International Journal of Advance Research in Science and Engineering
Vol. No.4, Special Issue (01), September 2015
www.ijarse.com
IJARSE
ISSN 2319 - 8354

IMPROVING THROUGHPUT DURING DENIAL OF SERVICE ATTACK BY USING INDIRECT TRUST IN RANDOM EARLY DETECTION

Reshma S[1], Sminesh C N[2]

[1,2]*Dept. of CSE, GEC, Thrissur, Kerala, (India)*

ABSTRACT

Most of the traffic in Internet is possible through packet switching technology; Random Early Detection (RED) algorithm is widely used for queue management in packet switching networks. RED algorithm perform well in normal scenario, however the performance degrades under an attack scenario like Denial of Service (DoS). The immediate effects of DoS attack on RED are decrease in throughput and increase in delay. A method, that use network flows trust to mitigate the effect of DoS attack in RED, tries to address this problem}. Trust is a numerical value indicating the trustworthiness of a node, lower the trust value greater is the possibility that it is a malicious flow. The result of this method is not considerably good since it uses only direct evidences in trust calculation. We propose a method that improves the trust computation used in the existing method by considering indirect evidences along with direct evidences. The proposed method computes indirect trust from indirect evidences, which is the direct trust provided by the neighboring nodes. The refined trust value computed from direct and indirect trust is used by the proposed system to calculate the packet drop probability in RED. As a result the proposed system provides a better throughput than the existing system.

Keywords: Networks Security, Performance, Denial of Service, Random Early Detection

I. INTRODUCTION

In packet switching networks a queue is an important resource and there are a number of algorithms for queue management. In a normal network scenario every flow fairly content for the bottle neck queue. However in an attack scenario, like a Denial of Service attack, a malicious flow would fill up the bottle neck queue. The goals of network-layer DoS attacks include the consumption of networks resources and to deny other users from legitimate access to systems information and services [1]. In packet switching networks there are many efficient queue management algorithms. By making these algorithms more robust we can reduce the effects of DoS attack.

Random Early Detection (RED) is one of the popular queue management algorithms. Unlike other algorithms of similar kind, RED uses the average queue size to decide whether to enqueue or drop a packet. The major goals of RED are congestion avoidance, avoidance of bias against bursty traffic and avoiding global synchronization. RED helps to keep the average size of the queue low while allowing occasional burst of packets [2].

The RED algorithm employs two threshold values to control the growth of the queue. On arrival of a packet it evaluates the average queue size against the minimum threshold and the maximum threshold. If the average

queue size is below the minimum threshold, the arrived packet is enqueued. If the average queue size is above the minimum threshold, the packet would be dropped with a probability. The probability that the packet will be dropped is calculated based on how close the average queue size is to the maximum threshold. The average queue length is calculated using Exponential Weighted Moving Average. Many modifications in RED algorithm were proposed over the years [3] [4].

Here we propose a method using flows trust value to mitigate the effects of DoS attacks by modifying RED. It employs the concept of trust to safeguard legitimate flows. Each node monitors network flows and calculate flow's trust value, which is used for the relevant queue management. Legitimate flows will have higher trust values and malicious flows will have lower trust value [5]. This improves the throughput in DoS attacking scenarios compared to existing queue management algorithm RED.

In Random Early Detection, when the average queue size exceeds a pre-set threshold then packets are dropped with a certain probability. Trust value computations can be used to decide which packets are to be dropped [6]. The higher the trust value, lower is the probability to drop that packet. In the proposed method we use a refined trust computation method in which both direct trust and indirect trust are used. The direct trust is computed by a node from the characteristics of the flow. The direct trust computed by the nodes are shared with their neighbor nodes. The direct trust values from neighbors are used as indirect evidence by each node. The indirect trust is computed from these indirect evidences. The aggregate of direct and indirect trust values is used in calculating the probability of dropping a packet. Thus malicious flows with low trust value will be penalized.

The rest of the paper is organized as follows. Section 2 presents some of the works done in the area of DoS detection. Section 3 describes the design of our proposed system. In Section 4, we discuss an implementation model of the system. We conclude the paper in Section 5.

II. RELATED WORK

An efficient DoS detection technique should be included into RED to make it robust. A detection method using randomization of minimum RTO counters LDoS attack [7]. Low-rate TCP targeted denial of service attacks are a subset of DoS attacks that exploit the retransmission timeout (RTO) mechanism of TCP. Such an attack can drastically reduce throughput while producing little evidence of its presence relative to traditional DoS attacks. The Low Rate Denial of Service (LDoS) attack exploits TCPs slow time scale dynamics of retransmission time-out (RTO) mechanisms to reduce TCP throughput. Basically, an attacker can cause a TCP flow to repeatedly enter a Retransmission Time Out (RTO) state by sending high-rate (R_b), but short duration bursts (T_b), and repeating periodically at slower RTO time-scales (T_a) as shown in fig 1. The TCP throughput at the attacked node will be significantly reduced while the attacker will have low average rate making it difficult to be detected. The variation in queue length during an LDoS detection is as shown in fig 2.

