J Supercomput (2006) 36:235–248
DOI 10.1007/s11227-006-8295-0

A novel approach to detecting DDoS attacks at an early stage*

Bin Xiao · Wei Chen · Yanxiang He

Abstract Distributed Denial-of-Service (DDoS) attacks pose a serious threat to Internet security. Most current research focuses on detection and prevention methods on the victim server or source side. To date, there has been no work on defenses using valuable information from the innocent client whose IP has been used in attacking packets. In this paper, we propose a novel cooperative system for producing warning of a DDoS attack. The system consists of a client detector and a server detector. The client detector is placed on the innocent client side and uses a Bloom filter-based detection scheme to generate accurate detection results yet consumes minimal storage and computational resources. The server detector can actively assist the warning process by sending requests to innocent hosts. Simulation results show that the cooperative technique presented in this paper can yield accurate DDoS alarms at an early stage. We theoretically show the false alarm probability of the detection scheme, which is insensitive to false alarms when using specially designed evaluation functions.

Keywords DDoS Attack · Cooperative Detection · Bloom Filter

1. Introduction

Distributed Denial-of-Service (DDoS) attacks are a serious threat to the Internet, particularly to Internet commerce. DDoS attacks exhaust the resources of the victim server and prevent the victim server from providing service to legitimate users. They do this by sending a large

*This work is partially supported by HK Polyu ICRG A-PF86 and CERG Polyu 5196/04E, and by the National Natural Science Foundation of China under Grant No. 90104005.

B. Xiao (✉)
Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
e-mail: csbxiao@comp.polyu.edu.hk

W. Chen · Y. He
Computer School, The State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, Hubei, China
e-mail: {chenwei, yxhe}@whu.edu.cn

number of malicious packets toward the victim server. The majority of attack packets are delivered using the TCP protocol [1]. Current TCP-based DDoS attacks most commonly exploit the TCP three-way handshake [2] and use SYN flooding [3, 4], sending numerous SYN request packets towards a victim server. This leads to many half-open connections on the victim server, which quickly depletes server resources and prevents the server from accepting legitimate user requests.

Current mechanisms for detecting and preventing TCP-based DDoS attacks can be classified into three categories according to where the defense is deployed at the source-end, at the victim-end, or in the intermediate network. Each mechanism has its limitations. Detection on the victim side of a server (by capturing abnormal traffic signals) has two drawbacks. First, it is not easy to detect abnormal deviations until the DDoS attack is at the final traffic-bursting stage. Consequently, victim-side mechanisms do not have the capacity to provide an effective early warning. Second, the aggregation of numerous malicious packets on the victim server makes it in any case difficult to launch an effective post-detection response. An approach that provides an early DDoS alarm either near the source or at the intermediate-network is also limited in that at these points it is difficult to capture the attack signature.

In this paper we propose a novel detection mechanism that makes use of valuable information obtained at the innocent host whose IP is utilized as the spoofed IP and which receives abnormal TCP control packets during the three-way handshake. Making use of the innocent host offers two distinct advantages. First, while attackers usually try to detect and deceive defenses deployed around the victim and the attacking source, it is difficult for attackers to be aware of the existence of defensive mechanisms operating on the innocent host. Second, while a defense at the victim server requires the monitoring of numerous attacking packets, leading to congestion and even making the defense system itself vulnerable to DDoS attacks, a defensive mechanism operating at the innocent host itself faces little risk of DDoS attacks.

The detection scheme proposed in this paper provides accurate alarms but, crucially, requires minimal storage space and imposes low computational overheads. This is crucial because the efficacy of the system depends on the number of participant Internet Service Providers (ISPs). As a result, it must use limited ISP resources and not significantly degrade an ISP's other services. This is not an easy result to achieve, given that accurate detection on the innocent host normally incurs higher storage costs. This is because the backscattered distribution of abnormal signatures [1] means that more packets have to be monitored, recorded and analyzed in order to capture a small quantity of attack signatures. The proposed modified Bloom filter-based detection scheme deals with this problem by adopting a space-efficient data structure so that all abnormal traffic can be monitored while nonetheless occupying a small storage space.

This paper makes three main contributions to the early-stage detection of a TCP-based DDoS attack. First, we introduce a novel cooperative detection system that is composed of a client detector and a server detector. Cooperation improves the alarm accuracy and shortens the detection time. To our best knowledge, this is the first detection system to be proposed based on detection on the innocent host side. Second, we present a modified Bloom-filter-based detection scheme that monitors abnormal handshakes. The proposed scheme has a space-efficient data structure that involves simple hash function operations and incurs a small computational overhead. Third, we theoretically show the probability of false alarms when using the modified Bloom-filter architecture to detect DDoS attacks. The probability is analyzed in terms of both false negatives and false positives.

