Disclosing the Element Distribution of Bloom Filter

Yanbing Peng, Jian Gong, Wang Yang, and Weijiang Liu

Department of Computer Science and Engineering, Southeast University
Sipailou 2, Nanjing, Jiangsu, P.R. China 210096
{ybpeng, jgong, wyang, wjliu}@njnet.edu.cn

Abstract. An algorithm named Reconstruction based on Semantically Enhanced Counting Bloom Filter (**RSECBF**) was proposed to disclose the distribution of original element from semantically enhanced Counting Bloom Filter's hash space. The algorithm deploys **DBSM,** which directly selects some bits from original string as the reversible hash function. The overlapping of hash bit strings in this paper brings the ability to confirm the homogenetic hash strings. The efficiency of **RSECBF** in the principal component analysis was verified by the DDoS detection in a published trace.

1 Introduction

Bloom Filter was created by Bloom in 1970 [1] to compress the searching space of string and cut down the error rate by multi-hash functions. The Bloom Filter became a popular tool in networking studies. In the studies on network [2], the Bloom Filter was implemented as an ingenious classifier. The newest paper on reversible sketches appeared in IMC 2004 [3]. The sketcher needs uniform distributed hash functions derived from the non-uniform distributed original strings, which makes it anfractuous.

The key issue of the reconstruction from different multi-hash strings to original string is to confirm that these strings are homogenetic. The Directly Bit String Mapping (**DBSM**) hash functions select some bits directly from the original string. The overlapped bit string between different hash functions confirms their homogenetic property, which means that they are from the same original string.

2 Properties of Semantically Enhanced Counting Bloom Filter

We found out that if the hash function is treated as aggregating rules in partition the original strings, the hash function can be any distributed. We call Counting Bloom Filter as semantically enhanced one when its hash functions select some bits directly from the original string. For example, a hexadecimal string, *0xabcd*, when using directly selected hash functions, the higher two octets is *0xab*, and the lower two octets is *0xcd.* The two hash functions, as a part of the original string, have semantical implication. These semantically enhanced hash functions are reversible and easy to be calculated, which are called as Directly Bit String Mapping (**DBSM**).

For **DBSM** in the space-independent Counting Bloom Filter:

Property 1: *One original string is mapped into the hash space once and only once.*

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part I, LNCS 3991, pp. 1022 – 1025, 2006.

Property 1 can be induced from the work principle of Bloom Filter. It suggests that, when an original string x_i appears C_{xi} times in set S, its hash strings will appears in each hash space at least C_{xi} times.

The ordinary length of hash string is a trade-off between space efficiency and accuracy. The more the amount of the hash functions are, the more the calculations are needed, but the longer the hash string is, the larger the space overhead is. The general length of the hash string is in the range of 8~24 bits, and a 16 bits' hash string is a suitable choice for most of the applications.

When the original string is unclear, it is a key point to confirm which hash strings are homogenetic for the different hash strings. The overlapped bits among different **DBSM** hash strings deployed in this paper really benefit the homogenetic judgement. The hash values from two hash functions may be homogenetic if their overlapped bits have the same value. The hash strings can be combined with each other to a longer one by the patchworks, i.e., the overlapped string. When the splice is failed, the shorter string would be the distribution of the original strings.

For 32-bit original string such as IP address, the highest 16-bit string is selected as hash function H_h, the middle 16 bits as H_m, and the lowest 16 bits string as H_l. The overlapping relationship can be described by **Property 2**:

***Property 2**: The sum of those counters in H_h with the same value overlapped string is equal to that in H_m. So is it between H_m and H_l.*

Property 2 can be induced from **Property 1** and the overlapping relationship between two hash functions. If a string, e.g. IP address a.b.c.d appears C times in set S, the short string a, b, c, d, which also expresses their location in the original string, must appears at least C times in respective hash space. When a.b.c.d is active, the short string in respective hash space would be active too. When a.b in H_h actively appears C times, b.c in H_m as an active homogenetic candidate of a.b in H_h, would appear at least C times in H_m, or the string a.b.0.0/16 should be an active aggregation.

3 Original String Distribution Discovery

If the hash strings' value is Pareto distributed in the Counting Bloom Filter's hash space, the principal component analysis in hash space will disclose the remarkable changes on the aggregation by *Top N*. This method is very suitable for those large scaled abnormality detections which change IP address greatly in distribution. For example, Eddie Kohler *et al.* [4] found that the active IP addresses are Pareto distributed, so the distributions of hash function IP_h, IP_m, IP_l are Pareto distributed too. The scanning behavior aggregates its source at the launchers' IP address, but the DDoS behavior focuses its aggregation character on the victims' destination IP address, which can be disclosed easily by **Property 2** and the principal component analysis in the Counting Bloom Filter's hash space.

