第30卷 第22期 计 算 机 工 程 2004年11月
Vol.30 №22 Computer Engineering November 2004
·博士论文· 文章编号：1000—3428(2004)22—0024—03 文献标识码：A 中图分类号：TP301.6

字符串匹配技术研究

李雪莹[1,2]，刘宝旭[2]，许榕生[2]
(1.军事医学科学院医学情报研究所网络信息中心，北京100850；2.中国科学院高能物理研究所计算中心，北京100039)

摘 要：简述了字符串匹配算法的研究进展，分析了Knuth- Morris-Pratt算法、Boyer-Moore算法以及Horspool、Wu & Manber和Aho-Corasick针对Boyer-Moore算法提出的多种改进算法，并基于网络安全应用中开放源码的NIDS系统——Snort2.0，对其中几个算法进行评测，指出了实际应用中字符串匹配技术的关键点和解决办法，探讨了应用字符串匹配技术的NIDS的研发方向。

关键词：模式匹配；字符串匹配；时间复杂性；空间复杂性；网络入侵检测系统

Research of String Matching Techniques

LI Xueying[1,2]，LIU Baoxu[2]，XU Rongsheng[2]
(1.Network Information Center, Institute of Medical Information, Academy of Military Medical Sciences ,Beijing 100850；
2.Computing Center, Institute of High Energy Physics, CAS, Beijing 100039)

【**Abstract**】The evolution of string matching algorism research is surveyed in the paper. It analyzes Knuth-Morris-Pratt algorithm, Boyer-Moore algorithm and the various changes of Boyer-Moore algorithm proposed respectively by Horspool, Wu & Manber and Aho-Corasick . The important points of string matching techniques in practice are presented by evaluating several algorithms in Snort2.0 which is an open source NIDS system used in the network security field. And identifies the direction of research and development in NIDS using string matching.

【**Key words**】Pattern matching； String matching； Time complexity； Space complexity; Network intrusion detection system

字符串匹配是模式匹配中最简单的一个问题，但在文本处理领域中字符匹配是一个非常重要的主题。它可用于数据处理、数据压缩、文本编辑、信息检索等多种应用中，大多数操作系统中软件实现的字符匹配算法是基本组件之一。字符串匹配技术通常也和其他字符问题有一定关联。在实际应用中字符串匹配技术不仅适用于计算机科学，在语义学、分子生物学等领域也具有相当重要的应用，在以模式匹配为特征的网络安全应用中也发挥了举足轻重的作用。

1 问题描述

定义1 一个文本(text)标识为y=y[0…n-1]，它的长度为n，text就是待匹配的文本。

定义2 用于进行匹配的字符串称为模式(pattern)，标识为x=x[0…m-1]，它的长度为m。

定义3 定义1和2中的字符串建立在有限字符集上，该字符集称为字母表(alphabet)，由Σ标示，大小为σ。该文是建立在这样一个假设上的，即从字母表Σ中选取的字符串strings对算法的有效性有着严格的影响。

定义4 对定义1中的文本进行扫描检测时需要运用的窗口window，它的大小为m。

定义5 将定义4中的窗口window和text的一端对齐，逐个比较window中和模式中的字符称为尝试(attempt)。

定义6 在text中发现一个和Pattern完全匹配的字符串或者发现window中的字符和Pattern不匹配时，将窗口右移，即shift。该机制称为滑动窗口机制(sliding window mechanism)。当对位置j进行尝试时，窗口的位置为y[j…j+m-1]。

定义7 如果存在两个字u和v使得w=uz=zv，则z是字w的边界(border)，z是w的前缀也是它的后缀。注意在这种情况下|u| = |v|，并且它们是w的一个period。

