DEFENSE AGAINST SYN FLOODING ATTACK USING PSO ALGORITHOM

Utpal Chandra Mohanta
CSE 04906421

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of BSc in Computer Science and Engineering.

DEPARTMENT OF COMPUTER SCIENCE
STAMFORD UNIVERSITY BANGLADESH

November, 2016

ABSTRACT

SYN flooding attack is a threat that has been designed based on vulnerabilities of the connection establishment phase of the TCP protocol. In this attack, some sources send a large number of TCP SYN segments, without completing the third handshake step to quickly exhaust connection resources of the victim server. Hence, a main part of the server's buffer space is allocated to the attack half open connections and incoming new connection requests will be blocked. This paper proposes a novel framework, in which, the defense issue is formulated as an optimization problem. Then it employs the particle swarm optimization (PSO) algorithm to solve this optimization problem. Our theoretical analysis and packet-level simulations show that the proposed defense strategy decreases the number of blocked TCP connection requests and cuts down share of attack connections from the buffer space.

ACKNOWLEDGEMENTS

Frist of all I would like to thank the almighty GOD. Today I am successful in completing my work with such ease because He gave me the ability, chance and co-operating supervisor.

I would like to take the opportunity to express our gratitude to Lecturer Maliha Mahbub, my respected supervisor. Although she was loaded with several other activities, she gave me more than enough time in this work. She not only gave me time but also proper guidance and valuable advice whenever I faced with any difficulties. Her comments and guidance helped me in preparing my thesis report.

I wish to express my sincere gratitude to Lecturer, Tarin Kazi for her constant guidance throughout the course of the work and many useful discussions which enabled me to know the subtleties of the subject in proper way.

And I would also like to take the opportunity to express our gratitude to our honorable Chairman, Prof. Dr. Kamruddin Md. Nur, Department of Computer Science and Engineering.

Last of all I am grateful to my family ; Who are , always with me in my every step of life

DECLARATION

I, hereby, declare that the work presented in this Thesis is the outcome of the investigation performed by me under the supervision of Maliha Mahbub, Lecturer, Department of Computer Science, Stamford University Bangladesh. I also declare that no part of this Thesis and thereof has been or is being submitted elsewhere for the award of any degree or Diploma.

Countersigned

……..

(Maliha Mahbub)

Supervisor

Signature

…….

(Utpal Chandra Mohanta)

Candidate

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

There are several types of important attacks, such as the worm, virus, Trojan horse and especially Denial of Service (DoS), each of which causes crucial problems to usual business operations. In spite of extensive efforts to provide robustness for the systems against DoS attack, this attack is yet a serious problem on the Internet. Traditionally, DoS attacks aim at degrading the availability and quality of services, by consuming the service resources to make it unavailable. In doing this, DoS attacks may send to the victim a high-rate traffic that exhausts service resources. Statistical evaluations show that DoS ranks at the fourth place in the list of the most venomous attack classes against information systems. Recently, many efforts have been made, in parallel with the evolution of DoS attacks, in the field of prevention and detection in networking security. In terms of prevention, some of the approaches that have been proposed include egress or ingress filtering, disabling unused services, and honey pots. In other works, a congestion pricing approach] and a router-based technique. have been employed to neutralize DoS attacks.

SYN FLOOD ATTACK:

The recent DoS attacks on popular web sites like Yahoo and eBay and their consequent disruption of services have exposed the vulnerability of the Internet to Distributed Denial of Service (DDoS) attacks. It has been shown that more than 90% of the DoS attacks use TCP. The TCP SYN flooding is the most commonly-used attack. Not only the Web servers but also any system connected to the Internet providing TCP-based network services, such as FTP servers or Mail servers is susceptible to the TCP SYN flooding attacks. A TCP connection is established in what is known as a 3-way handshake. When a client efforts to start a TCP connection to a server, first, the client requests a connection by sending a SYN packet to the server. Then, the server returns a SYN–ACK, to the client. Finally, the client acknowledges the SYN–ACK with an ACK, at which point the connection is established and data transfer commences. In a SYN flooding attack, attackers use this protocol to their benefit. The attacker sends a large number of SYN packets to the server. Each of these packets has to be handled like a connection request by the server, so the server must answer with a SYN–ACK. The attacker then has two options. One is simply not to answer to the SYN–ACK, which will cause the server to have a half-open connection [1]. This would allow the server to block any further packets from the attacker's IP address, ending the attack prematurely. Then again, the attacker spoofs the IP address of some unsuspecting client. The server logically answers to this IP address, but the legitimate client actually residing at this IP address will decline this SYN–ACK as it did not initiate the connection. The result is that the server is left waiting for a reply from a large number of connections. Since resource of any system is limited, then, there are a limited number of connections a server can handle. Once all of these connections are active, waiting for replies that will never come, no new connections can be established whether valid or not. Note that SYN flooding attacks aim to exhaust TCP buffer space and do not affect the parameters such as link bandwidth and processing resources.

There are some proposed defenses for this attack. Zuquete proposes SYN cookies to defense against SYN flooding attacks. Some other works have used mathematical models to analytically study of the DoS attacks. Chang proposed a simple queuing model for the SYN flooding attack. Long et al. proposed two queuing models for the DoS attacks in order to obtain the packet delay jitter and the loss probability. Wang et al. studies the DoS attacks analytically by using a more general queue model, a two-dimensional embedded Markov chain, which can more accurately capture the dynamics of the actual DoS attacks[2]. In , Gligor proposed that a Maximum Waiting Time (MWT) must be related with every service presented by a computer system. Response time refers to the delay between the request for a service and its provision, whereas, MWT equals to the maximum acceptable response time. In the nonexistence of DOS attacks, a user request should stop within the MWT on a machine. Warrender and colleagues, suggested a general host-based intrusion detection model, which analyzes system call sequences to discover anomalies. They claim that their system can detect DOS attacks. In this method, if a program uses up extra of a resource, then other programs will suffer. Hussain and Blazek evaluated traffic arrivals and corresponding ramp-up activities in. To detect attacks, packet rate against time is analyzed instead of only the packet header. This is done, in order that IP address spoofing cannot deceive the attack detection procedure. Khan and Traore analyzed the influence of DoS attacks on three parameters: the queue-growth-rate, the arrival rate, and the response time, which were used for the attack detection.