The remainder of the paper is organized as follows. Section 2 will introduce some related work in the spoofed DDoS detection. Section 3 discusses and analyzes TCP-based DDoS attacks. Section 4 describes the cooperative detection technique. Section 4.1 describes the client detector mechanism and Section 4.2 describes the server detector mechanism. Section 5 offers a theoretical analysis of the possible false alarms. Section 6 describes experiments evaluating the proposed technique. Section 7 offers our conclusions and outlines future work.

2. The related work

Most current DDoS attack detection and prevention schemes are deployed either at the victim server, at the attack source side, or between the two. In the following, we describe schemes representative of each of these three deployments and describe associated problems.

Victim server side detection of DDoS attacks has received the bulk of past research attention, doubtless because the main goal of researchers has been to protect the victim server. Wang et al. [4], detected SYN flooding attacks at leaf routers that connect end hosts to the Internet. They observed that the SYN-FIN packets pair each other in the normal network traffic and proposed a non-parameter CUSUM method to accumulate these pairs. Cheng [5] utilized the TTL (Time-To-Live) value in the IP header to estimate the Hop-Count of each packet. The spoofed packets could be distinguished from normal ones by the Hop-Count deviation. Lemon [6] incorporated SYN cache and cookies to prevent DDoS attacks, using cache or cookies to evaluate the security status of a connection before establishing the real connection with a protected server. Hussain et al. [6] proposed a framework for classifying DoS attacks based on the header content and the transient ramp-up behavior. Keromytis et al. employed the secure overlay service (SOS) [7, 8] to proactively prevent DDoS. SOS architecture is composed of SOAP, overlay nodes, beacon, secret servlet and filtered region, which makes it difficult for an attacker to target nodes along the path to a specific SOS-protected destination. Based on SOS, researchers from Columbia University continued their proactive defense research. MOVE [9] and WebSOS [10] are modified forms of the SOS architecture but with different emphasis. Puzzle based methods [11, 12] impose heavily overhead to zombies, which can mitigate attacking rate and make zombies exposed to host owners. Each of these must minimize resource usage while promptly responding and recording the states of numerous connections. At the same time, the method itself must be immune to DDoS attacks.

Source side mechanism for detecting and preventing of DDoS attacks can be difficult to deploy. Source-end deployed methods have some advantages but are difficult to deploy. For reasons related to performance, however, ISPs are disinclined to deploy source-end defenses in their domains. Mirkovic and Prier [13] introduced a DDoS defense system at the source-end in which attacks were detected by constantly monitoring two-way traffic flows and comparing them with normal flow models. The RFC2827 [14], for example, is designed to filter out spoofed packets with spoofed IP addresses at each ingress router and can drop a suspicious packet that does not belong to its routing domain. However, the fact that it may degrade routing performance makes ISPs reluctant to participate in this defense system.

After an attack is detected, it is possible to find the attacking source using traceback [15–18] and pushback [19] techniques. Traceback attempts to identify the real location of the attacker. Source IPs used during a DDoS attack are often forged and cannot be used to identify the real location of the attack source. Most traceback schemes respond to this by either marking some packets along their routing paths or by sending special packets [18]. By tracking these special marks, it is possible to reconstruct the real routing path reconstructed and locate the true source IP. After the real path of the spoofed packets has been identified, the pushback technique can perform advanced filtering and work at the last few routers before the malicious traffic reaches the target victim.

3. The TCP-based DDoS attack

Most DDoS attacks exploit TCP control packets by spoofing the three-way handshake between the source and the destination server. In this section we analyze the behavior of TCP control packets first in a normal three-way handshake and then in a spoofed three-way handshake.

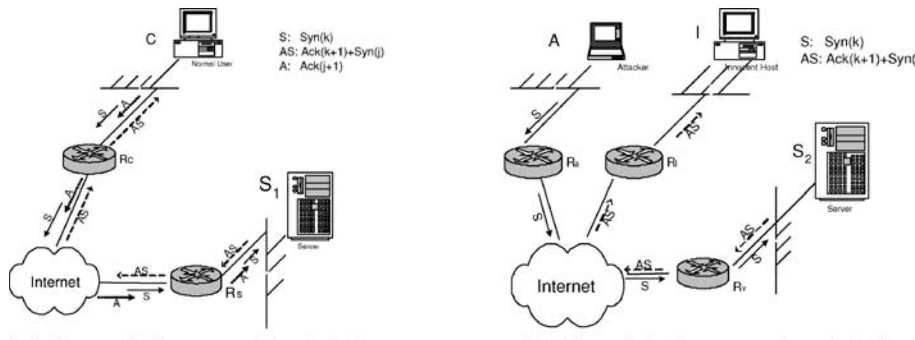
Figure 1(a) shows a normal three-way handshake. First client C sends a $Syn(k)$ request to the server S_1, which replies with a packet containing both the acknowledgement $Ack(k+1)$ and the synchronization request $Syn(j)$ and waits with a half-open connection in its memory space for the acknowledgement from the client C. Upon receiving both $Ack(k+1)$ and $Syn(j)$ client C will finish building the connection by sending $Ack(j+1)$. When server S_1 gets $Ack(j+1)$, it removes previously stored half-open connections in its memory space. The released memory space on server S1 makes it possible to handle further connection requests from clients and a network can run smoothely. k and j are respectively sequence numbers produced randomly by the server and the client during the three-way handshake.

