第27卷 第1期
2012年2月
郑州轻工业学院学报(自然科学版)
JOURNAL OF ZHENGZHOU UNIVERSITY OF LIGHT INDUSTRY(Natural Science)
Vol. 27 No. 1
Feb. 2012

文章编号:1004-1478(2012)01-0065-04

一种高效的字符串匹配算法

廖秀玲, 邵剑飞, 李小武

(昆明理工大学 信息与自动化学院, 云南 昆明 650050)

摘要:针对目前精确串匹配算法中模式右移次数多、算法运行时间长等问题,提出了一种新的高效算法——BMH2S. 该算法采用寻找真首子串和利用2个字符子串的方法来计算右移量. 测试结果表明,BMH2S是一种高效的模式匹配算法.

关键词:入侵检测系统;字符串匹配;模式匹配;BMH2S算法

中图分类号:TP391.1 **文献标志码**:A

An efficient string matching algorithm

LIAO Xiu-ling, SHAO Jian-fei, LI Xiao-wu

(*Faculty of Infor. Eng. and Auto., Kunming Univ. of Sci. and Tech., Kunming* 650050, *China*)

Abstract: Aiming at present the exact matching algorithm has problems that the model moves to the right more frequently and the running time is long, a new efficient algorithm of BMH2S was presented. Which uses a truth substring and the two characters of the substring to calculate the amount move to the right. The test results showed that BMH2S is an efficient model matching algorithm.

Key words: misuse intrusion detectoin system(MIDS); string matching; model matching; BMH2S algorithm

0 引言

随着计算机网络技术的不断发展,网络安全问题日益突出. 单纯的防火墙技术只能防御外来的攻击,而不能阻止来自网络内部的攻击. 入侵检测[1-2]是提供强大主动策略和增强网络安全性的有效手段. 对于入侵检测系统MIDS(misuse intrusion detectoin system)而言,检测过程中最费时间的部分就是字符串匹配,且字符串匹配在语言翻译、病毒检测、信息过滤、信息检索及计算生物学等领域也有相当重要的应用,甚至已经发展为一门相对独立的学科——字符串学(Stringology)[3]. 因此,对高效的字符串匹配算法的研究,具有重要的现实意义和理论价值.

字符串匹配可以分为3类,即精确串匹配、近似串匹配和正则表达式匹配[4],其算法有BM算法、KMP算法、BF算法、Sunday算法[5]等. 本文拟在精确串匹配方面[6]对几种常见算法进行分析,然后综合它们的优点并做出改进,提出一种高效的BMH2S算法.

1 相关算法简介

模式匹配的定义:设有给定长度为 n 的主串 $T[0\cdots n-1]$ 和长度为 m 的子串 $P[0\cdots m-1]$. T 的长度远远大于 P 的长度,即 $n \geqslant m$. 通常把主串 T 称为正文,把子串 P 称为模式. 在 T 中寻找 P 的过程,称作模式匹配. 在 T 中找到等于 P 的子串,则匹配成功;否则,匹配失败.

收稿日期:2011-09-07

作者简介:廖秀玲(1986—),女,湖北省黄冈市人,昆明理工大学硕士研究生,主要研究方向为网络通信与信息处理.

BM[7-8]算法的基本思想是:将正文子串$T[i\cdots i+m-1]$与$P[0\cdots m-1]$对齐进行自右至左的匹配,若不匹配,令正文中该位置字符为x,然后继续从正文的$i+m-1+d(x)$位置开始重新进行BM算法匹配.当发生不匹配时,算法用shift和skip数组将模式向右移至尽可能远的距离.shift数组通过考查匹配部分来决定模式右移量,skip数组则通过考查造成失配的正文字符在模式中出现的情况来决定右移量.$d(x)$的计算如下:

$$d(x)=\begin{cases} m & \text{若 } x \text{ 不在 } P \text{ 中出现,或 } x=P[m-1],\\ & \text{但 } x\neq P[j]\ (0\leqslant j\leqslant m-2)\\ m-j & \text{其他}\end{cases}$$

BMH[9]算法对BM算法进行了改进.算法将失配与计算右移量独立,计算右移量时只是简单地使用正文中与模式最右端对齐的字符来决定右移量,即只是计算skip数组.

BMHS算法在BMH算法基础上做了进一步的改进,在计算skip数组时考虑下一个字符,即利用下一个字符来决定右移量.

