Differentiating Malicious DDoS Attack Traffic from Normal TCP Flows by Proactive Tests

Zhiqiang Gao, *Member, IEEE,* and Nirwan Ansari, *Senior Member, IEEE*

***Abstract*— To defend against distributed denial of service (DDoS) attacks, one critical issue is to effectively isolate the attack traffic from the normal ones. A novel DDoS defense scheme based on TCP is hereby contrived because TCP is the dominant traffic for both the normal and lethal flows in the Internet. Unlike most of the previous DDoS defense schemes that are passive in nature, the proposal uses proactive tests to identify and isolate the malicious traffic. Simulation results validate the effectiveness of our proposed scheme.**

***Index Terms*— DDoS defense, proactive test, TCP.**

I. Introduction

DISTRIBUTED denial of service (DDoS) attacks are probably the most ferocious threats to the integrity of the Internet. It is well known that it is rather easy to launch, but difficult to defend against, a DDoS attack. The underlying reasons include (1) IP spoofing; (2) the distributed nature of the DDoS attack (a huge number of sources generate attack traffic simultaneously); (3) no simple mechanism for the victim to distinguish the normal packets from the lethal traffic.

Over the years, many DDoS defense schemes have been proposed. Mutaf [1] proposed a real-time anomaly detection scheme to identify TCP SYN flood attacks by analyzing daily maximum arrival rate. Similar work was done by Haggerty *et al.* [2]. The above two schemes were designed for SYN flood attacks, which occurred before connection establishment while our effort is to identify malicious TCP flows after successful TCP connections. Xu *et al.* [3] proposed to isolate malicious traffic via HTTP redirect messages. Since most of the attack flows employ spoofed source IP addresses, their sources cannot receive the redirect messages, and thus the subsequent packets from them will be blocked. This scheme is simple and may be readily implemented. However, their mechanism works only for web servers. Yau *et al.* [4] presented a QoS based DDoS defense scheme. Their scheme tries to maintain fair share among all competing flows by using the fair packet queueing technique. However, simply using fair sharing without traffic discrimination does not work. Consider the following two scenarios. 1) The high rate traffic is legitimate while the attack traffic is low-rate [5]. Rate-limiting is improper in this case. 2) Most flows carry attack traffic during a flood-based DDoS attack while good traffic is low-rate. Under this scenario, even imposing fair sharing of bandwidth does not help the good traffic much because the majority of bandwidth is consumed by malicious flows. To ensure good performance and accommodate as many normal users as possible, it is critical to differentiate traffic. However, it is by no means trivial to make such a distinction. Discrimination based on packet headers is vulnerable to IP spoofing; discrimination based on packet contents may be thwarted by the increasing use of end-to-end encryption.

We hereby propose to identify malicious traffic from their behaviors. We believe that aggressiveness is the salient feature of DDoS traffic, besides IP spoofing. One example of the aggressive behavior is that an attack source may not care about whether it may receive the response from the victim or not, and it can still conduct an attack by bombarding its target with a monstrous number of useless packets. Note that "aggressiveness" is not equivalent to "high-rate". It is possible that a high-rate flow is a normal TCP stream. The receiver may identify the aggressive behavior by intentionally testing the response of a source upon certain control signals from the receiver. Any source that fails to pass such tests is regarded as a lethal one and can be punished accordingly. However, a source, which passes the test, may not be necessarily benign. A sophisticated attacker may pass the test by behaving well initially, and perform deleterious operations later. To handle this case, the receiver may increase the frequency of such tests. A better solution is to introduce some dynamics into the test and randomly determine the frequency of the test for each flow, especially the high-rate ones. To accommodate high-rate legitimate traffic better, we set a threshold that defines the maximal number of successful tests for a flow. No more tests are conducted on packets from a flow once the flow successfully passes the specified number of tests. By actively testing a source, the receiver can determine with high confidence the nature of a flow from that source and react accordingly. Filtering based on behavior brings an attack source into a dilemma: sends packet aggressively at the risk of being identified and punished, or reduce the attack rate to meet the requirements of the receiver so that the effect of an attack is diminished. In so doing, the receiver may throttle the scope and impact of potential attacks.

