Source-Based Filtering Scheme against DDOS Attacks

Fasheng Yi[1,2], Shui Yu[1], Wanlei Zhou[1], Jing Hai [1] and Alessio Bonti[1]

[1] School of Engineering and Information Technology
Deakin University, Burwood, VIC 3125, Australia
[2] Department of Network Security, University of Electronic Science and Technology of China, Chengdu, Sichuan, 610054, P. R. China
{yifs, syu, wanlei, jinghai, abont}@deakin.edu.au

Abstract. IP address spoofing is employed by a lot of DDoS attack tools. Most of the current research on DDoS attack packet filtering depends on cooperation among routers, which is hard to achieve in real campaigns. Therefore, in the paper, we propose a novel filtering scheme based on source information in this paper to defend against various source IP address spoofing. The proposed method works independently at the potential victim side, and accumulates the source information of its clients, for instance, source IP addresses, hops from the server during attacks free period. When a DDoS attack alarm is raised, we can filter out the attack packets based on the accumulated knowledge of the legitimate clients. We divide the source IP addresses into $n(1 \le n \le 32)$ segments in our proposed algorithm; as a result, we can therefore release the challenge storage and speed up the procedure of information retrieval. The system which is proposed by us and the experiments indicated that the proposed method works effectively and efficiently.

Keywords: Network Security, DDoS, Packet Filtering.

1 Introduction

Distributed deny of service (DDoS) is one of the most significant threats to the security of the Internet. A key observation in defense against DDoS attacks is that DDoS attack traffic tends to use spoofed source addresses to disguise their true identities, which conceals attacking sources and dilutes localities in attacking traffic. The vulnerability and security deficiency of the TCP/IP suite brings the initiating DDoS attacks on easily, yet it is extremely hard to defend against them. The research on detection and defence against DDoS attacks has been explored widely in the research community [1], [2], [3], [4], [5]. Commonly, there are two categories in this field, invasion detection and attack packets filtering.

In general, the common IP spoofing types can be classified into three, which include, random spoofing, subnet spoofing, and fixed spoofing. It is critical to identify attack packets from legitimate Internet traffic, and further filter them when a DDoS attack occurs. Park and Lee [1] proposed the route-based packet filters as a form of

mitigating IP spoofing, which assumes that there is one single path between one source node and one destination node, so any packet with the source address and the destination address that appear in a router that is not in the path, should be discarded. Packet marking [2], [3] was also proposed to solve this problem. The basic idea is that routers on the attack graph mark the packets which are passing through the local router, a fingerprint will be established for an attack packet by the cooperation among the routers which are located on the attack path. The victim examines the fingerprints with the source IP address, and then identifies the spoofed packets. These methods have a number of essential flaws which limit their application in the real battle against DDoS attacks. First, it needs cooperation of all the routers on the attack path, which is obviously hard to be fulfilled on the Internet. In addition, victims are supposed to understand the whole network topology, this is impossible for large scale of DDoS attacks. Hence, in order to improve feasibility and efficiency against DDoS of spoofed address, an advanced method named IDPF [4] has been brought in. It is constructed from the information implicit in Border Gateway Protocol (BGP) route updates and is deployed in network border routers; however, this method does not have the strength to handle subnet spoofing address. Subsequently, a new method which is Hop-Count Filtering (HCF) [5] proposed another novel simplified scheme to identify packets whose source IP addresses is spoofed. The information about a source IP address and its responding hops from a server (victim) are recorded in a table at the server side when there are attacks free. Once an attack alarm is raised, the victim will inspect the incoming packets' source IP addresses and their responding hops to differentiate the spoofed packets. It is not compulsory for routers to collaborate mutually in this scheme; however, it is difficult to ensure the integrity and accuracy between the source IP addresses and their responding hops from the victim. Moreover, this method is subjected to subnet spoofing or fixed spoofing. Furthermore, this method requires a large memory to construct a rounded table because of the huge address space of 32 bits IP. The storage space can be saved by exploiting hashing or aggregating, but more occurrences are needed to query or locate.