OUR APPROACH:

As a novel approach to secure computer systems against the SYN flooding DoS attacks, this paper formulates the defense issue as an optimization problem and employs the PSO algorithm to find an optimal solution for this problem. For this purpose, I first use queuing theory to model the under-attack server and then formulate the defense strategy based on the optimization theory. The derived solution leads to a dynamic defense strategy which monitors the system performance continually and then tries to direct the system to the best defense position by appropriate setting of some system parameters. A preliminary version of this work has been published in. This paper takes one giant leap to complete this research. For this purpose, after that it designs a PSO-based defense mechanism, evaluates it from two different points of view. In one hand, it gives a theoretical analysis and shows that the proposed algorithm certainly converges to an optimal point in which the server achieves its possible best performance. On the other hand, it implements the algorithm in matlab environment by some modifications over TCP sink and TCP full modules and then evaluates it by a packet level simulative study which shows its success against SYN flooding attacks.

The rest of the paper is organized as follows. In Section I present an introduction to PSO method. In Section, we discuss how the under-attack server can employ PSO algorithm to defend against SYN flooding attacks. Section gives a theoretical analysis to examine convergence of the proposed algorithm. Implementation issues and the simulation setup are discussed in Section. Section brings simulation results and finally Section 7 presents concluding remarks.

Chapter 2

RELATED WORK

To counter SYN flooding attacks, several defense mechanisms have been proposed, such as:

J. Lemon et al. [3] 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. Using this result, we described a new attack detection method taking the time variance of arrival traffic into consideration. Simulation results show that our method can detect attacks quickly and accurately regardless of the time variance of the traffic.

D.J Bernstein et al. [4] 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.

C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram and D. Zamboni, et al. [5] offers protection against SYN flooding for all hosts connected to the same local area network, independent of their operating system or networking stack implementation.

S. Gavaskar, R. Surendiran and DR. E. Ramaraj et al. [6] have used three counters algorithm to detect and migrate against TCP flood Attack.

All of these defense mechanisms are installed at the firewall of the victim server or inside the victim server, thereby providing no hints about the sources of the SYN flooding. They have to rely on the expensive IP trace back to locate the flooding sources. Because the defense line is at, or close to, the victim, the network resources are also wasted by transmitting the flooding packets.

Moreover, these defense mechanisms are stateful, i.e., states are maintained for each TCP connection or state computation is required. Such a solution makes the defense mechanism itself vulnerable to SYN flooding attacks. Recent experiments have shown that a specialized firewall, which is designed to resist SYN floods, became futile under a flood of 14,000 packets per second [7]. The stateful defense mechanisms also degrade the end-to-end TCP performance, e.g., incurring longer delays in setting up connections. In the absence of SYN flooding attacks, all the overheads introduced by the defense mechanism become superfluous. We, therefore, need a simple stateless mechanism to detect SYN flooding attacks, which is immune to the SYN flooding attacks. Also, it is preferred to detect an attack early near its source, so that one can easily trace the flooding source without resorting to expensive IP trace back.

In our paper we have used Particle Swarm optimization to defense against SYN flood attack. We have used an objective value which was evaluated by the PSO by updating the present value of the time for which a request is holding idly in the buffer queue and is it completing the full data transmission or not in every iteration for each request. And based on this, we have maximized the buffer space so that more valid request can arrive and minimized the buffer time so that no request can stay in the buffer queue for a long time.

Chapter 3

OBJECTIVE AND SCOPE

3.1 SYN ATTACK

3.1.1 Introduction

These days many people do their job by using computers. These computers could connect each other through internet, which is based on TCP/IP. However, Transmission Control Protocol (TCP) has weakness when computers connected. Using this weakness anyone can attack the system. This attack is called TCP SYN flooding attack.

3.1.2 Transmission Control Protocol (TCP)

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite has become the industry-standard method of interconnecting hosts, networks, and the Internet. As such, it is seen as the engine behind the Internet and networks worldwide.

Although TCP/IP supports a host of applications, both standard and nonstandard, these applications could not exist without the foundation of a set of core protocols. Additionally, in order to understand the capability of TCP/IP applications, an understanding of these core protocols must be realized.

The TCP three-way handshake in Transmission Control Protocol (also called the TCP-handshake; three message handshake and/or SYN-SYN-ACK) is the method used by TCP set up a TCP/IP connection over an Internet Protocol based network. TCP's three way handshaking technique is often referred to as "SYN-SYN-ACK" (or more accurately SYN, *SYN-ACK*, ACK) because there are three messages transmitted by TCP to negotiate and start a TCP session between two computers. The TCP handshaking mechanism is designed so that two computers attempting to communicate can negotiate the parameters of the network TCP socket connection before transmitting data such as SSH and HTTP web browser requests.

This 3-way handshake process is also designed so that both ends can initiate and negotiate separate TCP socket connections at the same time. Being able to negotiate multiple TCP socket connections in both directions at the same time allows a single physical network interface, such as Ethernet, to be multiplexed to transfer multiple streams of TCP data simultaneously.

The three-way handshake mechanism is below.

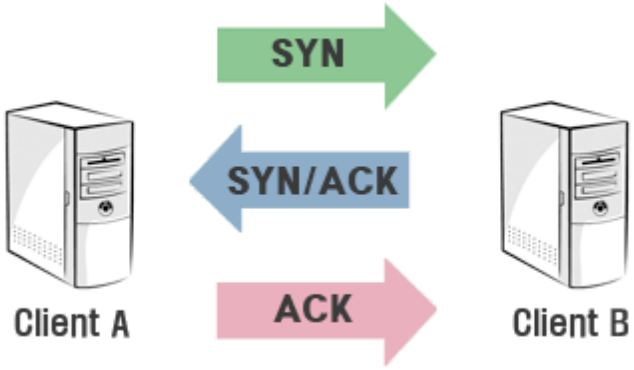

Fig-3.1: *TCP Three-way Handshake*

Step1: Machine A wants to initiate a connection with machine B, So machine A sends a segment with SYN(Synchronize Sequence Number). This segment will inform the machine B that Machine A would like to start a communication with Machine B and informs machine B what sequence number it will start its segments with.

Sequence Numbers are mainly used to keep data in order.

Step2: Machine B will respond to Machine A with "Acknowledgment" (ACK) and SYN bits set. Now machine B's ACK segment does two things; they are as below.
1. It acknowledges machine A's SYN segment.
2.It informs Machine A what sequence number it will start its data with.

Step 3: Now finally machine A Acknowledges Machine B's initial sequence Number and its ACK signal. And then Machine A will start the actual data transfer.

Initial sequence Numbers are randomly selected while initiating connections between two devices.

