Bloom Filter哈希空间的元素还原

彭艳兵, 龚 俭, 刘卫江, 杨 望
(东南大学计算机科学与工程系, 江苏省计算机网络技术重点实验室, 江苏南京 210096)

摘 要: 本文提出使用语义增强的 Counting Bloom Filter Reconstruction(RSECBF)算法来快速还原源串或给出源串的聚类特征. 它给每个哈希函数独立的哈希映射空间以消除哈希函数的内部冲突; 扩展哈希函数使其不受均匀性限制, 使得哈希函数可以带有语义; 利用哈希串的重叠和数量一致性来解决同源哈希串拼接成源串的问题, 为源串的还原创造了条件. 本文针对 Pareto分布的哈希函数, 为主成分的还原提出了一个简洁的源串还原算法. 对于直接选择部分比特的哈希映射而言, 如果主成分分析中的 RSECBF不能还原出源串, 则还原出来的最长串就是源串的聚类特征. 仿真及实际检验表明, Bloom Filter可以扩展其哈希函数来实现语义增强, RSECBF还原的结果是可信的. 本算法可以在异常行为发生的时候挖掘网络行为特征.

关键词: Counting Bloom Filter; 语义增强; 参数还原; 异常行为检测
中图分类号: TP393 文献标识码: A 文章编号: 0372-2112 (2006) 05-0822-06

Element Recovery from Counting Bloom Filters' Hash Space

PENG Yan-bing, GONG Jian, LIU Wei-jiang, YANG Wang
(Department of Computer Science and Engineering, Southeast University, Key Laboratory for Networking in Jiangsu Province, Nanjing, Jiangsu 210096, China)

Abstract An original element reconstructing algorithm named Reconstruction with Semantically Enhanced Counting Bloom Filter (RSECBF) is proposed to timely recover the original element in set S from Counting Bloom Filters' hash space. The semantically enhanced hash function is based the ideas that, the independent hash space preserved for each different hash function eliminates the internal confliction among hash functions; and the hash function could be extended from the uniform distribution to any distribution; the overlapping of hash bit strings bring the ability to recover the original string by the uniqueness of hash mapping process and the hits amount balance of overlapped hash string. The recovery algorithm is greatly simplified for the Pareto distribution when only the principal component is analyzed. For Directly Bit String Selecting, the reproduced longest string just is the distribution character of the original strings. The simulation and the validation with published data trace suggested that the recovery result of RSECBF is acceptable. It can be used to find the network behavior characteristics when abnormal behavior bursts in the real networks.

Key words counting bloom filter; semantically enhancement; parameter recovery; abnormal behavior detection

1 引言

最早在 1970年[1], Bloom 提出一种基于多哈希函数映射来压缩参数空间、实现快速参数查找判定的 Filter方法, 在计算机的其他领域得到了广泛的应用, 比如串匹配[2]、分布式的协同[3], 数据库领域[4]、路由查找[5]等. 由于网络中很多地方用到了串的比较, 使得 Bloom Filter成为最近在网络研究中比较热门的工具, 在网络抽样[6]、抽样的还原[7]、流分布估计[8]里有着广泛的应用.

Bloom Filter对哈希函数的假设是均匀分布的, 而很难找到多个对于源串集合符合均匀分布的哈希函数. 如果选择的哈希算法的可逆性不好, 会使得从哈希数组空间很难还原得到原始串的信息, 或者计算的复杂度太高. 而 Bloom Filter把所有的哈希串映射到同一个哈希空间也不可避免地导致了不同哈希函数间相互覆盖的内部冲突.

2004年, Robert Schweller等在 MC2004会议上提出一种可反向追溯的方法[9], 基于 Sketch Sketch的工作原理和 Counting Bloom Filter很像. 这种可反向追溯的方法

收稿日期: 2005-07-25; 修回日期: 2006-01-09
基金项目: 国家 973重点基础研究发展规划 (No 2003CB314803); 江苏省网络与信息安全重点实验室 (No BM2003201)

基于复杂的分类过程, 且哈希函数非常复杂, 整个还原过程不直观.

