A Novel Technique for Detecting DDoS Attacks at Its Early Stage

Bin Xiao[1], Wei Chen[1,2], and Yanxiang He[2]

[1] Department of Computing,
The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
{csbxiao, cswchen}@comp.polyu.edu.hk
[2] Computer School, The State Key Lab of Software Engineering,
Wuhan University, Wuhan 430072, Hubei, China
yxhe@whu.edu.cn

Abstract. Spoofing source IP addresses is always utilized to perform Distributed Denial-of-Service (DDoS) attacks. Most of current detection and prevention methods against DDoS ignore the innocent side, whose IP is utilized as the spoofed IP by the attacker. In this paper, a novel method has been proposed to against the direct DDoS attacks, which consists of two components: the client detector and the server detector. The cooperation of those two components and their interactive behavior lead to an early stage detection of a DDoS attack. From the result of experiments, the approach presented in this paper yields accurate DDoS alarms at early stage. Furthermore, such approach is insensitive to the false suspect alarms with adopted evaluation functions.

1 Introduction

Distributed Denial-of-Service (DDoS) attacks are one of the most serious threats to the internet. As more business and commerce services depend on the internet, DDoS attacks can bring numerous financial loss to these e-business companies. As Moore [1] reported, the majority of attack packets is the TCP type. In the TCP case, SYN flooding is the most common attack behavior [2,3].

Current TCP based DDoS attacks are usually performed by exploiting TCP three-way handshake [4]. The SYN flooding attack is launched by sending numerous SYN request packets towards a victim server. The server reserves lots of half-open connections which will quickly deplete system resource, thus preventing the victim server from accepting legitimate user requests.

Lots of work has been done to detect and prevent the TCP based DDoS attack. Current counteracting methods are mostly deployed at the victim server side, or the attacker side or between them. The information at the innocent host, whose IP is utilized as the spoofed IP, is totally ignored. Detection at the side of a victim server can is more practical but can hardly produce an alarm at the early stage of a DDoS attack because abnormal deviation can only be easily

J. Cao et al. (Eds.): ISPA 2004, LNCS 3358, pp. 825–834, 2004.

found until the DDoS attack turns to the final stage. Counteracting methods near the side of attack source mainly belong to the prevention mechanism, such as filtering suspicious packets before forwarding them to the internet. Providing an early DDoS alarm near the side of an attacker is a difficult task because the attack signature is not easy to capture at this side. The information at the innocent host side will be used in our approach because the innocent side will receive abnormal TCP control packets for the three-way handshake. These abnormal packets for three-way handshake provide valuable information to give alarms at the early stage.

In order to detect a TCP based DDoS attack at its early stage, in this paper we have made the contributions as follows:

- The client detector performs detection at the side of innocent hosts because this side can provide valuable information. Another benefit of such deployment is to make the detection system itself invulnerable to direct DDoS attacks. To our best knowledge, there is no literal report about detection at the innocent host side.
- A novel cooperative detection system composed of the client detector and the server detector is presented. The cooperation improves the alarm accuracy and shortens the detection time.
- The server detector can actively send queries to the client detector to confirm the existence of a DDoS attack. This active scheme enhances the earlier DDoS detection.

The remainder of the paper is organized as follows: Section 2 will introduce some related works in spoofed DDoS detection. In Section 3 the TCP-based DDoS attack will be discussed. The techniques for cooperative DDoS detection, including the client detector and the server detector, will be presented in section 4. Some experiment results will be given in Section 5 to evaluate the performance of the proposed method. In Section 6 we will conclude our work and discuss future work.

2 The Related Work

Most of current DDoS attack detection and prevention schemes can be classified into three categories according to the location of the detector: at the victim server side, at the attack source side and between them.

Detecting DDoS at the victim server side is more concerned by researchers. In [3] Wang detected the SYN flooding attacks at leaf routers that connect end hosts to the Internet. The key idea of their method is the SYN-FIN pair's behavior should show invariant in normal network traffic and a non-parameter CUSUM method is utilized to accumulate these pairs. In Cheng's work [5], their approach utilizes the TTL(Time-To-Live) value in the IP header to estimate the Hop-Count of each packet. The spoofed packets can be distinguished by Hop-Count deviation from normal ones. A method incorporating SYN cache and cookies method is evaluated in Lemon [6] work. The basic idea is to use cache

or cookies to evaluate the security status of a connection before establishing the real connection with a protected server. For those methods that provide detection at the victim server side, the main challenge is to consume the least recourse to record the state of numerous connections and evaluate the safety of each connection.