3.2 TCP SYN Flood Attack

TCP SYN Flood attack uses the three-way handshake mechanism. At the first of the attack client A, an attacker, sends a SYN packet to client B. Then client B sends a SYN/ACK packet to client A. As a normal three-way handshake mechanism client A should send an ACK packet to client B, however, client A does not send an ACK packet to client B. In this

case client B is waiting for an ACK packet from client A. This status of client B is called “half open”. This kind of incomplete connection is stored in Backlog Queue. After 75 seconds the incomplete connection is removed from Backlog Queue and it is disconnected.

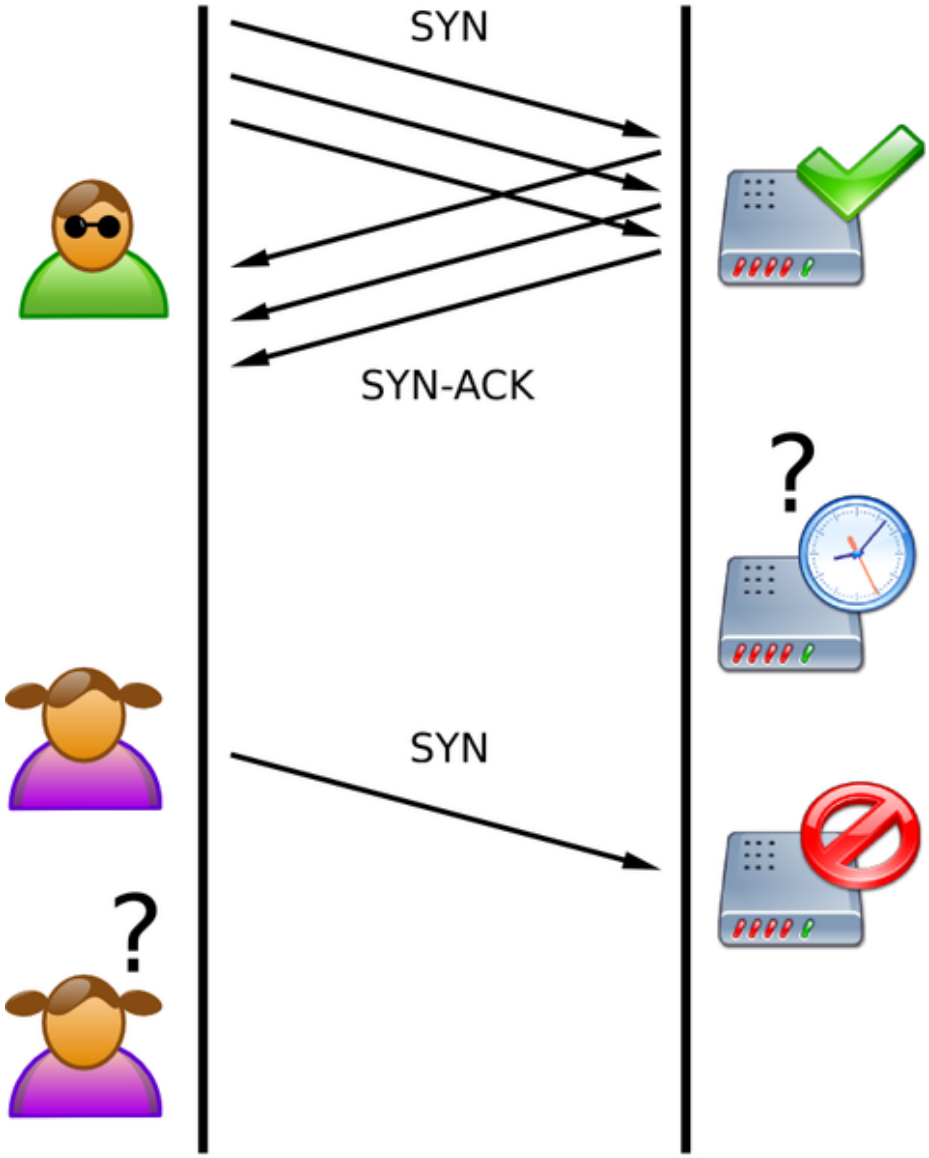

Fig-3.2: *TCP SYN flood attack*

However, if client A sends lots of SYN packets before client B removes incomplete connections from Backlog Queue, then Backlog Queue in client B is overflowed. In this case client B cannot accept TCP connection at all. This is called Denial-of-Service, and this type of attack is TCP SYN Flood Attack.

3.3 Particle swarm optimization algorithm

Particle swarm optimization is a search algorithm that has been inspired from bird flocking and fish schooling. This population based algorithm has been designed and introduced by Kennedy and Eberhart in 1995. The basic PSO has found many successful applications in a number of problems including standard function optimization problems, solving permutation problems and training multi-layer neural networks. The PSO algorithm contains a swarm of particles in which each particle indicates a potential solution. The particles fly through a multidimensional search space in which the position of each particle is adjusted according to its own experience and the experience of its neighbors. PSO system combines local search methods (through self-experience) with global search methods (through neighboring experience), attempting to balance exploration and exploitation. As in evolutionary computation paradigms, the concept of fitness is employed and candidate solutions to the problem are termed particles, each of which adjusts its flying based on the flying experiences of both itself and its companion. PSO is an approach to problems whose solutions can be represented as a point in an n-dimensional solution space. A number of particles are randomly set into motion through this space. At each iteration they observe the fitness of themselves and their neighbors and emulate successful neighbors (those whose current position represents a better solution to the problem than theirs) by moving toward them. The position and velocity of particle i at iteration k can be respectively expressed by notations (1) and (2).

$$X_i(K) = [X_{i1}(K), X_{i2}(K), X_{i3}(K), \dots\dots\dots\dots, X_{iN}(K)] \quad (1)$$

$$V_i(K) = [V_{i1}(K), V_{i2}(K), V_{i3}(K), \dots\dots\dots\dots, V_{iN}(K)] \quad (2)$$

Particle i keeps track of its coordinates in the solution space which are associated with the best solution that has achieved so far by that particle. This value is called local best Lbesti. Another best value that is tracked by the PSO is the best value obtained so far by any particle in the neighborhood of that particle. This value is called global best Gbest. The basic concept of PSO lies in accelerating each particle toward its local best and the global best locations. Various schemes for grouping particles into competing, semi-independent flocks can be used, or all the particles can belong to a single global flock. This extremely simple approach has been surprisingly effective across a variety of problem domains.

In the basic particle swarm optimization algorithm, particle swarm consists of “n” particles, and the position of each particle stands for the potential solution in D-dimensional space .the particles change its condition according to the following three principles :

(1)To keep its inertia

(2) to change the condition according to its most optimist position
(3)to change the condition according to the swarm's most optimist position .