本文给每个哈希数组独立的存储空间, 以少量的空间消耗消除了哈希函数在共享空间上的内部冲突; 论证了非均匀 hash也能够适应 Counting Bloom Filter, 使得哈希函数带有语义成为可能. 语义增强的哈希函数可以选择简单的映射方式如直接选择源串的部分比特位, 以直接反映源串的信息; 如何在冲突中还原源串是本文研究的主要话题. 充分利用不同的简单哈希函数的相互重叠来拼接和还原源串是本文工作的一个特色.

第二部分分析了各种 Bloom Filter的误差特性, 放开对 Bloom Filter的哈希函数的均匀性限制; 第三部分分析了带有语义的 Counting Bloom Filter的特性, 提出一类可以用于源串还原的哈希函数, 并讨论了其还原源串的依据; 第四部分给出参数还原的线性规划方程组, 并给出了利用重尾分布的参数还原算法和其算法的复杂度; 第五部分从仿真和实验的角度验证了扩展 Bloom Filter的可行性及还原源串的效果; 第六部分是结论和将来的工作.

2　Bloom Filter的特性

2.1　标准 Bloom Filter的误差分析

Bloom Filter算法使用多个短的哈希串来再现一个长的字符串所代表的空间, 其具体的工作原理可以参见[1, 10]. 标准的 Bloom Filter如图 1所示, 它描述了一个源串集合 $S = \{x_1, x_2, \cdots, x_n\}$共 n个元素, 里面的元素 x_i我们把它称为源串, 其取值范围为 $\{1, \cdots, M\}$; 把该集合里所有的元素用 m个比特单元构成的存储空间 A来表示, 这 m个比特单元被初始化为 0; 使用哈希函数集合 $H = \{H_1, \cdots, H_k\}$, 我们把 H_j作用得到的串记为 h_{ji}, 其表示的范围为 $\{1, \cdots, m\}$; 把所有源串 x_i通过这 k个独立的哈希函数映射到存储空间 (以后简称空间)A. 这样就可以通过哈希函数集合 H 的映射过程来检验空间 A, 以断定元素 x'是否属于集合 S. 把图 1中的 m 个比特换成 m 个计数器, 就成为了 Counting Bloom Filter

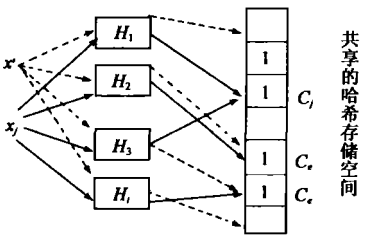


图 1　标准 Bloom Filter的工作过程

Bloom Filter应用的场合必须允许有一定误判的可能性. 根据文献[10], Bloom Filter不会错误地否定. 但是如果所有 hash都为 1, Bloom Filter肯定的源串 x'是否真的属于集合 S的概率, 即其错误肯定率 (False Positive Rate, *FPR*) 需要进行讨论. 根据文献[10]的原理, 可以推知 *FPR* 为:

$$FPR = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(\frac{kn}{m}\right)^k, \quad \text{if } kn << m \qquad (1)$$

可见 Bloom Filter中假设哈希函数能够随机均匀地把源串映射到哈希映射存储空间 A.

一般 hash串表示的范围 m 比源串表示的范围 M 小很多, 从 x_i到 h_{ji}的空间压缩过程必然使得 hash的过程存在冲突, 多个不同的源串可能被映射到相同的哈希映射空间的某一点, 我们把它叫做外部冲突 (External Confliction, *Ce*), 它不能通过哈希函数的选择来消除, 除非空间 A 的大小 m 大于等于 M, 但是合理选择的哈希函数可以降低 *Ce*. 图 1的 Bloom Filter使用了共享的哈希存储空间 A, 使得在一次映射过程中不同的哈希函数可能会映射到哈希映射空间 A 的同一个位置, 我们把它叫做内部冲突 (Internal Confliction, *Ci*), 它可以通过为每个哈希函数使用独立的哈希映射空间来消除. 我们仍把消除内部冲突后的哈希映射空间数组称为哈希映射空间 A, 相应函数 H_j的哈希映射空间统一表述为 A_j.