定义8 对于给定的text y=y[0…n-1]和pattern x=x[0…m-1],在text中寻找所有出现pattern的过程就是字符串匹配[3]。

2 字符串匹配问题的解决方法

2.1 解决方案的评价原则

算法的复杂性是评价算法优劣的重要依据，算法的复杂性有时间复杂性和空间复杂性之分。对任意给定的问题，设计出复杂性尽可能低的算法是设计算法中追求的一个重要目标；当给定的问题有多种算法时，选择其中复杂性最低者，是选用算法所遵循的一个重要准则。因此，算法复杂性分析对算法的设计或选用有着重要的指导意义和实用价值[3]。

一般而言，好的字符串匹配算法要有以下特点：

(1)速度快：这是评价一个字符匹配算法最重要的标准。通常要求字符匹配能以线性速度执行。目前，网络安全应用中的基于误用的NIDS使用字符串匹配算法进行入侵检测，因此，随着网速的提高，对字符串匹配速度的需求同样也在提高。

有几种时间复杂性的评价指标：1)预处理时间的复杂 性：有些算法在进行字符串匹配前需要对模式特征进行预处理；2)匹配阶段的时间复杂性：字符串匹配过程中执行查找操作的时间复杂性，它通常和文本长度及模式长度相关；3)最坏情况下的时间复杂性：对一个text进行字符模式匹配时，设法降低各算法的最坏情况下的时间复杂性是目前的研究热点之一；4)最好情况下的时间复杂性：对一个text进行字符模式匹配时的最好的可能性。

(2)内存占用少：执行预处理和模式匹配不仅需要CPU资源还需要内存资源，尽管目前内存的容量较以前大得多，但为了提高速度，人们常利用特殊硬件。通常，特殊硬件中内存访问速度很快但容量偏小，这时，占用资源少的算法将更具优势。

2.2 解决方案的分类

字符串匹配过程中,windows从左向右滑动，而windows中的text的字符和Pattern中的字符的匹配顺序在不同的算法

基金项目：国家“973”计划基金资助项目(G1999035806)；中国科学院知识创新工程基金资助项目(KJCX1-09)
作者简介：李雪莹(1974－)，女，博士生，主研领域为网络安全和信息处理；刘宝旭，博士、副研究员；许榕生，博导、研究员
收稿日期：2003-10-23 E-mail：lixy@mail.ihep.ac.cn

中是不同的。可以按照匹配的顺序将算法分为从左向右比较、从右向左比较、按特定顺序、顺序无关4大类。

3 几个经典的字符串匹配算法

介绍经典算法时不能不提到第一个线性字符匹配算法——Morris-Pratt算法，它是Morris和Pratt在对Brute Force算法进行严格分析的基础上提出的。该算法被Knuth，Morris和Pratt进一步改进为Knuth- Morris-Pratt算法(即KMP算法)。这两个算法按从左向右的顺序执行匹配操作。从本质上看KMP算法就是出现不匹配情况时带有智能指针初始化的brute force算法。为了在不匹配时确定重新定位的指针，KMP算法需要进行预处理计算出一个next表。

KMP算法在最糟情况下的时间复杂性为$O_{(n+m)}$。因为next数组的存在，KMP算法需要的额外的空间为$O_{(m)}$。在大多数情况下，KMP算法不比brute force好很多，但KMP确保了线性，并且其扩展性适合求解更难问题。为了在实践中得到比brute force快得多的算法，人们将眼光转向了Boyer Moore算法，即BM算法。BM算法按从右向左的顺序进行匹配。在通常应用中BM算法被看作是最有效的字符匹配算法。其主要思想是：首先在pattern的最后一个字符x_{m-1}和text中的第m个字符y_{m-1}之间进行比较，当不匹配(或模式完全匹配时)，用两个预计算函数来向右移动Windows。这两个移动的函数分别为good-suffix shift(称为匹配移动)和bad-character shift(称为事件移动)。Boyer-Moore算法在good-suffix shift 和bad-character shift中用途最大。

