Defense against SYN Flooding Attacks: A Scheduling Approach

Shahram Jamali*
Computer Engineering, Associate Professor, University of Mohaghegh Ardabili, Ardabil, Iran
jamali@uma.ac.ir
Gholam Shaker
Young Researchers Club, Ardabil Branch, Islamic Azad University, Ardabil, Iran
gholamshaker@gmail.com

Received: 29/Sep/2013 Accepted: 08/Feb/2014

Abstract
The TCP connection management protocol sets a position for a classic Denial of Service (DoS) attack, called the SYN flooding attack. In this attack attacker sends a large number of TCP SYN segments, without completing the third handshaking step to quickly exhaust connection resources of the victim server. Therefore it keeps TCP from handling legitimate requests. This paper 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, we present a scheduling algorithm 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.

Keywords: DoS Attack, SYN Flooding, Scheduling, TCP, Performance

1. Introduction

Security has been always an important issue in communication and computation systems. In these systems security has different aspects. Sometimes the threat is in the form of disclosing of our confident information. In this case we use some security algorithms and protocols to protect the confidentiality. Cryptanalysis [1] and robust key management and distribution algorithms [2, 3] play important roles from this point of view. On the other hand sometimes security is defined as continuous and uninterrupted service. This definition is taken when the threat is in form of a DoS attack. The goal of a DoS attack is to completely tie up certain resources so that legitimate users are not able to access a server. A successful DoS attack overpowers the victim and conceals the offender's identity [4]. A DoS attack can be regarded as an explicit attempt of attackers to prevent legitimate users from gaining a normal network service. 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 [5]. Anderson et al. rely on the use of a 'send-permission-token' to restrict DoS attacks [6]. Allen and Marin use estimates of the Hurst parameter to identify attacks which cause a decrease in the traffic's self-similarity [7]. Two statistical methods of analyzing network traffic to find DOS attacks are provided in [8]. One monitors the entropy of the source addresses found in packet headers, while the other monitors the average traffic rates of the 'most' active addresses. Many proposals have been made for IDSs intended to detect DoS attacks [9-12], most of them being based on the arithmetical detection of high traffic rate spending from the interloper or interlopers.

Generally DoS attacks could have two major forms. In the first one, the malicious user crafts very carefully a packet trying to exploit vulnerabilities in the implemented software (service or a protocol). In the second form, 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 [13]. One of the most common DoS flooding attacks is SYN flooding attacks. It works at the conveyance layer. A TCP connection is recognized in what is known as a 3-way handshake. When a client labors to start a TCP connection to a server, first, the client needs a connection by distribution an SYN packet to the server. Then, the server returns a SYN–ACK, to the client. Lastly, the client admits the SYN–ACK with an ACK, at which point the connection is recognized and data transfer commences. In an SYN flooding attack, attackers use this protocol to their advantage. The attacker directs a large number of SYN packets to the server. All of these packets have to be touched like a connection request by the server, so the server must response with a SYN–ACK. The attacker then has two choices. One is just not to response to the SYN–ACK, which will reason the server to have a half-open connection. This would let the server to chunk any further packets from the attacker's IP address, ending the attack hastily. Then again, the attacker parodies the IP address of some unwary client. The server rationally responses to this IP address, but the genuine client really

* Corresponding Author

exist in at this IP address will weakening this SYN–ACK as it did not pledge the connection. The result is that the server is left waiting for a reply from a large amount of connections. Since reserve of any system is imperfect, then, there are a limited number of connections a server can handle. Once all of these are in use, waiting for connections that will not ever come, no new connections can be made whether valid or not. It is clear that though this is a conveyance layer attack, it touches all TCP-based requests in the victim server. When the server cannot handle new connections, any request that tries to establish TCP connections with the server, fails in its effort. Note that SYN flooding attacks goal to use TCP buffer space and do not touch the parameters such as link bandwidth, dispensation capitals and so on.

