Defending Against TCP SYN Flooding Attacks Under Different Types of IP Spoofing

Wei Chen[†] [‡] Dit-Yan Yeung[†]

[†]Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

[‡]Computer School
Wuhan University
Wuhan 430072, Hubei, China

E-mail: {wchen,dyyeung}@cs.ust.hk

Abstract

TCP-based flooding attacks are a common form of Distributed Denial-of-Service (DDoS) attacks which abuse network resources and can bring about serious threats to the Internet. Incorporating IP spoofing makes it even more difficult to defend against such attacks. Among different IP spoofing techniques, which include random spoofing, subnet spoofing and fixed spoofing, subnet spoofing is the most difficult type to fight against. In this paper, we propose a simple and efficient method to detect and defend against TCP SYN flooding attacks under different IP spoofing types, including subnet spoofing. The method makes use of a storage-efficient data structure and a change-point detection method to distinguish complete three-way TCP handshakes from incomplete ones. Simulation experiments consistently show that our method is both efficient and effective in defending against TCP-based flooding attacks under different IP spoofing types.

1 Introduction

Distributed Denial-of-Service (DDoS) attacks are large-scale cooperative attacks typically launched from a large number of compromised hosts. DDoS attacks are bringing about growing threats to businesses around the world. While many methods have been proposed to counter such attacks, they are either not efficient or not effective enough. Some recent DDoS incidents show that such attacks continue to cause serious threats to the Internet.

DDoS attacks are even more difficult to fight against if IP spoofing is incorporated into such attacks. IP spoofing, or called source IP address spoofing, refers to the technique of lying about the return address (i.e., source address) of a packet. With IP spoofing, attackers can gain unauthorized access to a computer or a network by making it appear that a message has come from a certain trusted machine by "spoofing" the IP address of that machine. Strictly speaking, IP spoofing is not an attack by itself; it is merely a scheme used with DDoS attacks.

Spoofing techniques can be categorized into different types according to what spoofed source addresses are used in the attacking packets. The three common IP spoofing types are random spoofing, subnet spoofing, and fixed spoofing [8]. In random spoofing, the attacker randomly generates 32-bit numbers for use as source addresses of the attacking packets. In subnet spoofing, the addresses are generated from the address space corresponding to the subnet in which the agent machine resides. For example, a machine which is part of the 143.89.124.0/24 network may spoof any address in the range from 143.89.124.0 to 143.89.124.255. Another type of IP spoofing, called fixed spoofing, chooses source addresses from a given list. In this case, the attacker typically wants to perform a reflector attack or impose a blame for attack on several specific machines.

To defend against spoofed flooding traffic, especially that with subnet spoofing, we propose a scheme that is based on a storage-efficient data structure and a change-point detection method. The storage-efficient data structure, which is a variant of Bloom filter [3], is used to generate a hash digest of the traffic. The change-point detection method is based on the CUSUM algorithm [4], which is a nonparametric change-point detection method. After some information about the traffic is extracted and stored in the Bloom filter, CUSUM is then applied to detect abnormal changes in the digested traffic.

The remainder of this paper is organized as follows. We first review some basics about TCP handshakes in Section 2. In Section 3, a space-efficient data structure called Bloom filter is reviewed. Our method makes use of a modified Bloom filter to store a hash digest of the relevant portions of a packet. The CUSUM based change-point detection scheme is presented in Section 4. The experimental results reported in Section 5 show that our method can accurately detect spoofed flooding traffic under different IP spoofing types. Finally, Section 6 presents some related work in the area of DDoS research and Section 7 concludes the paper and outlines some further research issues.

2 TCP Handshakes

Most existing DDoS attacks exploit the Transmission Control Protocol (TCP) [9]. It has been reported that more than 90% of the existing DDoS attacks are TCP based [12], although we do expect that attacks of other types, such as UDP based, will increase in the future as more and more applications based on them become available in the Internet. It is well known that IP spoofing is one of the major security problems in the TCP/IP protocol suite, and hence it is commonly exploited by attackers to launch attacks. In what follows, we will first look at TCP handshakes for normal transactions and then those for spoofed ones.