使用独立的哈希映射空间并不能消除外部冲突, 但是由于消除了内部冲突, 使得 *FPR* 也减小了许多. 独立哈希映射空间的 Bloom Filter的误差分析没有文献报道. 与式 (2)类似, 由于是 H_j 都是随机均匀分布, 同理可以推知其 *FPR* 为:

$$FPR = \left(1 - \left(1 - \frac{1}{m}\right)^{n}\right)^k \approx \left(\frac{n}{m}\right)^k, \quad \text{if } n << m \qquad (2)$$

简化后的 *FPR* 与上面共享空间的 Bloom Filter相比, *FPR* 缩小了 k^k 倍, 显然, 使用独立哈希映射空间能够显著减少 Bloom Filter的错误肯定率.

2.2　非标准 Bloom Filter的误差分析

假设 Bloom Filter的哈希函数非均匀, 则在进行 *FPR* 推导的时候, 引入一个分布函数 $p_j(1, n)$, 它表示有 n个源串的时候, 函数 H_j 在哈希映射空间 A_j 出现 1的概率. 按照标准的 Bloom Filter的分析方法, 非标准哈希函数的 Bloom Filter的 *FPR* 为

$$FPR = \prod_{j=1}^{k}(1 - (1 - p_j(1, n))) = \prod_{j=1}^{k} p_j(1, n) \qquad (3)$$

基于共享空间的非标 Bloom Filter的 *FPR* 为:

$$FPR = \prod_{j=1}^{k}\left(\sum_{j=1}^{k} p_j(1, n)\right)_j = \left(\sum_{j=1}^{k} p_j(1, n)\right)^k \qquad (4)$$

对于均匀分布, $p_j(1, n) = n/m$, 代入即可得到上述均匀分布的 Bloom Filter的误差公式, 可见均匀分布的哈希函数只是这些分布中的特例.

这个 *FPR* 公式表明 Bloom Filter同样可以使用非均匀的哈希函数, 即哈希函数可以带有语义.

2.3　Pareto分布的 Bloom Filter的误差分析

对于 Pareto分布的哈希函数 H_j 有: $P_j(x) = a_j x^{-(\alpha+1)}$, $x \geqslant 1$. 对于 n个元素, 因元素不可再分, 最小可分辨的概率为 $1/n$, 小于此概率的项被认为是 0, 因此可以推导出大于

0的项数. 把 $p_j = 1/n$ 带入分布, 得到 x 的值即为大于 0的项数 X:

$$X_{x>0} = (a_i n)^{1/(r_j+1)} \quad (5)$$

$$p_j(1, n) = \frac{X_{x>0}}{m} = \frac{1}{m}(a_j n)^{1/(r_j+1)} \quad (6)$$

$$FPR = \prod_{j=1}^{k} p_j(1, n) = (1/m)^k (a_j n)^{\sum_{j=1}^{k} 1/(r_j+1)} \quad (7)$$

当 $r_j \to \infty$的时候, $a_j \to 1$, Pareto分布就趋向于均匀分布, 其 *FPR* 也趋向于均匀分布时的 *FPR*.

