# 基于自动机并操作的多目标AC-BM算法

王正才 许道云 王晓峰
(贵州大学计算机科学与信息学院 贵阳 550025)

**摘 要** AC-BM算法的优点在于能同时进行多个模式串的匹配搜索,且文本串的移位得到优化,但一次只能在一个文本串中进行搜索。为了实现一次可以同时在多个文本串中进行搜索,设计了多目标AC-BM算法。利用自动机并操作技术构造多目标多模式树自动机,借助BM算法的坏字符跳转技术来计算文本串集移位。在Snort系统中分别实现2-目标AC-BM算法和3-目标AC-BM算法。实验结果表明,新算法如果在多个文本串中找到模式串就停止(表示检测到攻击行为),其在时间性能上就明显优于AC-BM算法。

**关键词** AC-BM算法,模式串,匹配搜索,自动机,坏字符跳转技术,Snort
**中图法分类号** TP393.08 **文献标识码** A

## Multi-objective AC-BM Algorithm Based on Automata Union Operation

WANG Zheng-cai XU Dao-yun WANG Xiao-feng
(School of Computer Science, Guizhou University, Guiyang 550025, China)

**Abstract** The AC-BM algorithm has the advantages that multiple pattern strings are searched simultaneously and that number of characters of moving text string is optimized. But, they are searched only in one text string in one time. To search in multiple text strings simultaneously, this paper designed multi-objective AC-BM algorithm. By union operation of two automatons, multi-objective multi-pattern tree automata was structured, and by BM algorithm's bad character move technique, function of moving a set of text strings was designed. In the Snort, 2-goal AC-BM algorithm and 3-goal AC-BM algorithm were implemented. On the condition that if in multiple text strings a pattern string is found, the algorithm stops, the result shows the new algorithm is obviously superior to AC-BM algorithm in time.

**Keywords** AC-BM algorithm, Pattern string, Pattern matching search, Automata, Bad character move technique, Snort

在网络安全技术不断发展的今天,入侵检测系统已成为一个重要安全防护工具。但网络带宽的不断增加,数据流量的不断增大,对入侵检测系统的处理速度提出了越来越高的要求。文献[1]对经典的Snort系统进行研究,发现字符串匹配操作占据了总的执行时间的31%。可见,在提高入侵检测系统的处理速度中,研究它的模式匹配算法,具有十分重要的意义。Snort是基于规则、完全免费、提供开源代码的入侵检测系统,容易在网上获取,为研究者研究入侵检测系统带来了很大的方便[2]。早期版本的Snort系统采用BM算法。BM算法[3]是经典的单模式匹配算法,当每次模式匹配不成功时,它对文本串移动的位数进行优化,大大减少了文本串移动次数,提高了算法的时间性能。随着Snort系统需要检测的对象增多,规则数不断增大,模式串的数量也随之增多。单模式匹配算法中,一个模式串需要搜索一次文本串,多个模式串需要搜索同一个文本串多次,匹配搜索耗时相当大。在新版Snort系统中,采用AC-BM算法。AC-BM算法[4]结合了AC算法和BM算法的优点。即,将多个模式串构建一个相应的模式树,形成有穷模式树自动机,文本串的一次搜索可以进行多个模式串的匹配操作,同时,文本串移位也得到了优化。这样,在规则数很大时,可以大大降低重复搜索文本串的次数。AC-BM算法在Snort系统中的成功运用,使研究者对它特别感兴趣,出现了很多改进算法。目前出现的改进主要有以下几类:对文本串移位的优化,如文献[5,6];对模式树进行优化,如文献[7-9];利用并发技术,降低搜索时间,如文献[10]。我们结合文献[10-14]的思想,提出同时对多个文本串进行多模式串匹配搜索,并称它为多目标AC-BM算法。在算法设计中,结合自动机并操作技术,构造多目标模式树自动机,并对多个文本串的移位和匹配搜索进行相应的修改。通过实验证明,改进的算法在目标为2(同时对两个文本串进行匹配搜索)时,与文献[10]中的双向AC-BM算法的时间性能差不多,但它只需一个模式树自动机运行,与原AC-BM算法的系统开销差不多;当目标为3(同时对三个文本串进行匹配搜索)时,其时间性能又得到了很大的提高。不过目标数增多后,多目标模式树自动机的构造复杂度也急剧增加。但模式树自动机针对相应的规则只需一次构造,而且规则不会随时变化。