The bloom filter has been used for defences against TCP SYN flooding which is a regular DDoS attack method [6]. It only requires a small quantity of memory without increasing the time of processing. But, there are many parameters configured manually and this scheme is not appropriate to network changing. Concurrently, it is deployed through routers and makes the hash operation on source and destination addresses. It is difficult to implement and obtain a good effect. Based on the feature of source IP address spoofing, Peng et al. [7] proposed a packet-filtering scheme on historical packets information. This method is applied on ingress routers. Generally, an ingress router keeps a history record of all the legitimate source IP addresses which have previously appeared in the network. When the ingress router is overloaded, the historical record is used to decide whether to admit or deny an incoming packet. In most of the cases, a router is connected by a large number of hosts, and attackers may infiltrate filtering algorithms using low attack packet rates. Again, with the method, it is hard to achieve a good filtering result.

The methods stated previously work with the DDoS feature of randomly spoofing source addresses. For skilled attackers, they might conduct more serious attacks with subnet spoofing or fixed spoofing. In current DDoS attacking, one or more attackers

exploit a great number of zombies in a large botnet to launch attacks, so it is very complicated to find the actual attackers, even if the zombies exposed their position information. Yet, the statistics-based filter methods PacketScore [8] and ALPi [9] noticed the problem, and proposed the statistical methods to cope with it at the potential victim site. A range of typical traffic attributes is accumulated by a victim. Usually, as a server, the victim identifies a DDoS attack by comparing the current traffic profile with the normal traffic profile, which is according to the current event. It is using a scoring mechanism to make decisions based on Bayesian probability or the leaky barrel theorem. Still, the statistics-based method is complex, since there are too many features involved. Simultaneously, because the methods employ the feature of bursting, they are not capable to filter low rate attacks, regardless of the types of spoofing address.

In this paper, we extract the advantages of the previous work and propose a novel attack packets filtering method based on the source information. The source IP address and the TTL value are treated as information sources in an IP packet. We do not treat the port number of a packet as information of data source, because source port numbers are not fixed for most servers as victims. The current IP address space is divided geographically, and references [2], [3], and [5] have confirmed that there is a strong correlation between the TTL values and the source IP addresses for normal traffic. Our principal methodology is to construct a table based on the statistics of source IP addresses under hops from that source IP addresses to the server in normal condition. During the period of DDoS attacking, the attacking packets with random spoofing would be filtered for their source IP addresses. Their hops would not accord with mapping relations in normal conditions, and the attacking packets with subnet spoofing or no spoofing would also be filtered out for their statistics profiles overrun their accumulated knowledge of the normal traffic. In order to reduce the size of storage space, we propose an advanced counting bloom filter to save the statistics of source IP address under the certain hops. We refer to our method as source-based filter (SBF). SBF is deployed at victim without the assistance of routers, and is convenient to put into practice.

Compared with the previous works, SBF possesses the following features and novel contributions to the battle against DDoS attacks,

- SBF works on the potential victim side, which has a strong incentive to implement the filtering function, moreover, no cooperation among routers is required.
- We deploy very limited information, source IP addresses and hops, to filter attacking packets, which simplifies the requirements for implementation.
- SBF reduces the utmost pressure of storage space and accelerate the information retrieval processes dramatically.
- Our method is excellent against various IP address spoofing, which is the main method of DDoS attacks so far.

The paper is organised as follows, our model and analysis will be presented in Section 2, certain problems of the implementation of SBF are discussed in Section 3, Section 4 focuses on performance evaluation, and finally, Section 5 concludes the paper and sketches the ongoing work.

