Journal of Engineering and Applied Sciences 12 (11): 3002-3007, 2017
ISSN: 1816-949X

Threshold Based Active Queue Management (TBAQM) for Alleviating DoS/Flooding Attacks

Amogh Venkatanarayan, Inder Mohan, Mainul Hasan, Ninni Singh and Gunjan Chhabra
School of Computer Science Engineering, College of Engineering Studies,
University of Petroleum and Energy Studies, Dehradun, Uttarakhand, India

Abstract: Denial of Service (DoS) is one of the most effective and strong attacks in cyberspace. They tend to be frequent due to which optimization is being looked at. Over time various methods have been developed and proposed to reduce this problem but with the advent of high power and high ability devices the strengths of these attacks have also increased. In such cases, it is highly imperative that a mechanism be introduced in a manner that doesn't consume too many resources by itself and yet provides similar means of security. Flooding attacks are a hindrance to the most basic of security principles, the CIA triad. Availability of resource is highly impaired in such attacks and can be seen as a cause of distress to quite a few. This study presents a threshold based queue management system which creates a virtual upper bound on the resource allocation so that the entirety of the resources is never exhausted. Thereby, giving adequate reaction and response time to the high influx of packets. Proper recognition of malicious packets will result in proper marking of the sender, thereby helping the decision of dropping of packets. Furthermore, appropriate studies using a modular simulator have been made and the results show the efficiency of the proposed method in the aforementioned scenarios.

Key words: Distributed denial of service, queue management, flood attacks, CIA, traid

INTRODUCTION

Denial of Service (DoS) is a cyber-attack which is on the rise these days. It renders the victim's system useless, thereby affecting its availability. It is commonly performed by overwhelming the victim's system by sending a huge number of IP packets. Another manner of performing DoS is to get the victim's system to perform a heavy-duty task, thereby rendering it useless for anyone else. Flooding based DoS attack is most commonly performed on the transport layer and the application layer. Since, the transport layer is responsible for establishing the communication channel between two devices, it is more rewarding for an attacker to attack on this layer.

Multiple solutions have been proposed over time to solve this problem. Modifications of queuing algorithms (Lau *et al*., 2000) for the sake of better bandwidth utilization so that all packets reach the server. Changes to the network by adding a firewall (Safa *et al*., 2008) and an internal De-Militarized Zone (DMZ) so that the malicious traffic does not reach the main system. Implementing various Active Queue Management (AQM) strategies ensure that the memory and processor utilization is better (Anelli *et al*., 2014; Bedi and Shiva, 2014 Maurizio *et al*., 2015). However, these solutions have drawbacks like need for large memory, higher processing power, etc.

This study proposes a real-time detection method for DoS attacks. Making changes to the AQM and implementing a threshold based activation of the detection system. Using selective information from the incoming packets memory is conserved. Most of the current methods depend on continuous assessments which eat into memory and processing capacity.

Literature review: Denial of Service (DoS) is possible on many layers of the network. Use of TCP/IP and UDP packets are however the most commonly used for performing such an attack. There are two major types of DoS attacks that are prevalent, flood DoS and low-rate DoS. The flooding based DoS attack relies on the multitude of packets that are sent. The low-rate DoS attack exploits the homogeneity of the minimum Retransmission Output (RTO) of TCP flows and causes link saturation attack.

An internet firewall having some local and some global level firewalls could be implemented to stop an attempt of DoS before the malicious traffic reaches the victim's network however, this would not work if the attack is happening from inside the same network using remote access (Chang, 2002).

Corresponding Author: Amogh Venkatanarayan, School of Computer Science Engineering, College of Engineering Studies, University of Petroleum and Energy Studies, Dehradun, Uttarakhand, India

One method to avoid DoS on the main resource would be to filter all ingress packets outside the network, using a router neither on attacker's network nor victim's, to produce Address Resolution Protocol (ARP) request to victim and attacker to ascertain genuineness of the victim (Safa *et al*., 2008). A particle swarm optimization approach, using half-open connections and number of attacks in a given period of time as parameters has also been suggested (Jamali and Shaker, 2012). However, these methods rely on additional hardware thus increasing the complexity of managing the situation.

Behavioral analysis of network traffic gives an idea whether flood attack is happening or not (Noh *et al*., 2008). During a flood attack trace-back gives the source and Pi's marking scheme gives the overlapping of packets and gives a more holistic idea of the flood (Li and Shen, 2008). Automata based re-allocation of resources is also proposed, to minimize the impact of the attack on the legitimate user (Fradet and Ha, 2010). This is another process by which a flood based DoS attack can be mitigated.