## 1 模式匹配和自动机

模式匹配指在给定的两个串T(长度为$n$的文本字符串$T=T_{[0]}T_{[2]}\cdots T_{[n-1]}$)和$P$(长度为$m(m\ll n)$的模式字符串

到稿日期:2012-11-10 返修日期:2013-03-25 本文受国家自然科学基金项目(60863005,61011130038),贵州大学自然科学青年科研基金项目((2009)021),贵州大学研究生创新基金项目(省研理工2010005)资助。
王正才(1977-),男,博士生,CCF会员,主要研究方向为信息安全、网络安全、算法分析与设计,E-mail:xiaowangji@126.com。

$P=P_{[0]}P_{[2]}\cdots P_{[m-1]}$)中，查找 $T$ 中是否存在长度为 $m$ 的字符串 $P$。通常 $T$ 的长度远远大于 $P$ 的长度，若在 $T$ 中找到等于 $P$ 的子串，匹配成功，否则匹配失败。经典的模式匹配算法有单模式匹配算法和多模式匹配算法两类。该类算法一次只能在一个文本串中搜索，称它们为单目标模式匹配算法。提出了一次能在多个文本串中进行匹配搜索的算法，称之为多目标模式匹配算法。为了描述方便，给出如下定义。

**定义 1**（单目标多模式匹配，single-pattern matching in single string） 设字符集 $\Sigma$，且 $|\Sigma|=\sigma$，$\Sigma^*$ 为正闭包。文本串 $T=t_0t_1\cdots t_{n-1}$，模式串集合 $P=\{P_0,P_1,\cdots,P_{r-1}$ 其中 $P_i=p_{i0}p_{i1}\cdots p_{im_i-1}$，$0\leqslant i\leqslant r$，$m_i\ll n\}$，$T\in\Sigma^*$，$P\subset\Sigma^*$。称求解集合 $\{s|P_{i_k}=T_{s+k}$，$0\leqslant i<r$，$0\leqslant k<m_i\}$ 为单目标多模式匹配。

**定义 2**（多目标多模式匹配，multi-pattern matching in multi-string） 设字符集 $\Sigma$，且 $|\Sigma|=\sigma$，$\Sigma^*$ 为正闭包。文本串集合 $T=\{T_0,T_1,\cdots,T_{l-1}|$ 其中 $T_i=t_{i_0}t_{i_1}\cdots t_{i_{n-1}}$，$0\leqslant i<l$，$n=\max\{n_0,n_1,\cdots,n_{l-1}\}\}$，模式串集合 $P=\{P_0,P_1,\cdots,P_{r-1}|$ 其中 $P_j=p_{j_0}p_{j_1}\cdots p_{j_{m_j}-1}$，其中 $0\leqslant j<r$，$m_j\ll n\}$，$T\subset\Sigma^*$，$P\subset\Sigma^*$。称求解集合 $\{s|P_{j_k}=T_{i_{s+k}}$，$0\leqslant i<l$，$0\leqslant j<r$，$0\leqslant k<m_j\}$ 为多目标多模式匹配。

在 AC-BM 算法中，重要工具是自动机。一台确定型有穷自动机(DFA)可以用一个五元组来描述，即 $A=\{Q,\Sigma,\delta,s,F\}$，其中：$Q$：非空有限的状态集合；$\Sigma$：输入字符表集；$\delta$：转移函数($Q\times\Sigma\to Q$)；$s$：开始状态，且 $s\in Q$；$F$：接受状态集，$F\subseteq Q$。设有两台 DFA 自动机 $A_1=\{Q_1,\Sigma,\delta_1,s_1,F_1\}$，$A_2=\{Q_2,\Sigma,\delta_2,s_2,F_2\}$，它们的并自动机为 $B=\{Q_1\times Q_2,\Sigma,\delta_B,(s_1,s_2),F\}$，其中 $F=Q_1\times F_2\cup Q_2\times F_1$ 为接收状态集，$(s_1,s_2)$ 为 $B$ 的开始状态。$\delta_B$ 为 $B$ 的变迁函数，定义为：$\forall q_1\in Q_1,q_2\in Q_2,\sigma\in\Sigma$：$\delta_B((q_1,q_2),\sigma)=(\delta(q_1,\sigma),\delta(q_2,\sigma))$。并操作后的自动机每次仍只读取一个字符，我们需要一个能每次读取多个字符的自动机。下面给出一个读取双字符的并自动机的定义。

