsMatchingeq_2$; str_2=$TagsMatchingeq_1$;);
3: Traverse str_1 and str_2
4: *temp*=*score*+*max*(blosum[str_1[i]][str_2[j]],
blosum[str_1[i]][ ]);
5: *queue.add*(*min*(blosum[str_1[i]][str_2[j]],
blosum[str_1[i]][ ]));
6: **if** temp <score **then**
7: *queue.pollTail*();
8: **end if**
9: **return** *score*

将编码后的标签序列分支看成字符串，算法对两个需要计算对齐分值的字符串构建一个标签序列对齐树，用于记录每一步对齐所得到的对齐分值。使用分支限界法扩展标签序列对齐树，每一次扩展都保证当前扩展是局部最优的对齐分值，将次优的对齐分值与对齐策略保存到队列中。两个字符串对齐扩展有三种情况：1、在第一个字符串中增加空格，使得第二个字符串对应位置的字符与空格进行对齐，2、两个字符串对应位置的字符进行比较，3、在第二个字符串中增加空格，使得第一个字符串对应位置的字符与空格进行对齐。每一种对齐扩展后，查询增补 BLOSUM62 替换矩阵，得到对齐分值。对齐方案使用当前比较的两个字符在各自字符串的位置来记录，空格用当前位置-1 表示，队列中缓存对齐方案和对齐分值。

其中，BLOSUM62 替换矩阵来源生物信息学，用于处理基因突变的替换分值矩阵，其中包含 20 种氨基酸之间基因突变对应的替换分值。借鉴引用到字符串之间的对比，将两个对齐的字符看成替换，字符与空格之间的对齐看成是突变的一种情况，将 BLOSUM62 替换矩阵中每一个字符对应的 19 种突变情况相加除以 20，然后四舍五入得到每个字符与空格对齐的分值。观察 BLOSUM62 替换矩阵，除了对角线上的值表示两个字符是相同的得到的替换分值为正数，其他的都是基因突变对应的值为负数，空格与字符对齐从类比的角度也属于基因突变，应该是负值。而空格与空格之间的对齐设置为无穷小，程序实现时将其设置为-200，防止在两个对齐的字符串中不断添加空格，导致无限循环至无意义的对齐，增补 BLOSUM62 对齐矩阵。

扩展标签序列对齐树时，不在较长的字符串中增加空格，因为如果在较长的字符串中增加空格，必然会导致两个字符串中出现两个空格进行对齐，导致整体对齐分值下降。扩展标签序列对齐树对标签序列进行对齐，一直扩展至两标签序列字符串的末尾，当队列不为空时，需要将标签序列对齐树进行回退，回退至队列尾部的相应位置，然后从队列中记录的对齐方案开始扩展，再次扩展至标签序列尾部，若得到的对齐分值比之前的好，重新设置对齐最优的对齐方案，以便下次回退扩展得到的是最优的扩展。若得到的新的对齐分值比当前对齐分值小，则将新的对齐分值丢弃，再继续回退队列中的下一个对齐方案并扩展至两字符串末尾。当扩展至两字符串末尾且队列回退至空时，算法输出最后的最优对齐分值及对应的最优对齐方案。对齐计算算法对两个字符串进行处理的时间复杂度为 O(mn)，m 和 n 分别是两个字符串的长度。

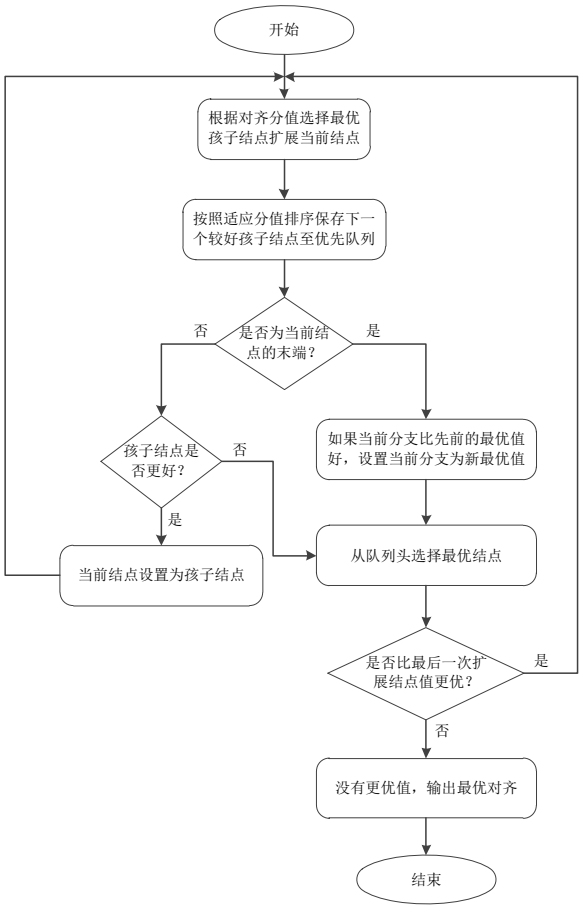

图 4 网页标签序列分支计算流程

使用 TagsAlignment 算法处理随机选择的网页源码，解析得到的网页标签序列树中对应位置的两个标签序列分支，两个标签序列分支编码得到对应的字符串为 "SRKKFFFFFFFFHHHHHETK" 和 "SRKKFFFFHHHETE"。首先设置一个空的缓存队

