Defense against SYN Flood Attack using LPTR-PSO: A Three Phased Scheduling Approach

Zonayed Ahmed
Lecturer, Department of Computer Science and Engineering
Stamford University Bangladesh
Dhaka, Bangladesh

Maliha Mahbub
Lecturer, Department of Computer Science and Engineering
Stamford University Bangladesh
Dhaka, Bangladesh

Sultana Jahan Soheli
Lecturer, Department of Information and Communication Engineering
Noakhali Science and Technology University
Dhaka, Bangladesh

***Abstract*—Security has become a critical factor in today's computation systems. The security threats that risk our confidential information can come in form of seemingly legitimate client request to server. While illegitimate requests consume the number of connections a server can handle, no valid new connections can be made. This scenario, named SYN-flooding attacks can be controlled through a fair scheduling algorithm that provides more opportunity to legal requests. This paper proposes a detailed scheduling approach named Largest Processing Time Rejection-Particle Swarm Optimization (LPTR-PSO) that defends the server against varying intensity SYN-flood attack scenarios through a three-phased algorithm. This novel approach considers the number of half-open connections in the server buffer and chooses a phase accordingly. The simulation results show that the proposed defense strategy improves the performance of under attack system in terms of memory occupancy of legal requests and residence time of attack requests.**

***Keywords*—SYN flood; Largest Processing Time Rejection-Particle Swarm Optimization (LPTR-PSO); three-phased algorithm; legal request; buffer**

I. Introduction

As far as security in data and telecommunications go, technology has sure come a long way, but it still seems to halt at some known stations. One of the most important aims in using of computer networks is to be able to share resources, and reduce network costs while ensuring total reliability. Therefore, it seems likely that these issues are often the most vulnerable while facing breach of security in various forms of attacks. One of the most common and also consistent threats to network reliability and resource availability is the Denial of Service (DoS) attack which can probably be dated back to the time data sharing and networking came into existence.

The goal of DoS attacks is to exhaust a system's resources such that it compromises its ability to provide the intended service and thus rendering it unavailable. DoS attacks typically trust on the misuse of exact susceptibility in such a way that it consequences in a denial of the service. New arithmetical assessment show that DoS positions at the quarter place in the list of the most poisonous attack classes in contradiction of information systems [1].

DoS attacks can be classified into two types. In one type, the malicious user crafts a packet very carefully trying to exploit vulnerabilities in the implemented software [2]. The second type is where the malicious user is trying to overwhelm system's resources of the provided service-like memory, CPU or bandwidth, by creating numerous of useless well-formed requests. This type of attack is well known as flooding attack [3].

One of the most common DoS attacks is called SYN-Flood attack. This flooding attack is caused by attackers through TCP three-way handshaking. It has been reported that more than 90 percent of the existing DoS attacks are TCP based [4].

Three-way handshaking procedure starts when the client sends a SYN request to the server. When the client receives the SYN request, it sends a SYN-ACK packet that contains the synchronization request and acknowledgement. Lastly the client receives it and sends an ACK packet to the server. This is how a connection is established through three way handshaking and then data transfer starts. The procedure is demonstrated in Fig. 1.

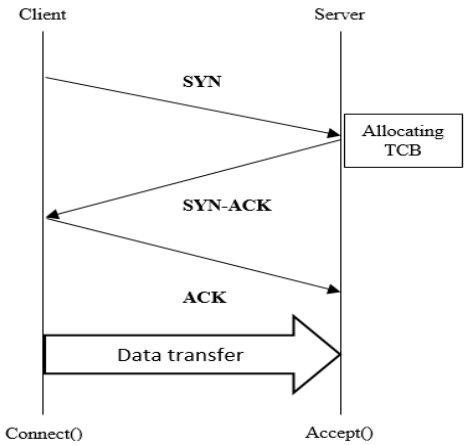

Fig. 1. TCP three way handshaking process.

Transmission Control Block (TCB) is a transport protocol data structure which contains all the information about a connection. Usually, each TCB exceeds at least 280 bytes, and in some operating systems currently takes more than 1300 bytes. The TCP SYN-ACK state indicates that the connection is only half open, and that the legitimacy of the request is still in question. The important aspect to note is that the TCB is

allocated based on reception of the SYN packet—before the connection is fully established or the initiator's return reachability has been verified [5].

The system attacker targets this particular aspect of this process. The attacker sends lots of requests to the server so that the server assigns TCB to each of these requests. The problem is in both ends. If the server continues to assign data to these attack requests, legitimate requests might not be getting resources. Again, as there is no easy way to detect legitimate requests beforehand, the server can't close all half-open connections beforehand.

Now, SYN-flooding attacks don't usually affect the factors such as the link bandwidth, dispensation capital, data rate and so on. Therefore, most of the defense against SYN flood attack can be conjured by an effective scheduling algorithm that helps detect the attack half open connections and discard them. A scheduling algorithm helps assigning resources to the requests in a particular order based on various parameters such as priority, processing time, etc. Since SYN flood attacks target to tie up the resource allocation process, a scheduling algorithm can be implemented which identifies the harmful requests while sorting arrived requests and rule them out. Although using scheduling approach to detect or simply identify the SYN attack requests has been proposed before but none of them has been very effective in successfully removing the attack requests and thus allowing more memory space for legal requests. This principle has been the foundation of the proposed algorithm **Largest Processing Time Rejection-Particle Swarm Optimization (LPTR-PSO)** which actually uses three separate algorithms for different phases based on the degree of attack to the server.

