

# 一种防火墙规则冲突快速检测算法

徐艳,董涛

(电子科技大学成都学院,四川 成都 611731)

**摘要:**目前,在防火墙规则冲突检测算法中,效率问题一直没有很好的解决,当防火墙规则数目较大时,检测规则冲突的速度很难满足客户的需要。为了能够快速检测出防火墙中的规则冲突,在目前使用较多的 ASBV 算法上提出一种规则冲突检测算法 (DBBV 算法)。该算法采用的方法是使用位向量和分治技术,该设计在检测规则冲突的时候,设计的算法只是进行了一次位运算。同时该算法采用的是范围形式的规则集。经过对算法详细的分析,以及通过实验方法的验证,改进的 DBBV 算法的规则冲突检测效率明显高于 ASBV 算法。

**关键词:**ASBV 算法;DBBV 算法;算法设计

中图分类号:TP309

文献标识码:A

文章编号:1673-629X(2013)09-0128-03

doi:10.3969/j.issn.1673-629X.2013.09.032

## A Fast Algorithm for Detecting Firewall Rule Conflict

XU Yan, DONG Tao

(Chengdu College of University of Electronic Science and Technology, Chengdu 611731, China)

**Abstract:** The current conflict detection algorithm has low efficiency, detection rule conflict speed can not meet the needs of users when existing large amount of firewall rule. In view of this situation, based on ASBV algorithm, put forward a kind of conflict detection algorithm DBBV. The algorithm used is divided conquer and bit vector technology, in the rule component process, DBBV algorithm will have a bit vector intersection. Simultaneously DBBV algorithm adopts range representation of a rule set. Based on the algorithm analysis detailed, verification with experiment way, found that DBBV algorithm for the detection of rule conflict is faster than ASBV algorithm.

**Key words:** ASBV algorithm; DBBV algorithm; algorithm design

## 0 引言

由于防火墙<sup>[1]</sup>技术的快速发展,针对配置的规则数量也在不断增加,从规则管理和运行的速度的角度,规则冲突的检测算法优劣显得尤为重要。很多的规则匹配算法采用了多种数据结构,这是为了加快算法的分类速度,可是这也带来了负面的作用,在这种情况下,就很容易产生规则的冲突,当规则数量增加到很多的时候,冲突的概率就会加大,这对数据包的匹配会产生很大影响,可能会放过错误的数据包,也可能丢弃正常的数据包,这对防火墙的性能无疑有很大的危害。并且规则冲突检查的快慢也是制约性能的重大因素,因此,有必要研究更加快速的防火墙规则冲突的检测算法。

因此,在目前使用较多的 ASBV 算法上提出一种规则冲突检测<sup>[2]</sup>算法 (DBBV 算法)。该算法思想是

采用位向量和分治技术,设计的算法只进行了一次位运算。同时该算法采用的是范围形式的规则集。最后经过对算法详细的分析,以及通过实验方法的验证,改进的 DBBV 算法的规则冲突检测效率明显高于 ASBV 算法<sup>[3]</sup>。

## 1 DBBV 算法

### 1.1 算法描述

在一个规则集合  $T$  里面,查出与其中某一条规则 ( $H_i$ ) 有冲突的集合,如果查找出了  $T$  中所有与之相冲突的规则,只需要循环使用 DBBV 算法即可。该算法首先是能够并行地处理规则中的每一维分量,找出该维中与  $H_i$  有冲突的所有规则的条数,再找出这些规则集合相交的部分,那么这个交集集中的元素就是与规定的规则有冲突的所有规则集合。这就是说,对有  $K$  维

的规则进行冲突检测时,是分步对每一维进行的处理<sup>[4]</sup>。

在规则集  $T$  中,与某条规则  $H_t$  有冲突的所有规则组成的集合定义为  $C(H_t)$ ,在第  $t$  维上,与  $H_t$  有冲突的规则集合定义为  $C_t(H_t)$ 。 $C(H_t)$ 、 $C_t(H_t)$  分别有对应长度为规则个数的位向量。在规则集合  $T$  中, $\forall t, 1 \leq t \leq m, t \in N$ ,都会存在  $t$  维分量的一个集合  $\{T1t, T2t, \dots, Tnt\}$ ,从这个集合中可以获取两个子集,第一个是  $t$  维分量的开始集合  $B(t) = \{Tb1, Tb2, \dots, Tbn\}$ ,还有一个是结束地址的集合  $E(t) = \{Te1, Te2, \dots, Ten\}$ 。这两个集合每一元素都对应着一个位向量。 $B(t)$  的位向量设定为  $BBV(t)$ ;  $E(t)$  的位向量设定为  $EBV(t)$ 。位向量的值就是该维对应的每条规则相同  $t$  维分量是否是冲突的。

