A Novel Router-based Scheme to Mitigate SYN Flooding DDoS Attacks

Changhua Sun, Jindou Fan, Lei Shi, Bin Liu
Department of Computer Science and Technology, Tsinghua University, China
{sch04, shijim}@mails.tsinghua.edu.cn, fanjindou@gmail.com, liub@tsinghua.edu.cn
http://s-router.cs.tsinghua.edu.cn/%7Esunchanghua

I. Introduction

Distributed Denial-of-Service (DDoS) attack remains a serious problem on the Internet today, 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, the TCP SYN flooding [1] is the most commonly-used one. It exploits TCP's three-way handshake mechanism and TCP's limitation in maintaining half-open connections. When a server receives a SYN packet, it returns a SYN/ACK packet and allocates resources (typically backlog queue in the system memory) to track the TCP state. Then the server would wait until either the half-open connection completes or the TCP connection times out. In the SYN flooding attack, the server will receive a large number of SYN packets but never receive the final ACK packets to complete the three-way handshake. Then the victim server's backlog queue can be easily exhausted, causing all the new incoming SYN requests to be dropped. Furthermore, many other system resources, such as CPU and network bandwidth used to retransmit the SYN/ACK packets, are occupied.

The most viable techniques [2] up-to-date to defend SYN floods include SYN cache [3] and SYN cookies [4]. SYN cache is to allocate minimal state when the initial request is received, and only allocate all the resources when the connection is completed. If the backlog queue is full, the oldest entry is removed. SYN cookies allocate no state for half-open connections. Instead, they encode most of the states and encrypt them into the sequence number transmitted in the SYN/ACK packet. The ACK packet that completes the handshake can be used to reconstruct the state to be put into the backlog queue. One problem with SYN cookies is not able to encode all the TCP options, and the other is that TCP protocol with SYN cookies would never retransmit the unacknowledged SYN/ACK packet. In addition, both of them do not handle application data piggybacked on the SYN segment, i.e., incompatible with Transactional TCP (T/TCP) [2]. Moreover, since deployed at the victim server, they can not alleviate bandwidth exhaustion, and not decrease the normal traffic's delay resulting from SYN flooding.

Router-based schemes can be complementary to victim modifications. Mitigation in [5] requires per TCP connection state in router, which would bring scalable problems. Wang et al. [6] proposed *SYN detection* to detect SYN floods, which is based on the fact that a normal TCP connection starts with a SYN packet and ends with a FIN or RST packet. When the SYN flood starts, there will be more SYN packets than FIN and RST packets. However, the attacker can avoid detection by sending a mixture of FIN (RST) and SYN packets.

In this poster, we propose a novel router-based scheme to mitigate SYN floods. Our scheme includes two phases: *detecting* and *mitigation*. In the detection phase, we use a counting Bloom Filter [7] to store the *4-tuple* (source and destination IP, source and destination Port) of counted SYN packets. Then a FIN or RST packet is counted only if its *4-tuple* is in the Bloom Filter. Therefore, any malicious FIN or RST packets would not be counted. In this case, the fact that there will be more SYN packets than FIN and RST packets during SYN flood would still hold. Through trace-driven simulations, we show our detection scheme is more efficient and robust in detecting various SYN flooding attacks. In the mitigation phase, we utilize *the client's persistence*, i.e., client's reaction to packet loss by subsequent retransmissions. We intentionally drop the first SYN packet of each connection request, and let go the second SYN packet. If a connection completes the three-way handshake, its subsequent SYN packets are passed. Otherwise, the subsequent SYN packets are passed with certain probability. The scheme can mitigate the SYN floods effectively, as just a small portion of SYN flooding packets reach the victim. In summary, our scheme is lightweight, stateless, but highly efficient. Moreover, it can be easily deployed at ISP's edge routers.

II. Methodology

A. Detection Scheme

We define *valid SYN packets* as the pure SYN and SYN/ACK packets, and *valid FIN packets* as the FIN and RST packets that close the TCP connections which either complete the three-way handshake or have a *valid SYN packet* in the same traffic direction before this packet. Then there are more *valid SYN packets* than *valid FIN packets* under SYN flooding.

This work is supported by NSFC (No. 60373007, 60573121 and 60625201), the Cultivation Fund of the Key Scientific and Technical Innovation Project, Ministry of Education of China (No. 705003), the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20040003048 and 20060003058), China/Ireland Science and Technology Collaboration Research Fund (2006DFA11170), and the Tsinghua Basic Research Foundation (JCpy2005054).

when we receive a SYN or SYN/ACK packet, the counter of *valid SYN packets* is increased. We extract its *4-tuple* as an item, and insert this item to a counting Bloom filter. A Bloom filter is a simple space-efficient data structure for representing a set in order to support membership queries. When we receive a FIN or RST packet, the item of its *4-tuple* is also extracted and queried from the Bloom filter. If this item is in the Bloom filter, the counter of *valid FIN packets* is increased, and this item is deleted from the counting Bloom filter. If not, this packet is not a *valid FIN packet*, and nothing is needed. Our detection scheme utilizes the change of the discrepancy between *valid SYN and FIN packets*. The detail is in [8].