We believe that in a SYN flooding attack an unfair scheduler gives more opportunity to attack requests but prevents legal connections from getting service. Hence, we propose a scheduling-based solution in which when an arriving request faces with a full queue, the oldest half connection is terminated and its resources are assigned to the new connection.

2. Related Work

There are many researches focusing on SYN flooding attack. S.H.C. Haris et al. in [14] used anomaly detection to detect TCP SYN flooding attack based on payload and unusable area. In order to detect TCP SYN flooding, the normal payload characters must be understand first unless the analysis will take times for those that are not expert in payload characters. Other researches [15, 16] proposed methods to throttle spoofed SYN flooding attacks at the sources. Long et al. [17] proposed two queuing models for the DoS attacks in instruction to get the pack postponement jitter and the loss probability. Wang et al. [12] educations the DoS attacks logically by using a more general queue model, a two-dimensional embedded Markov chain, which can more precisely capture the dynamics of the actual DoS attacks. Maciej Korczynski et al. [18] proposed an accurate sampling scheme for defeating SYN flooding attacks as well as TCP port scan activity. The scheme examines TCP segments to find at least one of multiple ACK segments coming from the server to validate legitimate connections. The method achieves good detection performance with false positive rate close to zero even for very low sampling rates. Hussain and Blazek assessed traffic arrivals and consistent ramp-up doings in [12, 19].Their approaches try to learn how a signal in a system and parameters explaining the system change. To detect attacks, packet rate against time is examined instead of only the packet header. This is done, in instruction that IP address spoofing cannot deceive the attack detection method. Haidar Safa et al. in [20] proposed a novel defense mechanism that makes use of the edge routers that are associated with the spoofed IP addresses' networks to determine whether the incoming SYN–ACK segment is valid. This is accomplished by maintaining a table that matches the incoming SYN–ACKs with outgoing SYNs and also by using the ARP protocol. Other investigates made in this area can be deliberate in [21, 22]. As another research S. Jamali and his coworkers have presented a PSO-based defense scheme against SYN flooding attacks [23], called PSO-SFDD. This algorithm formulates the defense issue as an optimization problem and then employs PSO algorithm to solve it. The achieved solution is a configuration for the under attack system that alleviates the attack effects.

3. Scheduling Algorithms

When a computer or network resource receives multiple service requests (jobs) at a given time, a scheduling algorithm is necessary to determine the order in which requests are serviced. Scheduling algorithms work based on several job features such as processing time, priority, due date, and so regarding their design goals [24]. In this section, we first present some of these scheduling algorithms and then present a novel scheduling algorithm which is used to design a defense strategy against SYN flooding attacks.

3.1 A Review over Scheduling Algorithms

There are many different scheduling algorithms proposed for scheduling in different operating systems or switches. Below we cover some popular algorithms, although there are some other algorithms such as Round Robin, Priority Scheduling, Multi-level Queue Scheduling and Real Time Scheduling [24] which will not be covered here.

FIFO-A common method of job scheduling for computer and network resources is First-Come-First-Serve (FCFS) or FIFO, where jobs are serviced in the order in which they arrive. Using FIFO, the job with the earliest arrival time is served first. Jobs with earlier arrival times are served before jobs that arrive later [25].

SPT (Shortest Processing Time)-Using SPT, jobs are processed in ascending order of processing times. It is well known that SPT minimizes the total completion times of a set of jobs. SPT produces an optimal job sequence for minimizing the total, and thus mean, of job waiting times [25, 26].

Hard fairness-Hard fairness [27] is also known as round-robin scheduling. Homogeneity of resources is not a requirement in this type of scheduling. It is the fairest scheme since each process is guaranteed exactly equal amount of time in order.

Max–Min fairness-Max–min fairness [28] allocates resources in order of increasing demand. The minimum amount of resources assigned to each process is maximized. So if there are more than enough resources for each process, every process gets what it needs. If there is not, the resources are split evenly. This means that the process which require fewer resources get a higher proportion of their need satisfied. This type of scheme works best when there is not large differences in amount of resources requested by different processes.

