布尔表达式匹配问题研究*

曹 京[1,2]，谭建龙[1]，刘 萍[1]，郭 莉[1]
（1. 中国科学院 计算技术研究所 软件室，北京 100080；2. 中国科学院 研究生院，北京 100049）

摘 要：提出了布尔表达式匹配问题，并给出了它的形式化定义；提出了一个解决布尔表达式匹配问题的通用算法框架，并在此框架上给出了一种算法及其改进；通过理论分析和实验数据给出了影响布尔表达式匹配算法性能的因素和它们之间的关系。

关键词：布尔表达式匹配；计数算法；最长过滤算法
中图分类号：TP301.6 **文献标志码**：A **文章编号**：1001-3695(2007)09-0070-03

Research of Boolean expression matching

CAO Jing[1,2], TAN Jian-long[1], LIU Ping[1], GUO Li[1]
(1. *Software Division, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100080, *China*; 2. *Graduate School, Chinese Academy of Sciences, Beijing* 100049, *China*)

Abstract: This paper proposed Boolean expression matching to solve these application, such as virus scan, spam mails filter, junk shot-message filter etc. First, formalized the definition of Boolean expression matching. Second, proposed an algorithm frame, then presented an algorithm and an improved algorithm based on this frame. At last, presented the performance factor of algorithms according to theoretic analyzing and experiment.

Key words: Boolean expression matching; count match algorithm; max filter match algorithm

所谓串匹配(string matching)，就是给定一组字符串集合 $P=\{P_1,P_2,\cdots,P_r\}$，对于任意一个字符串 P_i，找出它在文本 T 中的所有出现位置。称 P 为字符串集合，P_i 为字符串(或关键词)，T 为文本。字符串和文本中的字符都取自一个有限的符号集合 Σ，简称字母表或字符集。经典串匹配问题主要包括精确串匹配、正则表达式串匹配和近似串匹配。其应用领域包括信息检索、病毒检测、入侵检测、生物信息学等。在这些应用系统中，通常将规则描述为字符串形式，然后直接使用经典串匹配算法对规则进行匹配。随着信息量的急剧膨胀，应用系统要求更准确、更快速地对海量信息进行分析过滤，采用简单关键词来描述规则的方法已经无法满足需求。

Snort[1] 和 ClamAV[2] 都需要将多个关键词结合起来获得匹配结果，利用多个关键词之间的位置信息来进行特殊处理，但是无法满足垃圾邮件过滤等不依赖关键词位置的需求。除此之外，采用正则表达式串匹配也可以解决这种需求，目前正则表达式使用 k 表算法的空间复杂度是 $O(m\times 2^{2m/k})$，时间复杂度为 $O(kn)$ [3]（m 是正则表达式中的字符个数，n 是文本长度）。因此当 m 达到几千甚至上万的规模时，正则表达式匹配算法的空间复杂度就很高，无法满足实际应用的需要。为了解决上述这类问题，本文引入了布尔表达式匹配概念，给出了布尔表达式匹配的算法，并从不同角度来分析其性能。

1 布尔表达式匹配定义

定义 1 给定一个布尔表达式 $BE=\mathrm{KeyItem}_1\,\mathrm{op}_1\,\mathrm{KeyItem}_2\,\mathrm{op}_2\cdots\,\mathrm{op}_{k-1}\,\mathrm{KeyItem}_k$ 和一个数据文本 $T=t_1t_2\cdots t_n$。其中：$\mathrm{KeyItem}_i=\mathrm{Flag}_ip_i$；$\mathrm{op}_i\in\{\wedge,\vee\}$。$\mathrm{Flag}_i\in\{+,-\}$；$T$ 和 P_i 都是字符集 Σ 上的有限字符序列；$+p_i$ 表示 p_i 在数据文本中出现；$-p_i$ 表示 p_i 不在数据文本中出现；$\vee$ 为或关系；$\wedge$ 为与关系($\wedge$ 优先级高于 $\vee$)。当 $BE=\mathrm{True}$ 时，表示该布尔表达式在文本中出现。

在定义 1 中，采用标志(flag)来表示非关系，采用操作符(op)来表示与、或关系。

定义 2 在上述布尔表达式 BE 中，记 $\mathrm{occurence}(p_i)$ 为 p_i 在数据文本中出现的次数；记 $\mathrm{occurence}(BE)=\min(\mathrm{occurence}(p_i),\forall_i)$ 为布尔表达式 BE 在文本中出现的次数。