Active Queue Management (AQM) is a common method to address requests. FavorQueue suggests that certain connections which have already established and are in the active queue be given temporary priority (Anelli *et al*., 2014) which in case of small time-to-live could cause congestion and denial. An approach involving captcha based verification on application layer and MAC filtration and cryptography based authentication is proposed but this method requires higher memory even during normal conditions. An enhanced AQM using a smaller buffer is suggested (Parakash *et al*., 2016).

MATERIALS AND METHODS

Flooding based DoS attacks are dangerous because its longevity can be increased by escalating continuously after compromise of victim system. There are various kinds of flood based DoS attacks, namely, ICMP flood, Ping flood, SYN flood to name a few. Out of these, SYN flood works to confuse the server regarding the existence of the client which leads to server allocating resources to this phantom client.

SYN flood is done by misusing the 3-way handshake protocol at the TCP level. Figure 1 illustrates the 3-way handshake protocol. The client first sends a SYN (Synchronize) request to the server whenever the client system wants to connect to the server. The server, if available, responds with a SYN_ACK packet which means that the server is ready to connect and service the client's request. The last, 3rd packet is sent by the client

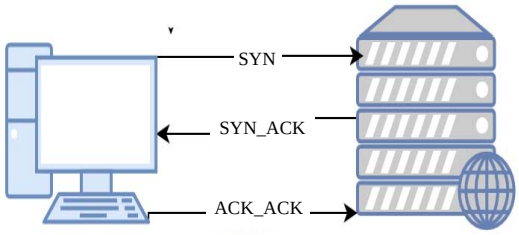

Fig. 1: Regular 3-way handshake protocol

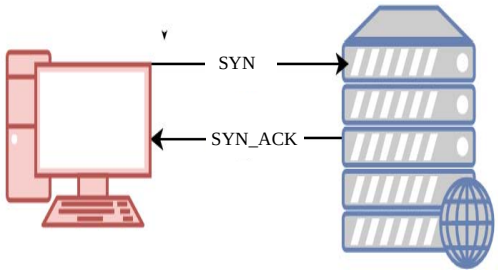

Fig. 2: Attacker's 3-way handshake protocol

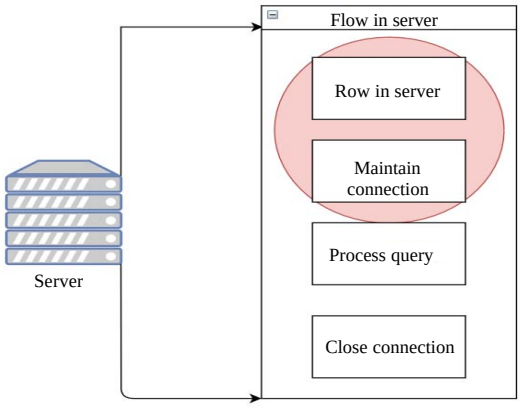

Fig. 3: Working of server

ACK_ACK, confirming the receipt of the server's acknowledgement. The use of 3 different types of packets is the reason why this protocol is called 3-way handshake protocol. Now, an attacker can misuse this protocol to fool the server. The attacker misuses the 3-way handshake protocol by not sending the final acknowledgement. The server, after sending the SYN_ACK is ready to receive the query/request and allocates certain memory so that the request can be served promptly. When the attacker doesn't send the ACK_ACK, this causes the server to wait till timeout to free the allocated resources. Repetition of such malicious packets, eats into the resources and results in a DoS attack (Fig. 2 and 3).