3 语义增强 Counting Bloom Filter的特性

根据上节的结论, Counting Bloom Filter允许非均匀的哈希函数, 独立的哈希映射空间可以保证不同的 hash代表的语义间不会产生冲突, 因此我们可以选用一些带有语义的哈希函数集合 H, 使得 hash后的串 h_{ji}与源串 x_i 具有很高的相似性, 并且非常简单, 对进行源串还原工作将会有很大的益处. 最直接的哈希函数是直接从源串里选择部分比特串, 比如源串为 0x*abcd*, 直接选取最高部分的比特串和最低部分的比特串的这两个哈希函数可以获得 0x*ab*和 0x*cd*; 如果知道它们属于同源串, 只需把这两个 hash值串起来就可以得到源串了. 我们把这种直接取源串若干比特的哈希函数叫做短比特串映射.

对于使用独立哈希映射空间的 Counting Bloom Filter, 有一个性质来描述从集合 S到哈希映射空间的映射特性:

性质 1 一个源串会分别在每个独立的哈希空间里有且仅有一次映射.

性质 1可以从 Bloom Filter的原理得到, 无须证明, 我们把它称为哈希映射的唯一性. 性质 1等效于, 如果源串在集合 S里出现了 1000次, 它对应的 hash串也会在其相应的空间至少出现 1000次. "至少"的含义是还有可能有外部冲突可能会使得 hash串的计数值偏大. 因此传统的 Counting Bloom Filter研究里通常取各 hash串中最小的数值作为对源串出现频数的极大似然估计.

从性质 1我们可以很直观地得出, 对于 DBSM 而言, 同源的 hash串出现的次数不小于源串出现的次数. 如果源串 0x*abcd*在集合 S里出现了 1000次, 则短比特串映射的 0x*ab*和 0x*cd*也应该至少出现 1000次. 因为使用独立的哈希映射空间, 可以不用担心 hash串 0x*ab*和 0x*cd*间会产生冲突. 但是进行源串的还原的最关键问题是如何确定两个彼此完全独立的短比特串来自于同一个源串.

哈希函数映射得到的比特串 h_j 一般不会比源串的比特位数差很多. 因为如果 h_j 比特位数太少, 会导致每个哈希函数的误差较大; 而 h_j的位数太大会导致比较大的空间开销, 导致空间压缩效率低. 比如 32比特的 IP地址, 一般会选择 8-24比特的短比特串为宜; 作为一个折中, 一般以 16比特串的空间压缩效率和降低误差效果最好.

采用多个哈希函数来保证可接受的 *FPR*, 会导致短比特串间有部分比特重叠, 降低了哈希函数间的随机性. 但是如果我们能够利用比特位的重叠特征做一些误差修正, 就像做图画的拼接的时候通常会利用一些重叠的部分来进行校正一样, 会有更好的效果, 同时我们也解决了如何确定两个比特串是否同源的问题, 并消除了随机性差的负面影响.

为了直观地讨论拼接还原过程, 我们这里只讨论 8比特整数倍的源串和 8比特的重叠关系; 为了直观地解说各种比特串, 借鉴 IP地址的表示方法. 我们以 32比特的 IP地址为例来展开我们的讨论, 其他任意长度的比特串和比特串的重叠关系都可以使用类似的方法分析.

对于 32比特的源串 (如 IP地址), 以取高 16比特位的哈希函数为 H_h, 以取中间 16比特位的哈希函数为 H_m, 以取低 16比特位的哈希函数为 H_l, 这样, H_h 和 H_m、H_m 和 H_l 间就有 8比特的重叠. 这种重叠关系也可以归纳为一个性质来描述:

性质 2 对于具有重叠关系的两个 DBSM 哈希函数而言, 取某一值的重叠串在数量上相等.

性质 2可以从性质 1和重叠关系导出, 同时也是通过短比特串的重叠比特串来拼接源串过程的真实写照. 这个性质我们称为重叠哈希串的数量一致性. 比如 IP地址 *a*. *b*. *c*. *d*在某个地方出现了 1000次, 则性质 1表明如果 *a*. *b*. *c*. *d*此时是活跃的, 则 hash串中相应的条目也应该是同样活跃的; 性质 2表明, 如果 *a*. *b*出现了 1000次以上, *b*. *c*作为一个候选, 如果它与 *a*. *b*是同源的, 它也应该至少出现 1000次以上; 如果存在两个活跃的 *a*. *b*. *c*和 *a*. *b*. *e*, 可以通过非重叠部分 *c*和 *e*把它们区分开, 而 *a*. *b*的出现次数应该比这两个短比特串出现的次数之和还要多 (性质 1).