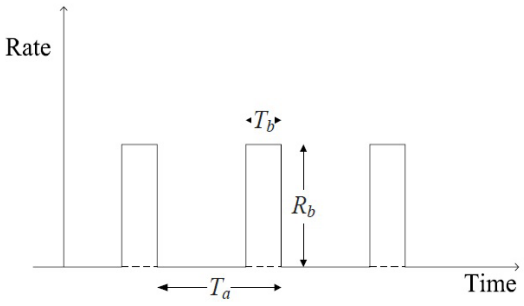

Figure 1 LDoS Stream[8]

A defense against this attack is made possible by randomizing the RTO. In doing so, information can still be transmitted while the attacker is waiting and a connection will be able to avoid timing out successively. If the minimum value of the RTO is not set to 1sec, but instead it is randomized around 1sec, the effect of the attack on the throughput is less severe. When the value of minimum RTO is randomized around 1sec, it is more difficult for the attacker to synchronize its square wave attack with the RTO expiration intervals. Thus the attacker can no longer predict the time the new packets will arrive. Randomizing the fixed minimum RTO will reduce the TCP connection performance in the absence of an attack. An unnecessary TCP timeout results in loss of useful throughput, and TCP begins a new slow start. TCP will also require a long time to adapt its RTT estimate after every timeout. The fixed minimum RTO of one second was selected because it eliminated unnecessary timeouts. Apart from these methods there are anomaly based DoS detection methods that make use of a counter to keep track of anomalous behavior [9].

Another detection method is based on burst length and time period [10]. A periodic stream in which the burst length is greater than or equal to RTTs of other connections with the same server, and the time period is equal to the fixed minimum RTO is considered. An attack flow satisfying these two important conditions can cause denial of service to other TCP flows. The flow objects maintain the arrival times of packets at the edge router in the pseudo transport layer. The malicious flow detection sub module computes the time difference of consecutive packets of each flow. The sub- module computes the average of the time difference values. The average high value of the time difference repeats periodically for the attack flow; other flows do not exhibit this property. The malicious flow detection sub module then estimates the burst length of a flow based on the packet arrival times, and compares it with the RTTs of other flows connected to the same server. The relative RTT of each connection is computed by the sub module from the packet arrival times maintained for both source and destination side packets. In a similar way, the time period is estimated, and compared with the fixed minimum RTO of one second. A flow meeting the above two conditions is marked as malicious.

In RED-FT each flow behaviors, such as the data rate and packets loss are monitored to check for anomalous behavior. Secondly, the collected data is quantified and the trust values are calculated. Furthermore, three strategies are selected according to the networks flow trust: a network flow is trusted and the RED operation is carried out, a network flow is distrusted and packets are dropped or a network flow is probabilistically trusted and the RED operation is probabilistically carried out. Finally, the execution result of the RED operation is returned to the monitoring module.

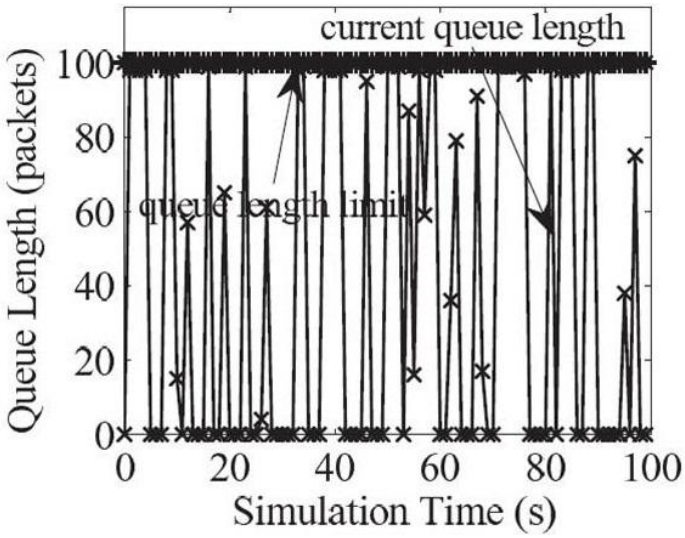

Figure 2 Queue Length in LDoS Attack Flow [11]