The rest of the paper is organized as follows: the previous approaches, studies and works related to the addressed premise of this paper are listed in Section 2, proposed algorithm and its working principle along with system model are demonstrated in Section 3, performance analysis and simulation results are included in Section 4 and the conclusion along with future works is stated in Section 5, followed by the references.

II. Related Works

Many researchers have been done focusing on SYN-flooding attacks. Some of them are briefly discussed here.

Shahram et al. [2] proposes that SYN flooding attack can be viewed metaphorically as result of an unfair scheduling that gives more opportunity to attack requests but prevents legal connections from getting services. In this paper, a scheduling algorithm named HRTE (Highest Residence Time Ejection) is proposed that ejects the half connection with the longest duration. When number of half open connections reaches to the upper bound. The simulation results show that the proposed defense mechanism improves performance of the under attack system in terms of loss probability of requests and share of regular connections from system resources.

In [6] the authors have analyzed the traffic at an Internet gateway and the results showed that we can model the arrival rates of normal TCP-SYN packets as a normal distribution.

Lemon et al. [7] proposed two queuing models for the DoS attacks in instruction to get the pack postponement jitter and the loss probability.

D.J. Bernstein et al. [8] presents a simple and robust mechanism, called *Change-Point Monitoring* (CPM), to detect denial of service (DoS) attacks. The core of CPM is based on the inherent network protocol behaviors, and is an instance of the Sequential Change Point Detection. To make the detection mechanism insensitive to sites and traffics patterns, a non-parametric Cumulative Sum (CUSUM) method is applied.

Another research offers protection against SYN flooding for all hosts connected to the same local area network, independent of their operating system or networking stack implementation [9].

Vasilios A. Siris et al. proposed the two algorithms considered are an adaptive threshold algorithm and a particular application of the cumulative sum (CUSUM) algorithm for change point detection [10]. The performance is investigated in terms of the detection probability, the false alarm ratio, and the detection delay. Particular emphasis is on investigating the tradeoffs among these metrics and how they are affected by the parameters of the algorithm and the characteristics of the attacks.

Gholam Shaker et al. [11] proposes a self-managing approach, in which the host defends against SYN flooding attack by dynamically tuning off its own two parameters, that is, *m* (maximum number of half open connections) and *h* (hold time for each half-open connection). In this way, it formulates the defense problem and optimization problem and then employs the particle swarm optimization (PSO) algorithm to solve it. The simulation results show that the proposed defense strategy improves performance of the under attack system in terms of BUE and PSA.

Chen et al. [12] addresses the issue of how to define the hash functions in Bloom filter to avoid threat of DoS attacks.

Compilation of an IP address database of previous successful connections is proposed in Peng et al. [13]. When a network was suffering from traffic congestion, an IP address that did not appear in the database was construed as more suspicious.

A similar approach called SYN cookie was proposed in Zuquete et al. [14]. This approach removes the backlog queue from the operating system. The SYN cookie receives an ACK packet it checks the sequence number to see if it is valid. If validated, the packet is accepted and the victim's host allocates resources for the connection, otherwise the packet is dropped.

Center Track [15] and SOS [16] both use overlay techniques with selective rerouting to prevent large flooding attacks.

The proposed algorithm LPTR-PSO is a combination of the principle of Highest Residence Time Ejection (HRTE) [2] and Particle Swarm Optimization (PSO) algorithms. The basic principle of HRTE is that it ejects the job with the highest residence time. However this algorithm does not hold up against multiple attacking half open connections and does not

provide any solution for the blocked legal requests. It only ejects one request with highest residence time. The proposed LPTR on the other hand sets a threshold value for determining the presence of all the requests which can pose as a threat and ejects them from the buffer queue. And even if the buffer still gets occupied by more attack half open connections, our algorithm then switches to self-adjusting optimization algorithm Particle Swarm Optimization (PSO) which effectively handles both the processing time of half open connections and buffer size.

III. Proposed Three Phase Defense Algorithm against SYN Flood: LPTR-PSO

The goal of TCP SYN flood attack is to consume up the TCP buffer space. It does not usually affect the factors such as the link bandwidth, dispensation capital, data rate and so on. Therefore, most of the defense against SYN flood attack can be conjured by an effective scheduling algorithm that helps detect the attack half open connections and discard them. This principle has been the foundation of our proposed algorithm LPTE-PSO which actually uses three separate algorithms for different SYN attack scenarios. When the server is not under any attack, the buffer queue is not fully occupied and the scheduling algorithm will go along with a traditional Round Robin algorithm for scheduling all the jobs (TCP service requests). However, if the buffer is full and the residence time for any job aka the turn-around time of a process exceeds a given threshold value (which in this case is the average turnaround time of all the process in the buffer queue), then the second phase of the algorithm starts which is the Largest Processing Time Rejection (LPTR) algorithm. In this phase, the jobs with residence time higher than the threshold value are considered to be attack half open connections and they are ejected from the queue. The released queue space is added to the buffer to accommodate more TCP service requests.

However, if the attacker keeps sending half open requests and the number of service requests exceeds the maximum TCP buffer size, the legal TCP service requests are blocked. Keeping this in mind, we augmented the third phase of the algorithm which uses a very useful optimization algorithm Particle Swarm Optimization (PSO). This phase starts if the number of service request exceeds the maximum buffer space in order to accommodate the legal request as well as ejecting attack half open connections.