During normal TCP handshakes, the client first sends a SYN request to the server. After receiving the request, the server replies with a packet which contains both the acknowledgement ACK and the synchronization request SYN (denoted as ACK/SYN hereinafter). Then the client sends an ACK back to the server to complete the establishment of the TCP connection.

Under IP spoofing, however, the three-way handshake will be very different from that of the normal case. Attackers usually use unreachable spoofed source IPs in the attacking packets to improve the attack efficiency [10]. These packets will not trigger the third round of an otherwise three-way handshake. Under random IP spoofing, most connections will not receive the second round of each handshake because the ACK/SYN packets are sent to other subnets. Under subnet spoofing, however, ACK/SYN packets are sent to the correct subnet but are destined to an incorrect host. The third round of the handshake is thus unsuccessful. As a consequence, a major difference between random spoofing and subnet spoofing is the different return paths of the ACK/SYN packets.

Our method tries to detect incomplete handshakes by monitoring the first and third rounds of each handshake. If either round is missing, it is regarded as an incomplete handshake. We use the first and third rounds because both of them belong to the outgoing traffic. Therefore, our method which requires only one-way traffic monitoring has the advantage of being flexible and hence can easily be deployed at the source side, the intermediate network or the victim side.

3 Traffic Information Digest

Accurate detection of anomalies has to rely on detailed information analysis. However, storing detailed traffic information for subsequent analysis is generally very expensive. In order to extract useful information about abnormal (i.e., incomplete) handshakes, we choose to record only information about TCP handshakes. Moreover, we use a storage-efficient data structure for this purpose. In this section, we first give a brief overview of such a data structure. We then discuss how to extract useful information from the network traffic.

3.1 Bloom filter

Bloom filter was first proposed by Bloom [3] in 1970. Recently, it has been adapted for use in some methods for defending against DDoS attacks [2, 5, 11]. A Bloom filter is composed of a vector $\mathbf{v}$ of m bits, initially all set to 0. We have k independent hash functions, $h_1, h_2, \ldots, h_k$, each with a range $0, 1, \ldots, m-1$. The vector $\mathbf{v}$ can show the existence of an element from some address space A. Given an element $a \in A$, the bits at positions $h_i(a), 1 \leq i \leq k$, in $\mathbf{v}$ are set to 1. Note that a particular bit may be set to 1 multiple times and hence may potentially lead to inaccurate results. Given a query on the existence of b in A, we check the bits at positions $h_i(b), 1 \leq i \leq k$. If any one of them is 0, then b is certainly not in A. Otherwise, we conjecture that b is in it. However, there is a certain probability that the Bloom filter will give a false result. This probability is referred to as the false positive rate.

A variant of the original Bloom filter, called *counting Bloom filter*, uses a table of counters to replace the n bits. Each counter represents the number of times the corresponding location has been hit. When a key a (such as an IP address) is inserted or deleted, the value of the corresponding counter in each row is increased or decreased by 1, respectively, according to $h_i(a)$ for all k rows. If an IP address b is already stored in the modified bloom filter, the counters at locations $h_i(b), 1 \leq i \leq k$, in the table should all be nonzero.

3.2 Traffic Digest

Two hash tables, which are based on the counting Bloom filter, are used in our scheme to record informa-

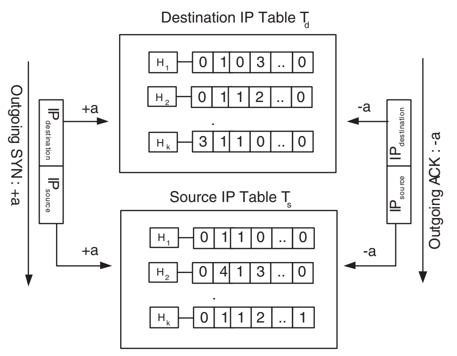


Figure 1. T_d and T_s together monitor TCP SYN and ACK packets.

tion about TCP handshakes. One table, called *destination table* and denoted as T_d, is used to record destination IP information. The other one, called *source table* and denoted as T_s, is used to record source IP information.