2 Source-Based Filter Scheme

2.1 Scheme Analyzing

We observed that clients for a given server are relatively stable. Jung et al. [10] indicate nearly 82.9% of all IP addresses in observed flash crowd events have sent a request before; however, only 0.6-14% of IP addresses in a Code Red attack had happened. Based on statistical analysis of Internet traffic collected in a medium size stub network, we have found that a large proportion of IP addresses emergence in Internet traffic consistently re-emerges based on daily observation. This implies that a reliable feature for identifying legitimate traffic is that it has appeared at the website regularly. In the meanwhile, the hop from a client to the server is fixed relatively. Based on this, a server can establish its own clients IP address space under a certain hop for each client, respectively. When hackers spoof the source IP addresses randomly and deliver the packets to the victim (server), therefore, the probability is low that the spoofed IP address locates in the specific clients IP address space under a given hop. Our method is inspired by this analysis. However, holding this information by creating a table could be very expensive. Because it occupies servers' immense memory spaces, and it also tardily conducts information retrieval in a huge table. Accordingly, in order to solve these problems, we introduce the bloom filter, which has been implemented [6]. Our method deployed at victim instead of router, in which the bloom filter is just constructed with source addresses, which replace the couples of source and destination addresses, the outcome would be improved compared with the method in paper of Chen et al. [6].

Bloom filter is likely to produce a bit of false positive rate, but it decreases rapidly the requirement of memory. Bloom filter was first proposed by Bloom [11] in 1970.Recently, it has been adapted for use in some methods for defending against DDoS attacks [12, 13, 14]. Counting Bloom filter is the variation of standard bloom filter by adding function of counting. For the sake of lightening the computing load of SBF at victim, we improve the counting Bloom filter algorithm named partial counting Bloom filter. The methodology in SBF is as follows.

we divide the IP addresses into *n* segments (n=1,2,3,4…32), each segment is 32/n bits, we regard the segment as Part_IP, which has the value from 0 to $2^{32/n}$. Commonly, when receiving a packet, the SBF in a victim counts n Part_IPs of source IP of the packet, easy to get the normal statistics profile of all the Part_IPs of legal source IP addresses. When attacking, according to the theory of Bloom filter, if the corresponding statistics value of one of the Part_IPs of the packet received was 0, the packet would be the attacking packet, should be discarded. Here, the hashing function turns into the direct mapping that acquires little CPU load of victim. In the other side, the method is useful to identify the attacks of subnet spoofing.

If the attackers exploit the legal IP addresses to spoof the source IP addresses of attacking packets, bloom filter can't filter these packets well. Thus, we were illumined from the idea of HCF [5], introduce hops to separate the source IP addresses. For a special server, the hops distribution of its clients takes on certain statistics laws [5]. On attacking, many zombies send attacking packets to the server, the counting

number of relating hops of these packets rises by a large amount. If the statistics of a hop-count value is found to exceed the normal statistical profile, we deduce that the sources of attacking fabrication in the distance of the hop. The particular step is as follows. As receiving a packet, we first calculate its hop-count according to HCF, and accumulate the counts of the hop-count. Then, we construct partial counting Bloom filter as stated previously. In this way, we solve the problem of spoofing legal address, for different zombies to a victim have the different hop-count.

Every TCP/IP attribute values of arriving packets of a victim has a nominal profile which is a set of baselines collected during a period in which the victim is free of attacks [8]. We only use both attributes, hop-count coming from TTL and Part_IPs from source IP address, to construct the traffic profile. Similar to PacketScore, by comparing the current profile against the nominal one, SBF is also able to distinguish legitimate packets from DDoS attacking packets of subnet spoofing or fix spoofing, even no spoofing. Relative to PacketScore, we just make use of two attributes and relieve the load of victims.