1.2 DBBV 算法数据结构

如何构造 DBBV 算法的数据结构是影响算法性能的一个关键因素,下面描述数据结构的建立,下面的讨论中设定防火墙的规则为  $K$  维。

DBBV 算法可以描述为:根据  $t$  维规则分量,建立起两棵二叉树,分别是由  $B(t)$ 、 $E(t)$  中的元素形成。两个集合中的元素构成了这棵二叉树的所有的节点,也就对应了一个位向量—— $BBV(t)$  或者  $EBV(t)$ 。

接下来,以表 1 中的规则集合描述。表中给出了规则集合中每个规则的  $t$  维分量,它的值设定在  $[0, 15]$ ,表中一共有 10 条规则,其建立的二叉树如图 1 和图 2 所示。二叉树中节点的值表示的是  $B(t)$  或者  $E(t)$  中的元素,矩形中的二进制串表示的是  $BBV(t)$  或者  $EBV(t)$ 。

表 1 规则集第  $t$  维分量

规则分量	规则分量的值
$T1t$	$[3, 7]$
$T2t$	$[4, 6]$
$T3t$	$[0, 3]$
$T4t$	$[0, 7]$
$T5t$	$[10, 13]$
$T6t$	$[5, 9]$
$T7t$	$[5, 8]$
$T8t$	$[8, 10]$
$T9t$	$[12, 15]$
$T10t$	$[14, 15]$

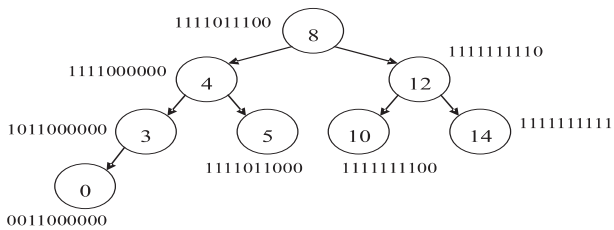


图 1  $B(t)$  生成的二叉树

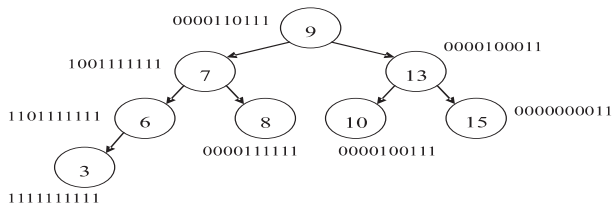


图 2  $E(t)$  生成的二叉树

1.3 DBBV 算法采用的冲突检测过程

(1) 为了找到一个合适的  $Tb$ ,通过  $B(t)$  对应的二叉树中搜索,就需要满足以下条件:如果  $Tbi \leq b$ ,且不存在  $p, 1 \leq p \leq n, BBVi(t)$  就是对应的位向量。假如没有满足这一条件的,就认为  $C_t(H_t) = \emptyset$ 。这一执行程序就结束了,算法执行完毕。

(2) 通过  $E(t)$  对应的二叉树来搜索  $a$ ,从而找到一个合适的  $Tei$ 。此时,  $Tbei$  需要满足的条件如下:如果  $a \leq Tei$ ,且不存在  $p, 1 \leq p \leq n$ ,从而得到的  $EBVi(t)$  这一位向量。但是如果满足此  $Tbei$  的条件不存在,就认为  $C_t(H_t) = \emptyset$ 。

(3) 对  $BBVi(t)$  和  $EBVi(t)$  进行位向量交集运算,被置为 1 的位所对应的规则,在  $k$  维上与  $H_t$  发生了冲突,那么这一  $k$  维对分量冲突进行检测的过程就结束了。

通过如下的例子来详细地阐述  $t$  维的处理流程。假设  $R1$  被检测的规则是  $Rt$ ,它的  $t$  维分量是  $[3, 7]$ ,在  $B(t)$  在二叉树相应的位置搜索 7,就会发现  $T6t$  的范围位于 5 与 7 之间,找到位向量  $BBV6(t)$ ,于是可以在二叉树的  $E(t)$  集合中找到,发现  $T3t$  和 3 相等。得到了  $EBV3(t)$  的位向量是 1111111111,最终可以得到如下结果:

$FBV(Rt) = BBV6(t) \cap EBV3(t) = 1111011000$

从这个值可以看出,  $R1$  在第  $t$  维分量,与下面一些规则是相冲突的:  $R1$ 、 $R2$ 、 $R3$ 、 $R4$ 、 $R6$ 、 $R7$ 。

1.4 规则更新操作

在所有的规则集  $T$  中, DBBV 算法既能够发现全部的发生冲突的规则,当再添加新的一些规则时,又能够发现先前的一些规则与刚才添加的一些规则发生冲突的部分。尽管前者和后者有相似之处,但是实际上也存在很多的不同之处,因为前者只需要进行  $n$  次冲突检测,后者不仅需要进行一次冲突检测,还需要对 DBBV 算法进行数据结构的更新。

DBBV 算法在进行更新操作时,不仅要要对二叉树进行修改,而且还要对位向量进行处理,此算法虽然可以利用重排序等优化措施,并且能够对全部的位向量进行处理,比方说修改处理,每当添加一些操作时,此算法就会浪费掉很多的时间。所以针对一些更新较为频繁,速度较快的应用程序,会表现出以下的状况:此算法对数据进行更新时,下一个新的规则就会立马进

行冲突的检验,这样的过程会造成后来添加的新规则不能够及时地进行冲突检验了。针对这一状况,此算法采用了一个较简单的方案。而此方案的基本思想就是:更新操作的个数积累到一定数量后,会一次性地完成对数据结构的修改处理。其具体的操作方法如下:

- (1) 在配置更新的过程中,会请求一个队列  $Q$ ,并设置一个阈值  $r$ 。
- (2) 在进行更新的操作中,会把更新的请求放入到此队列中,此队列既要记录这些规则的信息,还要对更新的请求类型加以记录。
- (3) 当  $Q$  中的更新请求的个数多于  $r$  的时候,那么有线程便开始运作了,这个线程就是负责更新操作的。开始复制目前使用的数据结构,称一个数据结构为当前缓存,这个数据结构就是冲突检测的数据结构,同时称另一个为备份缓存<sup>[5]</sup>。进而从  $Q$  中陆续地读出  $r$  个更新请求,从而删除这些请求。冲突检测完毕后,新规则就会被放入队列中,这样一来,当下一个新的规则进行冲突检测时,就能够进行及时地处理了。

2 仿真实验

文中进行了算法的对比,即 DBBV 与 ASBV 算法。主要是针对冲突检测的复杂度,算法所构造的数据结构的复杂度等方面。

由 DBBV 算法主要构造了二叉树和位向量的计算这两个步骤的数据结构。因为该算法能够对每一棵二叉树进行并行处理,构造二叉树所需要的时间复杂度为  $O(n\log n)$ ,而  $O(n)$  是构造位向量计算的时间复杂度,因此  $O(n\log n)$  是总的构造时间复杂度<sup>[7]</sup>。文中给出了这两个算法在构造数据结构时所需时间的对比。

由表 2 可知,DDBV 算法所需要的构造时间少于 ASBV 所需要的构造时间。比如,在处理 FW4 时,DDBV 算法所需要的平均时间是 117.66 ms,而 ASBV 所需要的平均构造时间为 403.44 ms。形成此现象的原因是:这两种算法在构造的过程中,ASBV 算法不仅要构造 Trie,还要为每一个二叉树的节点计算出两个位向量<sup>[6]</sup>。每一个节点关联了 4 个位向量,由此计算的复杂度为  $O(n^2)$ ,而另一个造成该算法速度慢的原因就是该算法只支持以前缀形式表示的规则集。文中所使用的规则集,许多的规则分量都以范围形式表示,所以此算法在构造的过程中,需要把范围式转化为前缀式<sup>[7]</sup>。因此,可以得知,即在最坏的情况下,一个范围形式可以转换成  $2w-2$  个前缀形式;然而对于  $m$  维的规则,在最坏的情况下,以范围式表达的规则可以转化为  $(2w-2)$  以前缀形式表达的规则。因为根据一系列的转换,ASBV 算法在处理 FW1 时,就多了 48 条规则,以此类推,再处理 FW4 时,就多了 1 651 条规则。