The next paragraph proposes an algorithm, which is called **R**econstruction with **S**emantic **E**nhanced **C**ounting **B**loom **F**ilter (**RSECBF**), to reconstruct the original string, or disclose the aggregation character of the original strings.

```
Get the Top N for $H_p$ and $H_q$ by their counters
Get the most active hash string $x.y$ and $y.z$ in $H_p$ and $H_q$
Analyze the principal component according to Property 2, for each $y$
        if there is no $y.z$ responding to $x.y$, it suggests that $x.y$ is an active prefix;
        else if y.z is the principal component, $x.y.z$ is an active original element;
        subtract x.y.*/x.y.z in their respective counter;
Repeat the analysis till the active hash string is exhausted
```

The access to hash space string is k times per original string; ordinarily k and N in the *Top N* are very small. The primary calculation of **RSECBF** is the overhead in the sorting of the *Top N* which can be carried out by some quick methods in paper [5]. The space size m is 2^{16} for 16-bits string; so the spatial complexity is $O(m + N)$, and the calculation complexity is $O((m + NlogN)^* k)$.

4 Actual Effect of RSECBF

The trace set published by Caida.org [6] was analyzed and the function of the distribution discovery was verified. Since the packets in this trace are known as backscatter packets which contains only responses from the attacked victim that went back to other IP addresses. It provided us a perfect chance to disclose the abnormal behavior and reconstruct their distribution of IP addresses and ports.

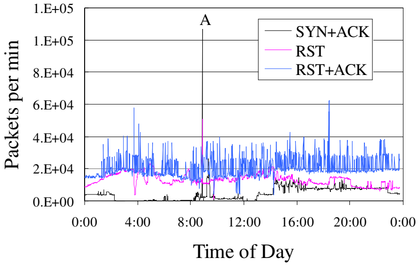

Fig. 1. The SYN+ACK, RST, RST+ACK packets' waves at Backscatter DDoS Trace [6]

The Backscatter's SYN+ACK/RST packet is a response packet for the spoofed SYN packet or other TCP packet (except RST packet) in DDoS attack, so its amount per minute can be used as an index to discover burst of abnormal behavior. Peak A in Figure 1 shows the burst of SYN+ACK/RST at 8:52 am, and then IP addresses and ports of related packets in Peak A are dug by **RSECBF** algorithm.

Table 1 shows that the two most active source IP hosts are 65.61.216.31 and 61.152.96.19 as well as the most active port is 80. The destination IP addresses are aggregated at 0.166.0.0/16 subnet at random in Table 3. The results suggest that 65.61.216.31 should be the victim, a web host.

The results imply that **RSECBF** has high efficiency in reconstructing the distribution characteristic of IP from their hash space when abnormal behavior occurs.

Table 1. Top N in SYN+ACK's Counting Bloom Filter hash space, 107005 packets

DBSM	H_h		H_m		H_l		*PORT*	
	Hash	Hits	Hash	Hits	Hash	Hits	Hash	Hits
Source IP and source port	**65.61**	**104585**	**61.216**	**104585**	**216.31**	**104585**	**80**	**106974**
	61.152	2214	152.96	2214	96.19	2214	21	24
	219.139	143	139.24	143	240.176	143	14364	5
Destination IP and Port	**0.166**	**104598**	166.33	1130	0.44	10	1336	189
	0.16	19	166.28	1078	2.237	10	1234	185

5 Conclusion and Future Works

The improvements of Bloom Filter in this paper extend the hash function to non-uniform distributed one with semantically implication. The reversible hash function, Directly Bit String Mapping (**DBSM**), makes the distribution discovery of original element easily. The overlapped bit string between different **DBSM** hash functions merges the homogenetic hash string into the original string by a simplified algorithm, which is called **RSECBF** for the Pareto distributed hash functions. The high efficiency of **RSECBF** in DDoS attacking detection is verified in a published trace.

Acknowledgement

This paper is supported by the National Basic Research Program of China (973 Program) 2003CB314804; the National High Technology Research and Development Program of China (2005AA103001)

References

1. Burton, H.B.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 13(7) , (1970):422–426
2. Andrei, B., Michael, M.: Network Applications of Bloom Filters: A Survey. Internet Math., no. 4, (2003)485–509
3. Robert, S., Ashish, G., Elliot, P., Yan, C.: Reversible Sketches for Efficient and Accurate Change Detection over Network Data Streams. IMC 2004, Taormina, Sicily, Italy, 207-212.
4. Eddie, K., Jinyang, L., Vern, P., Scott, S.: Observed Structure of Addresses in IP Traffic. Internet Measurement Workshop (2002).
5. Fabrizio, A., Clara, P: Outlier Mining in Large High-Dimensional Data Sets. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 2, (2005) 203-215
6. http://www.caida.org/data/passive/backscatter_tocs_dataset.xml