Defending Against SYN Flood Attack under Asymmetric Routing Environment

Jianxi Tao[*†§], Li Zhou[‡], Zhou Zhou[*†], Rong Yang[*†], Wei Yang[*†], Qingyun Liu[*†]

* Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
† National Engineering Laboratory for Information Security Technology, Beijing, China
‡ National Computer Network Emergency Response Technical Team/Coordination Center, Beijing, China
§ College of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing, China
Corresponding Author: yangrong@iie.ac.cn

***Abstract*—SYN Flood attack is still one of the major distributed denial of service attacks. Any network device or computer system with connection state table may have the possibility of suffering from this attack. Moreover, under asymmetric routing environment, unidirectional traffic problem makes it more difficult to defend against SYN Flood attack. In allusion to this problem, this paper presents DARE, a novel SYN Flood defense architecture. It consists of a statistical attack detection strategy and a dual connection management strategy. We verify the feasibility and effectiveness of our method through experiments in real network environment. The results show that our proposed method can filter SYN flood traffic, and mitigate the pressure of network infrastructure.**

Keywords-SYN Flood; asymmetric routing; connection management;

I. INTRODUCTION

With the development of streaming media business and the upgrade of user access bandwidth, the volume of Internet traffic is growing rapidly. The growth rate stays high for long time. Unfortunately, with the growth of Internet traffic, the DDoS attacks occur more and more rampantly. SYN Flood is one of the most salient problems. Any network device or computer system with connection state table, such as IDS (Intrusion Detection System), IPS (Intrusion Prevention System) *etc*., have the possibility of suffering from this attack. Nowadays, due to large-scale SYN Flood attacks, the connection state tables of network infrastructure are exhausted easily. They are not adequate to protect against SYN Flood attack effectively any more [1, 2]. What's worse, asymmetric routing becomes a universal phenomenon along with the complex structure of Internet. Asymmetric routing is a situation where for one reason or another packets flowing in *i.e*. TCP connections flow through different routes to different directions. Consequently, for network security appliance deployed between communicating parties, it makes original connection state management scheme invalid under the asymmetric routing environment.

The above observations reveal that we must find out a more effective SYN Flood defense solution. To this end, this paper presents a SYN Flood defense architecture, namely DARE, for intermediate network security appliance under asymmetric routing environment. Combined a statistical attack detection method with a dual connection management strategy, DARE can ease the influence brought by SYN Flood attack.

The remainder of this paper is organized as follows. Section II analyzes the limitations of existing approaches under asymmetric routing environment. Section III presents the proposed defense architecture. The experimental results reported in Section IV. Finally, Section V concludes the paper.

II. RELATED WORK

Many solutions have been proposed for defending against SYN Flood attack. They can be divided into two categories. One is that the server ensures the authenticity and legality of the SYN request via some assistant mechanisms, such as SYN Cookie, SYN Cache, SYN Proxy and SYN Kill. The other is that detecting the attack through control bits of TCP segment, then filter the attack traffic based detection result.

Although existing schemes can mitigate the damage of SYN Flood attacks to some extent, they can't conceal their inherent drawbacks and some application scenarios constraints yet. The first category can only protect end system, which provides some service (*e.g. web site*, *email*) for others [3-8]. Nevertheless, for network security appliance deployed between communicating parties, these methods are far from effectiveness. For the second category, they have a precondition that they can see all packets of a session in both directions[9-13]. However, under asymmetric routing environment, the precondition does not always hold.

III. THE DARE ARCHITECTURE

The defense architecture, namely DARE is shown in Fig. 1, traffic capturer gets traffic from network interface and transmits them to traffic dispatcher and attack detector, and then traffic dispatcher lead them to either of connection management with SYN or without SYN on the basis of detector's output, which has two possibilities: attacked state and normal state. The basic idea that we create connection state entry without SYN segment when detector's output is attacked state is innovative.

Before discussing the details of our architecture, we classify the TCP packets by packet header information as follows:

- SP (SYN packet): the SYN flag is set 1;
- AP (ACK packet): the ACK flag is set 1 and have no data;

This work is partially supported by The National High Technology Research and Development Program of China (863 Program), Grant No. 2011AA010703; The National Information Security Program of China (242 Program), Grant No. 2012A99; The National Natural Science Foundation of China (61303260).