2.2 Scheme Implementation

We build an information table according to the statistics of source IP addresses in case of hop-counts shifting between an Internet server and its clients. Given an IP address, we divide it into n segments, $S_0, S_1, \ldots S_{n-1}$, $1 \leqslant n \leqslant 32$, and the range of each segment is 0 to $N-1$, $N=2^{32/n}$. Based on reference [5], we know the hops from one computer to another on the Internet is able to be calculated by TTL, and the maximum hop is 31. we denote HC as the hop counter, therefore, $0 \leqslant HC \leqslant 31$. For a given server, we accumulate information for each of the segments on each hop value, respectively. For example, Figure 1 shows the part of the information table where $HC=m$.

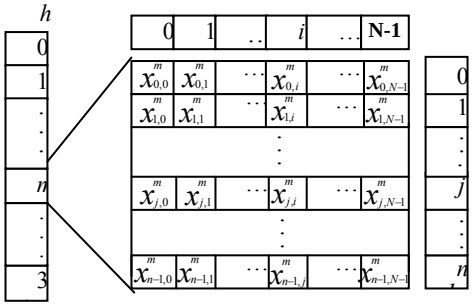

Figure 1. The snapshot for the information table with $hc=m$.

We denote $x^m_{j,j,}$ for statistical value of the values i of the j^{th} segment of the source IP address, we have $0 \leq j \leq n-1$, $0 \leq i \leq N-1$. For the bloom filters with the different hops are the same form, in order to make it clear, we only focus on one of the bloom filters in the rest of this paper, and therefore, omit the m in the variables. So $x^m_{j,i}$ is written $x_{j,i}$ for short. In fact, $x_{j,i}$ includes two parts, namely $a_{j,i}$, $b_{j,i}$, in which $a_{j,i}$ represents current statistics, and $b_{j,i}$ stand for normal profile. In normal situation, a victim adjusts

periodically its value of $b_{j,i}$ based on $a_{j,i}$, which is accumulated by the source IP addresses coming various legal clients.

The evolution suspends when attack occurs. Once a DDoS attack alarm is raised, we check every incoming packet S' according to it hops, and chop S' into n parts, then $S' = \{S'_0, S'_1, ..., S'_{n-1}\}$. For each part S'_j, we can get the score of the part as follows:

$$g(S'_j) = \begin{cases} a_{j,k} - b_{j,k} & a_{j,k} > b_{j,k} \\ \infty & b_{i,k} = 0 \\ 0 & a_{j,k} < b_{j,k} \end{cases} \quad (1)$$

In which k is the value of S'_j, We apply the following norm as a metric to distinguish between the attack packet and legitimate packet by identifying the incoming packet S'

$$\| f(s') \| = \sum_{j=1}^{n-1} g(s'_j) \quad (2)$$

We compare $\| f(s') \|$ with a given threshold δ. If the following holds, then it is an attack packet.

$$\| f(s') \| \geq \delta \cdot w \quad (3)$$

the intensity factor of attacks. Less w represents stronger attacks. It has something to do with the global statistics of the hop-count's packets. Set a_m and b_m as the current profile and nominal profile when hop-count equals to m, we have $w = b_m/a_m$. If w is 0, it shows that there is no legal packet come before, so we regard the arriving packet with the hop-count is attacking packet. When w is bigger than 1, it expresses that the arriving packet is the legal packet for being out of attacks level under its hop count.

We draw the threshold δ according to the equation (4) as follows:

$$\delta = n \max_{i=0, j=0}^{n-1, N-1} | a_{j,i} - b_{j,i} | \quad (4)$$

Apparently, δ is n times of the maximum of statistics' changing under certain hop_count. For example, if a packet with an IP address which never appears before, then $a_{i,j} > 0$ and $b_{i,j} = 0$, as a result, $||f(S')|| = \infty$, and it is an attack packet. This answers for the idea of Bloom filter.

3 System Analysis and Algorithm

3.1 Memory Requirement

SBF divides source IP address into n Part_IPs, and counts them separately. So we can get the needful space units as $y = n*2^{32/n} * y$ would decrease sharply with n increasing. In order to improve the effect of the filter, we build the filter under each

hop count. But it increases the memory needed. Now give the detailed analysis of the storage space consuming in our method.