When a SYN request packet, corresponding to the first round of a handshake, is captured in the outgoing traffic, the destination IP of the SYN packet is hashed using the k independent hash functions and the corresponding counters in T_d hit by the k hash functions are incremented by α where $0 < \alpha \leq 1$. At the same time, another hash table T_s works in the same way but records the source IP information.

When an ACK packet, corresponding to the third round of a handshake, is captured in the outgoing traffic, both the destination and source IP addresses are extracted again and hashed into T_d and T_s, respectively. This time the corresponding counters are decremented by α where $0 < \alpha \leq 1$. For a normal TCP handshake, both SYN and ACK are observed and hence the corresponding counters are first incremented and then decremented by α, leading to no resulting changes. Figure 1 summarizes how tables T_d and T_s are used.

All entries in both tables T_d and T_s are reset periodically to prevent them from growing indefinitely when it is under attack and hence many incomplete handshakes are encountered. Suppose R_{t-1} is the value of a counter at time $t-1$. Its value is reset to R_t at time t as follows:

$$R_t = (1-\alpha)R_{t-1}, \qquad 0 < \alpha \leq 1.$$

Although it is possible that two different IP addresses are mapped to the same counter in a row, the probability that they get mapped to the same counters in all $k > 1$ rows is very low even for a small value of k. As a result, the false positive rate caused by hash table collision is rather low.

4 Detection Mechanism

In this section, we address the problem of detecting change points in the probabilistic characteristics of random sequences. When a packet is identified as suspicious, we analyze its source IP and classify it into one of three categories: random, subnet or fixed spoofing.

4.1 Change-Point Detection Method

The essence of sequential change-point detection is as follows. Suppose the observations of a random process X_t (with discrete or continuous time) are received sequentially. At a certain moment (random or not, but unknown), some probabilistic characteristics of this process change. An observer must make a decision as quickly as possible to decide whether or not a change point has happened, while keeping the false alarm rate as low as possible.

Suppose a sequence $X_1, ..., X_r$ of independent random variables has been observed. For each $1 \leq v \leq r$, consider the hypothesis H_v that $x_1, ..., x_{v-1}$ have the same density function $f_0(\cdot)$ and $x_v, ..., x_r$ have another density function $f_1(\cdot)$. Denote by H_0 a hypothesis of stochastic homogeneity of the sample. Then the likelihood ratio statistic for testing the composite hypothesis H_v $(1 \leq v \leq r)$ against H_0 is:

$$\max_{0 \leq k \leq r} (S_r - S_k) = S_r - \min_{0 \leq k \leq r} S_k,$$

where

$$S_0 = 0, \; S_k = \sum_{j=1}^{k} \log \frac{f_1(x_j)}{f_0(x_j)}.$$

There is a nonparametric version of the CUSUM statistic:

$$y_r = (y_{r-1} + x_r)^+, \; y_0 = 0,$$

and the corresponding decision rule is

$$d_N(\cdot) = d(y_r) = I(y_r > N),$$

where $I(\cdot)$ is the indicator function and N is the threshold. The function d_N represents the decision at time r, which gives a value of 1 to indicate an attack and 0 to indicate a normal condition.

In general $E(X_r) = c$. We choose a parameter a that is the upper bound of c, i.e., $a > c$. Then we define $x_r = X_r - a$ so that it has a negative value under normal operation. When an attack takes place, the rate of increase will suddenly become larger and hence the value $x_r = X_r - a$ will become positive.

The two parameters a, the upper bound in case of normal operation, and N, the flooding threshold, influence the performance of sequential detection. We should try to strike a good balance by minimizing the detection delay subject to a certain false alarm tolerance level.