定义 3 给定一个布尔表达式 BE，一个数据文本 $T=t_1t_2\cdots t_n$，找到该布尔表达式 BE 在数据文本的出现次数，即为布尔表达式匹配问题。

定义 4 给定一组布尔表达式 $\{BE_1,BE_2,\cdots,BE_s\}$，一个数据文本 $T=t_1t_2\cdots t_n$，找到所有在数据文本中出现的表达式，即为多布尔表达式匹配问题。

由布尔代数理论可知，每个布尔表达式都可以表示成析取范式。定义 1 中的布尔表达式 BE 可以表示成 $BE=BE_1\vee BE_2\cdots\vee BE_n$。其中：$BE_i=\mathrm{KeyIten}_1\wedge\mathrm{KeyItem}_2\wedge\cdots\wedge\mathrm{KeyItem}_k$。

性质 1 任何一个布尔表达式可以表示成多个子布尔表达式的或关系。其中子布尔表达式只包含与关系和非关系。

根据性质 1 将布尔表达式变形拆分后，布尔表达式匹配问

收稿日期：2006-07-04；修返日期：2006-09-30 基金项目：国家“242”信息安全计划资助项目(2005C39)

作者简介：曹京(1981-)，男，江苏无锡人，硕士研究生，主要研究方向为算法设计、网络信息安全、中文信息处理(caojing@software.ict.ac.cn)；谭建龙(1974-)，男，湖南长沙人，副研究员，主要研究方向为数据流处理、网络安全、中文信息处理；刘萍(1972-)，女，山东济南人，助理研究员，主要研究方向为网络与信息安全、数据流管理；郭莉(1969-)，女，湖南人，研究员级高工，主要研究方向为网络与信息安全、高性能算法及其软件。

题只需处理与和非布尔关系。本文研究的是大规模多布尔表达式匹配问题。

2　布尔表达式匹配算法框架

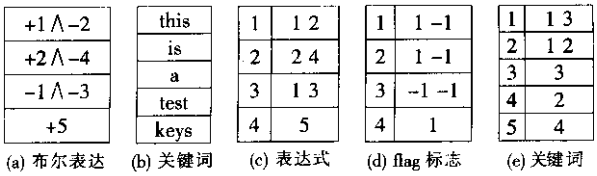
算法框架共有两层,即多串匹配层和逻辑计算层。多串匹配层位于算法框架的底层。它利用多串匹配算法(如 AC[4]、SBOM[5] 等)得到待扫描文本中出现的所有关键词。逻辑计算层位于多串匹配层之上。它根据底层匹配成功的关键词结果,通过快速计算得到最终的布尔表达式匹配结果。

对于一组布尔表达式 $\{BE_1, BE_2, \cdots, BE_s\}$ 来说,可以将其所包含的关键词构成数组 $P = \{p_1, p_2, \cdots, p_r\}$;然后用关键词数组中的序号来代替布尔表达式中的该关键词。因此算法的输入为一个三元组:(BE, P, T)。为了便于描述算法,并对算法进行理论分析,本文使用了如下符号:布尔表达式的个数为 s;关键词的个数为 r;每个布尔表达式中平均包含关键词的个数为 k;最短关键词长度为 m;测试文本数据长度为 n;字符集的大小为 $|\Sigma|$;包含匹配关键词的表达式数目为 q;最长关键词的个数为 p;包含匹配最长关键词的表达式数目为 t。

2.1　计数算法

根据上述的算法框架,算法的时间开销主要由多串匹配和逻辑计算两部分组成。多串匹配的时间基本上是一定的,所以尽可能简化逻辑计算是算法需要考虑的。由于在上述这些应用中,关键词在数据文本中出现很少[6],可以通过关键词来确定需要进行逻辑计算的布尔表达式。

计数算法主要通过构建两张映射表来简化逻辑计算。表达式映射表中索引为布尔表达式序号,内容为该布尔表达式包含所有关键词序号(不包括标志符号,标志符号用 flag 标志数组来表示);关键词映射表中索引为关键词序号,内容包含该关键词的所有布尔表达式的序号。映射表示例图如图 1 所示。