Assuming there are legal source IP addresses under a hop, we set the space units whose size is y. As described previously, every space unit requires saving current profile and nominal profile. The range of its value involves the interval of counting. Commonly, it is enough to have 2 bytes to represent the value for a usual server. Thus, the required space of SBF is

$$men(m) = m \cdot y \cdot (2+2) = 4m \cdot n \cdot 2^{32/n} (bytes) \quad (5)$$

Where m is the number of hop counts. The number is less than 32 from 0 to 31. More often, the number of hop counts of the packets arriving at a server is among from 10 to 20. Considering m maybe different in each server, SBF applies dynamically the storage space under a hop count, and avoid the waste of memory. For instance, if n is 4 and m is 16, the storage space is 64 KB. If we change n as 3, the needed space is 384 KB in other similar conditions. Comparing to HCF, if the memory is allocated statically, SBF occupy far less space than HCF; if HCF applies dynamic tree structure to store data, they use almost the same space, but HCF spend extra time querying data in the same situation.

3.2 Filter Efficiency

The false negative and the false positive represent efficiency of a filter, where the false negative is the probability of taking the attack packets as legal packets, and the false positive shows the probability of taking the legal packets as attack packets.

In our scheme, under the certain hop count, suppose T is the set of legal source IP addresses, and the number of entries of the set is t. the sets of its n Part_IPs are T_0, T_1, … T_{n-1}, which include the entries of t_0,t_1,…t_{n-1}. As the attacks with randomly spoofing are coming, the probabilities of each segment of the packets is located on corresponding segment of normal packets are t_0/N, t_1/N, … t_{n-1}/N. Hence, under randomly spoofing condition, the false negative is:

$$P_{fn} = \prod_{i=0}^{n-1} (t_i / N) = (\prod_{i=0}^{n-1} t_i) / 2^{32} \quad (6)$$

According to equation (6), P_{fn} is proportion with the product of Part_IP's distribution of legitimate packets' source IP addresses. Because SBF isolates source IP addresses to different fields based on their hops, and considering the district feature of IP, our method reduce the false negative more effectively than other methods.

On the basis of bloom filter theory, if all the regular packets have visited the server in learning stage, the false positive does not exist. But in the real environment, it is inevitable that some clients access the server in the first time during attacking periods. According to the rule described previously in this paper, if the clients are sited in where w>1 (the attacking sources don't situate the areas of the clients), SBF could not take the packets of these clients for attacking packets and filter them. In

some degree, our method protects the new clients in whose area there is no attacks source and decreases the ratio of false positive.

To subnet spoofing, the situation becomes complicated. Suppose the fixed part is k segments preceding the attack packets (k<n). When k is very small, for example k=1, we can use equation (6) to estimate the false negative, here the equation become:

$$P_{fn} = \prod_{i=k}^{n-1} t_i / N \tag{7}$$

When k is close to n, the false negative will get bigger according to equation (7). Our method uses the scoring way to reduce the false negative. With the attacks lasting, the attack packets will get high score which finally overrun the threshold to be filtered. But the Part_IPs of subnet spoofing will reach high score and the legitimate packets which have some of Part_IPs maybe get so high score that they are taken falsely for attack packets. In order to prevent this situation, we optimize the score rule of equation (1) to the follows

$$g'(S_j') = \min(\delta / r, g(S_j')) \tag{8}$$

The equation (8) sets the upper limit of scoring, in which r is a proportional factor that is configurable. Usually, we set it as n-1. Only when k is equal to r, the legitimate packets that have the same subnet of spoofing maybe erroneously identified as attacks packets. Assuming the number of Part_IPs of subnet spoofing is q_0, q_1,…q_{k-1} correspondingly , when k>1, the false positive is

$$P_{fp} = \prod_{i=0}^{k-1} q_i / t_i \tag{9}$$

It is easy to observe that the false positive is proportion to the ratio of Part_IPs distribution product of spoofed subnet to that of legal IP addresses. it is impossible for spoofed subnets to be excessive, so the false positive is limited to low proportion.