The detection at the attack source side seems more difficult than at the server side because the signatures at the attack source side are not easy to detect however prevention at the attack source side is more effective. For example the RFC2827 [7] is to filter spoofed packets at each ingress router. Before the router forwards one packet to the destination, it will check whether the packet belongs to its routing domain. If not, this packet is probably a spoofed one and the router will drop it.

Detection and prevention between these two sides mainly include traceback [8–11] and pushback [12]. Tracing back attempts to identify the real location of the attacker. During a DDoS attack, the source IPs are often forged and can not be used to identify the real location of the attack source. Most of the traceback schemes are to mark some packets along their routing paths or send some special packets [11], together with monitored traffic flow. With these special marks, the real routing path can be reconstructed and the true source IP can be located. With the identification of real path of the spoofed packets, pushback techniques can be applied to perform advanced filtering. The pushback is always performed at the last few routers before traffic reaches target.

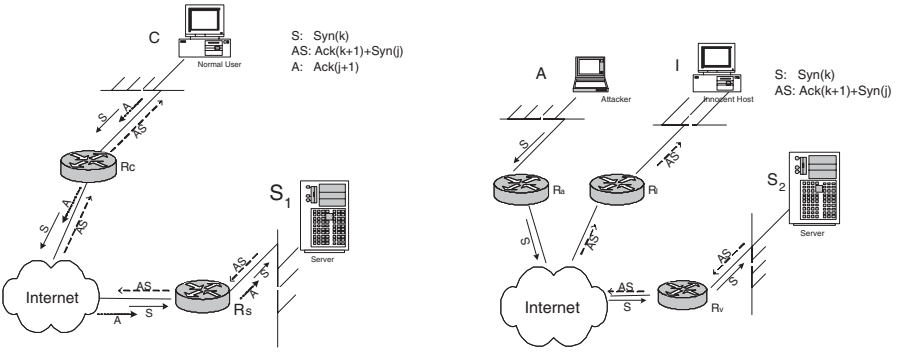
3 The TCP-Based DDoS Attack

Detecting DDoS attack at its early stage is a challenging problem. For a DDoS attacking packet, its source IP address is usually forged. The normal three-way handshake to build a TCP connection would be changed consequently.

The normal three-way handshake is shown in Figure 1(a). First the client C sends a $Syn(k)$ request to the server S_1. After receiving such request, server S_1 replies with a packet, which contains both the acknowledgement $Ack(k+1)$ and the synchronization request $Syn(j)$(denoted as $Ack(k+1)+Syn(j)$ in the following paper). Then client C sends $Ack(j+1)$ back to finish the building up of the connection. k and j are sequence numbers produced randomly by the server and client during the three-way handshake. During the normal three-way handshake procedure, $Syn(k)$, $Ack(k+1)+Syn(j)$ and $Ack(j+1)$ can be observed at the edge router(R_c in Figure 1(a)) near the client.

If the IP-Spoofed attack happens, the normal authentication process is modified. As Figure 1(b) shown, the innocent host I, whose IP is used as spoofed source IP, is usually not in the same domain with the attacker host A. In fact, to avoid being traced back, the attacker usually uses the IP address belonging to other domains to make a spoofed packet. In Figure 1(b) the edge router R_a in the attacker domain forwards the $Syn(k)$ packet with the spoofed address P_I, the IP address of the innocent host I, to the server S_2. The sever S_2 replies with an $Ack(k+1)+Syn(j)$ packet. This $Ack(k+1)+Syn(j)$ will be sent to

the innocent host I because the server S thinks the $Syn(k)$ packet is from I according to the spoofed source IP P_I in it. So the edge router R_I at the innocent host side will receive the $Ack(k+1)+Syn(j)$ packet from victim server S_2. But there is no previous $Syn(k)$ request forwarded by the client detector at R_I. This scenario is different from the normal one. Our approach is proposed on the base of this difference.



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

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

4 Techniques for DDoS Cooperative Detection

In order to detect DDoS at its early stage, the client detector and the server detector are introduced in the presented techniques. The client detector is deployed at the edge router of innocent hosts. For example, the client detector will be installed on the router R_I in Figure 1(b). It checks the TCP control packets flowing through the edge router. When it captures suspicious events, the alarm about the potential DDoS attack will be sent to the side of protected server. The server detector, employed by the protected server, can perform detection not only by passively listening the warning from the client detector, but also by actively sending queries to the client detector to confirm alarms.

4.1 The Client Detector

One of the main tasks for the client detector is to monitor the TCP control packets that comes in and goes out of the domain. Although a TCP connection may hold for several seconds or even for several minutes, most three-way handshake can be finished in a very short period(e.g., less than 1 seconds) at the beginning phrase of the connection. Compared with other stateful defense mechanisms which maintain states for the whole TCP connection, keeping the states only for the three-way handshake will reduce the computing and storage overhead.