A. Scheduling using LPTR (When n=m)

The proposed algorithm LPTR is based on the principle of Highest Residence Time Ejection (HRTE). The basic principle of HRTE is that it ejects the job with the highest residence time. However this algorithm does not hold up against multiple attacking half open connections and does not provide any solution for the blocked legal requests. It only ejects one request with highest residence time at a time. This could be highly ineffective during a distributed SYN flood attack since most of the time the attacker sends many attack half open request at a time to use up all the buffer space. LPTR on the other hand sets a threshold value for identifying tall the half open connections that has been occupying the buffer for too long and ejects them, thus freeing the memory space for arrived requests which otherwise would have been blocked.

The TCP buffer queue has a limited space it can allocate for incoming service requests and they are considered as half open connection waiting for their turn to get the resources from server they requested. The size of the queue i.e. the maximum number of arrival request that can be held at the buffer is **m**. The arrivals of the regular request packets and the attack packets are both Poisson processes with rates λ_1 and λ_2, respectively. The two arrival processes are independent. Obviously, when the system is under attack then number of pending connections increases and in a point in which there is no more space then the arriving requests will be blocked, which is commonly known as the SYN flooding attack.

The paper proposes that the defense against this attack can be considered as a queue scheduling algorithm LPTR that differentiates attack requests from regular requests by using a threshold value and then ejects the attack requests thus freeing up the space for more arrival requests. The current number of half open connections that are residing in the queue is n. When the queue is full, i.e. n=m, the newly arrived request faces a full buffer, then the algorithm switches to LPTR mode. This algorithm calculates the average turnaround time for all the half open connections or jobs that are currently waiting for their turn where,

Turnaround time of a process = Burst time of the process+ waiting time.

The turnaround time is therefore the processing time required by a connection to get the resource they asked for and thus completing their time at the queue. LPTR calculates the average turnaround time of each process present in the queue and sets this value as a threshold. It then compares the turnaround time of each process with the threshold. If the current turnaround time of a process exceeds the average turnaround time allotted for the process, then that process is considered to be an attack request and it is ejected from the queue, thus freeing the space occupied by the attack request to be allocated to the arrived requests.

However, even if the memory space released from the attack requests gets consumed by further attack request the queue gets full again where n>m. In this scenario, our proposed algorithm switches to PSO, popular swarm intelligence based optimizing algorithm which rather than deleting any requests, adjusts the burst time for each request and sets new size of the queue as to accommodate more arrived requests in the face of a serious SYN flood attack.

B. Scheduling using PSO (When n>m)

The last phase of the proposed algorithm includes PSO, a population based optimization algorithm that assigns possible solutions of a problem in a search space. The method of PSO optimized was introduced in 1995 by James Kennedy and Russel Eberhart [17]. It has been a noteworthy nature-inspired metaheuristics used in science and engineering specially in biotechnology. PSO is mostly used in real world data analysis, resource allocation, scheduling problems [18] and more.

The basic idea of PSO is to assign multiple solutions of a target problem to a search space, or “swarm” of particles. The basic idea of PSO is to assign multiple solutions of a target problem to a search space, or “swarm” of particles which

coexist and share information with the neighboring particles. PSO has an objective function which undergoes a number of iterations and the goal of this objective function is to provide optimal solutions. All the particles in the search space have fitness value that is obtained by the objective function. As the algorithm goes through multiple iterations, each particle moves in the problem search space by changing its velocity vector looking for the optimal solution provided by the objective function. Therefore, each particle has to adjust its position in the search space under the influence of its own top solution, known as the local best and the best solution found by the entire swarm which is also known as the global best position [19]. The particles are essentially characterized by two properties: the particle position, which defines where the particle is located with respect to other solutions in the search space, and the particle velocity, which defines the direction and how fast the particle should move to improve its fitness compared to the rest of the neighbor particles [18]. The proposed algorithm switches to PSO when the number of half open connection n exceeds the maximum buffer queue size m, which can be considered a heavy attack scenario. Two parameters are considered here: the maximum residence time each half open connection is allowed to hold in the queue, t and the maximum number of half open connection that the buffer can allow.

1) Objective Function of PSO

For defense against SYN flood attack PSO algorithm, an objective function has been selected which consists of the following parameters: t and m as mentioned earlier. The goal here is to minimize the occupancy of attack requests in the queue and preventing the loss of legal requests while simultaneously increasing the space occupied by legitimate requests. So, the objective function of this problem is formulated as,

*Objective function = min [(attack half open connections*rate of packet loss)/(regular half open connections)]*

By calculating the objective function at iterative steps and comparing it to the objective function obtained from the previous steps of all particles in swarm, the parameters t and m of the queue at any moment are calculated.

The basic PSO algorithm, which minimizes this objective function in a swarm consisting of a finite number of particles. Each particle *i* of the swarm is associated with a position in a continuous n-dimensional search space. Similarly, the velocity is also an n-dimensional vector. The position and velocity of each particle *i* at an iteration *k* is denoted as x^{ki} and V^{ki} respectively, the following equations are used to iteratively modify the velocities of the particles and positions:

$$V_t^{k+1} = wv_t^k + c_1r_1(Lbest_t^k - x^k) + c_2r_2(Gbest^k - x^k) \quad (1)$$

$$x^{k+1} = x^k + V_t^{k+1} \quad (2)$$

$$V_p^{k+1} = wv_p^k + c_1r_1(Lbest_p^k - p^k) + c_2r_2(Gbest_p^k - p^k) \quad (3)$$