SRTF (Shortest Remaining Time First)-Shortest Remaining Time First, also known as **Shortest Job First (SJF)**, is a scheduling method that is a preemptive version of shortest job next scheduling. In this scheduling algorithm, the process that needs the smallest amount of remaining time to be completed is selected to execute. Since the currently executing process has the shortest amount of remaining time and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a smaller amount of time.

3.2 Proposed Scheduling Algorithms: Highest Residence Time Ejection (HRTE)

This algorithm is a preemptive two-phase scheduling algorithm. This algorithm is useful for scenarios in which the service time of requests is unknown. According to this scheduling algorithm, while input queue isn’t full, HRTE is in its first phase and acts exactly like round robin algorithm. But upon queue becomes full and arriving requests are blocked, HRTE switches to its second phase during which ejects the job with the highest residence time and assigns the released capacity to the arriving requests. HRTE remains in this phase until a free capacity is available in the wafting queue. In other words, it has some similarities with SRTF, but when remaining time is unknown SRTF cannot be used. In this situation we assume that those requests that have the longest duration in the past will have the longest remaining time, as well and hence will be rejected.

4. Proposed Defense Scheme: Queue Scheduling

When a connection request arrives at a TCP-based server, receives a buffer space of the backlog queue upon finding an inactive buffer space and is blocked otherwise. Now, consider a server under the SYN flooding attacks. Assume that in this computer each half-open connection is held for at most a period of holding time (h), and at most a number of maximum half-open connections (m) are allowed. We assume that a half-open connection for a regular request packet is held for a chance time which is exponentially distributed with parameter μ. The arrivals of the regular request packets and the attack packets are both Poisson processes with rates λ1 and λ2, 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 room for pending connection to be saved the arriving requests will be blocked. In the other words, when a server is under SYN flooding attacks, half-open connections quickly consume all the memory allocated for the pending connections and prevent the victim from further accepting new requests, leading to the well-known buffer overflow problem.

We believe that the defense against this attack can be considered as a queue scheduling algorithm that differentiates attack requests from regular requests and then ejects the attack requests. This defense scheme, called HRT_SYN, tries to block attack connections and to prevent the system from allocating buffer space to attack connections. To this end, we use the proposed scheduling algorithm and keeps attack request from occupying system resources for a long time. According to this approach while there is free capacity for coming connection requests they will be accepted and inserted in the queue. But, when a connection arrives and faces with a full queue, then the HRTE scheduling algorithm is invoked and ejects the connection with the highest residence time. The ejected connection is likely an attack request that leaves the system to make the capacity free. This defense scheme can be described by flowchart of Figure 1. To measure efficiency of defense algorithms we use three parameters i.e. RT, AT and Ploss:

RT is Regular requests residence Time that can be described as mean ratio of the occupied resources by regular connections to total available connection resources. It is computed by using equation (1).

$$\mathrm{RT} = \frac{\sum_{\mathrm{i:\ all\ regular\ connections}} \mathrm{duration_i}}{\mathrm{simulation\ time} * \mathrm{m}} \quad (1)$$

In which, *m* is maximum number of connections that are allowed to be established in the system.

AT is Attack residence Time that can be described as mean ratio of connection opportunistic that is occupied by Attack connections. It is computed by using equation (2).

$$\mathrm{RT} = \frac{\sum_{\mathrm{i:\ all\ attack\ connections}} \mathrm{duration_i}}{\mathrm{simulation\ time} * \mathrm{m}} \quad (2)$$

Ploss *(Probability of connection loss)* is connection loss probability, a basic measure for assessing the performance of the system under DoS attacks. Each arriving connection request packet is rejected once there are *m* pending connections in the system. On the other hand some half-open connections are ejected after a period of time (in Linux and PSO_SFDD) or when there are *m* pending connections and a new connection request arrives (in HRT_SYN). Therefore *Ploss* can be described as ratio of the number of ejected or rejected requests to the all arrived requests.