The above design is feasible for TCP solely because TCP has the built-in congestion control and reliable transmission mechanism. Note that TCP is the dominant traffic in the Internet, and as much as 90% of DDoS traffic uses TCP [6]. Currently, TCP occupies 80% in terms of the number of flows,

Manuscript received May 1, 2006. The associate editor coordinating the review of this letter and approving it for publication was Dr. Chuan-Kun Wu. This work was supported in part by the New Jersey Commission on Science and Technology via the NJ Center for Wireless Networks and Internet Security.

Z. Gao and N. Ansari are with the Advanced Networking Laboratory, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: {zg4, nirwan.ansari}@njit.edu).

Digital Object Identifier 10.1109/LCOMM.2006.060669.

1089-7798/06$20.00 © 2006 IEEE

and 90% with respect to the number of packets. It is thus essential for DDoS defense schemes to accommodate TCP traffic effectively and efficiently.

The rest of the letter is structured as follows. Section II presents in detail our proposal. We then present in Section III some simulation results to validate our scheme. Finally, conclusion is presented in Section IV.

II. TCP Flow Differentiation

A. Connection Establishment

Whether a connection has been established has a significant implication to the receiver. A successfully established connection indicates that both ends have completed the three-way handshaking procedure, which implies that IP spoofing is not employed by the source. For an incomplete connection, on the other hand, the receiver shall be alert, and be conservative in its resource consumption. Possible measures to mitigate potential attacks include (1) tightening the total bandwidth allocated to all incomplete connections, and (2) significantly reducing the timeout value to avoid buffer occupied by half-open connections for a long time, or no buffer allocation at all for half-open connections.

B. Benign and Malicious Flows

TCP is an end-to-end solution that requires close orchestration between the sender and the receiver. To characterize the nature of a TCP flow (after a successful connection), the receiver can actively test the response of the sender by delaying the ACK packets intentionally. If the sender is normal, it will take action accordingly and reduce its sending rate. On the contrary, for a DDoS attack, two cases may occur. One is that the sender uses forged source IP addresses, and thus cannot receive the rate-reduction message from the receiver. It has no idea of the proper sending rate. The other scenario is that the sender does receive the notification, but it neglects it and just keeps sending packets, thus violating the protocol, and it may be punished by the recipient to reduce its share or even block its traffic. This procedure is dynamic. The protected site can decide the frequency and extent of rate-reduction so that no perpetrator can easily fool the system to believe that the traffic from the perpetrator is normal. Fig. 1 depicts the flowchart of the traffic differentiation procedure.

Upon the arrival of a new incoming packet, the receiver first determines to which flow the current packet belongs by checking the tuple of (source IP address, source port number, destination IP address, destination port number). If it is the first packet of a flow, the receiver examines whether the number of admitted flows reaches the maximum flow count, a threshold set by the receiver to ensure proper provisioning of quality of service. If this does occur, the packet is dropped. Otherwise, the new packet is admitted after updating the flow table maintained by the receiver, resulting in an increment of the flow count by 1, and initialization of several counters, such as the number of successful tests and the number of failure tests. The receiver then checks the behavior history of the flow. If the number of failure tests is no less than a threshold, f, the packet will be dropped. An integer larger than 1 is selected to prevent our scheme from falsely identifying the behavior