4 基于语义增强的 Bloom Filter的参数还原

4.1 非均匀分布 Counting Bloom Filter的参数还原

非均匀分布的 Counting Bloom Filter的一个重要特点是可以使用其非均匀分布的聚类特点来区分冲突项间的关系, 使得我们不用求解 4.1节中复杂的线性方程组. 特别是如果我们只是用来检验分布的主成分是否有聚类的时候, 我们只需对 Counting Bloom Filter的主要成分进行分析, 规模的缩减使得计算复杂度大大降低.

如果 Counting Bloom Filter中哈希函数是重尾分布的, 则可以使用重尾特性来进行源串的主成分聚类. 如果源串的分布有了显著的改变, 我们还可以很快还原出这些变化部分的信息. 如果这种异常的改变对分布来说是显著的, 那么我们只需使用 Top*N* 即可发现这些显著的异常改变. *N* 很小, 使得我们只需少量的运算即可完成我们的目标.

根据 EddieKohler等观察到的 IP地址结构[11], 活跃 IP的分布是非常不均匀的重尾分布, 相邻网段或者 IP的活跃度的差异更大, 因而 IP_h、IP_m 和 IP_l 的分布也是重尾分布的; 如果多个 IP的活跃度相差很小, 可以推断是一种非正

常的分布式的行为. 因此, 即使多个独立的源串对应同一个短比特串也没有问题, 他们的聚类特征为我们推断网络里的异常活动提供了依据, 因为异常活动通常会改变其聚类特征. 比如 DDoS攻击的源 IP分散, 端口分散, 而宿 IP和宿端口则集中在某个 IP和端口上; 再如, 大规模扫描的源 IP集中在少数几个 IP上, 宿 IP和端口分散, 通过聚类特征的分析很容易把它们与正常的 TCP/IP行为区别开.

所以对于主成分分析而言, 非均匀分布的 Counting Bloom Filter只需对 Top$N$ 进行分析, 并且只需少量的重叠的短比特串映射.

4.2 TCP/IP异常行为中聚类信息还原

4.1节给出了主成分分析的方向, 对于源串集合中的不符合分布的异常行为, 我们也同样能够迅速发现. 对于 H_h、H_m 和 H_l, 根据性质 1、性质 2和哈希函数的聚类特征可以推导出一个判断源串分布特征的算法, 这个算法如果不能还原出源串, 则能够返回占主导成分的最长前缀 H_h, 表明源串是否聚集在某些短比特串上, 以此判断其是否有聚类行为, 对于 IP来说就是该 IP前缀是否会聚类在某个网段. 如果有可能, 它将返回占集合 S 主导地位的源串的值. 这种聚类的判定方式对于大规模 TCP/IP异常行为的检测非常有用, 比如对于目标网段的扫描会使得目标 IP的 H_h 和/或 H_m 集中在少数的值上, 而对于 DDoS攻击而言, H_h、H_m 和 H_l 里必然能够反应出受害主机的 IP聚类.

下面的算法可以用来还原源串, 或者还原源串的聚类特征. 每个 Top$N$ 有两个值构成, 一个是保存 Counting Bloom Filter的哈希映射空间的位置索引 h_{ji}, 一个保存该哈希函数串的命中次数 Hits. 对原始哈希串数组的 Hits进行排序的时候, 需要带着该位置索引 h_{ji}, 它携带了源串的部分信息.