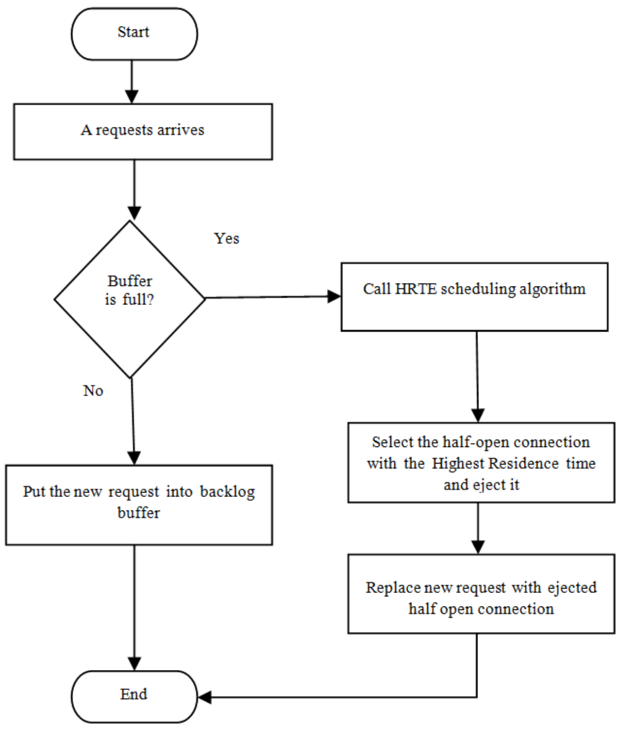

Start
A requests arrives
Buffer is full?
Yes
No
Call HRTE scheduling algorithm
Select the half-open connection with the Highest Residence time and eject it
Replace new request with ejected half open connection
Put the new request into backlog buffer
End

Fig. 1. HRT_SYN in operation

5. Implementation Issues and Simulation Results

In his section we use MATLAB software package to gather a group of simulation results to demonstrate the validity of our design. In this study TCP of the Linux operating system along with PSO-SFDD are considered as a comparison reference point to show efficiency of HRT_SYN algorithm.

5.1 Simulation Setup

As an important issue, note that the victim server doesn't have any information to determine whether the arriving requests are regular requests or attack requests. It accepts all arriving requests and then applies scheduling algorithm. Arrival rate of regular requests to the server has Poisson distribution with mean rate λ and attack requests has mean rate kλ, in which, k represents the ratio between arrival rate of the attack requests to arrival rate of the regular requests and is referred to as attack intensity. Duration of half-open connections for normal requests has exponential distribution with mean μ. In this simulation we consider a server in a network with λ=10/s and μ=100 s. Attack intensity i.e. k is set with different values in different simulation scenarios.

5.2 Simulation Results

In this section a typical TCP protocol is equipped with HRT_SYN defense capabilities and then is compared with Linux TCP which uses static values for maximum holding time and maximum number of half open connections (75 second and 125 connections respectively).

In this study three scenarios of simulations are presented. In the first scenario attack intensity is considered as k=0.1, in the second one it is considered as k=3 and finally in the third scenario, we consider a variable attack intensity that fluctuates between 0 and 3 as shown in Figure 2.