**定义 3**(双字符并自动机) 设有两台 DFA 自动机 $A_1=\{Q_1,\Sigma,\delta_1,s_1,F_1\}$，$A_2=\{Q_2,\Sigma,\delta_2,s_2,F_2\}$，它们并操作得到的 DFA 双字符并自动机为 $B=\{Q_1\times Q_2,\Sigma\times\Sigma,\delta_B,(s_1,s_2),F\}$，其中 $F=Q_1\times F_2\cup Q_2\times F_1$ 为接收状态集，$(s_1,s_2)$ 为 $B$ 的开始状态。$\delta_B$ 为 $B$ 的变迁函数，定义为：$\forall q_1\in Q_1,q_2\in Q_2,(\sigma_1,\sigma_2)\in\Sigma\times\Sigma$：$\delta_B((q_1,q_2),(\sigma_1,\sigma_2))=(\delta(q_1,\sigma_1),\delta(q_2,\sigma_2))$。

## 2 AC-BM 算法

AC-BM 算法包括 3 个主要部分：构造基于模式串集合的 AC 模式树自动机、确定移位函数、利用 AC 自动机和移位函数搜索文本串。

1. 用模式串集合构建模式树自动机。模式树自动机构建基于字符串的前缀，相同的前缀作为树的根节点。匹配搜索时，将模式树中最短的模式串右端的字符 $P_{[m-1]}$ 和文本串的最右字符 $T_{[n-1]}$ 对齐。

2. 匹配搜索时采取自后向前的方向。字符比较从左至右进行(从模式树自动机的根字符开始向节点方向按层逐个字符进行比较)。比较之前可以进行预处理，以减少不必要的比较。

3. 文本串移位同时使用坏字符和好后缀移位规则。坏字符移位：当模式树中的字符与文本串的字符 $Q$ 不匹配时，将文本串移到下一个 $Q$ 出现的位置，让模式树中的 $Q$ 和文本串中的 $Q$ 对齐，如果在模式树中不存在字符 $Q$，则将文本串向右移动模式串集中最小字符串长度。好后缀移位：当模式树中的字符与文本串中的字符 $Q$ 不匹配时，将自动机移动到模式树中最先与文本串部分匹配的下一位置。在移动过程中一定要保证自动机的移动量不大于模式串集中最短的模式串。为了简单性，我们只考虑坏字符移位情况。

设有模式串集合{BPMRDO，BPMGRDO，BPMVRDO}，要检查的文本串内容为 DISHGLBRWOZ…BSJMVGLOPID，在 AC-BM 算法中模式匹配搜索为：将模式树(模式串集合中位数最少的模式串右端)和文本串的右端对齐，如图 1 所示的字符 O(红色)和字符 D(红色)对齐。字符比较从模式树的根节点开始，图 1 中字符 B(粉红色)和字符 G(蓝色)不匹配，模式树的下一个 G(蓝色)出现在深度为 4 的位置。文本串向右移动位数计算为：设最短模式串长度为 min $l$，最长模式串长度为 max $l$，如果当前坏字符在模式树中不出现或者出现深度大于 min $l$，则向右移动位数为 min $l$，否则为坏字符在模式树中出现深度减 1。文本串移动后如图 2 所示。