All previous analysis shows that the false probability of our method is closely relate to the Part_IPs's distribution of legitimate IP addresses. The range of the number of legal IP addresses is

$$\max_{i=0}^{n-1}(t_i) \le t \le \prod_{i=0}^{n-1} t_i \tag{10}$$

Obviously, with the increase of n, even if t_0, $t_{1\ldots}t_{n-1}$ is constant, the value of t can be changed in a substantial scale. Therefore, the greatest impact of our method consists in the distribution of the segments of legitimate IP addresses, not its number. However, we separate the legitimate IP addresses with hop counts, and cut down the distribution product of the segments in the same number of IP addresses.

3.3 Adaptive Analysis

It is necessary for a good filter to be in possession of nice adaptability. The network environment is continually changing. It is a bad design if a filter needs to configure

manually in different environment to reach fine effect. We discuss the adaptability of SBF in the following.

Firstly, though network keep relative stabilization in certain period, but the topology of network will transform in a long time. The hops of some client to the server would vary. Furthermore, in different time, a server has variously nominal profile of the segments of source IP addresses of legal client. In order to adapt these changes, we apply the equation (11) to update the normal statistics profile.

$$b_{ji} = b_{ji\text{-}old} . \alpha + a_{ji} . (1-\alpha) \tag{11}$$

The operation works in an interval. Where α is a correlation factor, $0<\alpha<1$, which is decided by the interval and the changing regularity of the clients of a server. When the hop of a client or the number of clients to a server varies, the corresponding a_{ji} would reduce to zero. After some time, the nominal profile b_{ji} would decrease, even drop to zero, changing adaptively. On attacking, b_{ji} stops updating to carry out scoring and filtering; when attacks halt, a_{ji} is set as 0, and the updating operation works again.

4 Performance Evaluations and Algorithms

4.1 Performance Evaluations

In order to evaluate the proposed method, a prototype of DDoS attack system has been established, as an example, the SYN Flooding, which is the most well-known DDoS attacks, is employed in our method. We use the cusum algorithm [15] for attacking detection, and the proposed method works as the packet filter at the victim side. For contrast, we also implement the 32 bits strict filtering algorithm of HCF.

First of all, we examined the false negative and the false positive, when the number of segmentations varies. The number of clients simulated is kept at a fixed level of 20000. The results are shown as Figure 2.

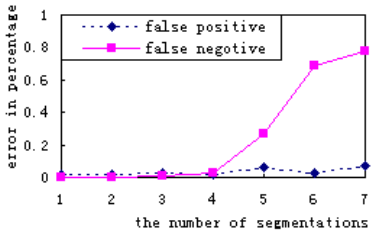

Figure 2. Errors versus number of segmentations

The simulation shows that when the number of segmentations is not greater than 4, both of the errors are quite low (both of them are not greater than 2.5%), and the false negative increases dramatically with the number of segmentations grows. With the growing of the number of segmentations, the proportion of the instances in a segment to the segment's space increases, so the false negative rises according to equation (4).

The false positive keeps stable, because it is related with the number of legitimate clients who never access a server before, which is independent from the number of segmentations.

In the case of random spoofing, Figure 3 and Figure 4 illustrate the relation between error ratio and the number of clients, comparing with HCF algorithm. Therefore, based on our method, When n is equal to 3, the outcome indicates SBF is the same as HCF. With the number of clients growing, the false negative increases relevantly. In fact, the false negative is related closely with the distribution of clients to the server. The false positive increases slowly along with the growing of the number of legitimate clients, because more clients maybe absent in the learning procedure of our algorithm.