The position of each particle in the swarm is affected both by the most optimist position during its movement (individual experience) and the position of the most optimist particle in its surrounding (near experience).when the whole particle swarm is surrounding the particle ,the most optimist position of the surrounding is equal to the one of the whole most optimist particle ; this algorithm is called the whole PSO. If the narrow surrounding is used in the algorithm ,this algorithm is called the partial PSO. Each particle can be shown by its current speed and position ,the most optimist position of each individual and the most optimist position of the surrounding . In the partial PSO. the speed and position of each individual and the most optimist position of the surrounding . In the partial PSO.the speed and position of each particle change according the following equality

The velocity and position of particle i at iteration k + 1 can be calculated according to the following equations:

$$V_i(K+1) = wV_i(K) + c_1 r_1\big(\ Lbest_i(K) - \ X_i(K)\big) + \ c_2 r_2\big(\ Gbest_i(K) - \ X_i(K)\big) \quad (3)$$

$$X_i(K+1) = X_i(K) + V_i(K+1) \quad (4)$$

where w is the inertia weight, c1 and c2 are constants which determine the influence of the local best position Lbesti(k) and the global best position Gbest(k). Parameters r1 and r2 are random numbers uniformly distributed within. In compare with other meta-heuristic, PSO has better search capacity, performance and accuracy. It has absolute advantages in the continuous variable function optimization problems. The calculation in PSO is very simple and hence its CPU requirement is quite low. It also is easy to implement and there are few parameters to adjust. Due to these advantages this paper uses PSO as its optimizer.

3.4 Scope of the problem

SYN flooding attack tries to exhaust the buffer space in order to maximize number of blocked connections. Hence, the defense scheme logically must try to prevent the system from allocating buffer space to attack connections and minimize number of lost connections. In this paper we have implemented the PSO algorithm in order to dynamically calculate very important parameters: holding time of the packet in the queue and the maximum number of half open connection that can reside in the queue. The optimized best two position of these parameters are provided to the server so that it can tune these values to get the best defense against the SYN flood attack

Chapter 4

METHODOLOGY

4.1 Framework

Consider a server offering some TCP-based services that is subject to SYN flooding attacks. As mentioned before, SYN flooding attack uses the 3-way handshaking protocol running in the TCP connection establishment phase. In a SYN flooding attack, attacker sends a large number of SYN packets to the server. Each of these packets has to be handled like a connection request by the server, so the server must answer with a SYN–ACK and must allocate a memory space to this half-open connection. In other words, attacker tries to exhaust the memory space allotted to the TCP protocol. By this background, in modeling of this attack, we consider only one resource i.e. the memory space of the victim server and since it has limited capacity we consider it as a queuing system. Employing queuing theory, we give an abstraction for modeling SYN flooding attack. In this model, all connection requests share a same backlog queue. When a request arrives at the system, receives a buffer space of the backlog queue upon finding an inactive buffer space and is blocked otherwise. Assume that in this computer each half open connection is held for at most the period of time h seconds and at most m concurrent half-open connections are allowed. It is also assumed that a half-open connection for a regular request packet is held for a chance time which is exponentially distributed with parameter l. The arrivals of the regular connection request packets and the attack connection request packets are both Poisson processes with rates λ_1 and λ_2, respectively. The two arrival processes are independent of each other and of the holding times for half-open connections.

Obviously, when the system is under SYN flooding attack, number of pending connections increases and in a point in which there is no more room for new connections to be saved, the arriving connection requests will be blocked. In the other word, 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. It goes without saying that the time duration at which the system reaches to this saturation point is affected by the values of m and h. For example, when h becomes larger, attack half open connections live long duration of time causing to consumption of the connection opportunities and increased number of blocked connections. In this case, less percentage of the buffer space is allocated to legal requests, and major part of this space will be occupied by the attacker requests.

Considering the fact that values of h and m play important roles in SYN flooding attack, this paper chooses h and m as its control parameters and employs the PSO algorithm to tune them dynamically toward the best defense position. An abstract model for this defense approach has been shown in Fig. 5.1. In this figure the TCP-based server has been modeled by a single queue, in which, the connection requests are queued, waiting for the service. This figure shows that the system performance is measured continually and then is formulated as an

objective function. PSO then uses this objective function to find the best values for h and m parameters.

Here is the abstract model for the Self-securing Server Running PSO_SYN:

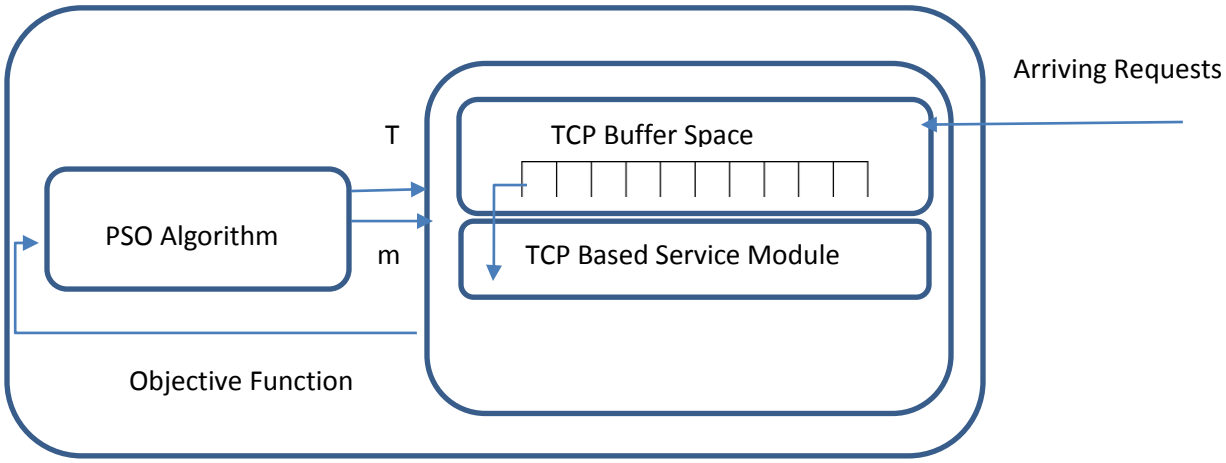

Fig-4.1: Abstract model for the Self-securing Server Running PSO_SYN.

It is clear that definition of an appropriate objective function is vital. Remember that SYN flooding attack tries to exhaust the buffer space in order to maximize number of attacking request. Hence, the defense scheme logically must try to prevent the system from allocating buffer space to attack connections and minimize number of attacking request.