(a) 布尔表达式数组　(b) 关键词数组　(c) 表达式映射表　(d) flag 标志数组　(e) 关键词映射表

图 1　映射表示例图

在进行多关键词匹配后,利用关键词映射表来获得包含匹配成功的关键词的所有布尔表达式序号。对于其中每个布尔表达式序号,再根据表达式映射表来判断该布尔表达式包含的其他关键词是否出现,从而得到该布尔表达式是否在数据文本中出现。

1)初始化阶段

a)根据性质 1,通过一定的变换操作得到布尔表达式数组 BE 和关键词数组 P。

b)构建布尔表达式映射表 KeyArray、关键词标志数组 FlagArray、关键词映射表 ExprArray。

c)根据关键词数组初始化多串匹配算法(文中实验采用 SBOM 算法)。

2)扫描匹配阶段

利用多串匹配算法扫描文本数据,结果保存在关键词匹配集合 KeyMatchItem 中。

3)构建结果阶段

a)判断 KeyMatchItem 是否为空,是则结束;否则转 3)-b)。

b)取出 KeyMatchItem 中一个关键词,通过关键词映射表找到相应的布尔表达式集合。

c)判断布尔表达式集合是否为空,是则转 3-a),否则转 3-d)。

d)取出布尔表达式集合中的一个,通过布尔表达式映射表得到关键词集合。

e)判断关键词集合是否为空,是则表示该布尔表达式匹配成功,转 3-c);否则转 3-f)。

f)取出关键词集合中的一个进行判断。如果该关键词匹配并且标志为 -1 或者关键词不匹配并且标志为 1,则该布尔表达式匹配不成功,转 3-c);否则转 3-e)。

初始化阶段属于预处理过程,这部分时间不予考虑。算法时间复杂度主要考虑扫描匹配阶段和构建结果阶段的时间开销。

上文中扫描匹配阶段采用 SBOM 算法来进行多关键词匹配,其时间复杂度为 $O[n\ \log_{|\Sigma|} mr/(m - \log_{|\Sigma|} mr)]$[7]。构建结果阶段的平均时间复杂度为 $O(kq)$,最坏时间复杂度为 $O(ks)$。因此算法扫描匹配阶段和构建结果阶段的总时间复杂度为 $O[n\ \log_{|\Sigma|} mr/(m - \log_{|\Sigma|} mr) + kq]$;扫描匹配速度等于文本长度除以时间。由此可以看出,影响计数算法速度的因素有关键词个数 r、布尔表达式平均包含关键词的个数 k、包含匹配关键词的表达式数目 q、最短关键词长度 m 和字符集大小 $|\Sigma|$。在具体应用中,最短关键词长度和字符集大小都是固定的,可以作为常量,因此本文实验不考虑这两个因素,均置为固定值。

当文本中出现的表达式个数较少时,上面表达式的时间复杂度的后一项就可以去掉。此时布尔表达式的匹配速度等价于 SBOM 算法的速度。

2.2　算法改进——最长过滤算法

通过上述算法分析可以得出,当测试文本较大时,计数法的匹配速度等价于 SBOM 算法的速度,而 SBOM 算法又与关键词数目有很大关系。实际应用中,在使用计数算法解决布尔表达式匹配问题时,很多关键词是不需要使用的,例如类似“!A and !B”的规则,只要任何一个关键词匹配成功,就可以给出规则匹配成功的结果。多串匹配的时间与模式串的个数成正比,是单调递增的。基于此,本文提出了一个改进算法,即最长过滤算法。

最长过滤算法采用两阶段扫描匹配过程:

a)初始化阶段选取每个布尔表达式中最长的一个关键词,构成一个最长关键词集合;然后再根据这个最长关键词集合构建一个下一次匹配关键词映射表(索引为最长关键词序号,内容为包含该最长关键词所有布尔表达式中的其他关键词序号)。

b)扫描匹配阶段采用两阶段扫描匹配过程。先用最长关键词集合构建的多关键词匹配算法扫描一遍测试数据文本;然后根据匹配成功的关键词和下一次匹配关键词映射表得到再次需要扫描的关键词集合;再初始化多关键词匹配算法来扫描一次测试数据文本。