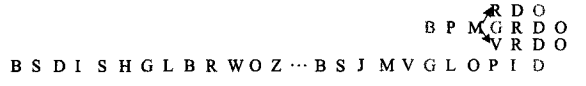



图 1 AC-BM 算法匹配前对齐

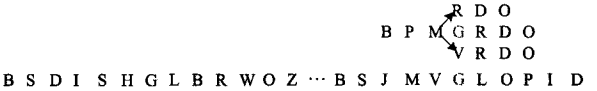


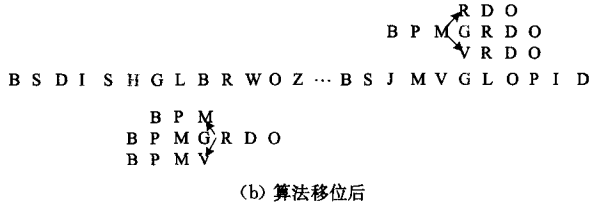

图 2 AC-BM 算法移位后

文本串长度为 $n$ 时，时间复杂度在最优情况下为 O($n$/min$l$)，在最坏情况下为 O($n$ * max$l$)。

## 3 多目标 AC-BM 算法

AC-BM 算法改进中，一种有效的方法是双向 AC-BM 算法。它在 AC-BM 算法的基础上，再构造一个反向模式树自动机(一般把原来的模式树自动机称为正向模式树自动机)。其基本思想是用两个模式树自动机对一个文本串同时进行匹配搜索。上面的例子在双向 AC-BM 算法中的匹配搜索如图 3 所示。


(a) 匹配前对齐




(b) 算法移位后

图 3 双向 AC-BM 算法运行示意图

双向 AC-BM 算法设计和 AC-BM 算法差不多，只是增加了一个模式树自动机，由两个自动机并发匹配搜索。当 $n$ 很大时，可以把对文本串的模式匹配搜索时间减小很多，以提高 AC-BM 算法的时间性能。但两个模式树自动机并发运行，增加了系统开销。当文本串数量很多，或者文本串长度很大时，其时间效率并不是最理想。这些算法都是每个文本串执行一次匹配搜索，多个文本串就执行多次匹配搜索。我们的思想是针对多个文本串只进行一次匹配搜索。利用自动机组合技

术，构造一个特殊的多目标多模式树自动机(我们将原来只能对一个文本串进行匹配搜索的模式树自动机称为单目标多模式树自动机)。这样不但可以对多个文本串同时进行搜索(长文本串可以预处理为多个子文本串)，减少文本串的搜索时间，而且只用一个模式树自动机，减小了系统开销。为了描述的简单性，下面的描述中只考虑两个文本串同时进行匹配搜索的情况，即只设计 2 目标 AC-BM 算法。其它目标数原理一样，首先对上面的例子增加一个文本串，即：模式串集合为{BPMRDO,BPMWRDO,BPMWRDO}，文本串集合为 $\boldsymbol{T}=T=\{T_1,T_2\}$，$T_1$ = DISHGLBRWOZ…BSJMVGLOPID，$T_2$ = FIDHBLBCWGZ…BEJMGGLGPDD。根据 AC-BM 算法的结构，对 3 个部分进行详细的设计。

### 3.1.2 目标多模式树自动机

在 AC-BM 算法中，模式串集合{BPMRDO,BPMWRDO,BPMWRDO}对应的模式树自动机如图 4 所示。状态集为 $Q=\{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15\}$，其中 0 为初始状态，{6,10,14,15}为终结状态集，且称 15 为失败状态(用黑色加粗圈表示。如果到达这个状态，匹配搜索失败，需进入文本串移位操作)，则称{10,14,15}为成功状态集(用红色加粗圈表示。如果达到它们，说明一个对应的模式串匹配成功)。除了初始状态和终结状态的其他状态为中间状态。对于中间状态，为了画图方便，都省略了一个到状态 15 的变迁，即当它们收到的字符不是通往下一个状态变迁的字符时，它们将到达失败状态 15。Σ 为输入字符集。当自动机运用结束时，如果处在成功状态集，则报告成功，并输出相应匹配串；如果处在失败状态，则进行文本串移位。为了描述方便，我们用树状自动机的根状态名加粗带下划线表示它，图 4 的自动机可以记为$\underline{\mathbf{0}}$,7为$\underline{\mathbf{0}}$中一个分支自动机，如图 5 所示。