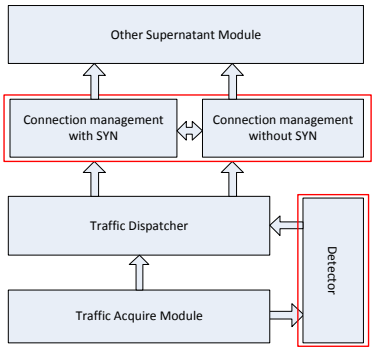

Fig. 1: The DARE Architecture

- DP (DATA packet): the ACK flag is set 1 and data field is not null;
- RFP (RST or FIN packet): the RST flag is set 1 or the FIN flag is set 1;
- 1stDP (first DATA packet): the DATA packet that is transferred firstly after connection establishing;

It must be specified that the SYN/ACK packet will be identified as SP. That's because, under the asymmetric routing environment, the traffic acquire module does not always get the complete three-way handshake packets. But as so long as either SYN packet or SYN/ACK packet is received, it must indicates that a connection is establishing. In the similar way, a connection will be closed when either a RST packet or FIN packet is received.

A. *Attack Detector*

Numerous studies show that the traffic generated by normal Internet behavior of Internet users is stable and smooth. However, in case of attacked state, many statistical properties become abnormal. Therefore, we can detect anomaly by statistical properties. Taking into account of actual feature of the traffic under the asymmetric routing environment, we choose SYN rate (the rate for which SYN packets account in all packets, express as α in follows) and destination IP address entropy (expressed as $E(DIP)$s) as the event to be detected.

The attack detection begins in the empty state, which is the original state before starting up. Empty state will transform into counting state immediately when booting finished. In that state it can receive the data from traffic acquire module and collect some statistics information, such as α, $E(DIP)$, etc. If α is larger than T_u (the upper bound of α), it will transform into attacked state. If α is less than T_l (the lower bound of α), it will transform into normal state. Otherwise, α is between T_l and T_u , it will transition to suspicious state. In that state it will make further judgment that checking the value of $E(DIP)$ whether it exceeds the threshold, which is represented as e in the diagram. If the value of $E(DIP)$ is less than or equal to e, detector will output attacked state. If not, it will output normal state. It will stay normal state or attacked state for a spell in one detection period, and then transition to counting state automatically. The attack detection procedure repeats as above, and it provides output result for traffic dispatcher continually.

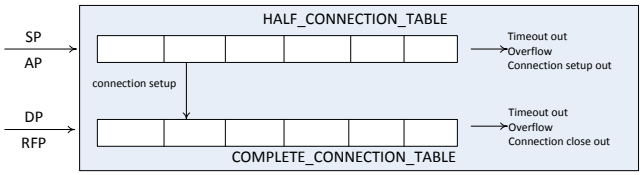

Fig. 2: Connection Management with SYN

B. *Connection Management with SYN*

As Fig. 2 shows, two hash tables are used to record the state of each connection, the one which is called HTC (Half Connection Table) keeps the half connection information, and the other which is called CCT (Completed Connection Table) maintains the state of completed connection. Upon receiving a TCP packet, we extract four-tuple (source IP address, destination IP address, source port number and destination port number) from packet header, and then use it to calculate a hash value, which is used as an index into a hash table. We used a fixed-length queue which implemented by linked list to resolve the hash collision. When a connection state entry is not found in the queue, a new entry is created and inserted into the tail of queue.

A new connection state entry is added into HCT only when a SP arrives and the entry with same four-tuple is not existed in that table. If an AP or DP arrives and the entry with same for-tuple is found in the HCT, this connection state entry is moved into CCT, which indicates that a new connection has established. Otherwise either AP or DP is dropped. The processing of DP has subtle difference from AP. When a DP is received, we look it up in CCT before HCT. In case a connection state entry is found in CCT, the connection state will be updated directly. But for AP, we searched in HCT immediately. The arriving of RFP denotes that a connection is end up, and we move the corresponding connection state out from both tables.

C. *Connection Management without SYN*

Previous method can only cope with the normal situation, but not with the situation suffering from attacking. Attacker exploits the flaw of three-way handshake to create intentionally a large number of half-open connections until the system memory resources are exhausted. We make a assumption that there is no three-way handshake, then there is no half-open connection, so there is no SYN Flood attack. The main role of three-way handshake is that communicating parties informed each other that their send-receive functions are intact. Since network security appliances are neither client nor server, but a third-party which is situated between communicating parties, it can ignore three-way handshake and only care about data transfer process. That's where our basic idea comes from. Based on this idea, we develop a connection management strategy without SYN packets.