构建结果阶段同计数算法。

最长过滤算法中，p 为最长关键词个数 $p<s$ 且 $p<r$，则第一次扫描阶段的时间复杂度为 $O[n\log_{|\Sigma|}mp/(m-\log_{|\Sigma|}mp)]$；为包含出现过的最长关键词的表达式数目，则第二次扫描阶段关键词的个数为 kt；多串匹配算法初始化过程时间复杂度为 $O(kt)$；扫描匹配的时间复杂度为 $O[(n\log_{|\Sigma|}mkt)/(m-\log_{|\Sigma|}mkt)]$；构建结果的时间复杂度同计数算法为 $O(kq)$。因此，扫描匹配阶段和构造结果的总时间复杂度为 $O[n\log_{|\Sigma|}mp/(m-\log_{|\Sigma|}mp)+n\log_{|\Sigma|}mkt/(m-\log_{|\Sigma|}mkt)+kq]$。

由此可以看出，影响最长过滤算法性能的主要因素是最长关键词个数 p，包含匹配最长关键词的表达式数目 t、平均包含关键词的个数 k、最短关键词长度 m。根据文献[6]，文本数据中关键词和表达式匹配的数目很少甚至没有出现，因此当文本数据很大时，上述总时间复杂度的后两项都可以忽略。这样扫描匹配阶段和构造结果的总时间复杂度可以写成 $O[n\log_{|\Sigma|}mp/(m-\log_{|\Sigma|}mp)]$。

2.3 两种算法比较

比较计数法和最长过滤法两种算法的时间复杂度，可将比较两者性能快慢的问题转换成比较 $\log_{|\Sigma|}mr/(m-\log_{|\Sigma|}mr)$ 和 $\log_{|\Sigma|}mp/(m-\log_{|\Sigma|}mp)+\log_{|\Sigma|}mkq/(m-\log_{|\Sigma|}mkq)$ 两个等式大小关系的问题。当关键词在文本中匹配的数目较少时，$\log_{|\Sigma|}mkq/(m-\log_{|\Sigma|}mkq)$ 可以忽略掉。由于 $p<r$，$\log_{|\Sigma|}mr/(m-\log_{|\Sigma|}mr)<\log_{|\Sigma|}mp/(m-\log_{|\Sigma|}mp)$，即此时最长过滤法比计数法的性能好。

3 实验结果和分析

实验环境为 CPU P4 2 GHz，内存 2 GB，Windows Server 2003 操作系统。

3.1 布尔关系的性能影响

为了验证布尔表达式中布尔关系对其性能的影响，作了如下的实验：布尔表达式中关键词均一样，连接两个关键词的布尔关系分成与、或、与非、或非以及上述四种布尔关系的随机混合五种；关键词长度为 2～8 Byte，测试文本数据为 100 MB，字符集大小均为 256；布尔表达式个数和关键词个数均为10 000；算法采用计数算法；每种布尔关系测试 30 次，最终结果取其平均。测试结果如表 1 所示。

表 1 布尔关系的性能影响表

布尔关系	匹配速度/MBps	构建结果时间/ms
与	39.53	2.869
或	39.90	6.046
与非	39.34	3.173
或非	39.54	3.172
混合	39.20	3.129

由表 1 中的数据可以看出，平均构建结果时间相差较大（最长和最短时间相差一倍），但是最后布尔表达式匹配的速度相差不大（最快和最慢相差不到 2%）。得出如下结论。

结论 1 布尔表达式中布尔关系的不同，对构建结果时间影响较大，对布尔表达式匹配速度影响不大。

3.2 其他性能影响因素

为了测试布尔表达式匹配算法性能的其他因素，作了如下实验：测试文本数据和关键词均随机生成，字符集大小为 32；测试文本数据长度为 100 MB；测试集分成三组（1～3、4～6、7～9）；关键词长度范围：测试集 1～3 为 4～30 Byte，测试集 4～6 为 5～30 Byte，测试集 7～9 为 6～30Byte；表达式个数均为 10 000。

每组中关键词是一样的，只是布尔表达式随机生成方式不一样：布尔表达式中关键词序号选取范围为 1～10 000；布尔表达式中关键词序号选取范围为 1～6 000。布尔表达式包含的关键词数目均为 3。具体实验结果如表 2 所示。