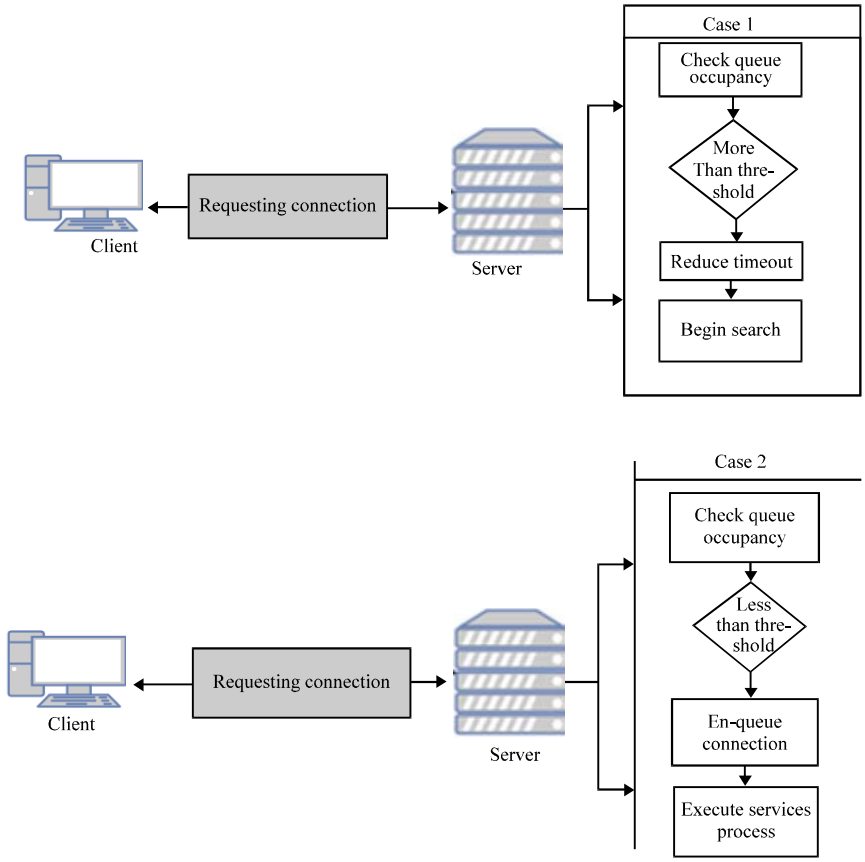

Fig. 4: Case 1 condition

The server has multiple functions to perform. Starting from receiving a packet, establishing a connection, maintaining the connection, processing queries and finally closing the connection. However, the focus of this study, as shown in the Fig. 4 will lie only until getting a connection and mainly, maintaining the connection.

Working: A server, when it receives a packet, first has to check for the kind of packet it has received and then the source of the packet. The packet is then filtered or processed accordingly. To keep a track of all the open or pending services, often a queue is used. A queue is used because of its FIFO (First in First Out) operation. The server also operates in a similar manner and processes requests in the same manner.

In a SYN flood, the queue begins to fill up and the resources that are allocated are wasted because of non-utilization. The wastage of this resource is detrimental as server systems need to be quick and agile in their operations. Many AQM methods have been introduced to solve this problem but such methods also end up eating into the resource pool as continuous checking is done to avoid DoS.

This study introduces a threshold based active queue management, so the continuous lookups into the queue, checking for a possible flood attack can be deferred.

Algorithm (Threshold based AQM):

```
1: procedure tBAQM Procedure
2:     if tBAQ.fillCap greater than tBAQ.Cap/0.8 then3:
timeout←timeout/3
4:     for each packet P ∈ tBAQ do
5:     if b←insertTemp(P,tempArr) then->Boolean return
6:     continue
7:         else
8:     break
9:         end if
10:    end for
11:    packet.Common←SearchPack(tempArr)
12:    end if
13:    IP←grabIP(packet.Common)->to get IP
14:    blacklist(IP)-> temporary blacklisting
15: end procedure
```

The above algorithm describes the mechanism that is proposed. When a packet first comes into the server system and is detected as a SYN packet, the server needs to add its details in the queue, allocate resources and also send a SYN_ACK packet to the source of the original packet.

Case 1: If the queue is not filled till 80% of its capacity. In this case, the SYN packet detail will be added into the AQM. The resource for this connection will be allocated and the acknowledgment will be sent to the original sender. There will be no change in the timeout duration and no other internal server process other than the one required by the packet will be started.

Case 2: If the queue has reached 80% capacity. In this case the packet will be added to the queue but the timeout duration will be one-third of the original. A new internal process to search for any kind of malicious packet(s) will be begun, which will follow the above proposed algorithm. The state of the active queue will be copied into another data structure in which search would be easy to do. In that data structure the frequency of similar packets based on various parameters like Source IP address or Sequence Number or Packet Pattern can be employed to detect the flood. Once the flood is detected, IP address of the found common malicious packet will be blocked temporarily and all the allocated resources would be freed. This blocking of IP address will be for limited duration (Fig. 4).

Hence, this algorithm will be able to successfully differentiate between flood attack and a genuine traffic surge. Such a mechanism is required to make sure that the service is always available for any legitimate user.

