A More Accurate Scheme to Detect SYN Flood Attacks

Changhua Sun, Chengchen Hu, Yachao Zhou, Xin Xiao and Bin Liu
Department of Computer Science and Technology, Tsinghua University, Beijing, China
{sch04, zhouyc06, xiaoxin00}@mails.tsinghua.edu.cn, {huc, liub}@tsinghua.edu.cn

I. Introduction

Distributed Denial-of-Service (DDoS) attack remains a serious problem on the Internet, as it takes advantage of the lack of authenticity in the IP protocol, destination oriented routing, and stateless nature of the Internet. Among various DDoS attacks, TCP SYN flooding is the most commonly used one and still dominates DDoS attacks according to the recent NANOG report [1] in 2008.

There are two main causes of SYN flood attacks. The first is the inherent asymmetry feature in TCP three-way handshake protocol, which enables the attacker to consume substantial resources at the server, while sparing its own resources. The other is the server cannot control the packets it receives, especially the SYN packets can easily reach the server without its approvement. Fig. 1 illustrates the three-way handshake protocol: 1) A client sends a SYN packet to a server to perform an active open request; 2) The server reserves connection resources (backlog queue) to track the TCP state on receiving a SYN packet and replies with a SYN/ACK packet in response; 3) Finally, the client sends an ACK back to the server as an acknowledgement, and the connection is established when receiving this ACK on the server side. We call the ACK packet in the third step as *CliACK*. During SYN flood attacks, an attacker generates a large number of SYN requests but never sends the CliACK packets to complete the connections. The victim server's backlog queue can be easily exhausted and all the new incoming SYN requests are dropped. Furthermore, other system resources like network bandwidth are occupied.

Among various SYN flood defense mechanisms [2], victim modifications, like SYN cookies, are the most viable techniques up-to-date to mitigate SYN floods. However, SYN cookies are not able to encode all TCP options, especially window scale and selective acknowledgment, which are believed to improve TCP's performance and are widely supported. Moreover, they only aim to solve the asymmetry feature in TCP protocol. Recently, leaf router-based SYN flood detections [3]–[6], have received much attention in current literatures with the advantage of protecting a batch of servers in the ISP's networks, and may solve the other bandwidth consumption problem here. These methods utilize the relationship between the control packets in the connection establishment and release, like SYN, SYN/ACK, CliACK, FIN and RST, and their ideas are very good. However, they all have at least one main drawback that the attacker can evade the detection by spoofing the control packets. SYN-FIN(RST) pair's behavior is widely used [3], [5], [6], but the attacker can easily spoof the FIN(RST) packet, especially the sequence number in the FIN(RST) packet has little relationship with that in the SYN packet. SYN-CliACK pair is also used [4] but has the same problem. The attacker can easily spoof the CliACK packet, *i.e.*, the attacker can use wrong y in Fig. 1

This work is partly supported by NSFC (60573121, 60625201, 60873250), China 973 program (2007CB310701), the Specialized Research Fund for the Doctoral Program of Higher Education of China (20060003058), the Cultivation Fund of the Key Scientific and Technical Innovation Project, MOE, China (705003) and 863 high-tech (2007AA01Z216, 2007AA01Z468).

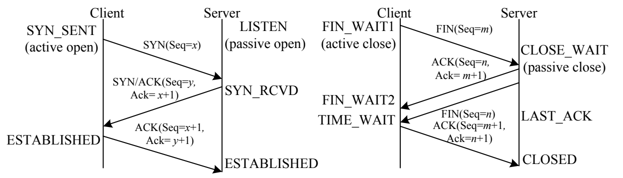

Fig. 1. TCP connection's establishment and release.

In this poster, we propose a more accurate SYN flood detection scheme which leverages the SYN/ACK-CliACK pair's behavior. As shown in Fig. 1, it has only one SYN/ACK packet and one CliACK packet for each normal TCP connection. During the SYN flood attack, since the SYN/ACK packets have no corresponding CliACK packets, the number of SYN/ACK packets is much larger than the number of CliACK packets. We employ a space-efficient data structure, counting Bloom filter [7], to store the full information (*6-tuple*) of per TCP connection attached in the SYN/ACK packet, including client and server's IP addresses, ports and initial sequence numbers, and then to recognize the corresponding CliACK packet. In this case, it is impossible for an attacker to generate the spoofed CliACK packets to evade the detection. The memory cost of our scheme is about 2MB even for 10Gbps link speeds, making it feasible and attractive for the hardware implementation in modern routers.

II. The Proposed Scheme

A. Overview

Our scheme exploits the unspoofable behavior of the SYN/ACK-CliACK pair and the key point is to recognize the CliACK packet efficiently. The time interval between the SYN/ACK and its CliACK is a RTT (round trip time), and it is shown that the RTT is less than 500ms for more than 90% of TCP connections. Therefore, the intuitive method is to store the 6-tuple of SYN packets for a certain time, like one second, and then to recognize the right CliACK packet. However, the implementation cost is very high. One 6-tuple item takes 20 bytes. The total number of items, n, stored in one second mainly depends on the sum of maximum SYN packet rate. According to the reports on a OC-192 backbone link from CAIDA's monitor [8], the maximum packet rate of the backbone link is far less than 700Kpps. In addition, the maximum SYN flood rate reported [9] is about 50Kpps. Therefore, it is very safe to suppose $n = 700000$. In this case, the memory cost of the intuitive method is about 14MB. Moreover, we need to devise a mechanism to find the right SYN packet item when receiving ACK packets, which is very challengeable. By contrast, our scheme can recognize the CliACK packet more efficiently with low cost.