$$p^{k+1} = p^k + V_p^{k+1} \quad (4)$$

where, V_t^{k+1} represents the distance that needs to be traveled by the *i*th particle from its current position in the *kth* iteration, x^{k+1i} represents the particle position in the *kth* iteration, *w* is the *inertia* parameter that weights the previous particles velocity, *pbest* represents its best personal position of the particle and *gbest* represents the global best position among all particles in the swarm. The parameters c_1 and c_2 enable the movement of the particle for its personal best position towards the global best position and their values should satisfy the condition, $c_1 + c_2 > 4$. In this case, the values of acceleration parameters have been chosen to be 2. Parameters r_1 and r_2 are two random numbers uniformly distributed in [0, 1] that are used to weight the velocity toward the particle personal best and toward the global best solution [18]. As stated earlier, PSO algorithm minimizes the objective function and each particle of the swarm tries to tune its positions to fit to the global best position, i.e. *t* and *m* seeks the best position in the whole swarm. The best defense positions of the parameters *t* and *m* are therefore utilized by the server to defend the flooding attack. When the half open connections exceed the maximum queue size, PSO with objective function including parameters *t* and *m* is executed. The proposed PSO reduces the residence time *t* of each request in the queue which allows the attack requests to be discarded quickly and simultaneously increases the buffer size so that incoming legal requests can be accommodated. The overall workflow of the algorithm is shown in the flow chart in Fig. 2.

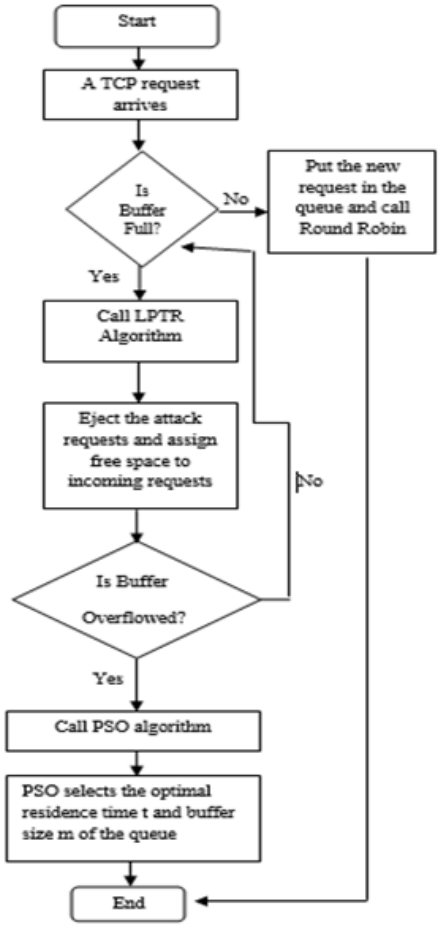

Fig. 2. Flow chart for LPTR-PSO algorithm.

C. Efficiency Measurement Parameters (ARTR and RRTR)

In order to effectively measure the efficiency of the algorithm in different attack scenarios, two parameters, Attack Request Time Rate (ARTR) and Regular Request Time Rate (RRTR) has been chosen.

RRTR is the ratio of the sum of all the regular connection duration in the queue to the total available resources or memory space. ARTR is the ratio of all the attack connections duration to the total available resources or memory space [2].

The efficiency of both LPTR and PSO algorithm has been measured using the same parameters to show that both the algorithms are capable of defending SYN flood attack. The goal here is to assure a lower ARTR and higher RRTR while the server is under attack.

IV. Performance Analysis, Comparison and Results

The working principle of this algorithm considered a server under SYN flood attack with buffer size m=15 dealing with TCP requests each with different burst time (time required to complete the task) and each set had variations in the turnaround time depending on the attack intensity. Three attack scenarios are considered, namely high attack intensity, medium attack intensity and low attack intensity. Comparative analysis of LPTR and HRTE and performance of PSO when request number is increased have been illustrated in this section.

For LPTR algorithm, n=15 sets of TCP requests at a time processed at a time which meets the requirement n=m.

In the three scenarios, length of turnaround time has been used as the threshold for selecting attack request and regular request. If turn-around time of a request is greater than average turnaround time, then it is considered an attack request. Otherwise it is a regular request. Three samples are shown for request sets for three scenarios. 10 such TCP request sets each containing n=15 requests are selected to measure the performance of LPTR. HRTE algorithm has been employed on the same data set to compare the results between them.

In case of PSO, n=16 sets of TCP requests has been used at a time processed at a time which meets the requirement n>m. PSO has reduced the total duration time of all the requests in the queue, simultaneously showing equal or even better performance than LPTR which is desirable as PSO will be used when attack risk is greater. PSO has also allotted different queue size m for different attack scenarios so that the incoming legal requests are not blocked from getting service. All the numerical analysis and simulations have been conducted using Matlab.

A. High Attack Intensity

In a high attack intensity, where k is the intensity factor determined by the ratio of attack request arrival rate λ_2, and regular request arrival rate λ_1 respectively, so $k = \lambda_2 / \lambda_1$. A sample set of requests in a low attack scenario may contain 1-4 attack requests with high burst time and 11-14 regular requests. The ARTR and RRTR have been plotted for each request set over the total duration of requests in each set.

1) Using LPTR(n=m):

For LPTR, the value of k=0.7-0.8. The total duration of requests ranges from 33 to 40. The performance of request set under high attack using LPTR is shown below.