表 2 布尔表达式匹配算法实验结果表

测试集	关键词数目	表达式产生方式	表达式命中次数	关键词命中次数	匹配速度/MBps	构建结果时间比例/%
1	10 000	A	25 642	45 136	11.21	0.36
2	10 000	B	26 972	45 136	9.97	0.32
3	6 000	B	26 972	28 135	14.23	0.46
4	10 000	A	808	1 391	13.74	0.44
5	10 000	B	913	391	13.66	0.43
6	6 000	B	913	808	18.60	0.95
7	10 000	A	36	52	17.64	0.28
8	10 000	B	35	52	17.60	0.28
9	6 000	B	35	36	21.42	0.34

表 2 中，每组测试集后两个测试集布尔表达式完全相同，只是前一个测试集比后一个关键词数目多了 4 000 个无用关键词，而且布尔表达式命中总次数是一样的，但是匹配速度却相差很大。由此可以得到如下结论。

结论 2 布尔表达式匹配速度随着表达式命中次数的增加而降低。

由于关键词数目与布尔表达式平均包含关键词个数成正比，布尔表达式平均包含的关键词越多，关键词数目也越多。可以得到如下结论。

结论 3 布尔表达式匹配速度随着布尔表达式平均包含关键词个数的增加而降低。

各组实验中的布尔表达式构建时间占总匹配时间的比例非常小。可以得到如下结论。

结论 4 当测试文本数据很大时，布尔表达式的匹配速度取决于多关键词串匹配算法的速度。

从组 1 到组 3，布尔表达式命中总次数和关键词命中总次数是越来越少，但处理速度越来越快。根据结论 4 可以得到：

结论 5 布尔表达式匹配速度随着关键词命中次数的增多而降低。

由此可以得出，当 n 很大时，布尔表达式的速度主要由关键词数目 r、布尔表达式平均包含关键词个数以及关键词命中次数决定。

上述结论与前面理论分析是一致的。至于关键词命中次数，在实际应用中会影响多关键词匹配速度[8]，因此同样也是影响布尔表达式匹配算法性能的主要因素之一。

3.3 两种算法对比实验

本文还给出了上面两种算法的对比实验。测试文本数据为随机生成，大小为 100 MB；关键词也随机生成；字符集大小为 32。关键词数目和表达式数目在同一测试集中相同，测试集 1～3 为 10 000，4～6 为 5 000，7～9 为 1 000。

从表 3 中可以看出，在关键词命中次数方面，最长过滤算法比计数算法要少得多，匹配速度最长过滤算法比计数算法要快很多。因此可以看出，最长过滤算法能够 （下转第 108 页）

3 恢复系统性能分析

在实验室内部网环境下搭建恢复系统环境。实验中,本地系统和远程系统均为1.70 GHz处理器。128 MB RAM、7 200转/s硬盘、100 Mbps网卡。由于远程服务器仅用于对外服务切换,其上发生的故障对数据的完整性和安全性没有影响,影响的只是业务的可用性和连续性。远程服务器发生故障时,重新镜像将更有效率[8],此种情况不采取对系统进行恢复的方式。此处仅对本地服务器、本地网关、远程网关的故障进行恢复测试。表1列出了四种灾难情况的恢复结果。

从表1可以看出,灾难发生点不同,系统恢复时间也有相应的差别。原因在于各灾难情况的恢复流程不同。如图3所示,本地网关和本地服务器同时发生灾难时,恢复分三个阶段进行;若只是本地服务器的LSDP发生灾难,恢复仅分两个阶段完成从本地网关到本地服务器的恢复。此外,影响系统恢复时间的因素还有所选择的恢复方式。图5以从远程网关到本地网关的数据恢复为例,显示了选择全恢复与快速恢复两种恢复方式在系统恢复时间上的差异。

表1 恢复时间统计表

灾难发生点	恢复时间/s	数据一致性
本地服务器	95	一致
远程网关	25	一致
本地网关	125	一致
本地服务器和本地网关	120	一致

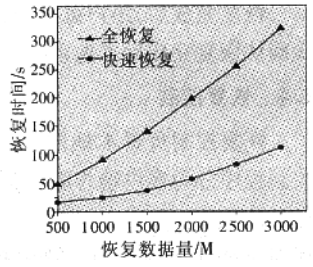


图5 全恢复与快速恢复时间比较

从图5可以看出,快速恢复具有更高的效率,并且需要恢复的数据量越大,其时间优势越明显。对以上系统恢复后进行的数据一致性检测表明,系统丢失的数据量为0。因此可得出该系统具有极好的恢复时间指标RTO[9]和快速恢复能力,可以在较短的时间内快速恢复系统关键数据。