列，当前的对齐位置为 0 和 0，当前对齐分值为 0，查询增补 BLOSUM62 替换矩阵得到三种对齐方案的对齐分值分别为-1、4、-1，选择对齐位置 1 和 1 为对齐方案，当前对齐分值设为 4，舍弃 0 和 1 的对齐方案，原因是不在较长的字符上增加空格，将 1 和 0 的对齐方案及对齐分值-1 缓存至队列中，局部最优的标签序列对齐扩展如图 5 所示。

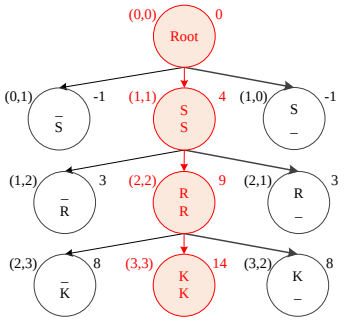

图 5 对齐树局部扩展

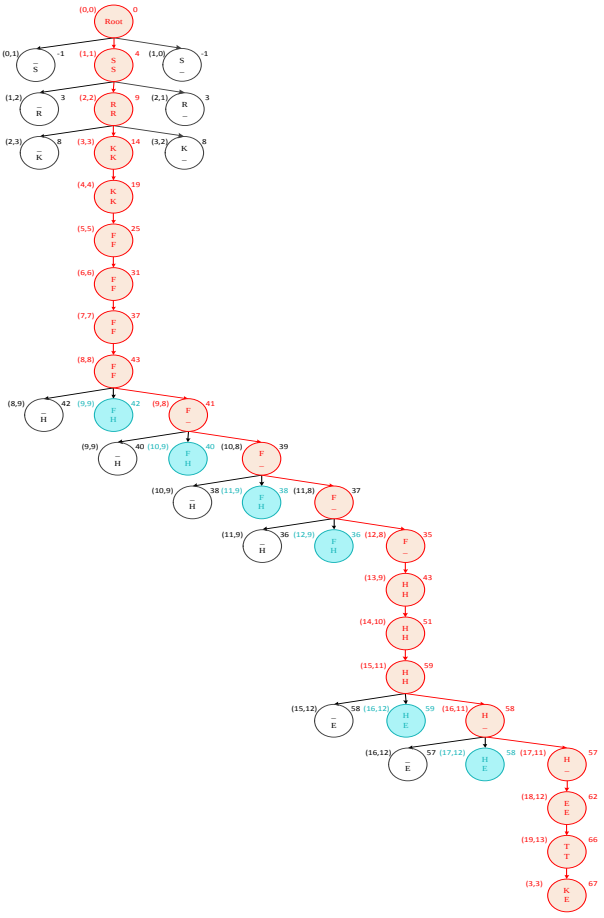

图 6 序列对齐方案

进行下一次扩展，直到两个字符串都到达末尾位置，完成第一次扩展。当一次扩展方案中，出现三种对齐情况的对齐分值相同时，优先级最高的相同位置的对齐，其次是在较短字符串中添加空格的对齐，最后是在较长字符串中添加空格的对齐。完成一次扩展之后，回退队尾的对齐方案，然后按照之前的扩展方案进行对齐，得到最后的最优扩展标签对齐树如图 6 所示，红色的为最优的对齐方案，绿色的表示中间过程被回退的局部最优扩展。对齐方案为 "SRKKFFFFFFFFHHHHHETK" 和 "SRKKFFFF_ _ _ _ HHH_ _ ETE"，在较短的序列中添加了六个空格，对应的对齐分值为 67。

4 实验与分析

我们使用 Mozilla 的 XML 用户界面语言和 Java、JavaScript 实现 TagsAlignment 计算，并且作为 Mozilla 浏览器的插件。如果当前的网页不在可信名单中，浏览器会调用该插件来计算该页面和基准页面的布局相似性，在这种情况下，会产生两个结果：

1. 用户在两个不同的合法 Web 页面输入相同的密码
2. 用户受到钓鱼攻击

相比已有的钓鱼检测方法，当且仅当布局相似度超过给定阈值时，该插件才会报警，大大减少了用户正在不同的合法 Web 页面输入相同密码时错误报警的次数。基于该插件，我们进行如下实验。