III. PROPOSED SYSTEM DESIGN

Our proposed system has a Network Traffic Monitoring Module, Trust Computation Module and Random Early Detection Module. The Network Traffic Monitoring Module receives packets and the traffic characteristics are measured by this module. For Example calculation of Average traffic over time a period is to be done in this module. It sends the detected data to the Trust Computation Module. The Network Traffic Monitoring Module also sends packets to the RED Module (REDM). The Trust Computation Module consists of Direct Trust Computation Module, Indirect Trust Computation Module and Trust Aggregation module. The Direct Trust Computation Module computes the direct trust according to the received data from Network Traffic Monitoring Module. Indirect Trust Computation Module receives values from neighboring nodes and computes the indirect trust. Trust aggregation Module computes the final trust value. This trust value is passed to the Random Early Detection Module. The REDM executes standard RED operations according to the current queue length and the trust value. Only direct evidences are used to calculate trust in the previous method. The trust calculation can be refined by calculated is to be considered in computing net trust value.

Indirect trust can be derived from indirect evidences. Trust values computed by neighboring nodes can be treated as indirect evidence. The nodes send the trust values it computed from direct evidences to the neighbors on demand. The weight given to the indirect evidence received is determined by the trust assigned to the sender of the evidence. Fig 3 represents the overall design of the system.

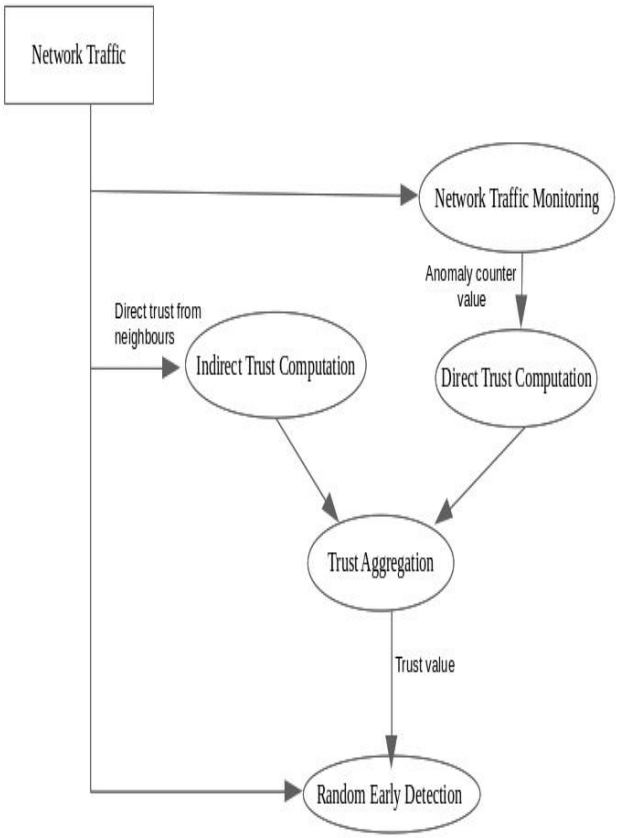

Figure 3 System Design

IV. TRUST COMPUTATION MODEL

For detection of LDoS attack, consider An as the arrival time of the nth packet dropped. Check if $A_n - A_{n-1} = t^*$ where t* indicates the period of packet drop. We use S to keep track of the number of intervals which show anomalous behavior:

$$S = \sum_{i=n-k+1}^{n} 1_{A_i - A_{i-1} = t^*}$$

T, the trust value corresponding to the LDoS, is calculated as:

$$T_d = \begin{cases} 0 \; if \; S \geq k \\ 1 \; if \; S < l \\ \dfrac{S-l}{k-l} \; if \; k > S > l \end{cases}$$

Indirect trust is calculated from the weighted average of evidences provided by neighbors. The evidences are the direct trust values calculated at the neighbors:

$$T_i = \frac{T_1 E_1 + T_2 E_2 + \cdots T_n E_n}{n}$$

Where E_1, E_2, . . .E_n are evidences from neighbor 1, neighbor 2, . . neighbor n and T_1, T_2, ..T_n are their corresponding trust values. For example if node A receives an e evidence from node B. Node Bs trust value computed in A is t. then the weight of the evidence e in the calculations in A is proportional to t. The direct and indirect trust values are then aggregated:

$$T_a = \lambda T_d + (1 - \lambda) T_i$$

Where T_a is the aggregate trust, T_d direct trust, T_i indirect trust, λ is the weight assigned to direct trust. The value of λ is between 0 and 1.The aggregated trust value is then passed to Random Early Detection module. When packets are to be dropped from nodes of lower trust value are dropped.