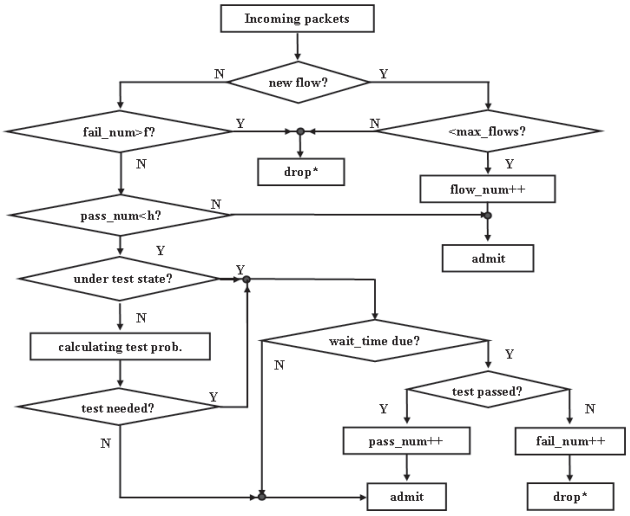

Fig. 1. Flowchart of the traffic differentiation.

of a flow. A low value of f may exacerbate packet dropping. In case of a false identification, subsequent packets from an innocent flow will be blocked. Selecting a too high value is unwise, either. A high f delays the packet dropping decision, and thus subsequent packets of a malicious stream may still consume system resources. Through extensive simulations, we found that f=3 provides a good balance between the proper identification rate and the acceptable performance impact.

For the flow whose behavior is not so bad in the past, our scheme further examines whether the flow has passed a certain number of tests, h. The receiver will admit directly any packet of flows having passed h tests successfully (Similarly, some tradeoff has to be made to determine a proper value of h. We set h to 6 by trials and errors). For other flows, we further check the current state of the flow. If the flow is under a test, its current rate shall not exceed one half of its previous one (the receiver enforces this constraint by manipulating the reverse ACK rate). If the flow conforms to that constraint, the flow passes the current test and its $pass\_num$ is incremented by 1. Otherwise, the flow fails one test. In the case that the flow is not in the state of testing, its sending rate is compared with that of the fair share of each flow. The result of the comparison is used to determine the test probability for that flow. Obviously, a flow with less bandwidth consumption is subject to less number of tests. The test probability p for a high-rate flow (over the fair share) is 1/($pass\_num$+1). At the very beginning, $pass\_num$ is 0 for all flows. Therefore, as long as a high-rate flow has not passed a test, its chance of being tested is 100%. As the number of successful tests of a flow increases, its test probability reduces. The test probability p for the less resource-consumption flow is 1/max(m, 2*h), where m is the total number of flows. For the normal case, m is far greater than $2h$; thus, p=1/m. We use the max(.) function to address the case that only a few flows exist in the system and ensure that the test probability for a low-rate flow is at most 1/2 of that of a high-rate one.

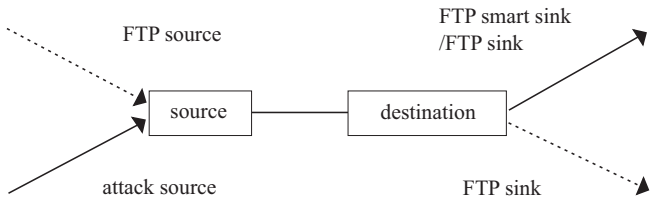

Fig. 2. Simulation setup for comparison study of the effectiveness of traffic differentiation.

The rate of a flow is calculated according to the following formula, num_pkt*sz_pkt/t, where t is the time interval (window), num_pkt the number of packets received during this period, and sz_pkt the packet size. It is worth mentioning that the flow rate calculated here is not the average rate of a flow, as normally used by others, because we update the starting time of a flow once it passes a test. In so doing, we can effectively thwart a low-rate DoS attack which sends a burst of attack packets to incite congestion and keeps silence for a much longer period to significantly lower its average rate in order to escape detection and filtering.

Four scenarios may happen. 1) An attack source always behaves well, and thus the effect of an attack is greatly diminished. 2) An attack source behaves well initially and misbehaves later. When tested, the constraint that the current rate is at most 1/2 of the previous rate will not be satisfied, and the source fails the test. 3) An attack source always misbehaves, that may be easily thwarted by the fail count. 4) An attack source misbehaves at first and behaves well later. In this case, the attack source is exposed to more chances of being tested because its $pass\_num$ is offsetted by the $fail\_num$ once it fails a test. Note also that a low-rate flow is also subject to test, though at a lower probability in our design. As time passes by, the chance that a low-rate flow has never been tested by the receiver is very low. We enforce this policy to contain the case that some low-rate streams are malicious.