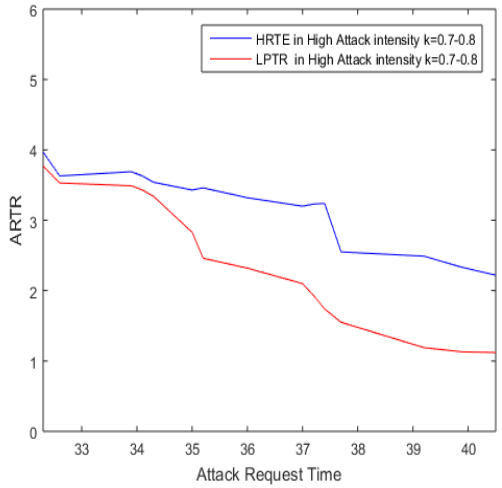

Fig. 3. ARTR for high attack intensity using LPTR and HRTE.

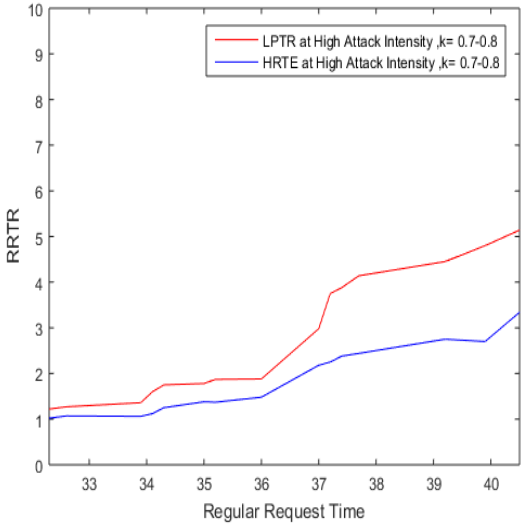

Fig. 4. RRTR for high attack intensity using LPTR and HRTE.

As seen from Fig. 3, the ratio of attack requests to the total requests gradually decreases as the total duration of each request set increases when the requests are sorted using LPTR and it shows a better performance than HRTE.

As seen from Fig. 4, the ratio of regular request to total requests increases as the request duration increases while LPTR is used and shows better performance than HRTE.

2) Using PSO(n>m)

For PSO, the value of k=2 in high attack scenario. As seen from the figures, PSO reduces the duration of each request in the queue thus reducing the total duration time is reduced from that in LPTR. The request duration ranges from 18 to 34 in case of attack requests and 8 to 20 in case of regular requests.

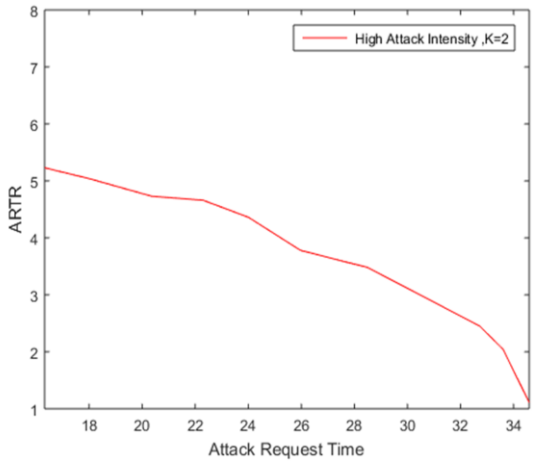

Fig. 5. ARTR for high attack intensity using PSO.

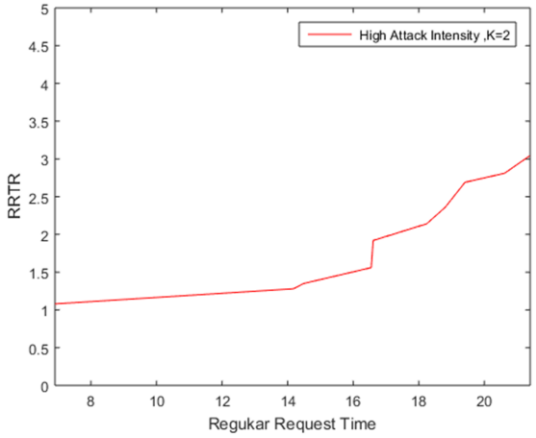

Fig. 6. RRTR for high attack intensity using PSO.

As seen from Fig. 5, the total duration of all request is reduced and the ratio of attack requests to the total requests gradually decreases as the total duration of each request set increases when the requests are sorted using PSO.

As seen from Fig. 6, the total duration of all request is reduced and the ratio of regular requests to the total requests gradually increases as the total duration of each request set increases when the requests are sorted using PSO.

B. Medium Attack Intensity

In medium attack intensity, a sample set of requests in a medium attack scenario may contain 7-8 attack requests with high burst time and 8-9 regular requests.

1) Using LPTR(n=m)

For LPTR, the value of k=0.5-0.6. The total duration of requests ranges from 19.5 to 22. The performance of request set under high attack using LPTR is shown below.

As seen from Fig. 7, the ratio of attack requests to the total requests gradually decreases as the total duration of each request set increases when the requests are sorted using LPTR, and shows a better performance than HRTE.

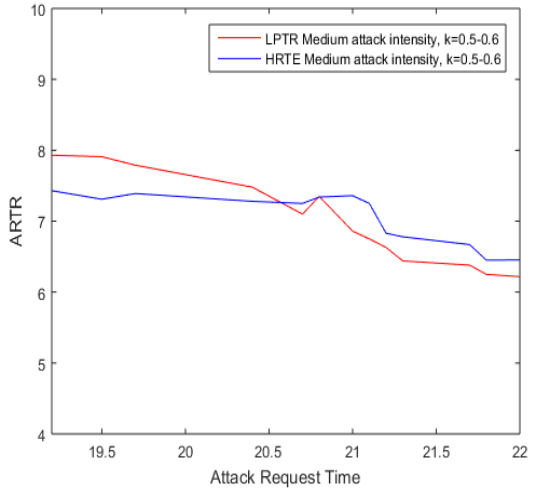

Fig. 7. ARTR for medium attack intensity using LPTR and HRTE.