4.2 Sequence Model

Since the k rows in T_d are independent of each other, for clarity, we choose one row for our discussions here. There are n counters in each row of table T_d and these counters are denoted as $C_1, C_2, ..., C_n$. The hash table is refreshed periodically. The value of counter C_i $(1 \leq i \leq n)$ forms a sequence $\{C_i^t\}$ according to the refresh period t. For each counter, we analyze the change-point characteristics of the sequence $\{C_i^t\}$.

Normal TCP traffic has symmetric SYN and ACK pairs and hence the counter values in T_d should be close to 0. When spoofed TCP handshake packets are sent to a victim host, there will be more SYN packets than ACK packets. The corresponding counter value in T_d will grow rapidly. The density function of sequence $\{C_i^t\}$ will change and CUSUM will detect this change. We define $x_i^t = C_i^t - a\,(a > 0)$ as the sequence of time t for the CUSUM method. The two parameters are set according to the network conditions. Thus x_i^t has a negative value during normal operation. When a spoofed flooding attack occurs, it will change to a positive value.

When a spoofed flooding attack occurs, each row is expected to have a counter with an abnormally high value. These abnormal counters are referred to as *heavy counters*. If a packet is hashed into T_d and hits all the k heavy counters in all k rows, this packet is regarded as a suspicious spoofed packet and hence an alarm will be launched.

4.3 Spoofing Type Classification and Response

Since random spoofing generates a wide range of IP addresses for the source IPs of the packets, the probability that two packets have the same source IP is very low. On the other hand, subnet spoofing has a much narrower range than random spoofing. During a spoofed flooding attack, the attacking source typically generates a large number of packets. This number is much larger than the number of candidate IP addresses used for subnet spoofing. For example, in order to bring down a victim server for 10 minutes, there should be at least 300,000 SYN packets [12]. However, a class C subnet only has 254 IP addresses which are available for a subnet spoofing attack. Therefore, quite a number of subnet spoofed packets are expected to have identical source IPs during the attack period.

We define two thresholds, θ_1 and θ_2 $(1 < \theta_1 < \theta_2)$, for T_s. When a packet hits k heavy counters in T_d, its source IP is checked in T_s. If all counters in T_s hit by this packet have values larger than θ_1 but smaller than θ_2, it is regarded as subnet spoofing. If the value is much larger than θ_2, it may be caused by fixed spoofing. Otherwise it is regarded as random spoofing.

Random spoofing may be throttled by ingress filters deployed at the edge routers. However, there has been a lack of efficient methods for fighting against subnet spoofing. To defend against attacks caused by fixed spoofing and subnet spoofing, we propose here a soft rate-limiting scheme. Specifically, the percentage of traffic to go through is equal to

$$R_{\texttt{pass}} = \gamma^{-\varepsilon C_s},$$

where C_s is value of a counter in T_s and γ, ε are parameters. The legitimate value of C_s is 0. Thus if C_s is close to 0, we essentially allow almost all traffic to pass through. According to the classification scheme above, a higher value of C_s means that the traffic has a higher probability of being fixed spoofed packets. The larger the value of C_s is, the more traffic will be blocked.

5 Performance Evaluation

We have carried out some simulation experiments to evaluate the performance of our proposed method. The DARPA off-line intrusion detection benchmark [6] is used for the experiments. We use the NS2 network simulator [1] to simulate the three IP spoofing types.

5.1 Attack Traffic Detection

Two datasets from the DARPA benchmark are used to evaluate the detection performance of our method. Specifically, one dataset contains SYN flood attack packets and the other dataset is free of flooding attack. In all the experiments, the number of rows k in each hash table is set to 4 and the number of counters n in each row is set to 1024. For attack detection, we only need to monitor the destination table T_d. As before, since each row in T_d corresponds to an independent hash function, we only discuss any one row here as other rows are the same.