In the remainder of this paper, SYN means a request sent to a server S inside the TCP control packet during the first round of the three-way handshake protocol; ACK/SYN will indicate a packet containing both $Ack(k+1)$ and $Syn(j)$ that is delivered back from the server S in the second round; and ACK will denote a control package representing $Ack(j+1)$ in the third round. During the normal three-way handshake procedure, SYN, ACK/SYN and ACK all appear at both the edge router R_c near the client and at the edge router R_s near the server, as shown in Fig. 1.

Figure 1(b) shows a spoofed three-way handshake and the implementation of a DoS attack. The packet at the first round of a valid authentication process is a malicious one with a spoofed IP address. The edge router R_a in the attacker domain forwards the SYN packet with the spoofed address P_I, the IP address of the innocent host I, to the server S_2. The server S_2 replies with an ACK/SYN packet and a half-open connection is pending. This ACK/SYN will be sent to the innocent host I because the server S_2 regards the SYN packet from I according to the spoofed source IP P_I. The edge router R_I on the innocent host side will receive the ACK/SYN packet but as no previous SYN request had been forwarded by the client detector at R_I, the ACK/SYN packet is dropped. The pending half-open connection on the server S_2 is maintained for a long time. More accumulated half-open connections will quickly consume all the memory space reserved for handling TCP requests and the server S_2 will deny any new requests. It is difficult to trace back the attackers true address because the innocent host I, whose IP is used as the spoofed source IP, is usually not in the same domain as the attacker, sender A.

4. The cooperative detection technique

In this section we present a cooperative technique for detecting a DDoS attack at an early stage. The technique makes use of a client detector and a server detector. The client detector, deployed at the edge router of innocent hosts, checks the TCP control packets flowing through the edge router. When it captures suspicious events, it notifies the protected server of a potential DDoS attack. The server detector, employed by the protected server, detects attacks not only by passively listening for warnings from the client detector, but also by actively sending queries to the client detector to confirm alarms.



(a) Normal three-way handshake (b) Spoofed three-way handshake

Fig. 1 The process of the TCP three-way handshake

4.1. The client detector

In this subsection, we describe the client detector and the detection scheme. The client detector is deployed at the edge router on the innocent host side. In Fig. 1(b), for example, the client detector can be installed on the router R_I. One of the main tasks of the client detector is to monitor the TCP control packets entering and leaving a domain. The detection scheme is developed from a modified hash table. We have designed the new hash table based on the Bloom filter method. States of each TCP three-way handshake are recorded in the hash table and the abnormal asymmetric three-way handshake can be clearly seen inside the client detector. The client detector thus can issue a DDoS attack after analyzing suspicious alarms.

The Bloom filter based hash table. The Bloom filter is a kind of space-efficient hash data structure. We propose using a modified Bloom filter in order to construct a hash table that can record three-way TCP control packets at a limited storage cost. The modified structure of the novel hash table makes it possible to capture abnormal handshakes even where the volume of traffic is large.

The original Bloom filter. The Bloom filter is first described by Burton Bloom [20] and originally used to reduce the disk access times to different files and other applications, e.g., spell checkers. Now it has been extended to defend against DDoS attacks [17, 21, 22]. The Bloom filter is composed of a vector 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 $\{0, \ldots, m-1\}$. The vector v can show the existence of an element in A. Given an element $a \in A$, the bits at positions $h_1(a), h_2(a), \ldots, h_k(a)$ in v are set to 1 (Fig. 2). Note that a particular bit might be set to 1 multiple times which may cause potential false results. Given a query of the existence of b in A, we check the bits at positions $h_1(b), h_2(b), \ldots, h_k(b)$. If any one of them is 0, then certainly b is not in the set A. Otherwise we conjecture that b is in it. Otherwise we presume that b is a member of that set. There is, however, a certain probability that the Bloom filter will give a false result, a "false positive". The parameters k and m should be chosen such that the probability of a false positive is small.