III. Simulations

To test the effectiveness of our proposed traffic differentiation, we set up a simulation scenario including 1 FTP source and an attack source, as shown in Fig. 2. These flows pass through the same bottleneck link. The difference is that one simulation uses a normal FTP sink to accept packets from both flows, and the other uses our developed TCP sink, called TCP smart sink. The simulation results are shown in Fig. 3 and Fig. 4.

Fig. 3 shows the throughput of the attack traffic using the FTP sink while Fig. 4 presents the throughput of the attack traffic using our proposed TCP smart sink, in which the throughput of attack traffic drops drastically after 3.2s. After 42.3s, the attack traffic is totally blocked. In contrast, using the FTP sink as the receiver, the attacker may keep the highest throughput during its lifetime. The result demonstrates the effectiveness of our proposed traffic differentiation.

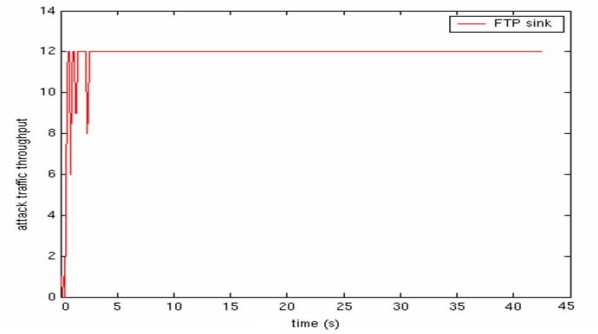

Fig. 3. Attack traffic throughput using FTP Sink

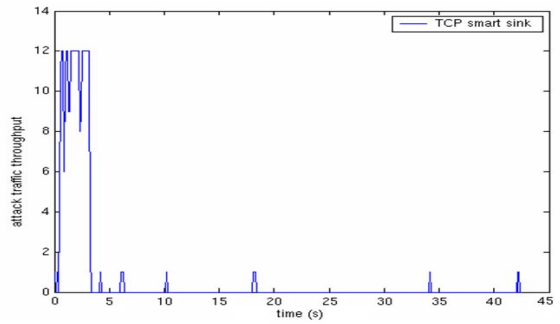

Fig. 4. Attack traffic throughput using TCP Smart Sink

IV. Conclusions

A novel DDoS defense scheme has been presented in this letter. The salient benefits of our proposal mainly lie in its capability of identifying malicious TCP flows by proactive tests. Preliminary simulation results have validated our design.

References

[1] P. Mutaf, "Defending against a denial of service attack on TCP," in *Proc. International Symposium on Recent Advances in Intrusion Detection (RAID 99)*.

[2] J. Haggerty, T. Berry, Q. Shi, and M. Merabti, "DiDDeM: a system for early detection of TCP SYN flood attacks," in *Proc. IEEE GLOBECOM 2004*, pp. 2037-2042.

[3] J. Xu and W. Lee, "Sustaining availability of Web services under distributed denial of service attacks," *IEEE Trans. Comp.*, special issue on reliable distributed systems, vol. 52, no. 2, pp. 195-208, Feb. 2003.

[4] D. Yau, J. Lui, F. Liang, and Y. Yan, "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles," *IEEE/ACM Trans. Networking*, vol. 13, no. 1, pp. 29-41, Feb. 2005.

[5] A. Kuzmanovic and E. Knightly, "Low-rate TCP-targeted denial of service attacks (the shrew vs. the mice and elephants)," in *Proc. ACM SIGCOMM 2003*, pp. 75-86.

[6] D. Moore, G. Voelker, and S. Savage, "Inferring Internet denial-of-service activity," in *Proc. 10th USENIX Security Symposium*, pp. 9-22.