Since our method uses evidences from neighbors, the malicious flows could be identified even before the node experience a sever attack. Malicious flows can be identified more accurately in our method due to the use of refined trust computation. Hence they are penalized more and the throughput of legitimate users is improved. The delay incurred in retransmitting the packets lost due to attack is reduced, since the packet drop is mostly from malicious flows.

V. SYSTEM SIMULATION

The proposed system implementation can be divided in to mainly three stages. First stage consists of setting up a simple wired network. After that attack is introduced to the network created. As a final stage we implement the indirect trust based algorithm to mitigate the attack.

We create a network with the configuration as shown in Table 1 with five attacker nodes. The bottleneck link has a bandwidth of 5 Mbps and a delay of 6 ms. All other links has a bandwidth of 10 Mbps and delay of 2 ms. The queue size is set to 50. The attacker sends UDP packets of size 50 bytes for a burst period of 200 ms. Then the attacker becomes idle. This is repeated periodically in every 1000 ms. The trace file records all the events during the simulation. The average throughput and end-to-end delay of the system is calculated by using awk script to analyze the trace file.

Table 1: Simulation Parameters

Parameter	Value
Bottleneck bandwidth	5 Mbps
Bottleneck delay	6 ms
queue size	50
Number of attackers	5
Attack period	1000 ms
Burst period	200 ms

The proposed method is implemented by creating a new queue type which inherits the red class. An array is used to store the anomaly counter for each flow. The dropAnomaly procedure checks for anomalous behavior and update the counter. The trust computed from this is used by the calculate_p procedure to calculate the drop probability. It is based on this probability that packets are dropped.

A new variable is added to packet to send trust value to neighbors. The trust value is set using the setTrust procedure and the trust value is received using the getTrust procedure.

VI. EVALUATION

Evaluate of the performance of the system is done by analyzing the trace file generated during the simulation in ns2 [12]. Trace file stores the information about each event in the simulation. Here the performance is evaluated based on two metrics, average throughput and average end to end delay. Average throughput is the average amount of data delivered in unit time. It is represented in Kbps. Throughput is calculated using the formula:

$$\text{AverageThroughput} = \frac{\sum \text{data bits received}}{\text{Total simulation time}}$$

The time taken by a packet to reach the destination is called end to end delay. The average end to end delay calculates the average time taken by a packet to reach destination over all the packets transmitted. It is calculated using the formula:

$$Avg\ e\ to\ e\ delay = \frac{\sum Endtime\,(packet) - Start\ time\,(packet)}{Total\ number\ of\ packets}$$

VII. OBSERVATIONS AND ANALYSIS

The performance of the system under attack scenario is analyzed in terms of average throughput and average end to end delay. Each measured for the same topology, varying parameters like attack rate, weight assign to direct trust and indirect trust, and weight assigned to trust value in calculating the drop probability.

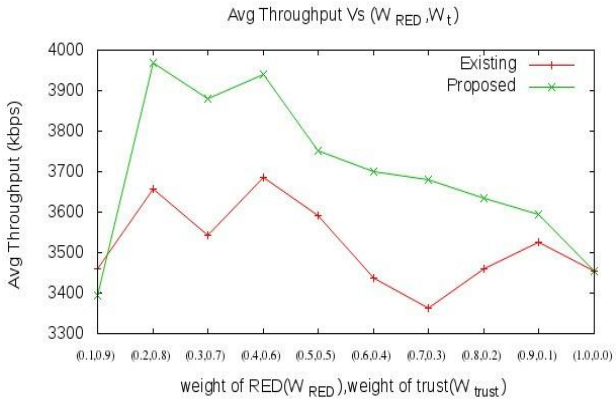

Figure 1: Variation of Throughput with (W_{RED},W_{trust})

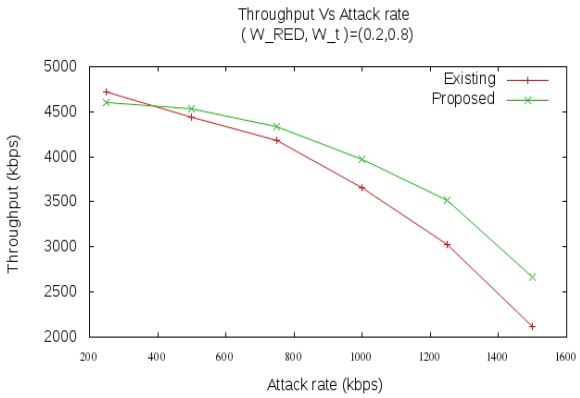

Figure 2: Variation of Throughput with Attack Rate

By analyzing the above simulation results, as shown in fig 4, we can see a better throughput than the existing

system is observed then the weight of trust in drop probability calculation is between 0.1 and 0.8. From fig 5 we can observe that the system has better throughput than existing system for attack rate of 500 kbps and above. A marginal increase in delay is incurred in the proposed system due to overhead in computing the indirect trust. Fig 6 shows the graph of average end to end delay against attack rate. From all these we can conclude that the proposed system performance better in terms of throughput for high attack rates. In future we will vary the DoS detection method used in the proposed system and study its effects on the end to end delay.