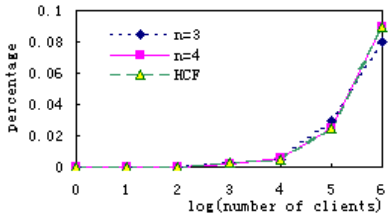

Figure 4. False Positive with IP random spoofing

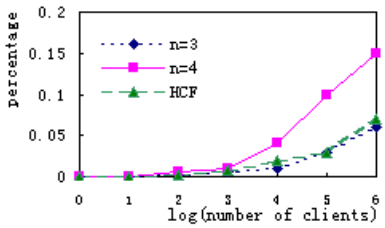

Figure 3. False Negative with IP random spoofing

Then, we contrast the impact of different methods with subnet spoofing. We deploy the attack packets producing randomly in some subnet of C class. The results are shown as figure 5 and figure 6. The figures reveal that our method improves apparently the false negative; yet, the false positive in our method is not as good as HCF. The main reason is that the scoring way influences the legal IP addresses which are identical to subnet spoofed. In the meanwhile, we find that the efficiency of filtering of our method is connected with time. As the time passes by, the false negative continually decreases, and false positive increases for a while to become steady.

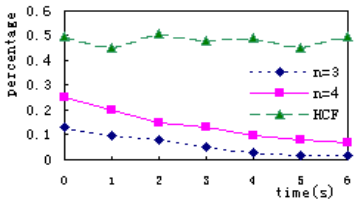

Figure 5. False negative with IP subnet spoofing

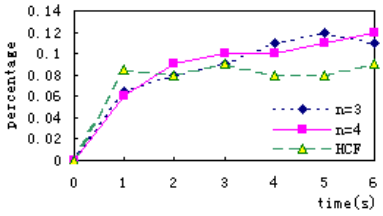

Figure 6. False Positive with IP subnet spoofing

Besides, our method accommodates automatically the attack intensity. For one hop of m, we add up all the incoming packets. In terms of the normal statistics of b_m and current statistics of a_m, the attack intensity w can be calculated. By equation (3), SBF improves the defense effect and decreases the false for the different hops have various attack intensity.

Finally, the scheme is capable of corresponding for the change of the attacks way. We previously analyze that SBF adapts to the attacks of random spoofing and subnet spoofing. In addition, if attackers fake the TTL values to deceive us by getting error hops, our method would get a good result as well. It is mostly because attackers are not able to alter the profile of the hop, as set forth, the attack packets would be filtered according to the regularity stated previously.

4.2 Filter Algorithms

After reviewing the previous statements, we consider the situation into two, first, in the normal condition, when there is free no attacks, we provide the algorithm of SBF as follows,

1. Calculate hop count *m* from the TTL field of the received packet 2. If there is no statistics under the hop, apply the space of $4.n.2^{32/n}$ bytes. If attacking stage has just end, set all of a_{ji} to 0 and set a_m to 0. 3. Carry out the statistics in terms of the segment values f the IP address. 4. If the updating time expires, make updating operation by equation (11).

Table 1 algorithm for attack free cases

In the other hand, we implement the filter algorithm in a different way, when attack occurs.

1. Calculate hop_count *m* from the TTL field of the received packet. 2. If the normal statistics b_m is 0, discard the packet as a attack packet and be over. 3. Carry out the statistics in terms of the segment values of the IP address. 4. If $b_m > a_m$, be over 5. Score the packet according to the equation (3). If the equation (3) holds, the packet is discarded.

Table 2 algorithm for ongoing attacks

5 Conclusions and Future Work

We proposed a novel DDoS attack packets filtering algorithm in this paper, which is based on source information statistics. Our method based on a server's clients source information, e.g. source IP addresses and responding hops, and follows some statistical patterns; however, it is hard for the spoofed packets to match the patterns. The proposed method accumulates the statistical information, e.g. source IP addresses and hops, during the non-attacking period, and therefore differentiates attack packets once the server is under attack. We divided the source IP addresses into n segments, rather than treat them as a whole. Compared with the previous works, our proposed method offers high efficiency of diversely spoofed packets filtering. Moreover, it works independently at the potential victim side, and there is no cooperation among

routers are needed in the proposed scheme. As the result, the proposed method releases the challenge on storage space and speed of information retrieval at victims.