Modified Bloom filter. The original Bloom filter is not able to handle the requirement that our client side detection scheme counts pairs of SYN and ACK/SYN control packets. We have therefore modified the original Bloom filter by substituting m bits with countable integers as

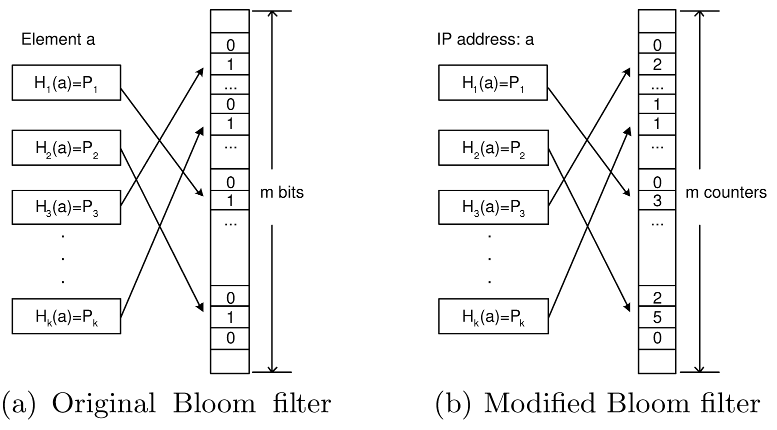


Fig. 2 Bloom filter uses independent hash functions to map input into corresponding positions

shown in Fig. 2(b). The counter can represent how many times the corresponding positions are hit.

The modified Bloom filter is composed of a table in which there are m counters from address 0 to $m - 1$. All counters are initialized to 0. When a key a (such as an IP address) is inserted or deleted, the value of the counters are increased or decreased by 1 accordingly at the table addresses $h_1(a), h_2(a), \ldots, h_k(a)$. A counter is turned on when its value changes from 0 to 1 and turned off from 1 to 0. If an IP address b is stored in the modified Bloom filter, the counters at addresses $h_1(b), h_2(b), \ldots, h_k(b)$ in the table are all non-zero. This allows us to monitor the current statistic of TCP control packets flowing through an edge router towards the innocent client host.

The Bloom filter based detection scheme. The detection scheme uses a modified Bloom filter. Inside the Bloom filter, a hash table records the destination IPs (the IPs of the protected servers). The operations to the hash table are triggered by outgoing and incoming traffic flows related to those IPs no matter the traffic contains attack packets or not. When the detection scheme observes a single ACK/SYN control packet, it can alert a Suspicious Alarm (SA) to the client detector. After the accumulation of SAs to a threshold score, a DDoS warning will be sent to the protected server.

The modified Bloom filter hash table takes actions when the client detector observes a SYN packet in the outgoing traffic or an ACK/SYN packet in the incoming traffic. When a SYN packet, the TCP control packet in the first round of handshake, is captured from the outgoing traffic, the destination IP (the server's IP) is hashed into the hash table using k independent hash functions. The output of each hash function denotes the address of a counter to be modified in the hash table. If the counter is 0, it is turned on to 1. If the counter is already turned on, its value is increased by 1. When an ACK/SYN packet, the TCP control packet in the second round of handshake, is captured from the incoming traffic, the source IP (the server's IP) is hashed into the hash table too. However, the counter is decreased by 1 in the hash table with address as the output of k hash functions. When a counter changes from 1 to 0, it is turned off. The actions upon the hash table are depicted in Fig. 3. The table can be reset to 0 to restart a monitoring process.

The normal and the abnormal TCP handshake control packet can trigger different results in the modified Bloom filter table. This difference provides the basis of our proposed detection

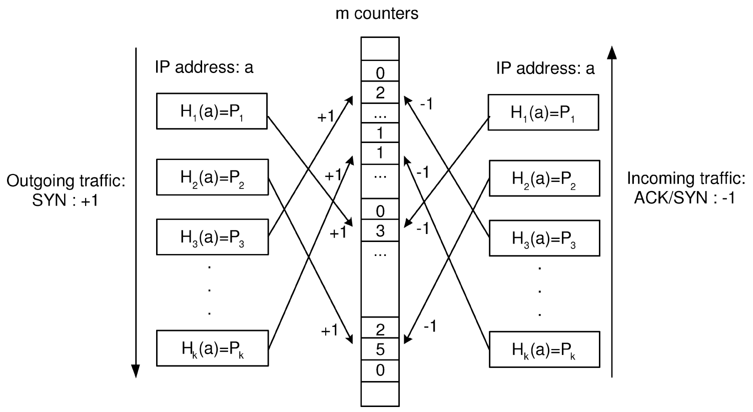


Fig. 3 The detection scheme increases or decreases the value of a counter according to the three-way handshake packet

scheme at the client detector. In a normal traffic flow, an ACK/SYN packet should hit all turn-on counters because those counters have already been turned on by a previous SYN packet. The value of a counter will remain the same if the paired SYN and ACK/SYN packets are completely captured at the client detector. In contrast, in the scenario of the spoofed handshake, a suspicious ACK/SYN packet could meet a turn-off counter since no previous paired SYN packet has turned it on (initially reset to 0). Among all k counters that are denoted from an abnormal ACK/SYN packet, it is possible that one of them is turned on because other SYN packets with different destination IP addresses could increase its value. However, the probability that all k counters are turned on at the same moment is rather low. If an ACK/SYN packet hits a turn-off counter, our detection scheme at the client detector will report a Suspicious Alarm (SA), as described in Fig. 4, and the details of this packet are recorded for further analysis. The detection scheme only requires addition and subtraction operations and is easily implemented.