RSECBF算法

分别查找 H_p 和 H_q 的 Top$N$

取得 H_p 和 H_q 的 Top$N$ 里最活跃的 hash串 $x.y$ 和 $y.z$

在重叠串中依照 3节的关系查找对应的主成分

如果 $x.y$ 没有 $y.z$ 对应, $x.y$ 是一种聚类行为

如果 $x.y$ 有 $y.z$ 对应, 并且是主成分, 表明 $x.y.z$ 是一种聚类行为

在 H_p 和 H_q 分别去掉 $x.y$ 和 $y.z$, 重复查找, 直到穷尽.

把这个算法分别作用于源 IP地址和宿 IP地址的 Counting Bloom Filter, 即可获得此时网络里源/宿 IP报文的活跃情况. 综合分析各种 TCP报文的活动规律, 还可判断出是否发生了大规模异常如扫描和 DoS攻击. 我们把本节提出的参数再现算法称为语义增强的源串再现 (RSECBF, Reconstruction with Semantic Enhanced Counting Bloom Filter).

对于两个独立 IP, 则 32比特的 IP作为标签冲突的概率为 2^{-32}. 若用短标签代表 2个独立 IP, 则 16比特哈希串作为标签, 冲突的概率为 2^{-16}. 这个误差比较大, 但结合重叠部分 8比特的拼接过程分析, 这个冲突的概率能被降低到 2^{-24}. 这样, 我们使用三个 16比特的短标签, 达到了使用 24比特标签的效果, 其资源开销与 16比特标签接近.

RSECBF算法的复杂性分析

本文使用的 Bloom Filter算法使用的 16比特哈希函数, 对哈希空间的访问开销为 $O(1)$; 算法的主要计算开销在于排序找出 Top$N$, Top$N$ 的计算方法有很多快捷的方法来实现[4]. 16比特串表示的空间大小为 $m = 2^{16}$, 算法的空间复杂度为 $O(m+N)$, 计算复杂度为 $O(m+N\log N)$.

5 对还原效果的仿真分析

5.1 Counting Bloom Filter的误差仿真

为了验证式 (7), 仿真分析选用源串在集合 S 里出现的频数分别符合随机均匀分布/Pareto分布. 为了获得很好的分布类型, 本文采用 GNU Scientific Library[12] 来获得均匀分布和 Pareto分布的样本. 表 1列出了仿真的结果.

表 1　Counting Bloom Filter的误差仿真, $m = 2^{16}$, $k = 4$

哈希特性	随机均匀		Pareto分布	
n	24×1024	1024	24×1024	24×1024
参数 r_0	–	–	0.5	0.1
理论 FPR	0.01978	5.96E-08	2.78E-08	5.01E-04
仿真 FPR	0.00954	5.81E-08	7.52E-10	2.12E-04

从表 1可以看出, 仿真的结果均比理论误差好, 可能是因为实际 hash分布的空间取 1的概率值与理论值有差距所致. 另外, Pareto分布的 FPR 要比均匀分布的 FPR 要好, 根据 Bloom Filter的工作原理可知, Pareto分布本身就具有一定的空间压缩特性, 使得取 1的概率与均匀分布相比很小的缘故.

5.2 Counting Bloom Filter的主成分还原仿真

对于 $r = 0.1$ 的 Pareto分布, 我们对其还原效果进行验证, 同样采用 GNU Scientific Library来获得相应分布, 样本数 $n = 1024$, $m = 65536$, $k = 3$, 哈希函数使用上述短比特串映射方法, 即 H_h、H_m 和 H_l, 对样本进行还原, 有表 2中的 Top5:

表 2　实验仿真效果分析, Top5

H_h		H_m		H_l	
0.0	1021	0.0	962	0.1	287
0.1	1	0.1	19	0.2	152
0.3	1	0.2	8	0.3	79
0.245	1	0.4	5	0.4	65
		0.3	3	0.5	34

根据 4.3节的算法, 可以推导出 0.0.0.1是最活跃的串, 共 287个, 占主要成份的串分别还有: 0.0, 0.2~10源串的分布也的确如此.