The future works can be explored in the following two promising directions.

- The storage structure can be improved further to reduce memory requirement. For example to deal with the sparse space in the information table, many spaces of $b_{ij}=0$ in our scheme could be handled nicely to release the challenge on storage space.
- The scoring model can be improved with finer granularity, e.g. models based on Bayesian theorem, Kullback-Leibler distance, etc.

References

1. K. Park and H. Lee, "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets," Proc. ACM SIGCOMM, Aug. (2001).
2. Abraham Yaar, Adrian Perrig and Dawn Song, " StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense", IEEE Journal On Selected Areas In Communications, vol. 24, no. 10, Oct. pp.1853-1863.(2006)
3. Fu-Yuan Lee, Shiuhpyng Shieh, "Defending against spoofed DDoS attacks with path fingerprint", Computers & Security, 24, pp.571-586.(2005)
4. Zhenhai Duan, Xin Yuan, Jaideep Chandrashekar. Controlling IP Spoofing through Interdomain Packet Filters. IEEE Trans. On Dependable and Secure Computing, Vol. 5, No. 1, JANUARY-MARCH , pp.22-36.(2008)
5. Haining Wang, Cheng Jin, and Kang G. Shin. Defense Against Spoofed IP Traffic Using Hop-Count Filtering. IEEE/ACM Trans. On Networking, vol. 15, no. 1, Feb, pp.40-53.(2007)
6. Wei Chen, Dit-Yan Yeung. Defending Against TCP SYN Flooding Attacks Under Different Types of IP Spoofing. in Proc. IEEE ICNICONSMCL, (2006).
7. Tao Peng, Leckie, C. Ramamohanarao, K. "Protection from distributed denial of service attacks using history-based IP filtering", ICC '03. May , Vol.1, pp: 482- 486.(2003)
8. Yoohwan Kim, Wing Cheong Lau, Mooi Choo Chuah and H. Jonathan Chao, "PacketScore: A Statistics-Based Packet Filtering Scheme against Distributed Denial-of-Service Attacks", IEEE Trans. On Dependable and Secure Computing, vol. 3, no. 2, Apr, pp.141-155. (2006)
9. Paulo E. Ayres, Huizhong Sun, H. Jonathan Chao, Wing Cheong Lau, "ALPi: A DDoS Defense System for High-Speed Networks", IEEE Journal On Selected Areas In Communications, vol. 24, no. 10, Oct, pp.1864-1876.(2006)
10. Jaeyeon Jung, et al. "Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites". WWW10, WWW2002, May 7-11, Honolulu, Hawaii, USA (2002).
11. B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 13(7):422—426, July (1970).
12. S. Abdelsayed, D. Glimsholt, C. Leckie, S. Ryan, and S. Shami. An efficient filter for denial-of-service bandwidth attacks. In IEEE Global Telecommunications Conference(GLOBECOM' 03), volume 3, pages 1353— 1357, Dec (2003)
13. E. Chan, H. Chan, K. Chan, V. Chan, S. Chanson, and etc. IDR: an intrusion detection router for defending against distributed denial-of-service (DDoS) attacks. In Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks 2004(ISPAN' 04)., pages 581—586 (2004)
14. Sertac Artan, N. Sinkar, K. et al. Aggregated Bloom Filters for Intrusion Detection and Prevention Hardware. Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE, Nov. pp. 349-354 (2007)
15. Haining Wang, Danlu Zhang, Kang G. Shin, "Detecting SYN Flooding Attacks", in Proc. IEEE INFOCOM, pp.1530-1539 (2002)