Monitoring States for the Three-Way Handshake. To keep the states for a three-way handshake, a record is created in the hash table. There exist three states for each record: 'syn', 'ack' and 'suspicious'. To illustrate the state transition well, R is denoted for the record mapped by the hash function using source and destination IP address values as input. The $Syn(k)$, $Ack(k+1) + Syn(j)$ and $Ack(j+1)$ are denoted for TCP control packets used by the three-way handshake. These packets will trigger the creation of a new record R or the state transition of a existing record R. Two sub-detectors, egress detector(ED) and ingress detector(ID), are designed to process the outgoing and incoming packets individually. When abnormal traffic flowing through the edge router is observed, Suspect Alarm(SA) will be generated and will be sent to the client detector. The final alarm will be decided by evaluating these SAs. The total state transition for R is shown in Figure 2.

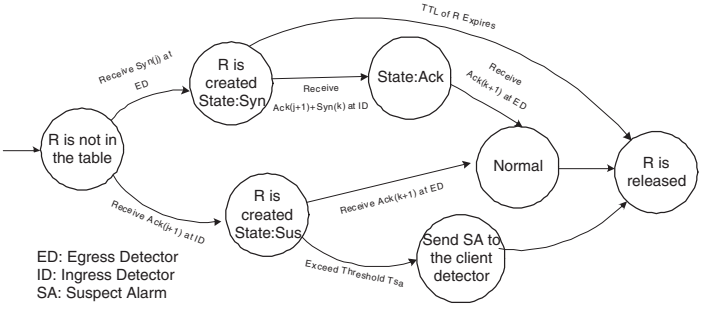


Fig. 2. States transition for R in the hash table

At the first phase of three-way handshake, A $Syn(k)$ packet will be sent from the client C to the server S to initialize a new TCP connection. When this packet is observed in outgoing traffic by ED, a new record R is created and the state of the record is set to 'syn'. This 'syn'-state record will be stored in the table with a Time To Live(TTL). If the TTL expires, this record will be deleted from the table.

When the server S receives the $Syn(k)$ packet from the client C and is ready to start the TCP session, it sends a $Ack(k+1) + Syn(j)$ packet to C. When the $Ack(k+1) + Syn(j)$ packet arrives, ID will check if there exists a corresponding 'syn'-state record previously created. If the matched one is found, the state 'syn' will be modified to 'ack'. Otherwise, a 'suspicious'-state record R will be created. It means there may exist IP spoofing behaviors, or the previous R has been deleted because the TTL of R has expired. It is 'suspicious' because it need to be verified in the third step of three-way handshake. Though the packet is suspicious, the edge router will continue to forward the packet to the destination C. If the $Ack(k+1) + Syn(j)$ packet is a legitimate one, the client C will reply with a $Ack(j+1)$ packet to finish the three-way handshake. If the $Ack(k+1) + Syn(j)$ packet is caused by a spoofed packet, there will be no any response for the handshake.

During the final stage of handshake, when the $Ack(j+1)$ packet arrives the edge router, the ED will search for the corresponding record of this three-way handshake. If it is in the table and the current state is 'ack' , it means a normal three-way handshake has been finished. The record will be released from the table. If the current state is 'suspicious', the record will also be deleted because this 'suspicious' state is caused by expiration instead of spoofed packets. As we have mentioned above, if the TTL of a 'syn'-state record expires, this record will be deleted from database and a 'suspicious'-state record will be generated. So this 'suspicious' record can be safely released from the hash table.

On the contrary, in the scenario of the spoofed handshake, no response will be generated because the spoofed IP is unreachable or the client C will send back a RST instead of a $Ack(j+1)$ packet. The 'suspicious'-state record will not be released from the table. If a 'suspicious'-state record stays for a certain time longer than the threshold, T_{time}, a Suspect Alarm(SA) will be generated and sent to the client detector.