Sunday算法的核心思想是:在匹配过程中,算法没有规定模式串从左到右还是从右到左比较,因此当发生不匹配时,算法能跳过尽可能多的字符以进行下一步匹配.假设在发生不匹配时,已经匹配部分为u,$T[i]$是紧跟在P和T对齐元素之后的元素,如图1所示,则:1)$T[i]$在P中没出现,则P向右移动m个单位;2)$T[i]$在P中出现,即$P[k]=T[i]$,则P向右移动$m-k$个单位.

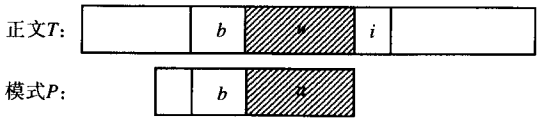

图1 Sunday算法不匹配的情况

2 BMH2S算法

BMH2S算法综合了BM算法、BMH算法、Sunday算法和BMHS算法的优点.该算法的基本思路是:第1步进行预处理,在Sunday算法和文献[10]的基础上,试图在模式P中找到一个最长的真首子串substring(substring < P).第2步进行匹配,把substring作为新模式与正文T进行匹配,在匹配过程中,使用$T[i]$和$T[i+1]$2个字符作为1个子串来决定模式的右移量.因此当发生不匹配时,能够移动尽可能远的距离;当发生匹配时,继续按顺序匹配endstring(endstring = P - substring).若endstring匹配失败,移动substring到刚刚endstring匹配失败的位置,继续第2步的匹配;若匹配成功,则1个字符串匹配过程完成,移动substring到endstring匹配成功的下一个位置,继续第2步匹配.如此匹配下去,直到正文T的最后一个字符.

2.1 预处理

预处理即寻找substring的过程.寻找substring串的目的是当模式与正文不匹配时,下一次匹配能移动尽可能远的距离.在开始寻找最长真首子串时,首先令substring = $P[0,1\cdots m-1]$作为新模式串,做相应的查找,若查找不到,再令substring = $P[0,1,\cdots m-2]$,依此类推.

寻找substring串的步骤:

1)设模式串最右边的元素为substring$[j]$($0\leqslant j\leqslant m-1$),正文$T[i]$($0\leqslant i\leqslant n-1$)紧跟在substring$[j]$与之对齐的元素后面的那个元素,如图2所示.

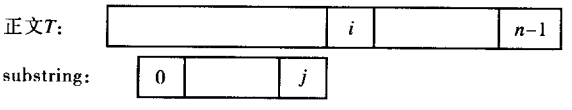

图2 正文T与substring串的位置

2)用$T[i]$按从右到左的顺序依次匹配substring$[0,1\cdots j-1]$元素,一旦发生匹配则将substring最后一个元素删去,同时将生成新的字符串赋给substring,substring串右移$j+1$个单位,进入下一轮寻找,即执行1);若直到$T[i]$与substring$[0]$都不发生匹配,则执行3).

3)此时分2种情况:如果$T[i]$ = substring$[j]$,则将substring串右移$j+2$个单位,然后执行1);如果$T[i]\neq$substring$[j]$,则将substring串右移$j+3$个单位,然后执行1).

4)当substring串移动到正文T最后一个元素时,如果substring串的长度小于模式串P的长度,则成功找到substring串;否则,将substring串的最后一个元素去掉,赋给substring,重新返回1),继续寻找.

2.2 匹配原理

2.2.1 substring串和正文匹配算法 用substring串和正文匹配时因为使用2个字符决定右移量,将正文T中与substring串最后一个字符及下一个字符相对应的2个字符作为1个子串twoChars = $T[i-$

$l][i]$. 当 twoChars 在 substring 中出现时,substring 向右移,使之与正文 T 中的 twoChars 对齐;否则,substring 直接右移 $m+1$ 个单位. 因为,$T[i]$有可能与 substring[0]相同,如果直接将 substring 右移 $m+1$ 个单位有可能漏掉一种匹配情况,因此只能右移 m 个单位,使 substring[0]与 $T[i]$对齐.

如下例所示:

1)右移 $m+1$ 个单位,漏掉一种匹配:

正文: We are Chinese. We love China!

模式: Chinese

模式移动后:　Chinese

2)右移 m 个单位,不会漏掉一种匹配:

正文: We are Chinese. We love China!

模式: Chinese

模式移动后: Chinese

相应算法如下:

```
String twoChars;//定义一个放2个字符的子串
boolean flag = false;
for(int j = m - 1;j > =0;j - - ){//开始匹配
  twoChars = t. substring(i - 1,i + 1);
  if(twochars. charAt(1) = = sub. charAt(j)){
    if(j! =0)
    if(twochars. charAt(0) = = sub. charAt(j - 1)){//
子串在模式中出现
      i + = i - j;
      flag = true;
      break;
  }
    if(j = =0){//T[i]与 substring[0]相同
      i + = m;
      flag = true;
      break;
    }
  }
  else//其他情况
  i + = m + 1;
  }
```

利用2个字符计算的右移量比用1个字符计算出的右移量大,因此减少了比较次数;另外,没有求最大值过程也减少了算法的时间消耗.