4.1 有效性测试

为获得真实可靠的分析结果，实验采用公开的钓鱼网页数据集 PhishTank[8]，这里选择相似度阈值 0.5 作为初始值。由于钓鱼网页会在基准网页上改变部分标签，所以相似度阈值太高会提高漏报率。在表 2 中，我们获得了在不同阈值下执行 TagsAlignment 对齐方法和简单标签对比方法从文献[9]随机获取得 200 多个不同网站测试的误报率(FP)和漏报率(FN)。简单标签对比不考虑 HTML 的全局结构，忽略了标签之间的关系，只计算基准页面的标签和潜在钓鱼页面标签相同的个数，该方法不断循环计算两个网页匹配的标签，直到一个网页的标签全部匹配完毕或者没有出现新的匹配标签。

实验结果与基准网页的布局特征紧密相关。事实上，如果基准网页在 TagsAlignment 中包含了很多特殊元素，那么 TagsAlignment 很容易区分它们。表 2 显示，选择 0.5 作为相似度阈值后，使用 TagsAlignment 方法的误报率为 16.89%，而使用简单标签方法的误报率为 30.29%，说明

TagsAlignment 具有较高的准确率和召回率。

表 2-5 不同阈值下的漏报率和误报率

阈值	TagsAlignment FP[%]	TagsAlignment FN[%]	TagsMatching FP[%]	TagsMatching FP[%]
0	100	0	100	0
0.1	88.31	0	90.86	0
0.2	62.19	0	75.96	0
0.3	45.20	0	55.29	0
0.4	30.16	0	40.86	0
0.5	16.89	0	30.29	0
0.6	7.54	0.03	18.27	0
0.7	0	18.42	12.5	0
0.8	0	39.47	5.29	21.05
0.9	0	73.68	0.48	50
1	0	100	0	100

4.2 性能测试

仍采用公开的钓鱼网页数据集 PhishTank，并随机选取 10000 个钓鱼网页。测试环境为一台曙光 A620r-F 服务器，配备双核 2.6GHz AMD2218CPU、4GB 内存，操作系统为 TurboLinux 3.4.3 版本。

实验结果显示 TagsAlignment 方法的对齐计算时间在钓鱼网页规模为 1000、5000、10000 时，比 TagsMatching 方法的匹配时间，分别减少了 31%、45%、75%，并且呈现出线性增长趋势，这也证明了使二维树结构转换成一维字符串结构后，通过 BLOSUM62 编码的替换矩阵快速计算对齐分值的有效性。

5 总结

随着网络服务的广泛应用，网络钓鱼已成为信息安全领域的重要安全威胁之一。本文提出了一种自动化、基于浏览器插件的客户端工具来防止没有经验的互联网用户被钓鱼网页攻击，该方法通过基准网页和潜在钓鱼网页的标签序列树对齐阈值来检测钓鱼网页，实验结果表明该方法的有效性。

参考文献

[1] Li, Zhou, Sumayah Alrwais, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures[C]. In Proceedings of IEEE Symposium on Security and Privacy (SP), ST. Francis, USA, 2013: 112-126.

[2] Canali, Davide, Marco Cova, Giovanni Vigna, and Christopher Kruegel. Prophiler: a fast filter for the large-scale detection of malicious web pages [C]. Proceedings of the 20th international conference on World wide web, Hyderabad, India, 2011: 197-206.

[3] Eshete, Birhanu, Adolfo Villafiorita, and Komminist Weldemariam. Binspect: Holistic analysis and detection of malicious web pages [J]. Security and Privacy in Communication Networks, 2013, 106: 149-166.

[4] Prakash P, Kumar M, Kompella R R, et al. Phishnet: predictive blacklisting to detect phishing attacks [C]. In Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM). San Diego, CA, USA, 2010: 1-5.

[5] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang. An empirical analysis of phishing blacklists [C]. In Proceedings of the 6th Conference in Email and Anti-Spam(CEAS). Mountainview, CA, USA, 2009.

[6] Liu, Gang, Bite Qiu, and Liu Wenyin. Automatic detection of phishing target from phishing webpage [C]. In Proceedings of the 20th International Conference on Pattern Recognition (ICPR). Istanbul, Turkey, 2010:4153-4156.

[7] Chuck Staben. BLOSUM62 Substitution Matrix [EB/OL]. 1998[2016-03-21].http://www.uky.edu/Classes/BIO/520/BIO520WWW/blosum62.htm.

[8] OpenDNS PhishTank [EB/OL]. http://www.phishtank.com. 2014.5.

[9] URoulette. Home Page. http://www.uroulette.com, 2007.