B. Architecture and Algorithm

As shown in Fig. 2, we use a counting Bloom filter (CBF) to recognize the CliACK packets. For each outgoing SYN/ACK packet, we extract its source and destination IP addresses, ports, sequence number and ACK sequence number and use the *6-tuple* <srcIP, dstIP, srcport, dstport, seq, ACKseq>, denoted as a, as the input for the k hash functions $h_1, h_2, ..., h_k$, hash value of each with range $\{0, 1, ..., m-1\}$. We increase the value by one for each bucket location $h_i(a)$. If the value of any $h_i(a)$ bucket would be overflowed to zero, we do not increase it and simply keep its maximal value. For each incoming ACK packet, we extract and use its 6-tuple <dstIP, srcIP, dstport, srcport, ACKseq -1, seq>, denoted as b, as the input for the k hash functions of CBF. If the value of all the bucket is greater than zero, *i.e.*, $h_i(b) > 0$ for $1 \leq i \leq k$, this ACK is recognized as the CliACK packet with some wrong probability, referred to as the *false positive* rate p_{fp}, since some buckets of $h_i(b)$ may be set to be greater than zero by other elements. For the recognized CliACK packet, we decrease the value by one for each bucket location $h_i(b)$. p_{fp} is approximately $(1 - e^{-kn/m})^k$. By optimizing the number of hash functions with minimized false positive rate, we get $k = \ln 2 \cdot (m/n)$. We choose $k = 4$, and then $m = 6n$, $p_{fp} = 0.0561$. The memory needed (denoted as M) is $M = 4m = 24n$ bits. If $n = 700000$, M is about 2MB.

C. Performance Evaluation

We use trace-driven simulations (public traces from NLANR) to evaluate the performance. And four metrics are used in the evaluations. 1) *Response time*. The detection (quiescence) response time is the time interval between the attack's start (end) and scheme's detection of its start (end). Fig. 3 shows the quiescence response time of our scheme (named SACK^2) and PCF [6] under one situation. It is illustrated that our scheme has a shorter response time, and the result is similar in other simulations. 2) *Implementation cost* is recognized as the memory and computation cost. As illustrated before, we only need 2MB memory cost. The computation

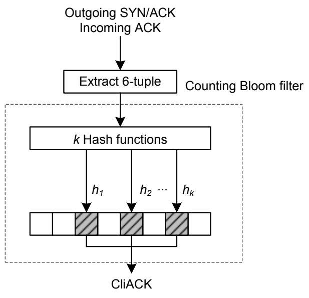

Fig. 2. Identify CliACK packet.

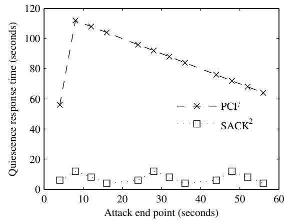

Fig. 3. Quiescence response time.

cost depends on the hash functions used in the Bloom filter, and can be very low using H_3 class. 3) and 4) *False positive (negative) rate* is the rate between the number of *reported attacks by mistake* (*unreported attacks*) and the number of the real attacks. The false positive rate of the Bloom filter means we may recognize a small portion (no more than 5.61%) of ACK packets as CliACK packet and may only bring a little false negative for our scheme. During all our simulations, we find no false positive rate and no false negative for our scheme, but find a few false positive for other mechanisms, partly because they cannot distinguish port scan from SYN flood [3]. Moreover, Existing mechanisms all have false negative since the attacker can spoof some control packets to evade the detection. However, the spoofing is not workable under our scheme as we utilize the full information of TCP connections.

III. Conclusion

We propose to use the SYN/ACK-CliACK pair's behavior to detect the various SYN flood attacks more accurately. The SYN/ACK packets carry the full information of the TCP connections and it is impossible for the attacker to evade the detection by spoofing the control packets. Moreover, we use a space efficient data structure, counting Bloom filter, to recognize the CliACK packet and the memory cost is 2MB even for 10Gbps link speeds. We need to fully compare our scheme with the existing detection mechanisms in future.

References

[1] C. Labovitz, D. McPherson, S. Iekel-Johnson, and M. Hollyman, "Internet Traffic Trends - A View from 67 ISPs," NANOG, 2008. [Online]. Available: http://www.nanog.org/meetings/nanog43/presentations/Labovitz_internetstats_N43.pdf

[2] W. Eddy, "TCP SYN flooding attacks and common mitigations," RFC 4987, 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4987.txt

[3] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," in *IEEE INFOCOM*, 2002.

[4] W. Chen and D. Y. Yeung, "Defending against TCP SYN flooding attacks under different types of IP spoofing," in *Fifth International Conference on Networking (ICN)*, 2006.

[5] C. Sun, J. Fan, and B. Liu, "A robust scheme to detect SYN flooding attacks," in *International Conference on Communications and Networking in China*, Aug 22-24 2007.

[6] R. Kompella, S. Singh, and G. Varghese, "On scalable attack detection in the network," *IEEE/ACM Trans. Networking*, vol. 15, no. 1, pp. 14–25, 2007.

[7] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: a survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[8] "equinix-chicago monitor Realtime reports," 2008. [Online]. Available: http://www.caida.org/data/passive/monitors/equinix-chicago.xml

[9] "SCO Offline from Denial-of-Service Attack," 2003. [Online]. Available: http://www.caida.org/research/security/sco-dos/