In the other words, the defense scheme can be defined as an optimizer that tries to minimize and Attack Requests Buffer Occupancy Percentage and maximize the Regular Requests Buffer Occupancy and the remaining occupancy for arrival requests.

OBJECTIVE FUNCTION:

$$\max \frac{Regular\ half\ open\ occupancy}{Attacking\ half\ open\ occupancy} * \text{Remaining occupancy for arrival request}$$

The connection loss probability, a basic measure for assessing the performance of the system under DoS attacks. Each arriving connection request packet must be blocked once there have

already been m pending connections in the system. Therefore, it can be described as ratio of the number of dropped requests to the all arrived requests.

The buffer occupancy percentage of attack requests that can be described as mean ratio of the number of attack half-open connections (those connections that are closed after T seconds) to all half-open connections.

The regular requests buffer occupancy percentage and is characterized by the mean ratio of the number of regular half-open connections (those connections that are closed prior to T seconds) to all half-open connections.

As the next step of our design we need to draw an appropriate mapping between the problem solutions and PSO particle. We consider the (T, m) parameters of the server as the PSO particle position. By this mapping, each particle tries to tune its positions i.e. (T, m) seeking the best performance. Particles' performance is formulated as the following objective function that links the optimization algorithm with the design requirements:

4.2 Objective Function

Let,
Regular packet arrival rate = λ_1
Attacking packet arrival rate = λ_2

So, attack intensity, k = $\frac{\lambda_2}{\lambda_1}$

Let, the maximum number of time for which packets can be held in queue = T second
And maximum number of half open connection = m

So the number of packets in a queue, n= λ_T
Number of regular packet in a queue, n_r=λ_1*T
Number of attacking packet in a queue, n_a=λ_2*T

At the arrival, the capacity of the TCP buffer, m = $n_r + n_a$

When, the more packet arrives, n_r and n_a increases and we have to increase the size of m
The difference between m and the updated ($n_r + n_a$) = |(n_r+ n_a)-m|

So, the new size of m should be = m +| m - (n_r+n_a)|
As we have to increase the regular request and the size of queue and decrease the attacking packet, so we have to maximize the objective function where

Objective function = (n_r/n_a)*(m+|m - (n_r+n_a)|)
= $(\lambda_1$*t/ λ_2*t) * (m+|m - $(\lambda_1$*t/ λ_2*t)|)
= $(r_1$*t/r_2* t) * (m+|m - $(r_1$*t/r_2* t)|) (5)

Where r_1 and r_2 is the poisson's ratio of respectively λ_1 and λ_2.

This objective function reflects the design intention i.e. maximization of arrival request and the TCP buffer m and at the same time minimization of the attacking packet.
According to the PSO algorithm described in Eqs. (3) and (4), values of T and m will be updated by the following equations:

$$V_h^{k+1} = WV_h^k + c_1 r_1\left(Lbest_h^k - h^k\right) + c_2 r_2\left(Gbest_h^k - h^k\right) \quad (6)$$

$$T^{k+1} = T^k + V_h^{k+1} \quad (7)$$

$$V_m^{k+1} = WV_m^k + c_1 r_1(Lbest_m^k - m^k) + c_2 r_2(Gbest_m^k - m^k) \quad (8)$$

$$m^{k+1} = m^k + V_m^{k+1} \quad (9)$$

where ($Lbest_h$, $Lbest_m$) is the local best position and ($Lbest_h$, $Lbest_m$), is the global best position. These best positions are selected according to the objective function of (5) and based on particles' experiences. The parameters c1 and c2 determine the relative pull of Lbest and Gbest and the parameters r1 and r2 lead to stochastically varying these pulls. These parameters should be selected sensitively for efficient performance of PSO_SYN. The constants c1 and c2 represent the weighting of the stochastic acceleration terms that pull each particle toward Lbest and Gbest positions. Low values allow particles to roam far from the target regions before being tugged back. On the other hand, high values result in abrupt movement toward, or past, target regions. Hence, the acceleration constants are set to be 0.5. Suitable selection of inertia weight, w, provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution. We set w to be 0.9. These values of the parameters have been selected through a try and error procedure. Fig. 2 shows more details about operation of the proposed defense scheme.

According to this flowchart, after an initial phase, PSO_SYN enters into an iterative loop in which three basic operations are performed continually. First, the system performance is measured and the objective function is computed. Second, local and global best positions are updated by using the value achieved for the objective function. Third, next position i.e. new value for (h, m) is computed by using Eqs. (6)–(9).

4.3 Flowchart

A flowchart is given bellow:

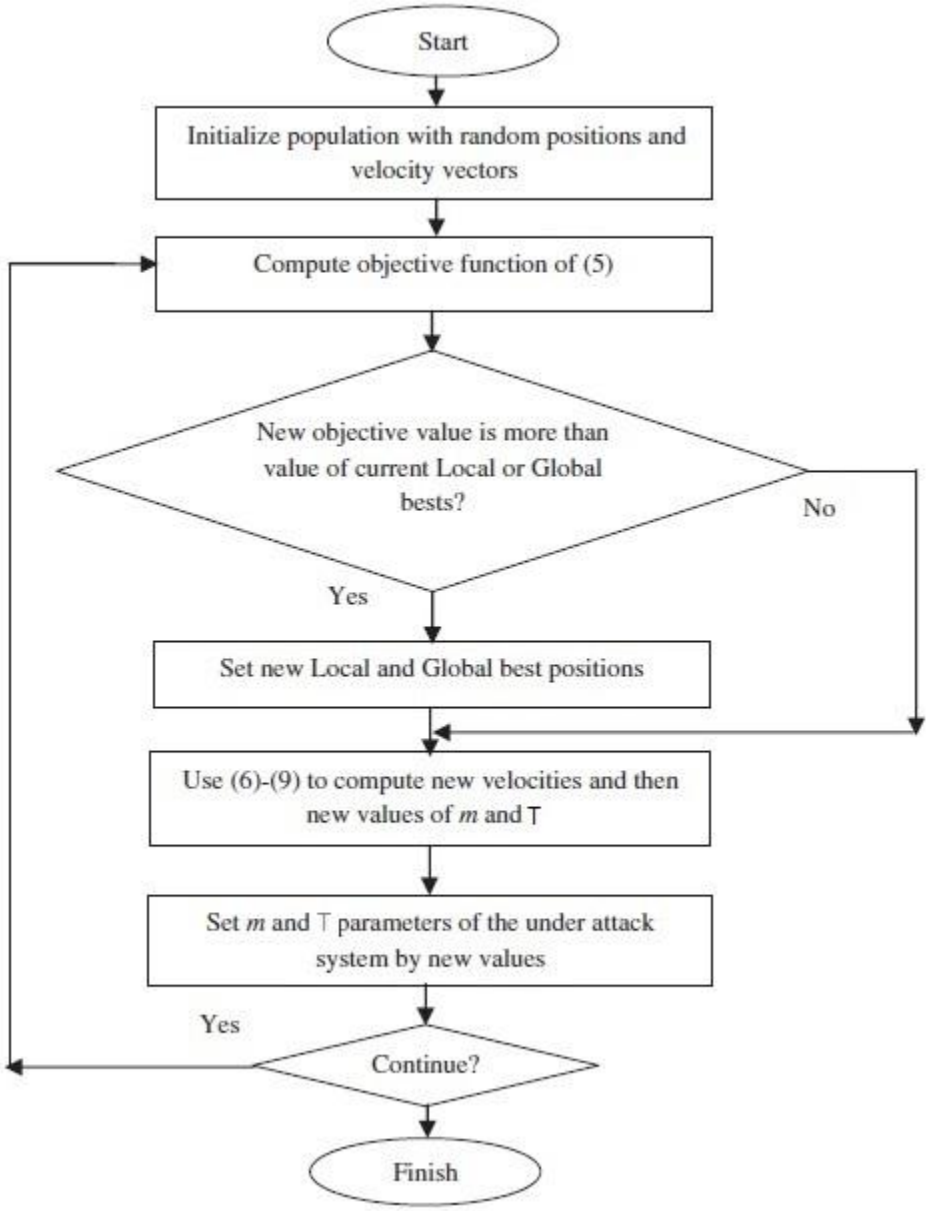

Fig. 4.2- PSO_SYN in operation.

4.4 Pseudo-code

Here is the pseudo code for PSO algorithm:

1: $k \leftarrow 1$ {initialization}
2: for m = 1 to M do
3: Generate_Solution(xm(k))
4: Initialize_Velocity(vm(0))
5: $f(xm(0)\_) \leftarrow \infty$
6: end for
7: {main loop}
8: repeat
9: {evaluate and update best solutions}
10: for m = 1 to M do
11: $f(xm(k)) \leftarrow$ Evaluate_quality(xm(k))
12: if $f(xm(k)) < f(xm(k-1)\_)$ then
13: $xm(k)\_ \leftarrow xm(k)$
14: else
15: $xm(k)\_ \leftarrow xm(k-1)\_$
16: end if
17: if $f(xm(k)) < f(x(k)\_)$ then
18: $x(k)\_ \leftarrow xm(k)$
19: else
20: $x(k)\_ \leftarrow x(k-1)\_$
21: end if
22: end for
23: for m = 1 to M do
24: for n = 1 to N do
25: $cmn(k) \leftarrow 0$
26: for all Km nearest neighbors (index p) of m do
27: $cmn(k) \leftarrow cmn(k) + (U(0, \phi)(xpn(k) - xmn(k))$
28: end for
29: $vmn(k) \leftarrow \chi * [vmn(k-1) + 1/Km * cmn(k)]$
30: $xmn(k+1) \leftarrow xmn(k) + vmn(k)$
31: end for
32: end for
33: stop_condition ← Check_stop_condition()
34: $k \leftarrow k + 1$
35: until stop_condition = false
36: return $f(x(k)\_)$, $x(k)\_$, k

Chapter 5

IMPLEMENTATION AND SIMULATION

5.1 Environment

Here we have used Matlab to execute our problem solution. LAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

5.2 Parameter Table

We have used several parameters: inertia, maximum time for a data to stay in a queue, maximum number of half open connection, correction factor etc. Here is the parameter table below:

Parameter	Symbol
Inertia	w
Regular packet	λ_1
Attacking packet	λ_2
Total number of regular request	n_r
Total number of attacking request	n_a
Total capacity in buffer	m
Poisson's distribution value for regular request	r1
Poisson's distributed value for attacking request	r2
Maximum time for a request in the queue	T
Attack intensity	k

Table-5.1: Table of Parameters

5.3 Simulation Setup

Our simulations use the topology in Fig. 3. In this network the victim server continually receives TCP connection request packets i.e. SYN packets from various sources. Some of them belong to attackers and others are for legal users.

Let Regular packet arrival rate,λ_1=100,20,30
And Attacking packet arrival rate, λ_2= 10,20,60 respectively
So, attack intensity, k = $\frac{\lambda_2}{\lambda_1}$ = 0.1,1 and 2

We used these values to show the change of time for each packet in buffer and the buffer capacity.

5.4 Simulation Result

In this section PSO_SYN is compared to TCP. As mentioned, while TCP uses static values of t = and m, PSO_SYN sets t and m dynamically based on attack intensity. For this purpose four scenarios of simulations are presented. In first scenario attack intensity is considered as k = 0.1 which means low attack intensity, in second one it is considered as k = 1 which means medium attack intensity, in third scenario it is set as k = 2 which means high attack intensityand finally in fourth scenario of our simulations, we consider a variable attack intensity that fluctuate between 0 and 2.

5.4.1 Scenario 1: Increase of t and m in Low attack intensity (k = 0.1)

The figure bellow is for time t in low attack intensity:

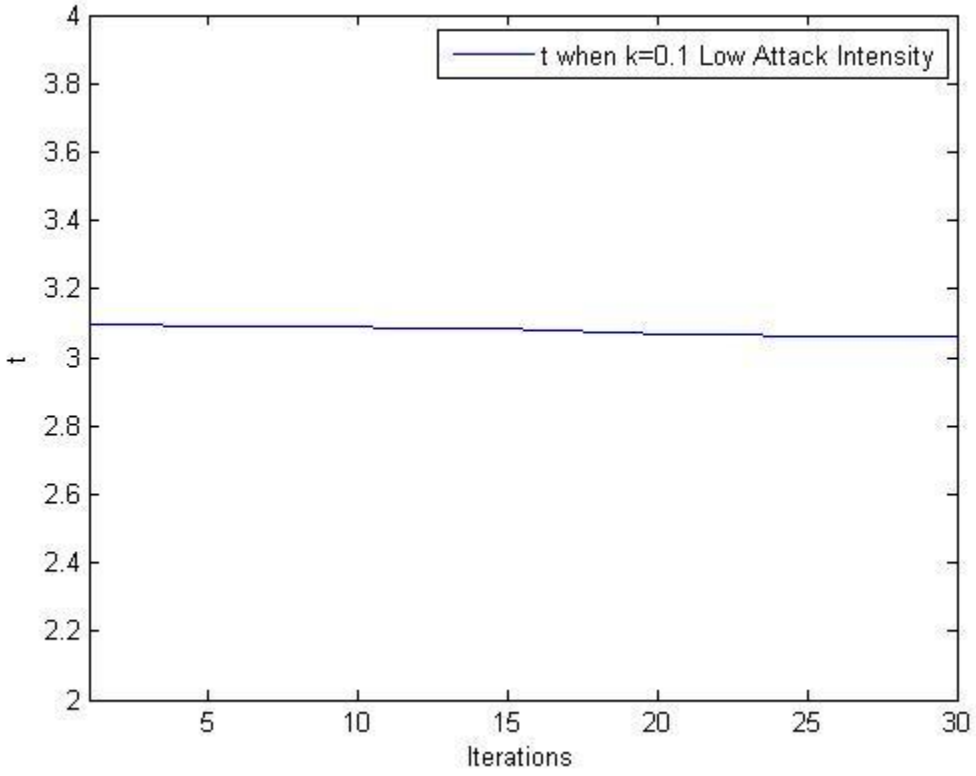

Fig-5.1: Variations of t computed by PSO_SYN in low attack intensity

In the figure the behavior of the system is shown where the victim server goes under a SYN flooding attack whose intensity is k = 0.1 i.e. low attack intensity. This figure shows the varying values of the parameter t, which is the holding time of each half open connection in the parameter. As seen in the figure, the value of parameter t is declining at a steady rate.

The figure bellow is for the capacity of buffer, m in low attack intensity:

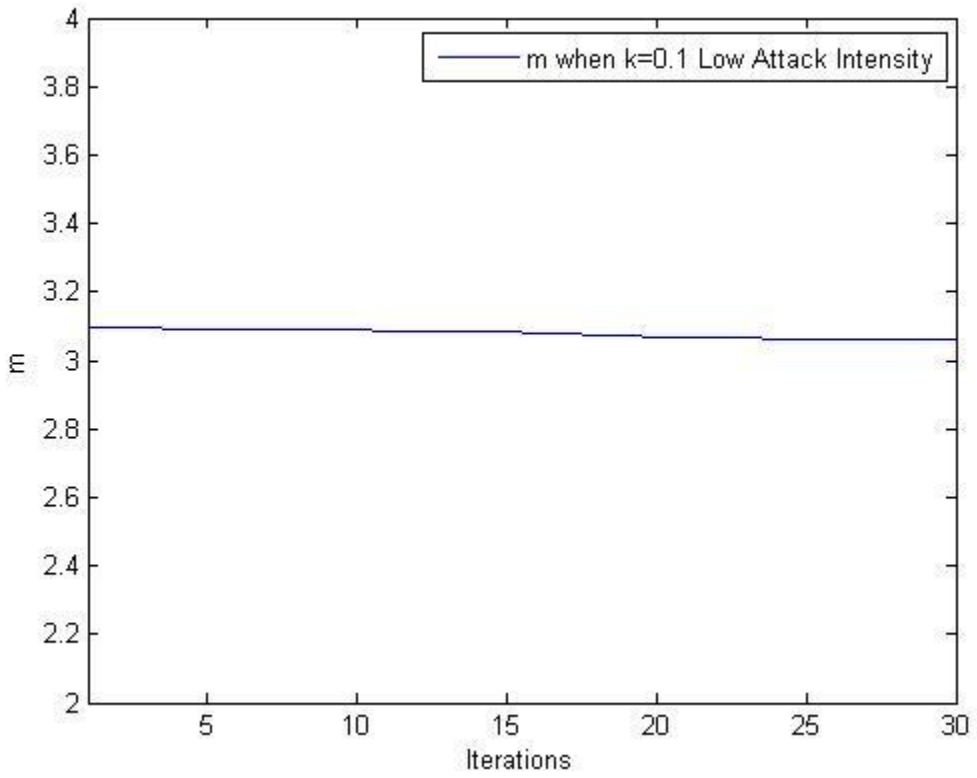

Fig-5.2: Variations of m computed by PSO_SYN in low attack intensity

The figure shows the varying values of the parameter p, which is the maximum number of half open connections allowed in the queue adjusted with PSO algorithm where the attack intensity is k= 0.1. This figure shows the varying values of the parameter m, which is the the maximum number of half open connections allowed in the queue. As seen in the figure, the value of parameter t is at a steady state.

5.4.2 Scenario 2: Increase of T and m in Medium attack intensity (k = 1)

The figure bellow is for time t in medium attack intensity:

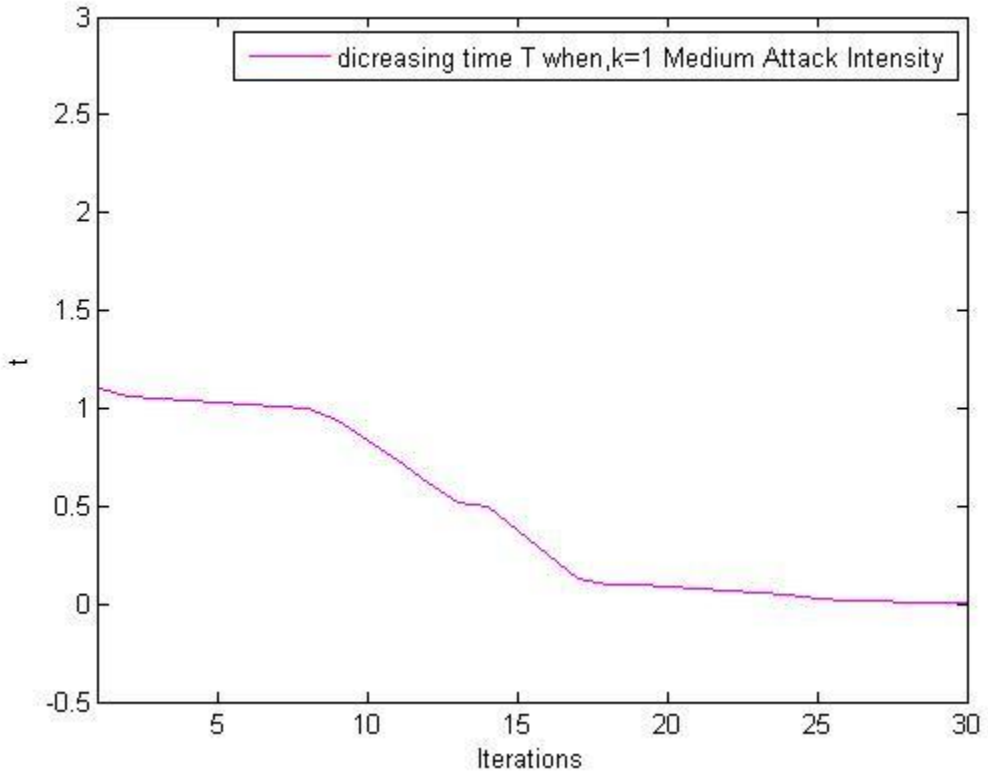

Fig-5.3: Variations of T computed by PSO_SYN in medium attack intensity

The figure shows the behavior of the system in medium attack intensity where k= 1. The figure clearly shows that when the attack intensity increases, the value of t is reduced by the PSO algorithm in order to defend the SYN attack requests.