5.3 对还原效果的实际分析

本小节使用在 Caida网站上公布的 trace[13] 对还原效果进行检验. Caida的这个攻击数据集给我们很好的检验还原效果的机会, 因为已知这个数据集全部是异常网络数

据报文.我们只对其 RST+ ACK报文进行重点研究,以时间粒度为 10min统计 RST+ ACK报文随时间变化的规律.

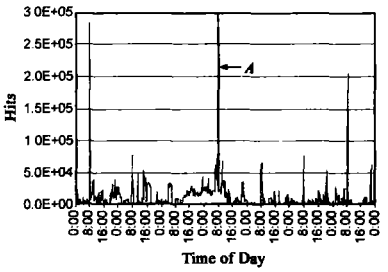


图 2 Caida异常报文 Trace里的 RST+ ACK 报文的时间曲线

RST+ ACK本来就是对异常的到达报文的回应,这里用它在时间粒度内的数量变化来揭示异常,如图 2所示.图 2中 A点(第 479个时间粒度)标志着大规模 RST+ ACK的爆发,我们选择使用了本文的算法对其主成分进行再现其源串的特征.

表 **3** A点的 RST_ ACK情况分析,共 **517714**个报文

(*a*)源 IP和端口 Top**5**

H_h		H_m		H_l		*PORT*	
Hash	Hits	Hash	Hits	Hash	Hits	Hash	Hits
200.251	379447	251.185	379447	185.3	379447	28	12074
64.124	79288	124.34	79288	34.175	79288	31	11033
194.102	15503	215.170	15120	170.8	15120	32	10987
200.215	15120	102.121	10864	121.97	10864	35	10937
4.17	10160	17.231	10160	231.154	10160	36	9906

(*b*)宿 IP和端口 Top**5**

H_h		H_m		H_l		*PORT*	
Hash	Hits	Hash	Hits	Hash	Hits	Hash	Hits
44.28	3768	108.68	1617	53.239	788	2291	2240
44.138	3749	160.94	1115	136.123	745	5786	418
44.75	3667	50.136	745	224.196	734	29876	398
44.243	3608	60.224	734	18.125	684	12597	378
44.242	3551	57.209	653	136.152	598	33765	348

表 3(*a*)中还原源串的分析表明, 200.251 185.3和 64.124 34.175是非常活跃的 IP,都占了总量十分之一以上.源端口的分布是均匀的.表 3(*b*)表明宿 IP分布非常均匀,目标网段为 44.0.0.0/8,端口非常均匀.由于其他类型的 TCP报文在此时的数量很少,这些特征表明这是一种大规模的扫描行为,与 hping工具使用 RST扫描的工作原理相同[14],而在 DDoS攻击中的伪造源 IP地址的报文在响应报文为 RST时,这些 RST报文也具有类似的特征.

通过对 RSECBF的主成分分析结果与直接归纳的结果的比较,发现这个算法在再现异常发生时的 IP地址分布特征的时候是高效的.

6 结论与将来的工作

本文提出一种 Counting Bloom Filter的改进,为每个哈希函数使用独立的哈希映射空间;并放开了对哈希函数的均匀性限制,使得可以让哈希函数带有语义,这样就可以使用简单的比特串映射作为哈希函数,直接反映了源串的部分特征;各短比特串间的重叠使得我们可以像拼接图画一样拼接源串.在深入的分析后,本文提出一个还原源串的算法,通过哈希串的重叠来让 Counting Bloom Filter能够还原出源串的特征,或给出源串的分布特征.

非均匀的聚类特性使得我们对主成分的特征分析只需进行 Top*N* 分析,从而大大减少还原的复杂度.利用网络上的某种异常如扫描和 DoS,DDoS攻击引发的 TCP某个标志报文数量的大幅改变,将显著改变 Counting Bloom Filter的哈希映射空间的特征,本文的算法可以迅速、准确地再现异常的受害者/发起者的 IP或 IP分布.对仿真和实际网络的详细分析表明,本算法对于揭示大规模 TCP异常行为的类型和参数非常有帮助.