We first observe the status of T_d during normal operation. Normal traffic begins at 21:00 on one day and ends at 18:32 on the following day. There are a total of 27877 TCP connections during this period. We observe that most of the counters have a zero value and only two counters (out of 1024 counters in the row) have nonzero values. The nonzero values are triggered by

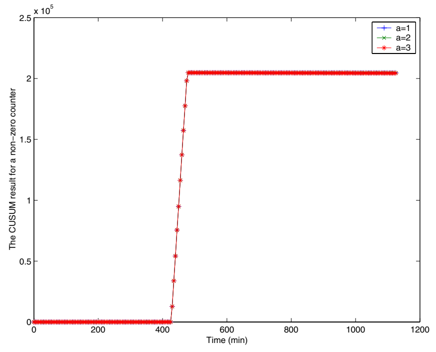


Figure 2. CUSUM result for a nonzero counter during a SYN flooding attack.

incomplete handshakes caused by occasional network errors.

In another experiment, the dataset with SYN flood attack packets is used. During the attack, there are many more SYN packets than ACK packets. The counters corresponding to the hashed values of the victim IP addresses grow dramatically and hence become heavy counters. While most of the counters still have a zero value, there are a few heavy counters with values significantly higher than those of the nonzero counters during normal operation as observed above. The CUSUM detection method can successfully detect these heavy counters. Figure 2 shows the CUSUM result for a nonzero counter when an attack is launched. Since there is very little traffic around midnight, the result for this period is not shown for clarity.

5.2 Spoofing Type Classification

We use NS2 to simulate the random spoofing, subnet spoofing and fixed spoofing attack scenarios. In the simulations, there are 10 server nodes and 1000 client nodes. One of the server nodes is selected as the victim. The client nodes randomly select one or more server nodes as destinations for establishing connections. Regardless of what spoofing type is used, we observe that the flooding traffic can trigger one counter in each row of table T_d to have an abnormally large value as above.

To classify the spoofing into one of the three types, we have to monitor the counter values in table T_s. Figure 3(a) shows the values of the counters in one row of T_s under random spoofing. The counters have relatively small values which are distributed somewhat uniformly across different counters because the spoofed IPs are generated randomly. For subnet spoofing as shown in Figure 3(b), since a much narrower range of IP addresses is used, we can see that only a portion of the counters have nonzero values. Moreover, these counter values are generally higher than those in Figure 3(a). If we examine more closely the two tables in Figure 3(a) and Figure 3(b), we can see that many different IP addresses are hashed into the same counter in Figure 3(a) due to hash table collisions, but a counter in Figure 3(b) almost always corresponds to the same IP address. Figure 3(c) shows the counter values under fixed spoofing. This type of spoofing is much easier to identify since only a few counters have nonzero values and they are heavy counters with extremely high values.

6 Related Work

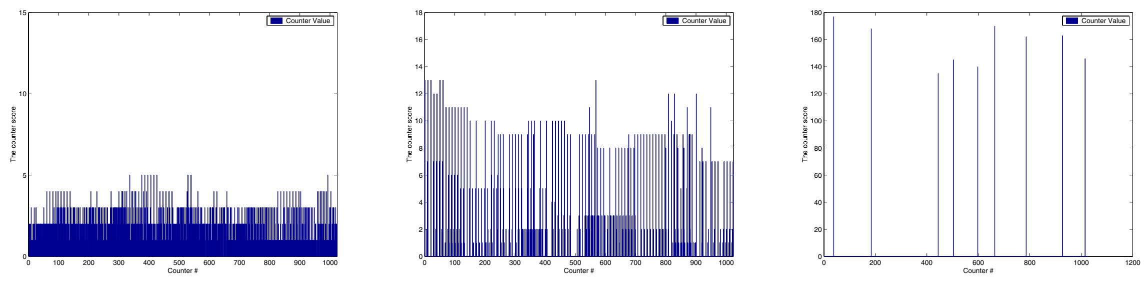
A hash table is a high-performance data structure that can be used for efficient table lookup. Recently, it has been applied to some applications for network packet processing. Snoeren [11] presented a technique based on hash table for IP traceback, which generates audit trails for the network traffic so that the origin of an IP packet delivered by the network in the recent past can be traced. Hash table has also been employed to look for imbalance between the incoming and outgoing traffic flows to or from an IP address [2]. More recently, a router equipped with DDoS protection capability called IDR [5] was proposed to detect DDoS attacks using a Bloom filter.