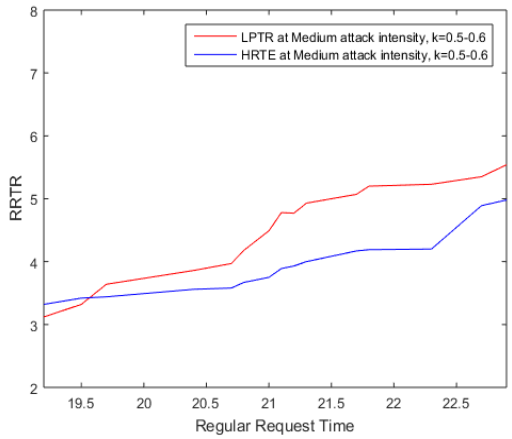

Fig. 8. RRTR for medium attack intensity using LPTR and HRTE.

As seen from Fig. 8, the ratio of regular requests to the total requests gradually increases as the total duration of each request set increases when the requests are sorted using LPTR and shows better performance than HRTE.

2) Using PSO(n>m)

For PSO, the value of k=1 in medium attack scenario. As seen from the figures, PSO reduces the duration of each request in the queue thus reducing the total duration time is reduced from that in LPTR. The request duration ranges from 9.75 to 20 in case of attack requests and 11 to 21.5 in case of regular requests.

As seen from Fig. 9, the total duration of all request is reduced and the ratio of attack requests to the total requests gradually decreases as the total duration of each request set increases when the requests are sorted using PSO.

As seen from Fig. 10, the total duration of all request is reduced and the ratio of regular requests to the total requests gradually increases as the total duration of each request set increases when the requests are sorted using PSO.

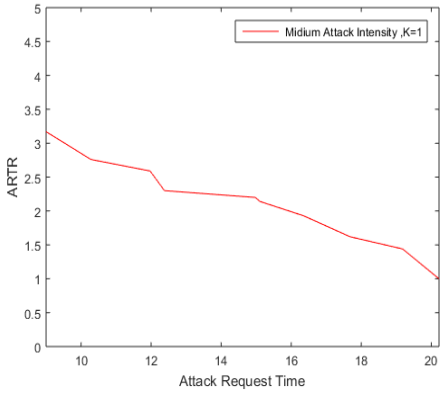

Fig. 9. ARTR for medium attack intensity using PSO.

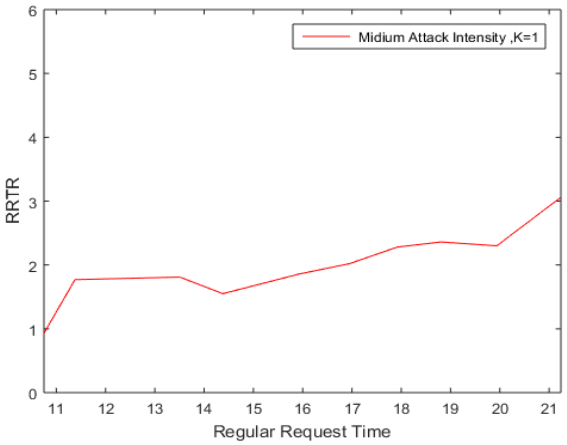

Fig. 10. RRTR for medium attack intensity using PSO.

C. Low Attack Intensity

A sample set of requests in a low attack scenario may contain 7-8 attack requests with high burst time and 8-9 regular requests.

1) Using LPTR(n=m)

For LPTR, the value of k=0.2-0.3. The total duration of requests ranges from 7.5 to 20.9. The performance of request set under high attack using LPTR is shown below.

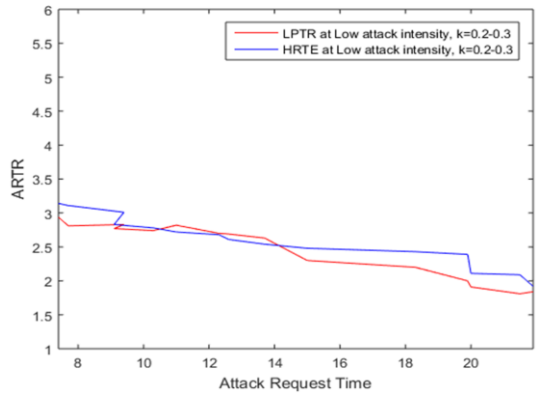

Fig. 11. ARTR for low attack intensity using LPTR and HRTE.

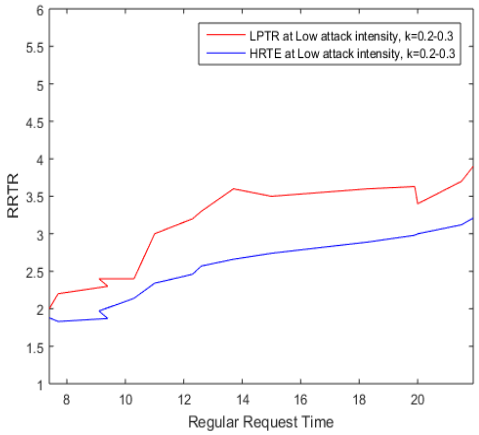

Fig. 12. RRTR for low attack intensity using LPTR and HRTE.