借助本文的算法,使用 Counting Bloom Filter的哈希映射空间来维护 TCP五元组和各类 IP信息,在分钟的粒度上,即使是主干网(1Gbps以上),维持一天的数据只需要数十兆的存储空间.而使用 Counting Bloom Filter记录下一天主干网的活动,还原的过程非常类似于给主干网的报文活动做细致的断层扫描,将更贴近 Tomography的概念.

参考文献:

[1] Burton B. Space/time trade-offs in hash coding with allowable errors[J]. Communications of the ACM, 1970, 13(7): 422-426.

[2] Chang C C, Lee T F, Leu J J. Partition search filter and its performance analysis[J]. Journal of Systems and Software, 1999, 47(1): 35-43.

[3] Little M C, Speirs N A, Shrivastava S K. Using bloom filters to speed-up name lookup in distributed systems[J]. The Computer Journal, 2002, 45(6): 645-652.

[4] Angiulli F, Pizzuti C. Outlier mining in large high-dimensional data sets[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(2): 203-215.

[5] Sarang D, Praveen K, David E T. Longest prefix matching using bloom filters[A]. Proceedings of SIGCOMM 2003[C]. New York: ACM Press, 2003. 201-212.

[6] Kumar K, Xu J, Jia W. Spatschek O, Li L. Space-code bloom filter for efficient per-flow traffic measurement[A]. INFOCOM [C]. New York: ACM Press, 2004. 1762-1773.

[7] Nicolas H, Darryl V. Inverting sampled traffic[A]. Proceedings of IMC[C]. Miami: ACM Press, 2003. 222-233.

[8] Kumar A, Sung M, Xu J, Wang J. Data streaming algorithms for efficient and accurate estimation of flow size distribution [A]. ACM Sigmetrics[C]. New York: ACM Press, 2004. 177-188.

[9] Robert S, Ashish G, Elliot P, Yan C. Reversible sketches for efficient and accurate change detection over network data

streams[A]. Proceedings of IMC[C]. Sicily, Italy: ACM Press, 2004. 207-212.

[10] Andrei B, Michael M. Network applications of bloom filters: a survey[J]. Internet Math, 2003, 1(4): 485-509.

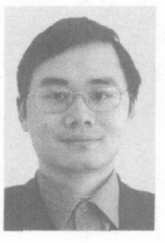
[11] Eddie K, Jinyang L, Vern P, Scott S. Observed structure of addresses in IP trafic[A]. Internet Measurement Work-shop[C]. Marseille: ACM press, 2002. 253-266.

[12] GNU Scientific Library GSL-1.4[DB/OL]. http://www.gnu.org/software/gsl/, 2005-04-15.

[13] Network Telescope[DB/OL]. http://www.caida.org/analysis/security/telescope/, 2005-01-25.

[14] hping tools[DB/OL]. http://www.hping.org/, 2005-03-11.

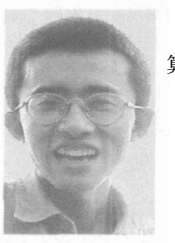
作者简介:



彭艳兵　男, 1974年生于湖北, 东南大学计算机科学与工程系 03级博士生, 主要研究方向为网络行为学. E-mail: ybpeng@njnet.edu.cn

龚　俭　男, 1957年生于上海, 博士, 东南大学计算机科学与工程系教授, 博士生导师, 主要研究方向为网络安全、网络行为学等.

刘卫江　男, 1969年 8月出生, 博士后, 东南大学计算机系、渤海大学计算机系教授, 主要研究方向为网络测量、网络行为学等.



杨　望　男, 1979年生于安徽, 东南大学计算机系博士生, 主要研究方向为网络行为模拟.