

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: taojianxi@nelmail.iie.ac.cn

Abstract—SYN Flood attack is still one of major distributed denial of service attacks. Any network device or computer system with connection state table have the possibility of suffering from this attack. Such attack not only occurs application server but also network infrastructure. 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 a novel SYN Flood defense architecture. It consists of a light-weight detection method and a hierarchical 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 mitigate the influence brought by SYN Flood attack.

I. INTRODUCTION

With the development of streaming media business and the upgrade of user access bandwidth, the volume of Internet traffic is growing rapidly, and the growth rate stays high for long time. Unfortunately, as Internet traffic growth, 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. With the growing of scale of SYN Flood attack, nowadays, the connection state tables of network infrastructure are exhausted easily, they are not adequate to protect against SYN Flood effectively any more [1]–[3].

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 have a new view on SYN Flood attack, and find out a more effective

defense solution. To this end, this paper presents a SYN Flood defense architecture for intermediate network security appliance under asymmetric routing environment. Combined a light-weight detection method with a hierarchical connection management strategy, our architecture can mitigate the influence of 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. These solutions can be divided into two categories, one is that server ensured the authenticity and legality of the SYN request via some assistant mechanisms, such as SYN Cookie, SYN Cache, SYN Proxy and SYN Kill; another is that detect the attack through control bits of TCP segment, then filter the attack stream based detection result.

Although those schemes mentioned before 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 of schemes can only protect end system, which provides some service (web site, email, *etc.*) for others [4]–[9]. 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 connection in both directions [10]–[14]. However, the precondition does not hold under the asymmetric routing environment, they can't apply to the case referred in this paper.

III. DEFENSE ARCHITECTURE

The defense architecture is shown in Fig. 1, traffic acquire module get traffic from network interface and transmits them to traffic dispatcher and 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.

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.

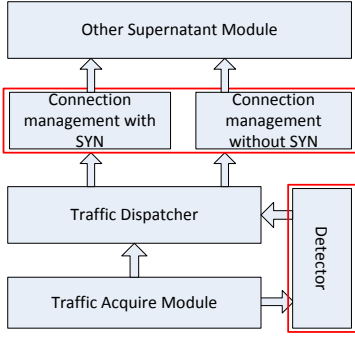


Fig. 1: Defense Architecture

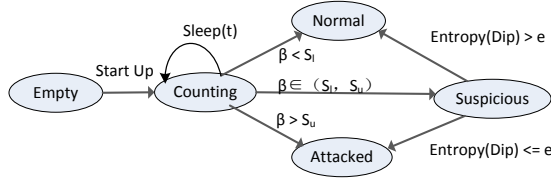


Fig. 2: State Transition Diagram

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;
- 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. 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 β in follows) and destination IP address entropy (expressed as $Entropy(DIP)$ as the event to be detected .

The state transition diagram of integrated detection procedure is shown in Fig. 2. It begins in the *empty* state, which is

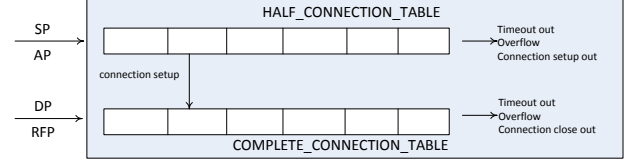


Fig. 3: Connection Management with SYN

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 β , $Entropy(DIP)$, etc. If β is larger than S_u (the upper bound of β), it will transform into *attacked* state. If β is less than S_l (the lower bound of β), it will transform into *normal* state. Otherwise, β is between S_l and S_u , it will transition to *suspicious* state. In that state it will make further judgment that checking the value of $Entropy(DIP)$ whether it exceeds the threshold, which is represented as e in the diagram. If the value of $Entropy(DIP)$ 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 detection procedure repeats as above, and it provides output result for traffic dispatcher continually.

B. Connection Management with SYN

As Fig. 3 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

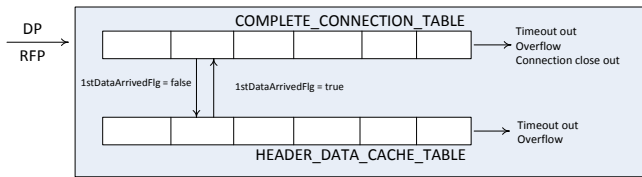


Fig. 4: Connection Management without SYN

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-recv 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.

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 $W = \max\{n_1, n_2, \dots, n_i, n_{i+1}, \dots\}$, then the 1stDPs of all connections must be captured. While $\max\{n_1, n_2, \dots, n_i, n_{i+1}, \dots\}$ 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., $N \geq \max\{n_1, n_2, \dots, n_i, n_{i+1}, \dots\}$. 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. 4, 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

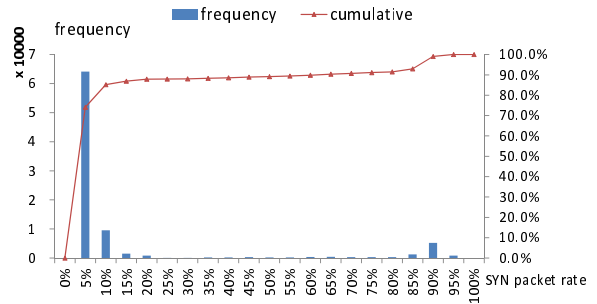


Fig. 5: SYN Packet Rate

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 RESULTS

In this section, we have carried out some experiments to evaluate feasibility and performance of our proposed method. Measuring β and the frequency of out-of-sequence 1stDP is used to provide evidence for the feasibility of this method. Refer to the performance of our method, we evaluate it 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. Feasibility Evaluation