Egress Detector and Ingress Detector. With the state transition description in Figure 2, the algorithms for ED and ID are defined individually in the Figure 3 and Figure 4. ED and ID process outgoing and incoming TCP control packets separately. They share the same hash table that contain the records for three-way handshakes. In the algorithm, the P is denoted to the TCP control packet observed at the edge router. The ED handles the $Syn(k)$ and $Ack(j+1)$ packets because these two kinds of packets are going out from the domain to the internet. The ID process the $Ack(k+1)+Syn(j)$ packets from the incoming traffic.

```
VOID Outgoing_Packets_Process (INPUT: P) {
if P is a Syn(k) packet then
  R=hash(source IP, destination IP)
  Create R
  Set R state = syn
else if  P is a Ack(j+1) packet then
  R=hash(source IP, destination IP)
  if R is found in the Hash Table AND R state == ack then
    Release R
  else if R is found in the Hash Table AND R state == suspicious then
    Release R
  end if
end if
}
```

Fig. 3. The algorithm for the Egress Detector

```
VOID Incoming_Packets_Process (INPUT: P) {
if P is Ack(k+1)+Syn(j) packet then
  R=hash(destination IP, source IP)
  if R is found in Hash Table AND R state == syn then
    Set R state=ack
  else if R is not found then
    Create R
    Set R state= suspicious
  end if
end if
}
```

Fig. 4. The Algorithm for Ingress Detector

Detection Scheme for Client Detector. If a 'suspicious' record in the hash table stays for a period of time longer than the threshold T_{sa}, a Suspected Alarm(SA) will be generated and will be stored in the local database. The client detector will analyze the source IP distribution of SAs in database. When SAs with the same source IP P_{victim} are reported in a short period, there probably exists a DDoS attack targeting the host P_{victim}. But if each SA has a different source IP, it is most likely caused by some other reasons. To evaluate the distribution of the source IP of alarms, an expression is presented below:

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

Where X_s stands for a subset of the total SA set. All the elements in X_s are SAs that have the same IP value s in a certain period. The score will increase dynamically when the number of SAs with the same source IP increases. On the other hand if each of the SAs has a different source IP, the score will reach minimum.

4.2 The Server Detector

The server detector is deployed at the protect server, such as S_2 in Figure 1(b). On the one hand, the server detector may passively wait for the potential direct DDoS alarm from the client detector. When enough potential DDoS attack alarms come, the server detector will give the confirmed direct DDoS attack alarm to server.

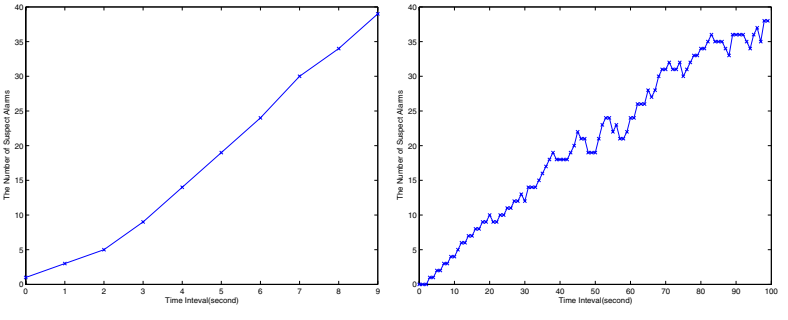
On the other hand it also can perform more active detection by sending queries to client detectors as soon as any suspicious event is found at the protected server. Sometimes the source IPs of spoofed packets is widely distributed. The number of SAs at one client detector is not enough for it to send a potential DDoS alarm to the server detector. In this scenario, the server detector will select several cooperative client detectors to query the number SAs. The client detectors will report the number of SAs with the special source IP, then the

server detector will analyze these SAs replied from different client detectors and confirm whether the half-connection is caused by spoofed DDoS packets or by some other reason.

Many other DDoS detection methods have to wait for capturing sufficient DDoS attack evidences before taking further action. This waiting delays detection and prevention against the forthcoming DDoS. In our approach, both the client detector and the server detector do not need to wait passively for special evidences, so this cooperative detection system can give DDoS alarm at the earlier stage.

5 Experiment

To evaluate the cooperative detection system, experiments are designed to test whether the cooperative detection approach can detect DDoS at the early stage. In the experiment, 10 zombies are simulated to perform SYN flooding attacks toward the server. The cooperative detection system include five client detectors and one server detector. The rate of the attack packets rises from 10 packets/sec to 1000 packets/sec in 10 seconds and 100 seconds respectively. The maximum rate is set to 1000 packets/sec because it is enough to shut down some services as Chang reported [13]. In simulation only 1% of $Ack(k+1) + Syn(j)$ packets replied by the victim server are designed to arrive client detectors. These packets will trigger client detectors to generate SAs. The number of SAs generated by the client detectors is shown in the Figure 5.



(a) Within 10 seconds (b) Within 100 seconds

Fig. 5. The number of SA generated by the client detector after receiving 1% of $Ack(k+1) + Syn(j)$ packets from the protected server. The attacking packets rate reaches to 1000 packets/sec within (a)10s (b)100s

Although only 1% spoofed attack packets can be received by client detectors, client detectors still give accurate SAs. The number of SAs is enough for client detectors to give a further potential DDoS alarm at the early stage of the

DDoS attack. The detection results are satisfying even when the DDoS attacker increases the attack packets slowly. From experiment results,the SAs number raises stably in the Figure 5(a) because the DDoS attack is launched in a short time. In the Figure 5(b) the number of SAs seems a little vibrated because the attack packets rise up at a much slower rate.