When a new Suspicious Alarm (SA) is reported, the client detector will analyze the source IP distribution of SAs in its database. During a DDoS attack, the client detector will find asymmetric ACK/SYN packets sent from a victim server. We denote P_{victim} as the IP of an ACK/SYN packet, which is also the IP of a victim server. When SAs are reported from packets with the same source IP (P_{victim}) in a short period, there probably exists a DDoS attack targeting the host P_{victim}. However, given that each SA comes from a different source IP, it is most likely they are triggered by other reasons other than a DDoS attack. To evaluate the distribution of the source IP of SAs, a score is calculated as follows:

$$\text{score} = \sum_{s \in \text{IP List}} (|X_s| - 1)^2$$

Where X_s stands for a subset of the IP list that contains all IPs from reported SAs. All elements in X_s have the same IP value s in a certain period. The score will increase dramatically when the number of SAs containing the same source IP increases. On the other hand if each of the SAs has a different source IP, the score will be 0. This score value can be an indicator to launch a warning of DDoS attacks. To save computation when a new SA comes, we use the

Fig. 4 The bloom filter based detection scheme to report a SA at the client detector

```
Client_Detection_Scheme (INPUT: P) {
if P is a $SYN$ packet then
    for $i = 0; i < k; i++$ do
        j=$Hash_i$(P)
        $Counter_j$++
    end for
else if P is a $ACK/SYN$ packet then
    for $i = 0; i < k; i++$ do
        j=$Hash_i$(P)
        //Check whether exists turn-off counter
        if $Counter_j == 0$ then
            Report Suspicious Alarm(SA)
            RETURN
        end if
    end for
    //If there is no turn-off counter, do subtraction
    for $i = 0; i < k; i++$ do
        j=$Hash_i$(P)
        $Counter_j$- -
    end for
end if
RETURN }
```

following expression to calculate the score value:

$$\text{score}_{\text{current}} = \begin{cases} \text{score}_{\text{previous}} & s \text{ not in the history IP list} \\ \text{score}_{\text{previous}} + 2 \times |X_s| - 1 & s \text{ in the history IP list} \end{cases}$$

The equation demonstrates that upon the arrival of a SA whose source IP is not in the history IP list, the score will remain unchanged while if it is in the history lists we have a new increased score. Because the score is the sum of $(|X_s| - 1)^2$, the new score is equal to the previous score adding the current $(|X_s| - 1)^2$ and minus the previous $(|X_s| - 1 - 1)^2$, i.e., $\text{score}_{\text{previous}} + 2 \times |X_s| - 1$. When the score exceeds a predefined threshold, the reported SAs with the IP of P_{victim} will be sent to the server detector at P_{victim} address. On the other hand, whenever a query is received from a server detector, the number of SAs with the IP of the server in the database of the client detector would be sent back. The server detection scheme will be introduced in the following subsection.

4.2. The server detector

The server detector is deployed at the protected server, such as S_2 in Fig. 1(b). With the assistance of client detectors, a server detector can detect a forthcoming DDoS attack at an early stage. Figure 5 illustrates the two parts of the server detection scheme. Both parts operate independently and concurrently and can issue a confirmed DDoS attack alarm. Since this confirmation has no negative effect on the protected server, the server can perform a query as soon as any suspicious half-open connections are observed. This is a distinct advantage over many other DDoS detection methods, which must wait to capture sufficient DDoS attack evidence before taking further action, a requirement which delays DoS attack detection and prevention. In our approach, cooperation between the client and server detectors ensures that the server detector launches DDoS alarms at a very early stage.

Fig. 5 Detection scheme at the server detector

```
Server_Detection_Scheme{
if A $SA$ comes then
   Store $SA$ in the server detector DB
   Sum the number $N$ of $SA$
   if $N >$ Threshold $T$ then
      Report confirmed DDoS attack alarm
   end if
end if
if Finding suspicious half-connections on the server then
   Send queries to client detectors
   Waiting for reply
   Sum the number $N$ of SAs replied from client detectors
   if $N >$Threshold $T$ then
      Report confirmed DDoS attack alarm
   end if
end if
}
```

The server detection scheme is composed of two parts. Part one shows that the server detector may passively wait for the potential direct DDoS alarm from client detectors. When enough potential DDoS attack alarms arrive, it will send the server a confirmed direct DDoS attack alarm. Part two indicates that the server detector also can perform more active detection by sending queries to client detectors when there are too many half-open connections observed. It is possible, however, that the source IPs of the spoofed packets are widely distributed with the result that the number of SAs at a client detector could be insufficient to provoke the sending of an SA to the server detector. In this scenario, the server detector will select several cooperative client detectors to query about the number of SAs. The selection of client detectors depends on the source IPs of current half-open connections reserved by the server. A query is first sent to a client detector that is in a routing domain containing the most pending connections. After receiving replies, the server detector can tell whether an investigated half-connection is caused by a spoofed DDoS packet and an alarm should be issued or whether it is caused by something else and no action is required.