RESULTS AND DISCUSSION

The above proposed mechanism was implemented on OMNET++ simulator, as shown in Fig. 5 and 6. OMNET++ provides the framework to create modular and discrete events in order to study them in detail. A basic template of the event and network model to be used was made. The basic template consisted of fixed number of clients and one attacker in the model. The attacker was programmed to create a flood attack on the server. All the clients were programmed to function according to the 3-way handshake protocol. The server allocated resources and added all packets it received into the active queue.

In order to make the study more extensive various measures like delay, bandwidth congestion and variable client number were simulated. All these contributed to factors that could increase the chance of making the DoS attack successful.

The attacker kept the flow of the stream of packets continuous for the entire duration of the simulation, whereas all the clients were programmed to not keep the flow continuous.

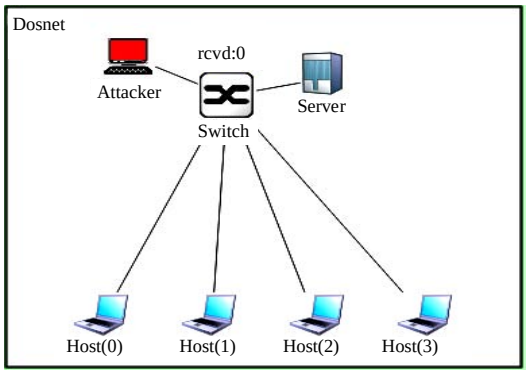

Fig. 5: Simulation-1

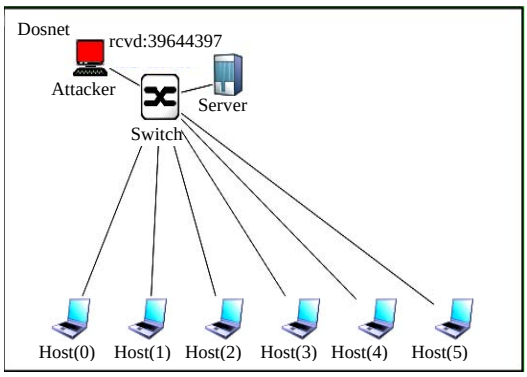

Fig. 6: Simulation-2

Without the implementation of algorithm, i.e. normal client-server architecture under attack, the server shut itself owing to insufficient resources. Depending on the rate of influx of packet, the graph was as shown in Fig. 7. After the implementation of the algorithm, without the blocking of the source IP, the algorithm removed all the common packets from the active queue and continued regularly until again crossing the threshold. The efficiency of this algorithm is shown in the graph as shown in Fig. 8. The algorithm achieved its goal of being able to search and identify the common packets before the server's entire resources were exhausted.

The result was measured for variable number of clients along with mainly two different bandwidth speeds for the attacker. 100 mbs; in this attacker packet took around 100 msec to reach the server, during which the clients with similar bandwidth also operated. But due to the incessant stream of the attacker's packets, soon a congestion condition was achieved, after which the proposed mechanism kicked in and effectively recognized and reduced the congestion by identifying the malicious

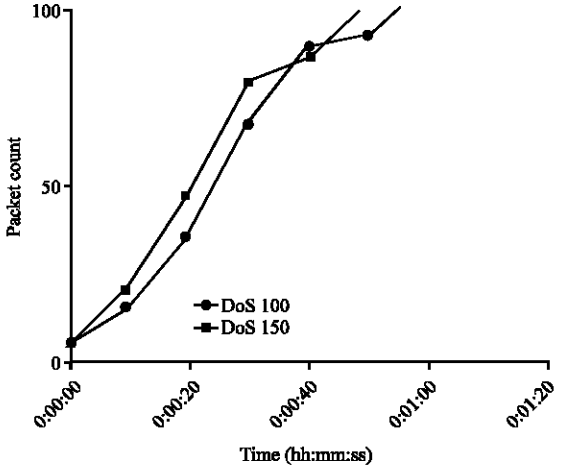

Fig. 7: Graph before algorithm implementation

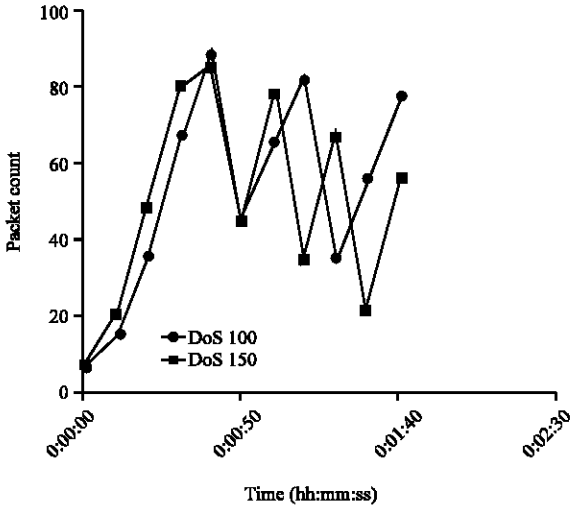

Fig. 8: Graph after algorithm implementation

packets. 150 mbs; in this the attacker packet took around 75 msec to reach the server. The bandwidth for the clients however was kept at 100 mbs. The rate of increase of the filling of queue was faster and the threshold congestion was achieved faster. In this case the number of packets that were removed from the active queue was also higher as the number of the malicious packets was higher this time.