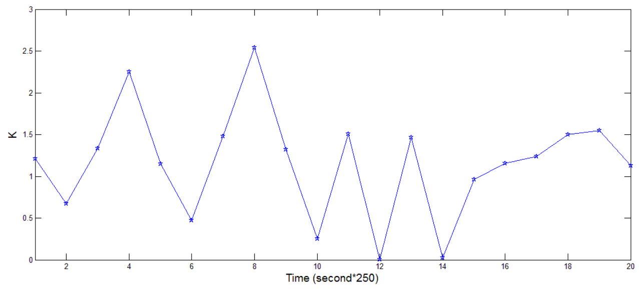
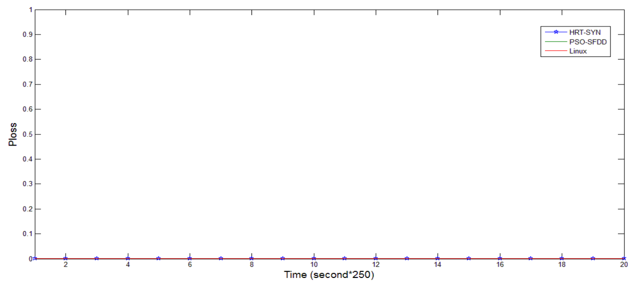


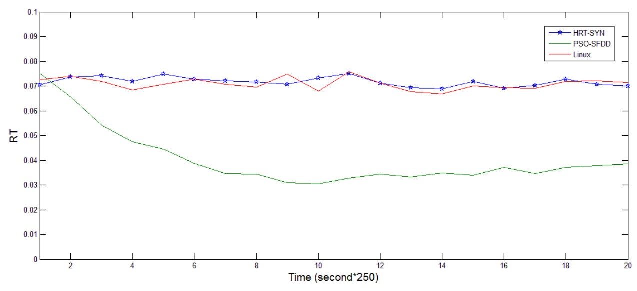
Fig. 2. Variable intensity of the attack in scenario 3

Scenario 1: Low attack intensity (k=0.1)

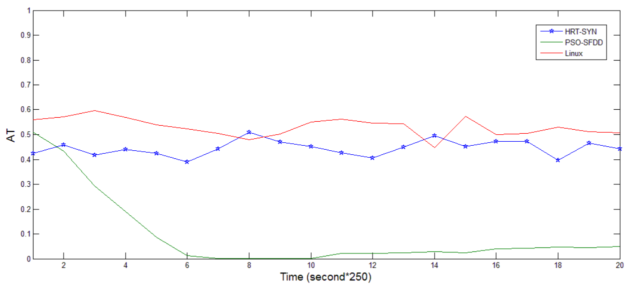
In order to study behavior of HRT _SYN against a low rate attack, we consider a scenario in which attack intensity is considered as k=0.1. Simulation results are given in Figure 3. This figure shows that defense performance of HRT_SYN are as performance of Linux TCP. According to Figures 3.a Ploss is zero for HRT_SYN, PSO_SFDD and Linux TCP during the simulation time. This means that since attack requests have low rate in this scenario, backlog buffer will never be full and hence no request will be blocked. Figures 3.b and 3.c shows interesting differences among HRT_SYN and Linux TCP in one side and PSO_SFDD on the other side. PSO_SFDD decreases both AT and RT. This has root in this fact that when PSO_SFDD faces with any attack connections it strictly reduces the allowed residence time of all half open connections. Obviously this affects both regular and attack connections and hence reduces both RT and AT. But since in this scenario attack rate is low always there is some available unused capacity for coming connection requests, and HRT_SYN will not be activated to eject the oldest half-open connection and hence RTT and AT will be increased.