Change-point detection methods have been applied to DDoS detection due to their simplicity and effectiveness. Wang et al. [12, 13] proposed a method for detecting SYN flood attacks at leaf routers that connect end hosts to the Internet. Based on the observation that SYN and FIN packets form pairs in normal network traffic, they proposed using a nonparametric CUSUM method to accumulate the pairs. Luo and Chang [7] proposed a two-stage scheme to detect the so-called pulsing DoS attacks. The first stage uses wavelet transform to extract the desired frequency components of the traffic data and the second stage tries to detect change points in the extracted components.

7 Conclusion

Although IP spoofing is not an attack in itself, it is commonly used with real TCP-based attacks by exploiting the characteristics in the design of the TCP/IP suite.

To defend against spoofed flooding attacks, we propose in this paper an efficient method that can detect all three types of spoofing: random, subnet and fixed spoofing. Based on the Bloom filter, we propose



(a) Random spoofing (b) Subnet spoofing (c) Fixed spoofing

Figure 3. Counter values in one row of table T_s under different spoofing types.

a storage-efficient data structure which only requires a fixed-length table for recording relevant traffic information. A change-point detection method, CUSUM, is then applied to detect abrupt changes in the traffic characteristics which correspond to the occurrence of flooding attacks. When malicious events are detected, they can further be classified into random spoofing, subnet spoofing or fixed spoofing type by analyzing a hash table for the source IP characteristics. Simulation experiments show that our proposed method yields very accurate detection and classification results yet with low computational demand.

There are some parameters in our method. Currently these parameters are set manually based on experience. A future extension is to devise an automated scheme for setting or adapting the parameters. Another interesting direction to pursue is to design adaptive hash functions that maximize the utilization of the hash table entries and hence minimize the false positive rate. Moreover, we plan to evaluate our method in a reasonably large real network.

Acknowledgment

The research work presented in this paper has been supported by a research grant from the Research Grants Council (RGC) of the Hong Kong Special Administrative Region, China under the Area of Excellence (AoE) scheme (Project No. AoE/E-01/99).

References

[1] Network simulator NS2.

[2] 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.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[4] B. Brodsky. *Nonparametric Methods in Change-Point Problems.* Kluwer Academic Publishers, The Netherlands, 1993.

[5] 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.

[6] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. W. andR. Cunninghan, and M. Zissman. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *the 2000 DARPA Information Survivability Conference and Exposition*, January 2000.

[7] X. Luo and R. K. C. Chang. On a new class of pulsing denial-of-service attacks and the defense. In *Network and Distributed System Security Symposium 2005(NDSS2005),San Diego, California*, February 2005.

[8] J. Mirkovic and P. Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communications Review*, 34(2):39–54, April 2004.

[9] J. Postel. Transmission Control Protocol : DARPA internet program protocol specification,RFC 793, September 1981.

[10] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on TCP. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 208–223. IEEE Computer Society, May 1997.

[11] A. C. Snoeren. Hash-based IP traceback. In *Proceedings of the ACM SIGCOMM Conference*, pages 3–14. ACM Press, August 2001.

[12] H. Wang, D. Zhang, and K. G. Shin. Detecting SYN flooding attacks. In *Proceedings of Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM)*, volume 3, pages 1530–1539, June 23-27 2002.

[13] H. Wang, D. Zhang, and K. G. Shin. Change-point monitoring for the detection of dos attack. *IEEE Transactions on Dependable and Secure Computing*, 1(4):193–208, October-December 2004.