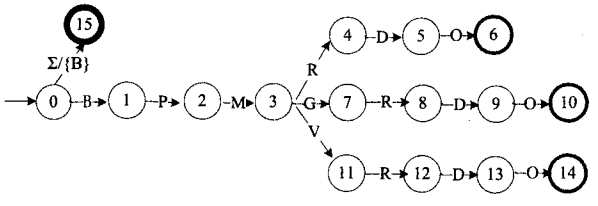



图 4 模式树自动机

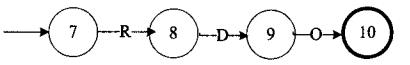



图 5 7表示的自动机

在自动机变迁中，变迁的条件是读到相应字符。在双目标多模式匹配搜索中，每次读取的是一对字符序列，因而将自动机的变迁分为双字符变迁和单字符变迁，单字符变迁又可分为 $T_1$ 单字符变迁和 $T_2$ 单字符变迁。由不同变迁构成的自动机分别称为双字符变迁自动机和单字符变迁自动机。

**定义 4**(双字符变迁和单字符变迁) 由输入字符序列中的两个字符同时确定的变迁称为双字符变迁，只根据输入字符序列中的一个字符发生的变迁称为单字符变迁。只根据 $T_1$ 文本串中字符变迁的单字符变迁称为 $T_1$ 单字符变迁，同理，$T_2$ 单字符变迁指只根据 $T_2$ 中的字符变迁。

**定义 5**(双字符变迁自动机和单字符变迁自动机) 由双字符变迁构成的自动机称为双字符变迁自动机，由单字符变迁构成的自动机称为单字符变迁系统。

例子的模式串集的双字符变迁自动机由两个图 4 的单目标多模式树自动机的双字符并可得，如图 6 所示。$T_1$ 单字符变迁自动机和 $T_2$ 单字符变迁自动机分别如图 7 和图 8 所示。

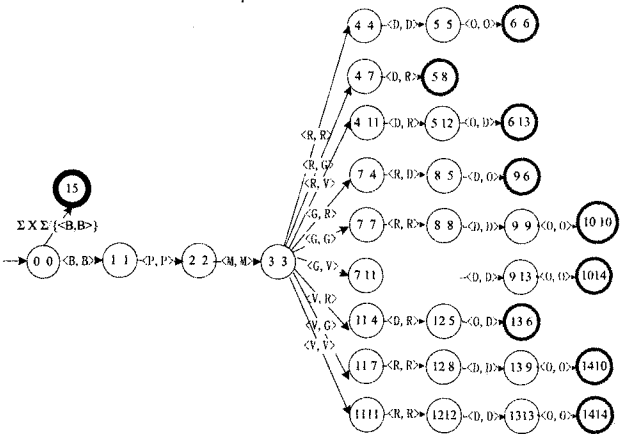



图 6 双字符变迁自动机

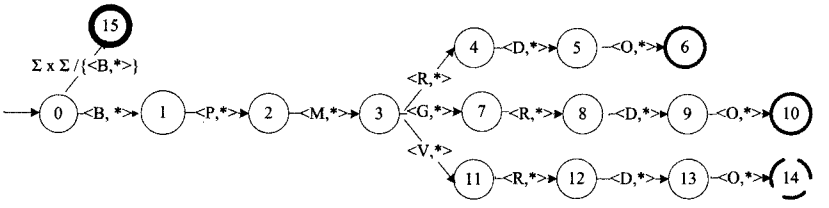



图 7 $T_1$ 单字符变迁自动机

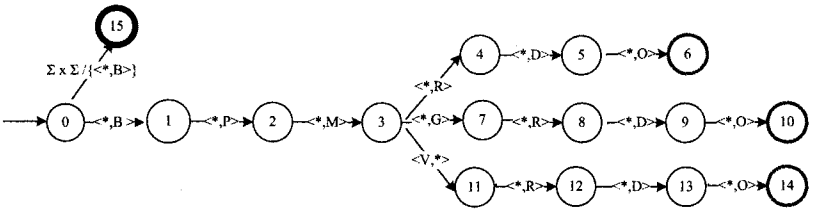



图 8 $T_2$ 单字符变迁自动机

单字符变迁自动机和图 4 的模式树自动机一样，唯一区别在于它们的输入是一对字符序列，而不是一个字符。在双字符变迁自动机中，初始状态为(0 0)，终结状态用加粗圈表示。其中黑色加粗圈状态 15 表示失败，红色加粗圈状态表示成功，其它状态称为中间状态。图中也省略了中间状态接收到的字符序列不是通往下一状态的字符序列，而将到达失败