5. False alarm analysis

An ideal detection method is expected to raise an alarm if and only if an attack occurs. Unfortunately, existing solutions can fail to raise accurate alarms in two different ways: by issuing false negatives and false positives. A false negative occurs when an attack event is either not detected by the IDS (Intrusion Detection System) or is incorrectly declared benign. A false positive occurs when a benign event is declared as an attack.

Effective detection requires the maintenance of low false negative and false positive rates. In this section, we analyze the probabilities of false negative and false positive of our client detection scheme for classifying suspicious events using a modified Bloom-filter table. The results show that the scheme has only a small probability of producing false negatives and false positives.

5.1. False negative

A false negative occurs when a detection method fails to count an attack packet. In our method, a false negative is said to have occurred if the client detector fails to launch an SA upon the arrival of a suspicious packet. This would occur if a malicious ACK/SYN packet did not hit any k non-zero counters in the modified Bloom-filter table and thus was regarded as legitimate. Thus, the probability of a false negative is equal to the probability of an ACK/SYN packet hitting all non-zero counters denoted by all k hash functions. First, we analyze the probability (denoted as $\mathrm{P}_{\mathrm{zero}}$) that a counter is still 0 after inserting n keys (i.e., n different IP addresses of SYN packets) into the counter table of size m. The probability analysis of a false negative in our method is similar to that of a false positive analysis in the original Bloom filter [20], which tries to find the certain probability that an element hits non-zero counters. Though we use the modified Bloom filter, which use counters to substitute for a bit array, the analytical result is the same since only non-zero counters are involved. Note that each key will be hashed into k counters in the table. The probability $\mathrm{P}_{\mathrm{zero}}$ is:

$$\mathrm{P}_{\mathrm{zero}} = \left(1 - \frac{1}{m}\right)^{kn}$$

The probability of a false negative is the probability of all k hash functions hitting non-zero counters simultaneously. Thus the probability of a false negative $P_{negative}$ is:

$$P_{\mathrm{negative}} = (1 - \mathrm{P}_{\mathrm{zero}})^k = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k$$

The above result is approximately equal to:

$$P_{\mathrm{negative}} \approx \left(1 - e^{-\frac{kn}{m}}\right)^k$$

5.2. False positive

A false positive occurs when a detection method wrongly identifies a normal packet as an attack packet. In a normal network traffic environment without any network errors, our detection scheme does not generate any false positives because SYN and ACK/SYN packets appear in pairs. Our scheme decreases k counters by 1 in the modified Bloom-filter table when receiving an ACK/SYN packet. Those k counters should be non-zero counters because they have been increased by 1 when receiving previous paired SYN packet and there is no possibility of hitting a zero counter. Thus the risk of our method declaring a false positive in an attack-free environment is zero.

False positives can however be triggered by a false negative during a DDoS attack. When a false negative happens, a malicious ACK/SYN packet is treated as a legitimate one. Consequently, there are k counters in the table that have decreased by 1 and some of these may as a result have been turned off. We refer to these counters as contaminative counters. When a normal legitimate ACK/SYN packet hits one of these contaminative counters, a counter which has been turned off can issue a SA, which creates a false positive.

A false negative can contaminate k counters. These counters could potentially trigger false positives. In other words, the number of false positives arising from one false negative can be between 1 and k. Assume that the distribution for the number of false positives due to a false negative is a uniform distribution. Then the probability of false positive is $\frac{k}{2}$ times of the probability of false negative. Hence, the probability of the false positive in our client detection scheme is:

$$P_{\text{positive}} = \frac{k}{2} \times P_{\text{negative}} = \frac{k\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k}{2}$$

This can be approximately represented as:

$$P_{\text{positive}} \approx \frac{k\left(1 - e^{-\frac{kn}{m}}\right)^k}{2}$$

The occurrence of false positive and false negative is acceptable when m and k are set appropriately. For instance, if the length of the modified Bloom-filter table is 8 times of the number of input entries, i.e., $\frac{m}{n} = 8$, and there are $k = 4$ hash functions, the probability of a false negative is 2.4% and the probability of false positive is 4.8%. The probability of false negative and false positive drops quickly when more memory is allocated and more hash functions are used. Furthermore, the collateral damage caused by false positives can also be mitigated by adopting the SA evaluation function in Section 4.1.