a. Connection Requests Loss Probability



b. Regular Requests Residence Time

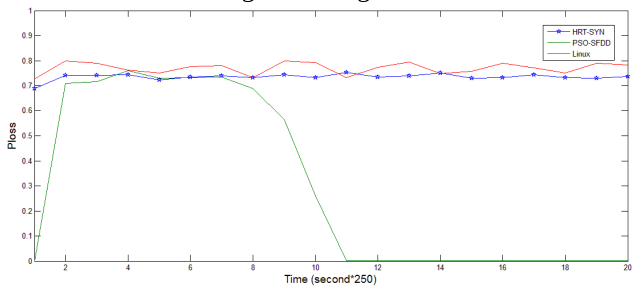


c. Attack Requests Residence Time

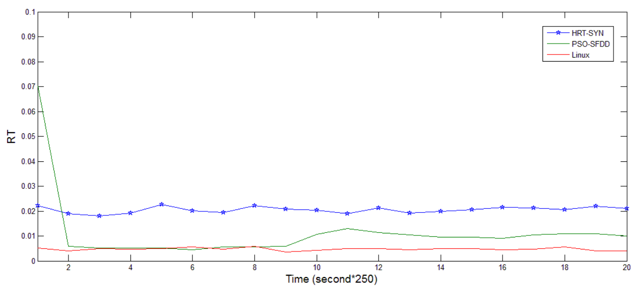
Fig. 3. Comparision of PSO_SFDD, HRT_SYN and Linux TCP for k=0.1

Scenario 2: High attack intensity (k=3)

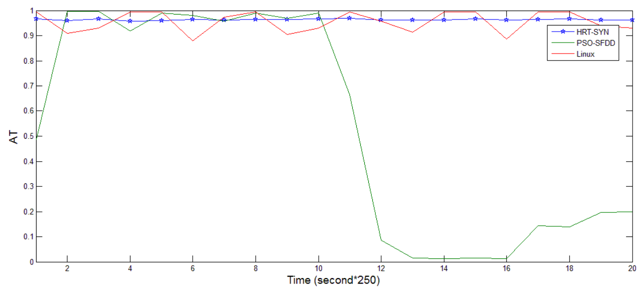
In this scenario the victim server goes under a heavy SYN flooding attack in which the number of attack requests is thrice the number of regular requests. Simulation results are given in Figure 4.



a. Connection Requests Loss Probability



b. Regular Requests Residence Time



c. Attack Requests Residence Time

Fig. 4 Comparision of PSO_SFDD, HRT_SYN and Linux TCP for k=3

This figure shows that PSO_SFDD outperforms Linux TCP and HRT_SYN in terms of Ploss and AT, but the best RT is achieved by HRT_SYN. Since HRT_SYN eject only the connection with highest residence time, it likely affects only attack connections and hence it increases share of regular connections share from connection capacity. This is due to the fact that HRT_SYN terminates those half-open connections that have been established by attackers and occupy the victim resources for lomg. As shown, HRT_SYN has lower *Ploss* in compare with Linux TCP, because it doesn't allow attack requests to remain inside the system and ejects them to provide free capacity for coming connections requests. Since PSO_SFDD decreases maximum holding time of half open connections, attack connection will be terminated sooner and hence incoming requests will have more chance to establish a connection. This is the reason for lower *Ploss* of PSO-SFDD in Figure 4.

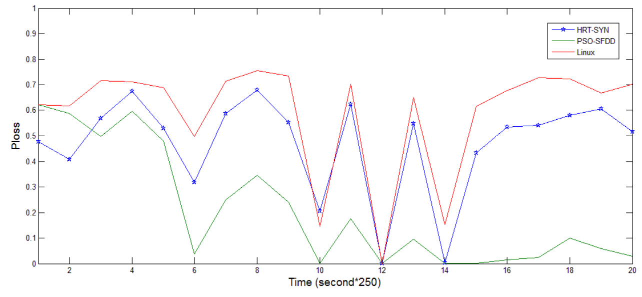
Scenario 3: Variable attack intensity (k fluctuates between 0 and 3)

In this scenario, the attack intensity changes during the time. As shown in Figure 2 the attack intensity oscillates in range of [0, 3] during the simulation time. Figure 5 shows that HRT_SYN improves the server performance in term of RT in compare with PSO_SFDD and TCP Linux. But as the previous scenario PSO_SFDD outperforms Linux TCP and HRT_SYN in terms of *Ploss* and AT. Note that RT shows share of regular connections from TCP resources and hence, it can be said that HRT_SYN provide better performance and serves regular connections more efficiently.