good-suffix shift被存储在大小为m+1的表bmGs中。定义的两个条件为：

(1)Cs(i,s)：对于每一个k，$i < k < m$, $s \geq k$或$x[k-s]=x[k]$并且，

(2)Co(i,s)：如果$s<i$，则$x[i-s] \neq x[i]$。

这样，对于$0 \leq i < m$：$bmGs[i+1]=\min\{s > 0$：Cs(i,s)和Co(i,s)$\}$，定义bmGs[0]为x的周期的长度。运用表suff来计算bmGs，具体的定义为：$1 \leq I < m$, $suff[i]=\max\{k$：$x[i-k+1\cdots i]=x[m-k\cdots m-1]\}$。

bad-character shift函数被存储在一个大小为σ的表bmBc中。因此Σ中的c为：如果c出现在x中，则$bmBc[c]=\min\{i$：$1 \leq i<m-1$并且$x[m-1-i]=c\}$。

在执行匹配操作前bmBc和bmGs以时间$O_{(m+\sigma)}$进行预计算，该计算过程的空间复杂性为$O_{(m+\sigma)}$。匹配阶段的时间的复杂性是二次项的，但当匹配一个非周期化的模式时至多需要匹配3n个text字符。在一个大的字母表中(相对于pattern的长度而言)，该算法极快。

4 改进算法

为追求更高的性能，人们不断对已有的算法进行改进。以下就几个主要的改进算法进行介绍。

4.1 Boyer-Moore-Horspool算法

BM算法中的bad-suffix shift对于小的alphabets不是特别有效，但当alphabets比pattern的长度大时，它就很有用。Horspool提出仅用Windows中最右边的字符的bad-suffix shift来计算BM算法的shift[5]，即Boyer-Moore-Horspool(BMH)算法。BMH较BM算法容易实现，其预处理的时间复杂性为$O_{(m+\sigma)}$，空间复杂性为$O_{(\sigma)}$，匹配执行的时间复杂性为$O_{(mn)}$，对于text中的一个字符的平均比较次数在$1/\sigma$和$2/(\sigma+1)$之间。

4.2 Wu & Manber的BM改进算法

BM算法中首先比较pattern的最后一个字符x_{m-1}和 text 中的第m个字符y_{m-1}，如果不匹配就看x_{m-1}在模式中最右边出现的位置并移动到相应的位置，在大多数情况下，不匹配的可能性远大于匹配的可能性。在自然语言的text中，shift的值为m或接近于m的情况非常多，这就使得算法执行得很快。Wu & Manber将这种方法运用到多模式匹配问题中，存在许多模式时，可能碰巧出现text中的许多字符和某些模式中的最后一个字符匹配，这时shift值很小，必须克服该问题来保证BM算法的速度。

Wu & Manber的BM改进算法[5](MWM算法)对pattern集进行了处理。许多在多项搜索中运用了固定pattern集的应用得益于将预处理结果存放在一个文件或内存中，该步骤十分有效。大多数情况下预处理阶段都执行得很快。该算法在预处理阶段构建了3个表，即SHIFT表、HASH表和PREFIX表。其中SHIFT表和BM算法相似但不完全相同，该表被用于确定当对text执行匹配时有多少text中的字符能被跳过。HASH和PREFIX表在shift值为0时使用，它们被用于确定候选匹配的模式并校验匹配。

4.3 Aho-Corasick 算法

Aho-Corasick算法[6]是同时搜索多个pattern的经典算法。该算法是Unix的fgrep的基础。笼统地讲，Aho-Corasick算法运用有限自动机结构来处理模式集合中的所有字符串，构建自动机使得每个prefix仅用一个状态来表示。检索时，当text中的下一个字符不是模式中所期待的字符时，一个失败的连接给出最长的前缀。Aho-Corasick算法的时间复杂性为$O_{(n)}$，预处理和pattern的大小为线性相关$O_{(m)}$。