6. Experiment

This section describes two experiments. The first tests the ability of proposed cooperative detection technique to detect DDoS attacks. The second tests the ability of the proposed detection scheme to tolerate false negatives and false positives in a real network environment.

6.1. Evaluation of the detection technique

Ten zombies were created to simulate SYN flooding attacks on a server. The attack packet rate ranged between 10 and 1000 packets/sec. A maximum rate of 1000 packets/sec was set because this is enough to shut down a server's services [23]. There were two different attack scenarios: in one the maximum attack rate was reached in 100 seconds and in the other the maximum was reached in 600 seconds. The simulation was designed so that only 1% of ACK/SYN packets sent from the victim server arrived at a client detector near the innocent host. These packets would trigger the detectors' generation of SAs.

Figure 6 shows the number of SAs generated by the client detectors in the two scenarios. Despite the fact that the experimental setup allowed the client detector to receive only 1% of the spoofed attack packets, it still produced accurate SAs in quantities sufficient for the detector to issue a DDoS alarm at the early stages of the DDoS attack. More impressively, as Fig. 6(b) shows, even when the DDoS attacker increased the number of attack packets slowly, the detection scheme produced accurate warnings, eight SAs in the first 100 seconds. This is enough to alert the protected server to the forthcoming DDoS attack. Note that in the Fig. 6(a) the number of SAs rises steadily because the DDoS attack takes place over a short period of time. In contrast, Fig. 6(b) shows fluctuation in the number of SAs, a result of the slower rate of increase in the number of attacking packets.

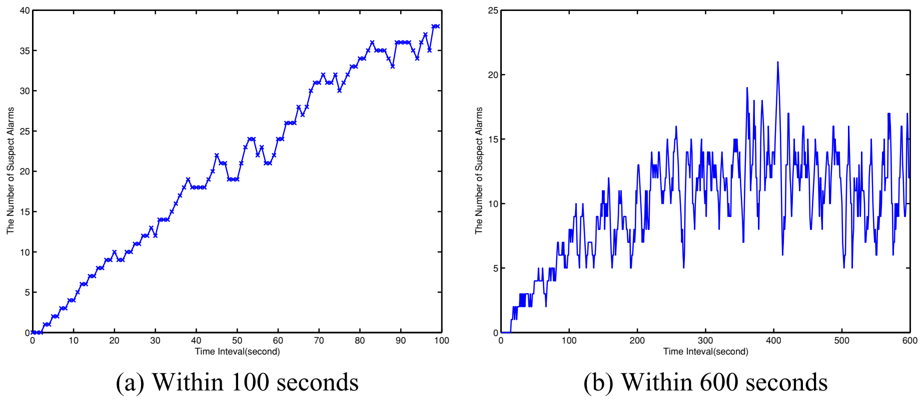


Fig. 6 The number of SA generated by the client detector within (a) 100 s (b) 600 s

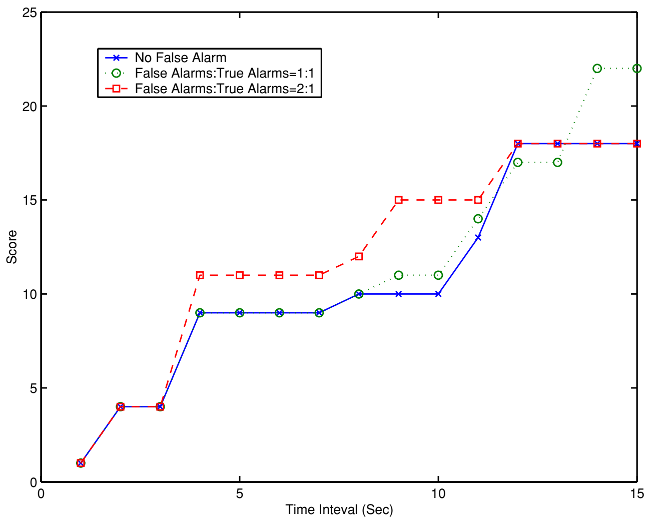


Fig. 7 The detection scheme is insensitive to false alarms

6.2. Tolerance of false negatives and false positives

We tested the ability of the proposed detection scheme to tolerate false negatives and false positives in a real network environment. Three different data sets were used. The first contained no false alarms, the second was 50% false alarms, and the third was 66% false alarms. In other words, many false negatives and false positives were in the third data set while none of them was in the first set. The results can be seen Fig. 7. Remembering that, as discussed in Section 4.1, an equation is used to calculate SA scores in order to determine whether a protected server will be sent a warning, we can see that even when false SAs in the third set make up more than 66% of total, the score value was very close to the result of the first set. The reason is that the score value rises rapidly when the SAs have the same IP and rises slowly when the SAs have distributed IP addresses. This specially designed evaluation function to compute the score value ensures the detection system is insensitive to false alarms and can be applied in the real network.