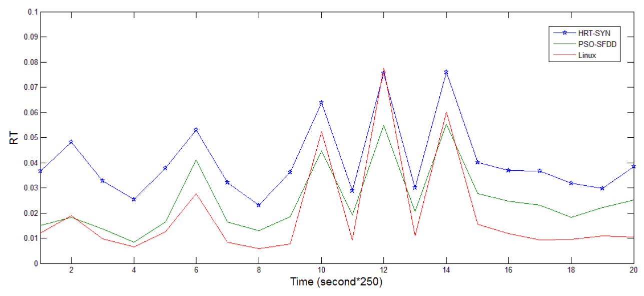
6. Conclusions

This paper proposed a scheduling-based approach to defend against SYN-flooding DoS attacks. We used a simple queuing model, to show important metrics of a computer system which is under SYN flooding attacks. Then, we used scheduling algorithm to draw a successful defense against SYN flooding attack requests. This scheduling algorithm ejects the half connection with the longest duration, when number of half open connections reaches to the maximum value. Simulation result showed that the proposed defense strategy behaves extremely better than Linux TCP and PSO-SFDD algorithms. Since HRT_SYN accepts all arrived connection requests and ejects the oldest half open connection its success is reflected by RT.

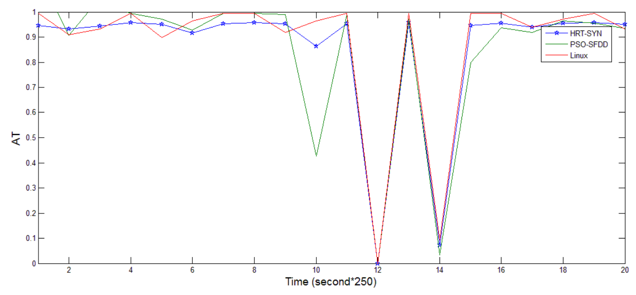
As a direction for future works we can focus to design a more efficient scheduling algorithm to provide better performance for the scheduling based defense scheme. This scheduling algorithm can consider other parameters such as statistical characteristics of previous regular and attack connections.



a.Connection Requests Loss Probability



b. Regular Requests Residence Time



c. Attack Requests Residence Time

Fig. 5. Comparision of TCP Linux, PSO_SFDD and Linux TCP for variable traffic intensity k= [0, 3]