状态 15 的变迁。

对于 2-目标多模式树自动机，既要考虑双字符发生的变迁，又要考虑单字符发生的变迁。对于双字符发生的变迁可以由图 6 所示的自动机操作，单字符变迁由图 7 和图 8 的自动机操作。图 6 到失败状态变迁的发生，一种是由于输入的字符序列中两个字符都与状态到达下一状态的字符序列中的字符不相同，另一种是只有一个相同。对于前者，说明当前两个文本串的搜索窗口中的字符串不包含模式串集中的任一模式串，不需要继续搜索下去；后者还需要对字符相同的文本串继续进行匹配搜索。因而我们将到达失败状态的变迁分为半失败变迁和全失败变迁。对于半失败变迁，它应该进入一个单字符变迁自动机。这里用 T1-1、T1-2、T1-3…表示针对图 7 的自动机1、2、3，用 T2-1、T2-2，T2-3…表示针对图 8 中的自动机1、2、3…。对模式串集{BPMRDO，BPMWRDO，BPMWRDO}和两个文本串同时搜索的模式树自动机如图 9 所示。为了画图方便，省略了一部分，但不会影响对 2-目标多模式串自动机构造的理解。

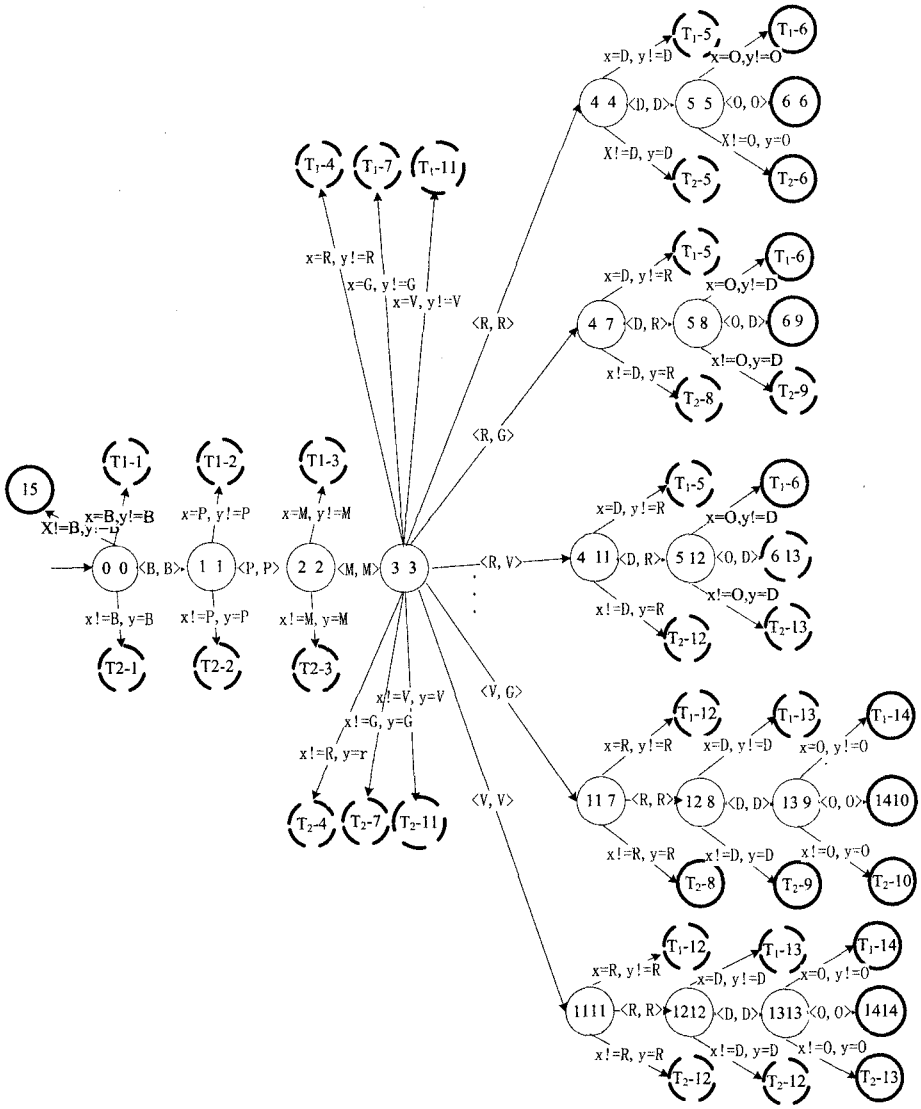



图 9　多目标多模式自动机

### 3.2　移位函数

2-目标 AC-BM 算法的文本串移位计算方法与 AC-BM 算法差不多，唯一的区别是，它处理的是两个文本串的移动。在 AC-BM 算法中，当失败变迁发生而停止自动机运行时，只需记住当前读入的字符，用它作为坏字符计算移位数。在双目标多模式匹配搜索中，有全失败变迁和半失败变迁，只有全失败变迁发生时，才需对文本串移位。我们设置一个数组{fai-lure1，failure 2}，并将它们初始化为 null。坏字符的确定过程如下：

当半失败变迁发生时：

Failure1 被赋予当前读入的第一个字符，如果自动机进入标有 $T_2$ 的单字符变迁自动机；

Failure2 被赋予当前读入的第二个字符，如果自动机进入标有 $T_1$ 的单字符变迁自动机。

当全失败变迁发生时：

坏字符为当前读入的字符，如果 failure1＝null 且 failure2＝null；

坏字符为当前读入的第一个字符和 failure2，当 failure1＝null；

坏字符对为 failure1 和当前读入字符序的第二个字符，当 failure2＝null。

当坏字符序列确定后，分别按坏字符跳跃移位计算法算出相应的移位数，获得移位序列。

### 3.3 文本串的搜索

文本串的搜索过程如下：

1. 将两个文本串右对齐；

2. 将模式串集中最短串与文本串右端对齐；

3. 从模式树的根节点开始，从左往右进行模式匹配操作。这步通过调用构造的 2-目标多模式树自动机来完成。