5 评测与结论

实践证明，BM及其几个变种算法在应用中性能较高。下面构建一个测试环境对实践中性能较好的3个典型算法在网络入侵监测系统中的工作状况进行测试，并分析比较实验结果，寻求提高字符串匹配性能的研发方向。实验环境采用Pentium4 的PC计算机作为硬件平台，其中CPU为1.5GHz，内存为256MB，操作系统用Linux8.0，NIDS选用开放源代码的Snort2.0，用gprof [7] 进行评测。

在NIDS中，检测数据量的多少、检测规则的数目及构成等的不同将使NIDS花费不同的检测时间。为保证测试结果的准确性，测试过程中采用不同字符串匹配算法的Snort对同一个在局域网出口处抓取的dump格式的文件进行读取检测。该文件的大小为794 327 604字节，共有1 183 161个数据包，其中TCP数据包为1 078 698个，占总数的91.171%；UDP数据包为30 208个，占总数的2.553%；ICMP数据包为8 278个，占总数的0.7%；ARP数据包共22 775个，占总数的 1.925%；其他数据包为43 202个，占总数的3.651%。测试中分别调整Snort中设置的规则的数目，来测试BMH、MWM和AC算法所用的时间和占用的计算机的资源(CPU占用率和内存使用量)。

首先，检测Snort2.0中字符串匹配处理所占用的处理时间。Snort2.0基于Wu和Manber的论文[6]实现。检测指定的测试数据文件花费41.26s,其中字符串匹配所用时间为61.6%。可见，在该评测条件下Snort2.0中字符串匹配占用了大部分执行时间。可以说字符匹配是Snort等入侵检测应用中的主要性能瓶颈，提高字符匹配速度是提高NIDS总体性能的关键问题之一。

改变Snort2.0中的字符串匹配算法为BMH、MWM和AC，用全部规则进行测试，具体数据见表1。

表1 全规则集下各算法的检测时间和资源利用率

	Rule = 1 331 Chains =139 alerts= 7 095		
	Snort的检测时间(s)	CPU占用率(%)	占用内存量(MB)
BMH	149.24	77~92	45.3~45.5
MWM	41.26	80~93	38.7~39.1
AC	40.20	70~84	117~118

改变检测规则集的构成，对各算法进行测量，结果见表2和表3。

表2 没有符合条件攻击时的匹配时间和资源利用率

	Rule =769 Chains=11 Alert=0		
	Snort的检测时间(s)	CPU占用率(%)	占用内存量(MB)
BMH	24.95	88~91	8.8~9.0
MWM	8.38	85~88	8.8~9.1
AC	13.02	83~87	17.9~18.2

表3 选用部分规则时的时间和资源利用率

	Rule =328 Chains=86 Alert=270		
	Snort的检测时间(s)	CPU占用率(%)	占用内存量(MB)
BMH	59.14	85~92	24.1~24.3
MWM	30.50	82~84	21.3~21.6
AC	34.90	74~78	41.6~42

表中的Rule表示检测规则的数目，Chains指规则链的条数，Alert表示的是警报的条数。

由测试结果可见，在速度上MWM算法和AC算法在Snort中实际的速度差别不大，在检测模式条数较多情况下AC算法略微快一些，而BMH算法的执行时间远大于这两个算法，检测模式条数越多该差距就越明显；由资源利用率的测量结果可见AC算法对内存要求较高，其内存的占用量基本为MWM和BMH的2~3倍，而MWM算法的CPU利用率较AC算法高10%~15%。

从以上对各算法的分析和Snort中的评测结果，不难得出以下的结论：

(1)在Snort应用中字符串匹配在处理时间中占相当大比重，它是网络入侵检测应用中主要的性能瓶颈；

(2)算法在执行时并不完全按数据量和模式数目呈线性增长，不同算法适应不同的数据流和模式集；

(3)资源占用和速度在实际应用中同样重要。

通过对网络数据流构成和系统资源等的分析，智能地选用合适的字符匹配算法是应对高速网络传输速率的方法之一。针对Snort中检测原理可以实现对不同规则检测的并行执行，并行化地引入将提高字符串匹配的总体效能。