表 2 数据结构的构造时间对比

规则集	规则个数	DBBV 算法平均构造时间/ms	ASBV 算法平均构造时间/ms
FW1	94	1.88	3.93
FW2	340	8.44	18.12
FW3	946	25.47	58.43
FW4	1 961	117.66	403.44

在规则冲突检测的过程中,DDBV 这一算法,将分为两个步骤来完成。首先,对每一维规则的分量进行并行处理,当然这其中还包括位向量的交集运算和树形搜索。这一步的时间复杂度为  $O(\log n + n/\text{wordsize})$ 。然后就是对产生的  $m$  个位向量进行交集运算。这步产生的时间复杂度为  $O(mn/\text{wordsize})$ 。第一步的时间复杂度远远小于第二步,因此 DBBV 算法的时间复杂度可以近似地等于  $O(mn/\text{wordsize})$ 。

文中这两种算法在规则冲突检测时,位向量的计算是对时间复杂度影响的重要因素。因此在进行的仿真实验的过程中,就对这一信息做了重点记录。

采用实验方法是使用 C++ 语言编写程序实现两种算法,对规则集中每一条规则都进行一次冲突检测,记录所有的位向量的运算次数,实验结果如表 3 所示。

表 3 算法性能对比

规则集	规则个数	DBBV 算法位向量次数	ASBV 算法位向量次数
FW1	94	2 538	8 630
FW2	340	33 660	100 512
FW3	946	255 420	1 020 261
FW4	1 961	1 094 238	7 285 562

从表中可以看出,随着规则条数的逐渐增加,两种算法相对应的位向量的运算次数也在不断增多,可是 ASBV 算法所需的位向量运算远超过 DBBV 算法<sup>[8]</sup>。从 FW1 开始,ASBV 算法的位向量运算次数是 DBBV 算法的三倍多,随着规则条数的增加,到 FW4 时,前者的运算次数已经增长到超过 DBBV 算法的六倍。经分析后得出结论,造成这种状况的原因如下:在对每一维进行处理的过程中,ASBV 算法做了多次的位向量并集的运算,而 DBBV 算法只是采用一次的位向量交集运算,从最坏的情况下来看,在每一维的处理中,ASBV 算法的规则冲突时间复杂度就大大增加。

下面分析 ASBV 算法和 DBBV 算法各自的空间使用量。DDBV 算法的空间复杂度为  $O(n^2)$ ,ASBV 算法的空间复杂度也是  $O(n^2)$ ,因为  $2m$  棵二叉树是由 DBBV 构造出来的,而二叉树的每个节点都关联一个位向量,每一个位向量又有  $n$  个 bit。ASBV 算法构造  $m$

(下转第 134 页)



加而受影响,表明基于颜色场结构和高斯金字塔的信息隐藏算法的信息隐藏算法抗击此类检测分析。

## 4 结束语

文中将颜色场结构和高斯金字塔理论分别用于载体的空间特性和能量解析,利用高斯金字塔方法,对图像进行能量集中化处理,通过高斯低通滤波和亚采样将载体图像分层,随着图层高度增加,图像的低频成分集中,即能量集中。据此可选择满足信息隐藏鲁棒性和不可见性的图层进行不同内容的嵌入。对分层图像进一步做颜色场结构处理,利用方向值代表载体图像的数据信息。该算法具有良好的系统特性,不可见性提高了 29.21%,对 JPEG2000 压缩、随机剪切、旋转、[3,3]均值滤波、[3,3]二次维纳滤波均具强的鲁棒性。通过应用 RS 和 HOSWC 检测分析算法对基于颜色场结构和高斯金字塔的信息隐藏算法进行抗分析性能检测,得出算法具有较强的抗分析性能。另外,文中提出的载体预处理方法对信息隐藏算法性能有明显的改善作用。

### 参考文献:

- [1] 袁琦,闵栋,邹俊伟. KML 文件的信息隐藏技术[J]. 北京邮电大学学报,2011,34(1):140-144.
- [2] 曹玉强,龚卫国,柏森,等. 基于 Curvelet 变换的鲁棒信息隐藏算法[J]. 计算机工程,2011,37(5):137-139.
- [3] Zhang T, Mu D, Ren S. Information hiding algorithm based on

(上接第 130 页)

棵二叉树,二叉树的每个节点会关联四个位向量。而一个 bit 只能对应于一条规则,尽管空间复杂度是平方的关系,然而实质上所消耗的不是很大。可见,DBBV 算法在冲突检测方面是优于 ASBV 算法的。