4. 根据移位函数计算出的移动位数序列，分别对文本串向右移动。当某个文本串长度不够时，在文本串左端添加空字符 null；当两个文本串的字符都是 null 时，匹配搜索结束。

## 4 系统测试

根据上面的工作，我们使用 VC6.0 对 Snort 里面的 Importkey()函数、MakeMr()函数和 PatternMatch()函数进行相应的修改。测试环境为：Snort2.0 默认值规则集，测试机 CPU 为 AMD5200+，内存 2G，操作系统为 WinXP。测试数据采用 1999 年美国高级研究计划局(DARPA)做 IDS 评估时所使用的数据集。分别对 AC-BM 算法、双向 AC-BM 算法、2-目标 AC-BM 算法和 3-目标 AC-BM 算法进行测试。我们主要是考虑算法的时间效能。测试结果如表 1 所列。

表 1 测试结果(单位 s)

| 时间 | 测试数据 | AC-BM | 双向 AC-BM | 2-目标 AC-BM | 3-目标 AC-BM |
|---|---|---|---|---|---|
| 周一 | inside. tcpdump(325M) | 208.9967 | 169.4463 | 170.5543 | 137.4578 |
| | outside. tcpdump(308M) | 196.1908 | 152.2133 | 149.4321 | 101.4376 |
| 周二 | inside. tcpdump(325M) | 141.0486 | 100.2637 | 110.2477 | 85.6512 |
| | outside. tcpdump(310M) | 131.4404 | 92.2342 | 98.8960 | 77.9874 |
| 周三 | inside. tcpdump(367M) | 285.5676 | 213.2512 | 200.4333 | 169.0012 |
| | outside. tcpdump(351M) | 272.8887 | 201.1235 | 189.7216 | 145.5622 |
| 周四 | inside. tcpdump(527M) | 219.8750 | 175.1251 | 164.3234 | 137.2345 |
| | outside. tcpdump(493M) | 203.2156 | 167.4234 | 174.6643 | 140.6775 |
| 周五 | inside. tcpdump(294M) | 144.3443 | 102.2134 | 98.4311 | 80.4499 |
| | outside. tcpdump(271M) | 131.0623 | 98.1234 | 102.4452 | 82.4457 |