6 未来的工作

字符串匹配工作一直是IDS研究领域中的热点之一。在网络速度迅速发展的今天，基于字符匹配技术的网络应用存在着性能瓶颈，提高系统的整体性能是研究和设计者的主要工作。字符匹配是其实现网络入侵检测的主要技术之一，因此，高效的算法将是提高系统总体性能的手段之一。在字符匹配算法中存在这样的几个定理：一个是“任何的字符串匹配算法在最糟的情况下必须检查至少n-m+1个text中的字符”，这说明没有哪一个算法会比KMP或BM有更好的计算复杂性。算法的平均性能可以得到提高，但最糟情况下的结果是一个严格的限制；另一个定理是“任何字符匹配算法至少检查n/m个字符”，这个是显而易见的。基于以上的结论，接下来研究的主要方向是设法提高算法的平均性能和减少资源耗费：主要的途径可以采用在实际的网络应用中设法减少m或n的值来减少实际匹配的次数；设法将部分匹配算法移植到硬件的实现中或在并行的体系结构下实现等，以减少匹配所耗费的时间。

参考文献

1 Roesch M.Snort——Lightweight Intrusion Detection for Networks.In Proceedings of the 13th Systems Administration Conference, Usenix, 1999

2 Breslauer D. Efficient String Algorithmics[Ph.D.Thesis].Computer Science Department, Columbia University, NY, 1992

3 傅清祥, 王晓东.算法与数据结构. 北京:电子工业出版社, 1998

4 Horspool R N. Practical Fast Searching in Strings. Software Practice and Experience, 1980,10(6):501-506

5 Sun W, Manber U. A Fast Algorithm for Multi-pattern Searching. Tech. Rep. TR94-17, Department of Computer Science, University of Arizona, 1994-05

6 Aho A,Corasick M.Efficient String Matching: An Aid to Bibliographic Search.Communications of the ACM, 1975,18(6):333-343

7 Graham S L, Kessler P B, McKusick M K.GPROF: A Call Graph Execution Profiler. In Proceedings of the ACM SIGPLAN '82 Symposium on Compiler Construction,Boston,MA,1982,17(6):120-126

☆☆

(上接第20页)

4 结论

本文提出了基于DCT域的音频信号中嵌入隐藏信息的算法，通过反复试验，音频分段的长度过大，过小隐藏的效率都不高，实验研究取16位最为合适。关于步长q的取值分析，q取值越大，隐藏信息相对于覆盖数字音频失真越大，不易察觉性越差。另外试验表明，通过伪随机码加密过的隐藏信息，具有良好的完整性和安全性。

笔者近期通过对CDMA无线通信技术的研究，发现其数字音频的传输过程中，有很大的信息隐藏的利用空间。这也是笔者下一步的研究方向，将在CDMA协议中融入此算法。

参考文献

1 Podichuk C I, Delp E J. Digital Watermarking: Algorithms and Applications [J]. IEEE Signal Processing Mag., 2001,18(4): 33-46

2 Patterson R D. A Pulse Ribbon Model of Monaural Phase Perception [J]. JASA,1997,82(5): 1560-1586

3 Bassia P, Pitas I, Nikolaidis N. Robust Audio Watermarking in the Time Domain [J]. IEEE Trans. on Multimedia, 2001, 3(2): 232-241

4 Craver S, Memon N, Boon-Lock Yeo. Resolving Rightful Ownerships with Invisible Watermarking Techniques: limitations, Attack, and Implications[J]. IEEE Journal on Selected Areas in Communication, 1998,16(4): 573-586

5 Craver. Technical Trials and Legal Tribulations[J]. Communications of the ACM, 1998, 41(7): 45-54