## 3 结束语

DBBV 算法提高了 ASBV 算法中的时间复杂度,在处理规则冲突检测<sup>[9]</sup>上采用的是一维运算后再进行交集运算,使得规则检测更加简单。在规则表达灵活性方面,DBBV 算法明显要比 ASBV 算法好,不会产生多余的位运算,造成时间复杂度的升高。通过对算法理论的分析,可以表明 DBBV 算法在冲突检测方面的性能是优于 ASBV 算法的。

### 参考文献:

- [1] 吕海涛,梁祖华. 基于防火墙规则匹配优化算法的研究[J]. 计算机安全,2008(3):17-19.
- [2] 梁建武,龙晓梅,刘军军. 基于 LE-Trie 的防火墙策略检测

Gaussian pyramid and GHM multi-wavelet transformation [J]. International Journal of Digital Content Technology and Its Applications,2011,5(3):210-218.

- [4] 徐凯平,郑洪源,丁秋林. 一种基于 LSB 和 PVD 的图像信息隐藏方法研究[J]. 计算机应用研究,2010,27(3):1068-1070.
- [5] 赵彦涛,李志全,董宇青. 基于排序和直方图修改的可逆信息隐藏方法[J]. 光电子·激光,2010,21(1):102-107.
- [6] 蒋晓瑜,黄应清. 基于小波变换的多分辨模板匹配[J]. 中国图象图形学报,2000,5(4):304-308.
- [7] 黄伟,王书文,杨筱平,等. 基于图像分解的敦煌壁画图像修复方法[J]. 山东大学学报(工学版),2010,40(2):24-27.
- [8] 牛夏牧,陆哲明,孙圣和. 基于多分辨率分解的数字水印技术[J]. 电子学报,2000,28(8):1-4.
- [9] 刘贵喜,杨万海. 基于多尺度对比度塔的图像融合方法及性能评价[J]. 光学学报,2001,21(11):1336-1342.
- [10] Wang Y S, Sun J, Wang C J, et al. Prediction of the chaotic time series from parameter-varying systems using artificial neural networks[J]. Acta Physica Sinica,2008,57(10):6120-6131.
- [11] Lou Der-Chyuan, Wu Chen-Hao. LSB steganographic method based on reversible histogram transformation function for resisting statistical steganalysis [J]. Information Sciences, 2012, 188:346-358.
- [12] Ren S, Zhang T. Study of Reversible Information Hiding Scheme Based on CARDBAL2 and Color Field Structure[J]. Advanced Materials Research,2011,225:275-279.

算法[J]. 计算机工程,2010,36(22):134-136.

- [3] 刘军军. 基于决策树的防火墙策略算法研究[D]. 长沙:中南大学,2009.
- [4] 杜德超,姚庆栋. 多维过滤规则无冲突的高速分组分类算法[J]. 电子学报,2002,30(11):1676-1680.
- [5] Baboescu F, Varghese G. Scalable Packet Classification[J]. ACM SIGCOMM Computer Communication Review, 2001, 31(4):199-210.
- [6] Acharya S, Wang Jia, Ge Zihui. Simulation Study of Firewalls to Aid Improved Performance[C]//Proc. of the International Conference on 39th Annual Simulation Symposium (ANSS'06). [s.l.]:[s.n.],2006:18-26.
- [7] Oppliger R. Internet Security: Firewall and Beyond[J]. Communications of the ACM,1997,40(5):92-102.
- [8] 姚亚锋,方贤进,赛文莉. 新型内容过滤防火墙的研究[J]. 计算机技术与发展,2010,20(11):158-161.
- [9] 蒋宁,廉东本. 包过滤防火墙相关规则的排序及向无关规则的转化[J]. 小型微型计算机系统,2004(8):1550-1553.

# 一种防火墙规则冲突快速检测算法

作者：[徐艳](#)，[董涛](#)，[XU Yan](#)，[DONG Tao](#)  
作者单位：[电子科技大学成都学院, 四川 成都, 611731](#)  
刊名：[计算机技术与发展](#)

英文刊名：[Computer Technology and Development](#)

年，卷(期)：2013(9)

本文链接：[http://d.g.wanfangdata.com.cn/Periodical\\_wjfz201309032.aspx](http://d.g.wanfangdata.com.cn/Periodical_wjfz201309032.aspx)