改进的算法在 2-目标下，时间性能和双向 AC-BM 算法差不多，但它只运行一个模式树自动机，系统开销和 AC-BM 算法差不多。当目标增加到 3 时，可以看到时间性能有了很大的提高。随着目标的增多，相应的模式树自动机的构造会越复杂，但不会带来严重的影响：(1)模式树自动机需要提前构造；(2)规则不会太频繁变化。

## 参 考 文 献


[1] Navaro G R M. Flexible Pattern Matching in Strings[M]. Cambridge University Press, 2002

[2] Roesch M, Green C. Snort users manual [OL]. https://www.Snort.org

[3] Boyer R S, Moor J S. A fast string searching algorithm[J]. Communications of the ACM, 1977, 20(10): 762-772

[4] Fan Jang-jong, Su K. An Efficient Algorithm for Matching Multiple Patterns[J]. IEEE Transactions on Knowledge and Data Engineering, 1993, 5(2): 339-351

[5] 周四伟，蔡勇. AC-BM 算法的改进及其在入侵检测中的应用[J]. 微计算机应用，2007，28(1)：27-31

[6] 万国根，秦志光. 改进的 AC-BM 字符串匹配算法[J]. 电子科技大学学报，2006，35(4)：531-534

[7] 姚亚锋，蒋毅. 模式匹配算法及其优化[J]. 南通职业大学学报，2011，25(4)：98-100

[8] Hou Zheng-feng, Zhang Xiao-le. Research and improvement of AC-BM algorithm[J]. Chinese Journal of Scientific Instrument, 2011, 3(2): 216-221

[9] Wu Pei-fei. The research and amelioration of pattern-matching algorithm in intrusion detection system[C]// Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications(HPCC-2012). 2012: 1712-1715

[10] 杨超. 双向 AC 算法及其在入侵检测系统中应用[J]. 计算机应用，2011，20(3)：222-225

[11] 黄海. 字符串匹配算法通用并行技术研究[D]. 西安：西安建筑科技大学，2010

[12] Miao Chang-sheng, Chang Gui-ran, Wang Xing-wei. Filtering Based Multiple String Matching Algorithm Combining q-Grams and BNDM[C]// Proceedings of the 2010 Fourth International Conference on Genetic and Evolutionary Computing. ACM, 2010: 582-585

[13] Zhang Wei-zhe, Zhang Yuan-jing, Zhang Hong-li, et al. A Memory-Efficient Multi-pattern Matching Algorithm Based on the Bitmap[C]// Proceedings of the 2009 Fourth International Conference on Internet Computing for Science and Engineering. ACM, 2009: 1-5

[14] Chen Xin-ming, Ge Kai-lin, Chen Zhen. AC-Suffix-Tree: Buffer Free String Matching on Out-of-Sequence Packets[C]// Proceedings of the 2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems, 2011: 36-44


(上接第 99 页)


[19] Chaum D, Van Heijst E. Group signatures[C]// Brighton, UK. Proceedings of EUROCRYPT 1991. Berlin/Heidelberg: Springer, 1991: 257-265

[20] Chase M, Lysyanskaya A. On signatures of knowledge [C]// Santa Barbara, California, USA. Proceedings of CRYPTO 2006. Berlin/Heidelberg: Sprin-ger, 2006: 78-96

[21] Belenkiy M, Camenisch J, Chase M, et al. Randomizable proofs and anonymous credentials [C] // Santa Barbara, California, USA. Proceedings of CRYPTO 2009. Berlin/Heidelberg: Springer, 2009: 108-125

[22] Wen Ya-min, Zhang Fang-guo. Delegatable secret handshake scheme[J]. Journal of Systems and Software, 2011, 84(12): 2284-2292

[23] Zhang Fang-guo, Chen Xiao-feng, Susio W, et al. A new signature scheme without random oracles from bilinear pairings [C]// Hanoi, Vietnam. Proceedings of VIETCRYPT 2006. Berlin/Heidelberg: Springer, 2006: 67-80

[24] Brands S. An efficient off-line electronic cash system based on the representation problem [R]. CS-R9323. CWI (Centre for Mathematics and Computer Science) Amesterdam, The Newthelands, Apr. 1993

[25] Pointcheval D, Stern J. Security proofs for signature schemes[C]// Saragossa, Spain. Proceedings of EUROCRYPT 1996. Berlin/Heidelberg: Springer, 1996: 387-398