References

[1] X. Wang, W. Guo, W. Zhang, M. Khurram Khan, Cryptanalysis and improvement on a parallel keyed hash function based on chaotic neural network, Telecommunication Systems, 2013.
[2] J. Kim, K. Kim, A scalable and robust hierarchical key establishment for mission-critical applications over sensor networks, Telecommunication Systems, 2013.
[3] J. Teo, Ch. How Tan, J. Mee Ng, Denial-of-service attack resilience dynamic group key agreement for heterogeneous networks, Telecommunication Systems, 2007.
[4] Sh. Chen, Y. Ling, R. Chow, Y. Xia, AID: A global anti-DoS service, Computer Networks, Vol. 51, 2007.
[5] L.A. Gordon, M.P. Loeb, W. Lucyshyn, R. Richardson, CSI/FBI computer crime and security survey, Computer Security Institute, 2005.
[6] T. Anderson, T. Roscoe, D. Wetherall, Preventing internet denial-ofservice with capabilities. Computer Communications Review, 2004.
[7] W. Allen, G. Marin, The LoSS technique for detecting new denial of service attacks. In: SoutheastCon; 2004.
[8] L. Feinstein, Statistical approaches to DDoS attack detection and response. In: DARPA information survivability conference and exposition proceedings, 2003.
[9] R. B. Blazek, H. Kim, B. Rozovskji, A. Tartakovsky, A novel approach to detection of denial of service attacks via adaptive sequential and batch sequential change point detection methods, in: IEEE Workshop on Information Assurance and Security, 2001.
[10] P. Mell, Donald Marks, Mark McLarnon, A denial-of-service resistant intrusion detection architecture, Computer Networks, 2000.
[11] A. Nadeem, M. Howarth, Protection of MANETs from a range of attacks using an intrusion detection and prevention system, Telecommunication Systems, 2013.
[12] Y. Wang, Ch. Lin, Q.L. Li, Y. Fang, A queuing analysis for the denial of service (DoS) attacks in computer networks, Computer Networks, 2007.
[13] D. Geneiatakis, N. Vrakas, C. Lambrinoudakis, Utilizing bloom filters for detecting flooding attacks against SIP based services, Computers & Security, 2009.
[14] S.H.C. Haris, R.B. Ahmad, "M.A.H.A. Ghani,"Detecting TCP SYN Flood Attack based onAnomaly Detection," Second International Conference on Network Applications, Protocols and Services, 2010.
[15] W. Chen, D. Yeung, Throttling spoofed SYN flooding traffic at the source, Telecommunication Systems, 2006.
[16] B. Xiao, W. Chen, Y. He, An autonomous defense against SYN flooding attacks: detect and throttle attacks at the victim side independently, Parallel and Distributed Computing, 2008.
[17] M. Long, C.H. Wu, J.Y. Hung, Denial of service attacks on network-based control systems: impact and mitigation, IEEE Transactions on Industrial Informatics, 2005. [6] P. Mel, D. Marks, M. McLarnon, A denial-of-service resistant intrusion detection architecture, Computer Networks , 2000.
[18] M. Korczy´nski, L. Janowski, and A. Duda, "An Accurate Sampling Scheme for Detecting SYN Flooding Attacks and Port scans," IEEE ICC, 2011.
[19] A. Hussain, J. Heidemann, C. Papadopoulos, A framework for classifying denial of service attacks, USC Information Sciences Institute, 2003.
[20] H. Safa, M. Chouman, H. Artail, M. Karam, A collaborative defense mechanism against SYN flooding attacks in IP networks, Network and Computer Applications, 2008.

[21] M. Hamdi, N. Boudriga, Detecting denial-of-service attacks using the wavelet transform, Computer Communications, 2007.
[22] V.A. Siris, F. Papagalou, Application of anomaly detection algorithms for detecting SYN flooding attacks, Computer Communications, 2006.
[23] S. Jamali, G. Shaker, PSO-SFDD: Defense against SYN flooding DoS attacks by employing PSO algorithm, Computers and Mathematics with Applications, 2012.
[24] P. Brucker, Scheduling Algorithms, Fifth Edition, Springer-Verlag, 2006.
[25] M. Pinedo, Scheduling theory, algorithms and systems. Englewood Cliffs, NJ: Prentice-Hall; 1995.
[26] S. Eilon, I. Chowdhury, "Minimizing waiting time variance in the single machine problem", Management Science, 1977.
[27] D. Koutsonikolas, S.M. Das, Y.C. Hu, An interference-aware fair scheduling for multicast in wireless mesh networks, Journal of Parallel and Distributed Computing, 2008.
[28] S. Singh, U. Madhow, E.M. Belding, Beyond proportional fairness: a resource biasing framework for shaping throughput profiles in multi hop wireless networks, INFOCOM, 2008.

Shahram Jamali is an associate professor leading the Autonomic Networking Group at the department of computer engineering, University of Mohaghegh Ardabili. He teaches on computer networks, network security, computer architecture and computer systems performance evaluation. Dr. Jamali received his M.Sc. and Ph.D. degree from the Dept. of Computer Engineering, Iran University of Science and Technology in 2001 and 2007, respectively. Since 2008, he is with Department of Computer Engineering, University of Mohaghegh Ardabil and has published more than 60 conference and journal papers.

Gholam Shaker received his B.Sc. from Payam Noor University Ardabil, Iran, a M.Sc. (2011) from IAU (Islamic Azad University-Zanjan branch) University. All are in computer engineering. He is currently teaching in department of computer science at IAU University. His research interests include Quality of service, security systems and communication networks.