Considering there exist network errors or latency which may cause false SA at the client detector, the false tolerance ability of the approach is evaluated. To test the sensibility to true SA and tolerance to false one , three different data sets are involved in the second experiment. The first data set contains no false alarms. The number of false alarms contained in the second data set is as many as the true alarms. The number of false alarms in the third one is as many as two times of the true alarms. The experiments result is given in Figure 6.

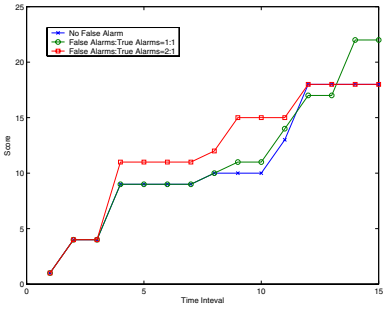


Fig. 6. Insensitive to false alarms

The experiment results show that even the false SAs are more than 50% of total SAs, the score will not be influenced much. The reason is that the score evaluated by the client detector raises rapidly when the SAs have the same IP while reach minimum when the SAs have different IP addresses. So SA evaluation expression is insensitive to false alarms. In the real code running on real machine, more than 50% suspect alarms are expect to be true because false SA caused by the network errors or latency is rather occasional.

6 Conclusion and Future Work

In this paper a novel detection method against the direct DDoS attack is presented. The detection system consists of the client detector and the server detector. The client detector performs detection at the side of innocent hosts and this is quite different from current methods which are often deployed at the sides of the victim server or the attacking source. The server detector is employed by the protected server and can perform both passively and actively DDoS detection. The benefit of this method is to yield accurate alarms at the early stage of DDoS attack. The key idea in the cooperative detection is based on the difference

between the normal TCP three-way handshake and the spoofed one. The experiment results are given in section 5 and the proposed approach is effective to detect DDoS at its early stage. The results also show the SA evaluation method is insensitive to the false SAs.

In the future research work we will apply method to real environment to test the memory and computing cost for actual running because the presented method will need more storage and computing cost than some other stateless methods [14] [3]. Some advanced hash algorithms for recording the states of three-way handshake will be researched to compress the storage space and improve the query efficiency.

References

1. Moore, D., Voelker, G., Savage, S.: Inferring internet denial of service activity. In: Proceedings of USENIX Security Symposium. (2001)
2. Chen, Y.: Study on the prevention of SYN flooding by using traffic policing. In: Network Operations and Management Symposium 2000 IEEE/IFIP. (2000) 593–604
3. Wang, H., Zhang, D., Shin, K.G.: Detecting SYN flooding attacks. In: Proceedings of IEEE INFOCOM. Volume 3. (2002) 1530–1539
4. Postel, J.: Transmission control protocol : DARPA internet program protocol specification,RFC 793 (1981)
5. Jin, C., Wang, H.N., Shin, K.G.: 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 (2003) 30–41
6. Lemon, J.: Resisting SYN flood DoS attacks with a SYN cache. In: In Proceedings of the BSDCon 2002 Conference. (2002)
7. Ferguson, P., Senie, D.: (Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing,RFC2827)
8. Song, D.X., Perrig, A.: Advanced and authenticated marking schemes for IP traceback. In: INFOCOM 2001. (2001) 878–886
9. Sung, M., Xu, J.: IP traceback-based intelligent packet filtering: A novel technique for defending against internet DDoS attacks. IEEE Transactions on Parallel and Distributed Systems **14** (2003) 861–872
10. Snoeren, A.C.: Hash-based IP traceback. In: Proceedings of the ACM SIGCOMM Conference, ACM Press (2001) 3–14
11. Bellovin, S.M.: ICMP traceback messages. Technical report (2000)
12. Ioannidis, J., Bellovin, S.M.: Implementing pushback: Router-based defense against DDoS attacks. In: Proceedings of Network and Distributed System Security Symposium, Catamaran Resort Hotel San Diego, California, The Internet Society (2002)
13. Rocky.K.Chang: Defending against flooding-based distributed denial-of-service attacks: a tutorial. Communications Magazine, IEEE **40** (2002) 42–51
14. Yaar, A., Perrig, A., Song, D.: SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. In: Proceedings. 2004 IEEE Symposium, Security and Privacy. (2004) 130–143