The value of β is the key point of our detection method. We collect statistics from a certain ISP network. The statistical interval is one second, and it start from 2013/3/17 10:23 to 2013/3/18 11:14. In 24 hours, we get approximately 88000 sets of data depicted in Fig. 5. As can be seen from this cumulative histogram, in a day, β is within 5% at about 80% of the time, the time when β is less than 15% takes up more than 90% of the time. As a result, β varies within the range between 5% and 15%. The result is in accordance with Seung-won *et al's* findings [15].

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 set, the frequency of out-of-sequence 1stDP is listed in Table I. It is clear that from the statistics, 99.05328% of the 1stDP have

TABLE I: Frequency of 1stDP Out-of-sequence

Sequence	Connection Number	Percentage	Accumulation percentage
1	2907594	99.05328%	99.05328%
2	24700	0.84146%	99.89473%
3	1894	0.06452%	99.95926%
4	837	0.02851%	99.98777%
5	240	0.00818%	99.99595%
6	118	0.00402%	99.99997%
7	0	0.00000%	99.99997%
> 7	1	0.00003%	100.00000%

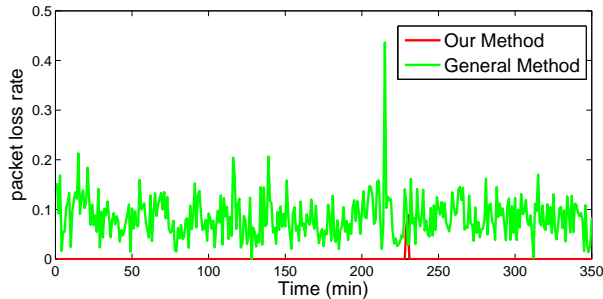


Fig. 6: Packet Loss Rate

arrived in order, the connection which the 1stDP arrives in first six data packets accounts for 99.99997%. 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 6.

B. Performance Evaluation

According the previous experiment, we set the upper bound of β as 15% and the lower bound as 5%, the size of buffer window is set as 6. Additionally, the threshold of $Entropy(DIP)$ is set as 4, which is proved to be effective by AT&T Lab [16].

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. The packet loss rate of both systems are shown in Fig. 6. Obviously, the packet loss rate of the system with 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.

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 ten-thousandth, the volume of traffic injected to the system with our method is 2.6 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 a method on basis of light weight detection and hierarchical 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 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," <http://www.computerworld.com/s/article/9232487>, 2012.
- [2] "Prolexic quarterly global ddos attack report q4 2012," <http://www.prolexic.com/knowledge-center-ddos-attack-report-2012-q4.html>, 2012.
- [3] "Summary of china's internet network security situation 2012," http://www.cert.org.cn/publish/main/46/2013/20130320093925791767941/20130320093925791767941_.html, 2013.
- [4] W. M. Eddy, "Tcp syn flooding attacks and common mitigations," 2007.
- [5] D. J. Bernstein, "Syn cookies," <http://cr.yp.to/syncookies.html>.
- [6] A. Zuquete, "Improving the functionality of syn cookies," in *Advanced Communications and Multimedia Security*. Springer, 2002, pp. 57–77.
- [7] B. Hang, R. Hu, and W. Shi, "An enhanced syn cookie defence method for tcp ddos attack," *Journal of Networks*, vol. 6, no. 8, pp. 1206–1213, 2011.
- [8] J. Lemon *et al.*, "Resisting syn flood dos attacks with a syn cache." in *BSDCon*, vol. 2002, 2002, pp. 89–97.
- [9] 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*. IEEE, 2006, pp. 1–5.
- [10] H. Wang, D. Zhang, and K. G. Shin, "Syn-dog: Sniffing syn flooding sources," in *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*. IEEE, 2002, pp. 421–428.
- [11] T. Nakashima and S. Oshima, "A detective method for syn flood attacks," in *Innovative Computing, Information and Control, 2006. ICIC'06. First International Conference on*, vol. 1. IEEE, 2006, pp. 48–51.
- [12] W. Chen and D.-Y. Yeung, "Defending against tcp syn flooding attacks under different types of ip spoofing," in *Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on*. IEEE, 2006, pp. 38–38.
- [13] C. Sun, 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 International Conference on*. IEEE, 2009, pp. 1–6.
- [14] H. Wang, D. Zhang, and K. G. Shin, "Detecting syn flooding attacks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2002, pp. 1530–1539.
- [15] S.-w. Shin, K.-y. Kim, and J.-s. Jang, "Analysis of tcp syn traffic: an empirical study," in *Advanced Communication Technology, 2005, ICACT 2005. The 7th International Conference on*, vol. 1. IEEE, 2005, pp. 652–657.
- [16] 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*. Springer, 2008, pp. 101–122.