7. Conclusion and future work

This paper presents a novel system for detecting direct DDoS attacks. The system consists of a client detector and a server detector. The client detector detects on the innocent host side using a modified Bloom-filter detection scheme. The protected server uses a server detector that can both passively and actively detect DDoS attacks. The client detector and the server detector cooperate to efficiently issue accurate alarms at an early stage of a DDoS attack. The system makes use of a modified hash table, derived from the Bloom filter, in order to monitor the three-way handshake. The modified table has a low rate of false negatives and false positives. In future work we plan to design the hash function so as to reduce the occurrence of hash collision, which should also reduce potential false negatives and false positives. We also plan to test the memory and computing cost of the modified hash functions in a real environment as well as investigating the reduction of false alarms by optimizing parameters k and m.

References

1. Moore D, Voelker G, Savage S (2001) Inferring internet denial of service activity. In Proceedings of USENIX Security Symposium, Washington, DC, USA, pp 9–22
2. Postel J (1981) Transmission control protocol: DARPA internet program protocol specification, RFC 793
3. Chen Y (2000) Study on the prevention of SYN flooding by using traffic policing. In: Network Operations and Management Symposium 2000 IEEE/IFIP, pp 593–604
4. Wang H, Zhang D, Shin KG (2002) Detecting SYN flooding attacks. In: Proceedings of Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM), vol. 3, pp 1530–1539
5. Jin C, Wang HN, Shin KG (2003) Hop-count filtering: An effective defense against spoofed DDoS traffic. In: Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS), ACM Press, pp 30–41
6. Hussain A, Heidemann J, Papadopoulos C (2003) Denial-of-service: A framework for classifying denial of service attacks. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), Karlsruhe, Germany, pp 99–110
7. Keromytis A, Misra V, Rubenstein D (2002) SOS: Secure overlay services. In: ACM SIGCOMM Computer Communication Review, Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Pittsburgh, PA, vol. 32, pp 61–72
8. Keromytis A, Misra V, Rubenstein, D (2004) SOS: An architecture for mitigating DDoS attacks. IEEE Journal on Selected Areas in Communications 22:176–188
9. Stavrou A, Keromytis AD, Nieh J, Misra V, Rubenstein D (2005) MOVE: An end-to-end solution to network denial of service. In: Proceedings of the 12th Symposium on Network and Distributed System Security (NDSS)
10. Morein WG, Stavrou A, Cook DL, Keromytis AD, Misra V, Rubenstein D, (2003) DOS protection: Using graphic turing tests to counter automated DDoS attacks against web servers. In: Proceedings of the 10th ACM Conference on Computer and Communications Security,Washington, DC, USA, pp 8–19
11. XiaoFeng Wang MKR (2004) Mitigating bandwidth-exhaustion attacks using congestion puzzles. In: Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04), Washington DC, USA, pp 257–267
12. Waters B, Juels A, Halderman JA, Felten EW (2004) New client puzzle outsourcing techniques for DoS resistance. In: Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04), Washington DC, USA, pp 246–256
13. Mirkovic J, Prier G (2002) Attacking DDoS at the source. In In: 10th Proceedings of the IEEE International Conference on Network Protocols, Paris, France, pp 312–321

14. Ferguso P, Senie D (2000) Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing
15. Song DX, Perrig A (2001) Advanced and authenticated marking schemes for IP traceback. In: Proceeding of Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), pp 878–886
16. Sung M, Xu J (2003) IP traceback-based intelligent packet filtering: A novel technique for defending against internet DDoS attacks. IEEE Transactions on Parallel and Distributed Systems 14:861–872
17. Snoeren AC (2001) Hash-based IP traceback. In: Proceedings of the ACM SIGCOMM Conference, ACM Press, pp 3–14
18. Bellovin SM (2000) ICMP traceback messages. Technical report
19. Ioannidis J, Bellovin SM (2002) Implementing pushback: Router-based defense against DDoS attacks. In: Proceedings of Network and Distributed System Security Symposium (NDSS), Catamaran Resort Hotel San Diego, California, The Internet Society
20. Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13:422–426
21. Abdelsayed S, Glimsholt D, Leckie C, Ryan S, Shami S (2003) An efficient filter for denial-of-service bandwidth attacks. In: IEEE Global Telecommunications Conference (GLOBECOM'03), vol. 3, pp 1353–1357
22. Chan E, Chan H, Chan K, Chan V, Chanson S (2004) 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), pp 581–586
23. Chang RK (2002) Defending against flooding-based distributed denial-of-service attacks: a tutorial. Communications Magazine, IEEE 40:42–51