As seen from Fig. 11, the ratio of attack requests to the total requests gradually decreases as the total duration of each request set increases when the requests are sorted using LPTR and shows better result than HRTE.

As seen from Fig. 12, the ratio of regular requests to the total requests gradually increases as the total duration of each request set increases when the requests are sorted using LPTR and shows better result than HRTE.

2) Using PSO(n>m)

For PSO, the value of k=0.5 in low attack scenario. As seen from the earlier figures, PSO reduces the duration of each request in the queue thus reducing the total duration time is reduced from that in LPTR. But since here the attack intensity is lower, PSO seems to extend the duration time a little more as opposed to reducing it too much so that every request can stay a little longer. The request duration ranges from 5 to 25.75 in case of attack requests and 11 to 30.5 in case of regular requests.

As seen from Fig. 13, the total duration of all requests is reduced but not to a great extent since this is a low attack scenario. The ratio of attack requests to the total requests gradually decreases as the total duration of each request set increases when the requests are sorted using PSO.

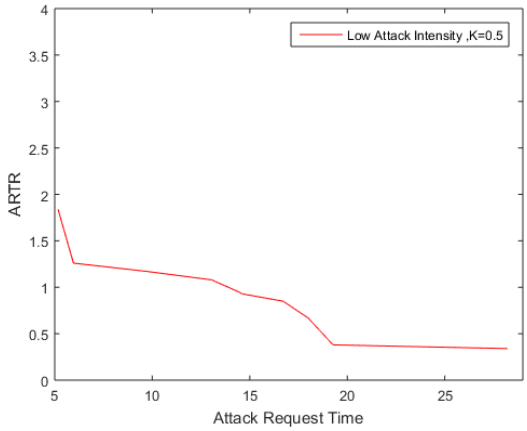

Fig. 13. ARTR for low attack intensity using PSO.

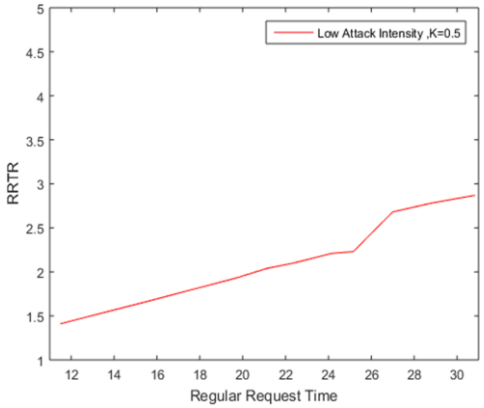

Fig. 14. RRTR for low attack intensity using PSO.

As seen from Fig. 14, the ratio of regular requests to the total requests gradually decreases as the total duration of each request set increases when the requests are sorted using PSO.

D. Variations of Buffer Size m using PSO

As stated earlier, the proposed objective function of PSO, reduces the duration of half open connection in the queue as well as increases the maximum number of half open connections that can reside in the buffer or buffer size m based on different attack scenario. This allows the victim server to accommodate more legal requests even if the server is under attack. The gradual rise in the buffer size of m of the server under different attack scenario is shown in next subsections.

1) High Attack Scenario

When the attack intensity is much high, the PSO increases the maximum capacity of the buffer m from 20 up to 65 to prevent blockage of arrived requests. In Fig. 15, as the total duration of half open connection in the buffer increases gradually, so does the maximum capacity of buffer.

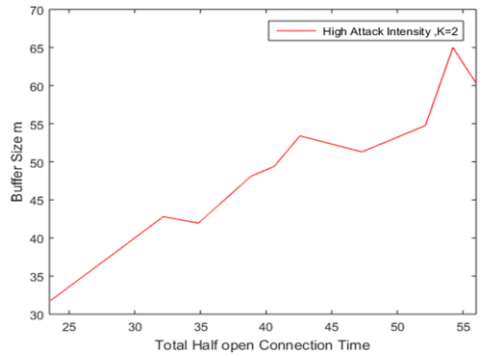

Fig. 15. Increase of maximum buffer size m in high attack intensity.

2) Medium Attack Scenario

When the attack intensity is medium, the PSO increases the maximum capacity of the buffer m from 20 up to 45 to prevent blockage of arrived requests. In Fig. 16, as the total duration of half open connection in the buffer increases gradually, so does the maximum capacity of buffer.

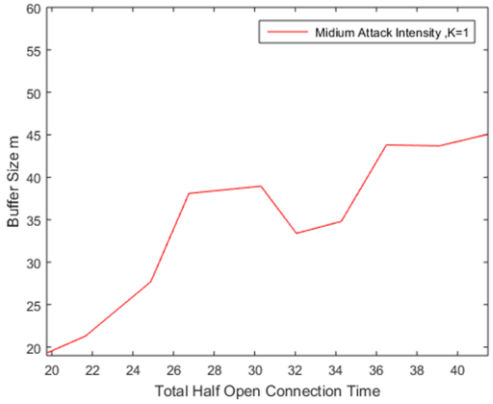

Fig. 16. Increase of maximum buffer size m in medium attack intensity.

3) Low Attack Scenario

When the attack intensity is much low, the PSO still increases the maximum capacity of the buffer m from 20 up to 41 to prevent blockage of arrived requests. In Fig. 17, as the total duration of half open connection in the buffer increases gradually, so does the maximum capacity of buffer.