4 结束语

本文设计并实现的异地灾难恢复系统在较低档次的平台上为用户搭建较高等级的远程灾难恢复系统,实现了灾难发生后系统的快速恢复,并保证了恢复后本地系统与远程系统间数据的一致性。用户可通过友好的界面进行灾难恢复的控制。整个系统具有很高的性价比。此系统对于研制具有自主知识产权的恢复技术及其相关产品具有积极意义。

参考文献:

[1] 李涛. 网络安全概论[M]. 北京:电子工业出版社,2004.
[2] LEWISW Jr, WATSON R T, PICKREN A. An empirical assessment of IT disaster risk[J]. Comm ACM,2003, 46(9):201-206.
[3] 王德军,王丽娜. 容灾系统研究[J]. 计算机工程,2005, 31(6):43-45.
[4] TESTA S, Chou W. The distributed data center: front-end solutions[J]. IT Pro, 2004, 6(3):26-34.
[5] WANG Kun, CAI Zhen, LI Zeng-xin, *et al*. A disaster recovery system model in an e-government system[C]//Proc of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies. 2005.
[6] FALLARA P. Disaster recovery planning[J]. IEEE Potentials, 2004,22(5):42-44.
[7] SHAO B B M. Optimal redundancy allocation for information technology disaster recovery in the network economy[J]. IEEE Transactions on Dependable and Secure Computing, 2005, 2(3):262-267.
[8] FROLUND S, PEDONE F. Dealing efficiently with data-center disasters[J]. Journal of Parallel and Distributed Computing, 2003, 63(11):1064-1081.
[9] 徐志发,汪海鹏. 电信级容灾系统关键技术分析[J]. 数据通信, 2003(1):9-11.

(上接第72页)减少关键词匹配次数,从而提高布尔表达式匹配速度。通过表3的实验数据,可以进一步证明3.2节中结论的正确性。

表3 计数法和最长过滤法性能比较表

测试集	表达式命中次数		关键词命中次数		平均匹配速度/MBps	
	计数算法	最长过滤算法	计算算法	最长过滤算法	计数算法	最长过滤算法
1	0	0	40 477	69	13.04	25.14
2	0	0	1 394	0	17.88	31.42
3	0	0	61	0	21.87	34.40
4	31	31	18 669	155	21.60	32.95
5	0	0	584	0	26.06	48.24
6	0	0	31	0	29.87	48.40
7	12	12	4 263	115	51.20	66.33
8	0	0	130	0	71.44	106.01
9	0	0	0	0	85.56	116.98

4 总结和展望

本文通过布尔表达式匹配算法性能的实验结果,发现其算法思想可以推广到wildcard通配符串匹配以及允许gap的扩展串匹配问题,而如何推广则是下一步需要做的工作。

参考文献:

[1] Snort 2.4.x[EB/OL]. http://www.snort.org.
[2] ClamAV[EB/OL]. http://www.clamav.org.
[3] NAVARRO G, RAFFINOT M. New techniques for regular expression searching[J]. Algorithmica, 2004, 41(2):89-116.
[4] AHO A, CORASICK M. Efficient string matching: an aid to bibliographic search[J]. Communications of the ACM, 1975, 18(6):333-340.
[5] ALLAUZEN C, RAFFINOT M. Factor oracle of a set of words[R]. [S.l.]: Institute Gaspard-Monge, University de Marne-la-vallee, 1999.
[6] KYTOJOKI J, SALMELA L, TARHIO J. Tuning string matching for huge pattern sets[C]//Proc of CPM2003. 2003:211-224.
[7] LIU Ping. Research of string matching for internet content Filtering [D]. Beijing: Institute of Computing Technology, Chinese Academy of Sciences, 2005:21-27.
[8] NAVARRO G, RAFFINOT M. Flexible pattern matching in strings [M]. [S.l.]: Cambridge University Press, 2002:69-76.
[9] NORTON M. Optimizing pattern matching for intrusion detection [R]. Columbia: Sourcefire Inc, 2004.