B. Mitigation Scheme

In SYN floods, attacker would send a quick barrage of SYN packets from IP addresses (often spoofed) that will not generate replies to the SYN/ACKs. To remain effective, attacker needs to send new barrages of bogus connection requests frequently. Most of the SYN flooding packets would not be retransmitted. On the other hand, If a legitimate client's SYN packet is lost, it would retransmit the SYN packet several times before giving up. Our mitigation scheme utilizes the characteristic of SYN floods and client's persistence. We use three counting Bloom filters to record related information:

- **BF-1**: to record the *4-tuple* of the first SYN packets of each connection;
- **BF-2**: to record the *4-tuple* of SYN packets, whose connections have completed the three-way handshake;
- **BF-3**: to record the *4-tuple* of other SYN packets.

The mitigation scheme starts working once detecting SYN floods. If a SYN packet is received, its *4-tuple* is extracted as an item and queried from the three BFs. The results are: 1) The item is not in any of the three BFs. This TCP connection is new, then we drop this SYN packet and insert the item to **BF-1**; 2) The item is in **BF-1**. This is the second SYN packet. We pass it and move the item from **BF-1** to **BF-3**; 3) The item is in **BF-2**. We pass the packet; 4) The item is in **BF-3**. We pass the packet with a certain probability p. We insert the item to **BF-3** and obtain the number, n, of this item in **BF-3**. Let $p = 1/n$, then p is smaller as the increasing of n.

If a ACK packet is received, its *4-tuple* is also extracted as an item and queried from the three BFs. The result is used as follows: 1) The item is not in any of the three BFs, or in **BF-1**. We drop this packet; 2) The item is in **BF-2**. We pass this packet; 3) The item is in **BF-3**. This TCP connection is completed. Then we pass this packet and move the item from **BF-3** to **BF-2**.

If the attacker uses different *4-tuple* of SYN packets, these SYN packets would be classified as the first SYN packets of each connection, and would be dropped. If some SYN packets with the same *4-tuple* are used in the attack, a small portion of SYN flooding packets would reach the victim (such as the second SYN packets). If these SYN packets are retransmitted again and again, they are dropped with higher and higher probability. Therefore, our mitigation scheme can drop most of SYN flooding packets and protect the victim.

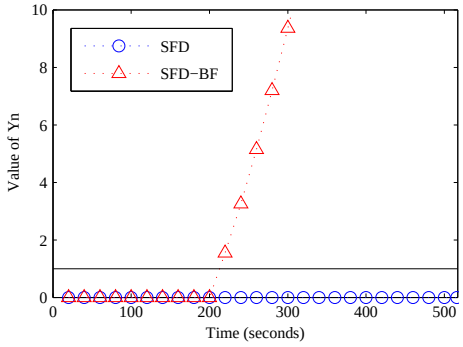

Fig. 1. SYN flooding detection under a complex attack.

C. Simulation Results

We carry out trace driven simulations to evaluate the performance of detection scheme, and details are in [8]. Fig. 1 shows the result of our detection scheme (*SFD-BF*) and the scheme in [6] (*SFD*) under a complex SYN flooding attack. The value of y_n greater than 1 reports the attack. It is shown that *SFD-BF* can detect the attack in a single observation period while *SFD* can not.

III. Conclusion And Open Issues

We propose a novel router-based SYN flooding defense scheme. Our scheme includes *detection* and *mitigation*. The detection scheme utilizes the inherent TCP *valid SYN–FIN* pairs behavior, hence is capable of detecting various SYN flooding attacks with high accuracy and short response time. The mitigation scheme further drops most of SYN flooding packets and protect the victim. Our scheme is stateless and requires low computation overhead (Bloom filter can be easily implemented by H_3 hash functions [9]), making itself immune to SYN flooding attacks.

However, sophistic attackers would retransmit every SYN packet two times to destroy the function of mitigation scheme. It is necessary to make it more robust and adaptive. At the same time, we are working on evaluate the proposed scheme in real network system and study its impact on legitimate users.

References

[1] CERT Advisory CA-1996-21 TCP SYN flooding and IP spoofing attacks. [Online]. Available: http://www.cert.org/advisories/CA-1996-21.html

[2] W. Eddy, "TCP SYN flooding attacks and common mitigations," February 2007. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-tcpm-syn-flood-02.txt

[3] J. Lemon, "Resisting SYN flood DoS attacks with a SYN cache," in *USENIX BSDCon*, 2002.

[4] "SYN cookies." [Online]. Available: http://cr.yp.to/syncookies.html

[5] B. Al-Duwairi and G. Manimaran, "Intentional dropping: A novel scheme for SYN flooding mitigation," in *Global Internet Symposium*, 2005.

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

[7] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[8] [Online]. Available: http://s-router.cs.tsinghua.edu.cn/%7Esunchanghua/publication/THUTR200703SYN.pdf

[9] M. V. Ramakrishna, E. Fu, and E. Bahcekapili, "Efficient hardware hashing functions for high performance computers," *IEEE Transactions on Computers*, vol. 46, no. 12, pp. 1378–1381, 1997.