The result was that the threshold based active queue management system worked effectively to reduce the load on the main server and the task was not computationally intensive. The server could effectively ward off the malicious packets. Under a legitimate traffic, the algorithm reduced the timeout duration thereby reducing the time an average client could take but none of the clients faced unavailability of the service.

CONCLUSION

In this study, a threshold based active queue management system has been proposed. It actively keeps track of all influx packets by keeping only some of the essential information as and when required. The large timeout time ensures that during normal usage the detection mechanism doesn't get activated. This algorithm helps reduce the load on the processor of the server as well as the mechanism is activated only when the threshold is crossed. This mechanism will help protect systems' resources. Any different kind of 3-way handshake exploiting attack can be detected by just modifying the search parameters of the function. This would be a prudent way to keep such attack in check. Any kind of attempt to attack this system could be thwarted by this proposed method. Through the results it was shown how this AQM performs better than the existing ones. The graphs that show continuous improvement are a testament to this fact.

ACKNOWLEDGEMENTS

This research was supported by the University of Petroleum and Energy Studies. We would like to specially acknowledge the support and guidance of Mr. Gunjan Chhabra, Asst. Professor, UPES. We would also like to thank Ms. Ninni Singh for her guidance and support throughout the project.

NOMENCLATURE

b	=	Check for completion of data transfer from queue to array
IP	=	IP address to be blacklisted
P	=	Packet in TBAQ
packet.common	=	Common Packet
tBAQ	=	Active Queue
tBAQ.Cap	=	Total capacity of Queue
tBAQ.fillCap	=	Capacity filled in the queue
tempArr	=	Temporary data structure to search
Timeout	=	Time after which connection will be closed if no action

REFERENCES

Anelli, P., R. Diana and E. Lochin, 2014. FavorQueue: A parameterless active queue management to improve TCP traffic performance. Comput. Netw., 60: 171-186.

Bedi, H., S. Roy and S. Shiva, 2014. Mitigating congestion based DoS attacks with an enhanced AQM technique. Comput. Commun., 56: 60-73.

Chang, R.K.C., 2002. Defending against flooding-based distributed denial-of-service attacks: A tutorial. IEEE Commun. Magaz., 40: 42-51.

Fradet, P. and S.H.T. Ha, 2010. Aspects of availability: Enforcing timed properties to prevent denial of service. Sci. Comput. Comput., 75: 516-542.

Jamali, S. and G. Shaker, 2012. PSO-SFDD: Defense against SYN flooding DoS attacks by employing PSO algorithm. Comput. Math. Appl., 63: 214-221.

Lau, F., S.H. Rubin, M.H. Smith and L. Trajkovic, 2000. Distributed denial of service attacks. Proceedings of the IEEE International Conference on Systems, Man and Cyberbetics, October 8-11, 2000, IEEE, Nashville, Tennessee, ISBN:0-7803-6583-6, pp: 2275-2280.

Li, L. and S.B. Shen, 2008. Packet track and traceback mechanism against denial of service attacks. J. China Univ. Posts Telecommun., 15: 51-58.

Maurizio, C., C.A. Grazia, M. Klapez and N. Patriciello, 2015. QRM: A queue rate management for fairness and TCP flooding protection in mission critical networks. Comput. Netw., 93: 54-65.

Noh, S., G. Jung, K. Choi and C. Lee, 2008. Compiling network traffic into rules using soft computing. Appl. Soft Comput., 8: 1200-1210.

Prakash, A., M. Satish, T.S.S. Bhargav and N. Bhalaji, 2016. Detection and mitigation of denial of service attacks using stratified architecture. Procedia Comput. Sci., 87: 275-280.

Safa, H., M. Chouman, H. Artail and M. Karam, 2008. A collaborative defense mechanism against syn flooding attacks in ip networks. J. Network Comput. Appl., 31: 509-534.