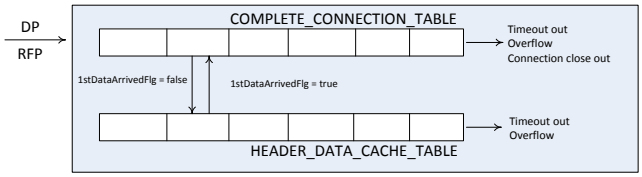

Fig. 3: Connection Management without SYN

However, a key question at this point must be worked out. It is how to ensure that the first TCP packet with payload (we call it first data packet in follows, 1stDP for short) has arrived. 1stDP is important because many application-level protocol features are able to be collected from it. Therefore, this question is the key point of connection management without SYN.

The size of two windows (receive window and send window) have reached an agreement when the connection established, and the value of both sizes are limited. The number of packets sent by client before it received the first acknowledgement of the first data packet does not exceed the receive window size of server. To resolve this question, we design a buffer window to buffered the first N arrived packet of a connection, and check their sequence numbers and packet length. If the reassembled data stream has no gap, we think the 1stDP has arrived.

Supposing that arrival sequence of 1stDP of connection i is n_i , and the size of buffer window is W . Take W=$max\{n_1, n_2, \ldots, n_i, n_{i+1}, \ldots\}$, then the 1stDPs of all connections must be captured. While $max\{n_1, n_2, \ldots, n_i, n_{i+1}, \ldots\}$ is hard to derive theoretically, we conjecture that the value of n_i is finite. Hence, a enough large N must be existed, for any i, $N \geq n_i$, i.e., $N \geq max\{n_1, n_2, \ldots, n_i, n_{i+1}, \ldots\}$. Therefore, take $W = N$, this question can be resolved. In Section IV-A, the value of $N$is probed.

This method is implemented as Fig. 3, which has a little difference from Fig. 3. CCT is used to maintain the completed connection state, and HDCT (Header Data Cache Table) is designed to record sequence number and length of the first N arrived packets. A new entry of CCT is created when a DP arrives and there is no same four-tuple connection state entry in it, and an existing entry is moved out from both tables when a RFP is received. The element in HDCT is created along with the creation of connection state entry. Compared with connection management with SYN, the state entry has an additional flag named 1stDPArrivedFlg, which indicates whether the 1stDP has arrived or not. It is initialized to false. Before 1stDPArrivedFlg becomes true, for a connection, the sequence number and packet length of all DPs are put into the corresponding buffer window until the buffer window is full. If the buffer window is full and there is no gap according to the series of sequence number and packet length, 1stDPArrivedFlg is set to true and this entry in HDCT is eliminated.

Someone may think that the connection management without SYN strategy can deal with all situations, and the strategy with SYN described in Section III-B can be replaced. Although strategy without SYN can work in normal situation, its effectiveness is lower than the strategy with SYN, because the former strategy needs to check whether the 1stDP have arrived or not. Therefore, we still use the connection management with SYN to cope with normal situation.

IV. Experimental Result

In this section, we carry out some experiments to evaluate our proposed method. We first measure the frequency of out-of-sequence 1stDP. Then we evaluate the performance of our method through two contrast tests in the same network environment, one is the comparison of packet loss rate under the same background traffic, and the other is the comparison of processing capacity in the case of specific packet loss rate.

A. Out-of-Sequence 1stDP Evaluation

With the purpose of getting the size of buffer window mentioned in Section III-C, we design this part of experiment. The data set is captured from a certain ISP network, totally 121G. There are nearly 3,000,000 connections in this data sets, the frequency of out-of-sequence 1stDP is listed in Fig. 5. It is clear that from the statistics, 99.05% of the 1stDP have arrived in order, the connection which the 1stDP arrives in first five data packets accounts for about 100%. Therefore, we think that the problem that 1stDP arrived out of order could be solved reasonably if the size of buffer window is set as 5.

B. Performance Evaluation

From a certain ISP network node, we mirror the traffic, which contains attack traffic, and led the same traffic into two systems, the one uses our defense method and the other uses normal method, which is similar to the method described in Section III-B. We set the upper bound of α as 20% and the lower bound as 10% according to the network traffic environment we measured. The size of buffer window is set as 5 according to previous discussion. Additionally, the threshold of Entropy (DIP) is set as 4, which is proved to be effective by AT&T Lab[14].

The packet loss rate of both systems is shown in Fig. 5. Obviously, the packet loss rate of the system which used our proposed method is lower than the system with normal method. Note that, during the test, the event that our detector outputs attacked state occurs 28 times.