2.2.2　匹配过程　经过预处理找到 substring 串之后进行匹配. 首先用 substring 串与正文匹配. 若匹配成功,则用 endstring 继续与正文匹配. 若 endstring 也匹配成功,则整个模式串 P 匹配成功;否则,substring 向右移动 L(substring 的长度)个单位,继续与正文匹配.

下面是一个用 BMH2S 算法匹配的例子:P = China,T = We are Chinese. We love China! 经过寻找 substring 算法查找到 substring = Chin,则 endingstring = a. substring 和 endingstring 与正文匹配过程如图3所示.

1. We are Chinese. we love China!
   Chin(ar 没在 substring 中出现)
2. We are Chinese, we love China!
   Chin(hi 在 substring 中出现)
3. We are Chinese. we love China!
   Chin(substring 匹配成功,则匹配 endstring. 若 endstring 匹配失败,继续匹配 substring)
4. We are Chinese. we love China!
   Chin(we 没在 substring 中出现)
5. We are Chinese. we love China!
   Chin(ve 没在 substring 中出现)
6. We are Chinese. we love China!
   Chin(in 在 substring 中出现)
7. We are Chinese. we love China!
   Chin(substring 匹配成功,则匹配 endstring. endstring 也匹配成功)

图3　匹配过程

3　算法测试结果及结论

为了对各算法的性能、效率做具体的测评,随机指定一段正文 T 和模式 P,在同一台计算机上用不同的算法进行匹配. 测试正文如下:

Little jay really hates to deal with string. But moondy likes it very much, and she's so mischievous that she often gives jay some dull problems related to string. And one day, moondy gave jay another problem, poor jay finally broke out and cried. So what is the problem this time? First, moondy gave jay a very long string A. Then she gave him a sequence of very short substrings, and asked him to find how many times each substring appeared in string A.

匹配模式为:how many times

在同一台计算机上分别用 BM 算法、BMH 算法、BMHS 算法、Sunday 算法及 BMH2S 算法单次匹配和 10 000 次循环匹配,分别统计每种算法单次匹配时右移次数和1万次匹配所消耗时间. 测试结果

见表1.由表1可以看出,BMH2S算法能够减少比较次数,有效提高匹配速度.

表1 各模式匹配算法的性能测试结果

算法	单次匹配中模式右移的次数/次	进行1万次匹配消耗时间/ms
BM	55	27
BMH	55	21
BMHS	58	24
Sunday	62	31
BMH2S	43	18

4 结语

在综合了BM算法、BMH算法、Sunday算法和BMHS算法优点的基础上,提出了一种新的高效算法——BMH2S. BMH2S算法采用寻找substring串和利用2个字符子串计算右移量的方法,在模式匹配过程中,每一次匹配不成功都能跳过尽可能多的字符,从而使模式右移次数大大减少.测试结果表明BMH2S是一种高效的模式匹配算法.

参考文献:

[1] 臧露.入侵检测技术在网络安全中的应用与研究[J].信息技术,2009(6):41.

[2] Mukherjee B, Heberlein L T, Levitt K N. Network intrusion detection[J]. IEEE Network, 1994, 8(3):26.

[3] Dan Gusfield. Algorithms on String Trees and Sequence—Computer Seience and Computational Biology[M]. Cambridge: Cambridge University Press, 2005.

[4] Gonzalo N, Mathieu R. Flexible Pattern Matching in Strings [M]. Cambridge: Cambridge University Press, 2002.

[5] Crochemore M, Czumaj A, Gasienniec L, et al. Speeding up two string matching algorithms [J]. Algorithmica, 1994, 12(4/5):247.

[6] 俞文洋,张连堂,段淑敏. KMP模式匹配算法的研究[J].郑州轻工业学院学报:自然科学版,2007,22(5):64.

[7] 郭军,笹尾勤.入侵检测中模式匹配算法的FPGA实现[J].科技创新导报,2007(14):3215.

[8] Franek F, Jennings C G, Smyth P W F. A simple fast hybridpattern-matching algorithm[J]. J of Discrete Algorithms, 2007, 4(5):682.

[9] 刘胜飞,张云泉.一种改进的BHM模式匹配算法[J].计算机科学,2008(11):164.

[10] 赵一浸.一个改进的BM串匹配算法[J].计算机研究与发展,1998,35(1):45.