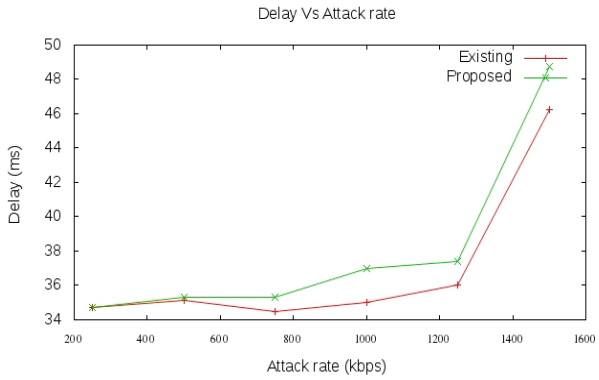

Figure 3: Variation of Delay with Attack Rate

VIII. CONCLUSION

DoS attack is a great threat to network security and has a significant impact on the throughput of the network. The effects of DoS attack in the network can be mitigated by making the queue management algorithms, such as RED, more robust. This will ensure that the communication of legitimate users is preserved as much as possible. The proposed method refines the trust computation used for mitigating the effects of DoS attacks in Random Early Detection. This improves the performance of RED in terms of throughput. Since the proposed method uses evidences from neighbors, the malicious flows could be identified even before the node experience a severe attack. The effects due to malicious flows are reduced in the proposed method due to the use of refined trust computation. They are penalized more and the throughput of legitimate users is improved. The proposed system's performance was evaluated based on average throughput and end to end delay. The effect of varying parameters was evaluated. Maximum throughput is obtained when direct and indirect trust is given equal weight.

REFERENCES

[1] P. Tao, L. Christopher, and R. Kotagiri, “Survey of network-based defense mechanisms countering the DoS and DDoS problems,” ACM Computing Surveys, vol. 39, no. 1, pp. 1–42, 2007

[2] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance”, IEEE/ACM Transactions Networking, vol. 1, no. 4, pp. 397–413, 1993.

[3] Changwang Zhang, Jianping Yin, Zhiping Cai, and Weifeng Chen, "RRED: Robust RED Algorithm to

Counter Low-RateDenial-of-Service Attacks", IEEE Communications Letters, vol.14, no. 5, pp 489-491, 2010.

[4] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: an algorithm for increasing the robustness of RED's active queue management", AT&T Center for Internet Research at ICSI, 2001. Available: http://www.icir.org/floyd/papers/adaptiveRed.pdf.

[5] Xianliang Jiang, Jiangang Yang, Guang Jin, and Wei Wei, "RED-FT: A Scalable Random Early Detection Scheme with Flow Trust against DoS Attacks", IEEE Communication Letters, vol. 17, no. 5, 2013.

[6] Ningning Cheng, Kannan Govindan, Prasant Mohapatra, "Rendezvous Based Trust Propagation to Enhance Distributed Network Security", International Journal of Signal and Imaging Systems Engineering, vol. 1, no. 3, 2011.

[7] G. Yang, M. Gerla, and M. Y. Sanadidi," Randomization: Defense against Low-Rate TCP-targeted Denial-of-Service Attacks", in Proc. IEEE Symposium on Computers and Communications, 2004, pp. 345-350.

[8] Changwang Zhang , Zhiping Cai , Weifeng Chen, Xiapu Luo d , Jianping Yin ,"Flow level detection and filtering of low-rate DDoS", Computer Networks 56, pp 3417–3431, 2012

[9] Vasilios A. Siris, Fotini Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks", Computer Communications, vol 29, pp 1433–1442, 2006.

[10] Amey Shevtekar, Karunakar Anantharam, and Nirwan Ansari, "Low Rate TCP Denial-of-Service Attack Detection at Edge Routers", IEEE Communication Letters, Vol. 9, No. 4,pp 363-365,2003.

[11] Aleksandar Kuzmanovic and Edward W. Knightly,"Low-Rate TCP-Targeted Denial of Service Attacks", SIGCOMM'03,2003, Germany.

[12] The Network Simulator version 2,Available: http://www.isi.edu/nsnam/ns