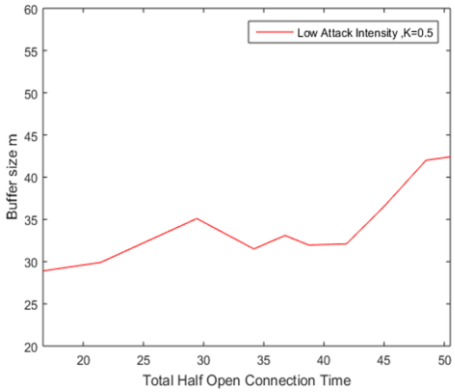

Fig. 17. Increase of maximum buffer size m in medium attack intensity.

V. Conclusion and Future Works

The proposed scheduling approach to detect and defend SYN flood attack executes a three phase scheduling algorithm based on three different situations a server can handle. While the server is not under attack and the buffer is not fully occupied, the newly arrived requests are allotted into the queue and traditional round robin approach is used to schedule their resource allocation. If the buffer is full, then the proposed novel LPTR algorithm is called which compares each half open connection in the queue with a threshold value and ejects the connections that exceed this threshold. If the buffer is still overflowed, PSO algorithm is called to schedule the existing and arrived requests by optimizing the residence time of each half open connection in the queue and the maximum number of connections that the queue can hold. As a result, the incoming requests can be allotted into the queue and the duration of half open connection in the queue is reduced which reduces the presence of attack requests in the queue. This novel approach takes various aspects of the server under

attack instead of one and shows effective results in all the cases. Instead of using different approaches to defend the attack, this mechanism can serve as both scheduling and defending framework that could ensure maximum defense with efficient scheduling at the same time. While the ongoing work focuses on the theoretical framework and simulated analysis, there are strong considerations of moving to the directions of implementing the model on real network traffic and study the effects and scope of this approach to defend similar Distributed DoS attacks on clustered and virtual networks as well. This approach can be further extended to defend other security issues and common attacks on virtual networks, especially in cloud computing which might be a future scope for the premise to be explored.

REFERENCES

[1] Gordon L.A., Martin P. L., Lucyshyn W., Richardson R., "2005 CSI/FBI computer crime and security survey", Computer Security Journal, 2005.

[2] Jamali S., Shahram, and Gholam Shaker. "Defense Against SYN Flooding Attacks: A Scheduling Approach." Information Systems & Telecommunication , pp.55, 2014.

[3] D. Geneiatakis, N. Vrakas, C. Lambrinoudakis, "Utilizing Bloom Filters for Detecting Fooding Attacks Against SIP Based Services", Computers & Security, 2009.

[4] H. Wang, D. Zhang, and K. G. Shin., "Detecting SYN flooding Attacks", Proceedings of Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM) volume 3, pages 1530-1539, June 23-27 2002.

[5] Eddy, Wesley M. "Defenses against TCP SYN flooding attacks." The Internet Protocol Journal 9.4, pp. 2-16, 2006.

[6] Divakaran, Dinil Mon, Hema A. Murthy, and Timothy A. Gonsalves. "Detection of SYN Flooding Attacks Using Linear Prediction Analysis." Networks, 2006. ICon'06. 14th IEEE International Conference on. vol. 1. IEEE, 2006.

[7] Lemon, Jonathan. "Resisting SYN Flood DoS Attacks with a SYN Cache." BSDCon. vol. 2002, 2002.

[8] Bernstein, Daniel J. "Cache-timing attacks on AES." On PALMS-Princeton University, 2005.

[9] 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", Proceedings of IEEE Symposium on Security and Privacy, May 1997.

[10] Vasilios A. Siris and Fotini Papagalou et al. "Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks", Journal of Computer Communications, 2006.

[11] Jamali, Shahram, and Gholam Shaker. "PSO-SFDD: Defense against SYN flooding DoS attacks by employing PSO algorithm." Computers & Mathematics with Applications 63.1, pp. 214-221, 2012.

[12] Chen, Wei, and Dit-Yan Yeung. "Throttling Spoofed SYN Flooding Traffic at the Source.", Telecommunication Systems 33.1, pp. 47-65, 2006.

[13] Peng, Tao, Christopher Leckie, and K. Ramamohanarao. "Protection from Distributed Denial of Service Attacks Using History-Based IP Filtering.", Communications, 2003, ICC'03, IEEE International Conference on vol. 1, IEEE, 2003.

[14] Zuquete, Andre, "Improving the functionality of SYN cookies." Advanced Communications and Multimedia Security. Springer US, pp. 57-77, 2002.

[15] Stone, Robert. "CenterTrack: An IP Overlay Network for Tracking DoS Floods.", USENIX Security Symposium, vol. 21, 2000.

[16] Keromytis, Angelos D., Vishal Misra, and Dan Rubenstein. "SOS: Secure Overlay Services." ACM SIGCOMM Computer Communication Review. vol. 32, no. 4, ACM, 2002.

[17] J. Kennedy and R. Eberhart, Particle Swarm Optimization. In Proceedings of IEEE International Conference on Neural Networks, vol. IV, doi: 10.1109/ICNN.1995.488968 pp. 1942–1948, 1995.

[18] Pacini E., Mateos C., Garcia C., Dynamic Scheduling based on Particle Swarm Optimization for Cloud-based Scientific Experiments, Technical Report, University of Maryland at College Park. Clei Electronic Journal, Volume 14, Number 1, Paper 2, 2014.

[19] Szymon L, Piotr A. Kowalski, Fully Informed Swarm Optimization Algorithms: Basic Concepts, Variants and Experimental evaluations. Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, pp. 155-161, 2014