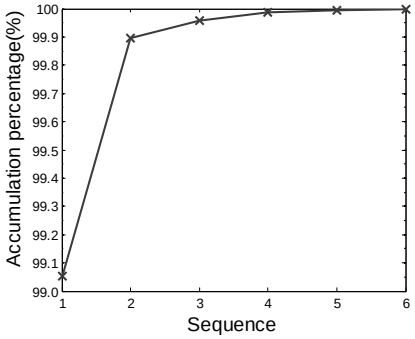

Fig. 4: Frequency of 1stDP out-of-sequence

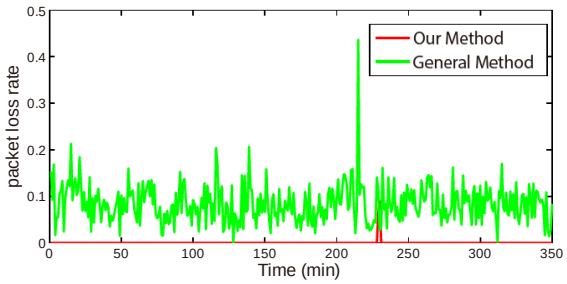

Fig. 5: Packet loss rate

In the last experiment, we inject concurrently real traffic including attack traffic into the two systems, the volume of traffic increased gradually until the packet loss rate reached ten-thousandth. When the packet loss rate reaches one thousandth, the volume of traffic injected to the system with our method is 3.4 Gbps, but that of system with normal method is 0.9 Gbps less than ours.

To sum up, our approach is able to solve the problem that existing approach can't defend against the SYN Flood attack under the asymmetric routing environment, and improve the processing capacity of network security appliance deployed between communicating parties.

V. Conclusion and Future Work

In allusion to the problem that existing method can't effectively protect network infrastructure under the asymmetric routing environment from SYN Flood attack, this paper presented DARE, a defense architecture on basis of statistical attack detection and dual connection management strategy. The main contribution of this paper is that a new idea that connection management without SYN segment. Experimental results show that our proposed method can filter SYN Flood traffic, and mitigate the pressure of network infrastructure.

In the future, we will apply our approach to real network and improve the attack detection method. In addition, we plan to give theoretical evidence for the conjecture that the first packet with payload arrives in the first N packets definitely.

References

[1] "High bandwidth DDoS attacks are now common, researcher says - Computerworld,". 2013.

[2] "Prolexic Quarterly Global DDoS Attack Report Q4 2012," Prolexic Technologies, Hollywood 2013.

[3] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations,". 2007.

[4] D. J. Bernstein, "SYN cookies,".

[5] A. Zúquete, " Improving the Functionality of Syn Cookies," *Advanced Communications and Multimedia Security*, pp. 57-77, 2002.

[6] L. Jonathan, "Resisting SYN flood DoS attacks with a SYN cache," in *Proceeding BSDC'02 Proceedings of the BSD Conference 2002 on BSD Conference*, CA, USA, 2002, pp. 89-97.

[7] Z. Wu and Z. Chen, "A three-layer defense mechanism based on web servers against distributed denial of service attacks," in *Communications and Networking in China, 2006. ChinaCom'06. First International Conference on*, Beijing, 2006, pp. 1-5.

[8] 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 *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, 1997, pp. 208-223.

[9] H. N. Wang, D. L. Zhang and K. G. Shin, "SYN-dog: Sniffing SYN Flooding Sources," in *Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria, 2002, pp. 421 - 428.

[10] T. Nakashima and S. Oshima, "A Detective Method for SYN Flood Attacks," in *Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06)*, Washington, DC, USA, 2006, pp. 48-51.

[11] W. Chen and D. Yeung, "Defending Against TCP SYN Flooding Attacks Under Different Types of IP Spoofing," in *Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies.*, Mauritius, 2006, pp. 38-43.

[12] C. H. Sun, C. C. Hu, Y. Tang, and B. Liu, "More Accurate and Fast SYN Flood Detection," in *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, San Francisco, CA, USA, 2009, pp. 1-6.

[13] H. N. Wang, D. L. Zhang and K. G. Shin, "Detecting SYN flooding attacks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies.*, New York, NY, USA, 2002, pp. 1530-1539.

[14] W. K. Ehrlich, K. Futamura and D. Liu, "An entropy based method to detect spoofed denial of service (DoS) attacks," in *Telecommunications Modeling, Policy, and Technology*. vol. 44 US: Springer US, 2008, pp. 101-122.