The figure bellow is for capacity in the buffer space, m in medium attack intensity:

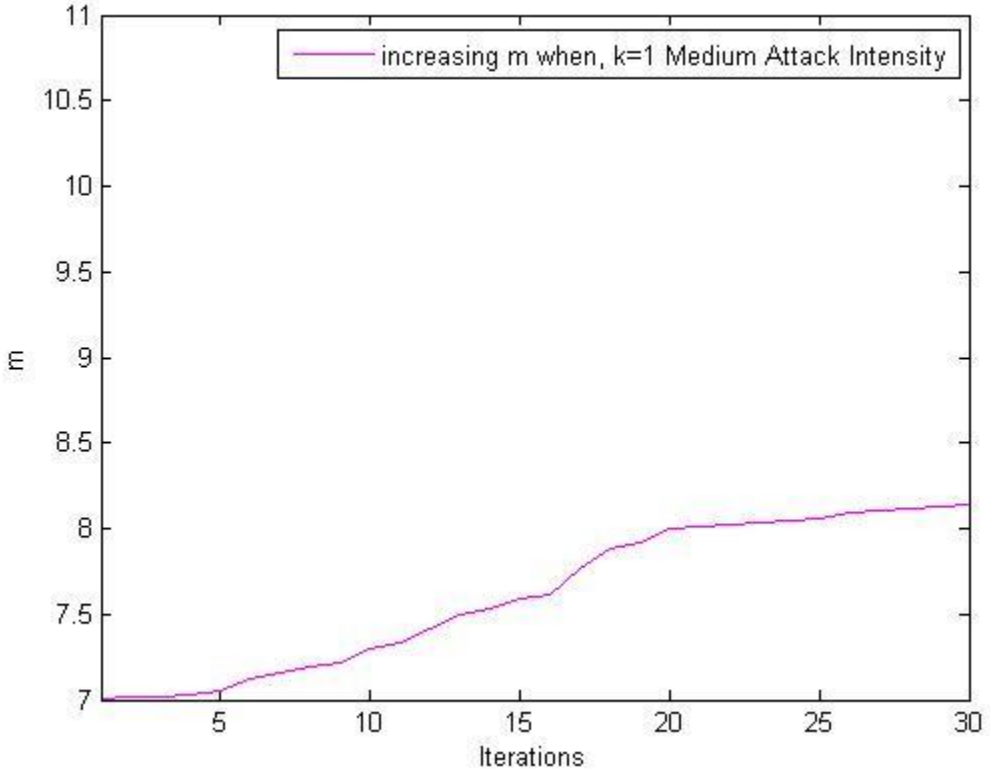

Fig-5.4: Variations of m computed by PSO_SYN in medium attack intensity

The above figure shows that PSO algorithm increases the value of number of half open connections in queue, m is increased when the attack intensity increases so that new requests can be allocated. .

5.4.3 Scenario 3: Increase of T and m in High attack intensity (k = 2)

The figure bellow is for time t in High attack intensity:

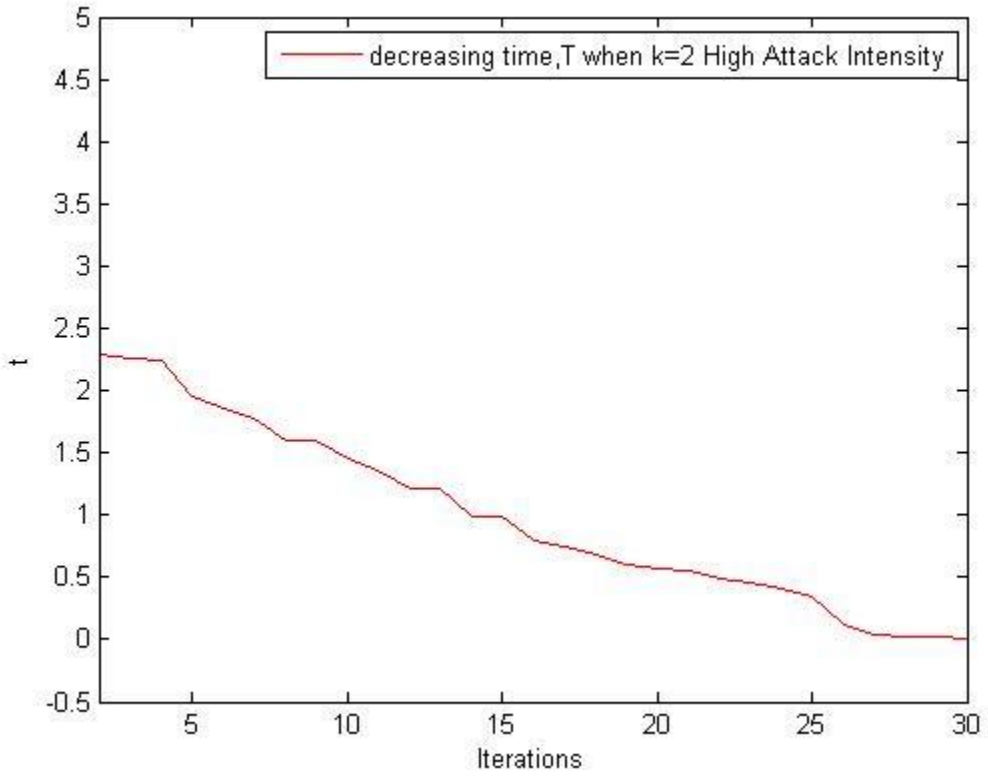

Fig-5.5: Variations of T computed by PSO_SYN in high attack intensity

The above figure shows that when the system is under high attack intensity, then the parameter t is further reduced by the PSO algorithm to eliminate the attack requests from the queue.

The figure bellow is for the capacity of buffer, m in medium attack intensity:

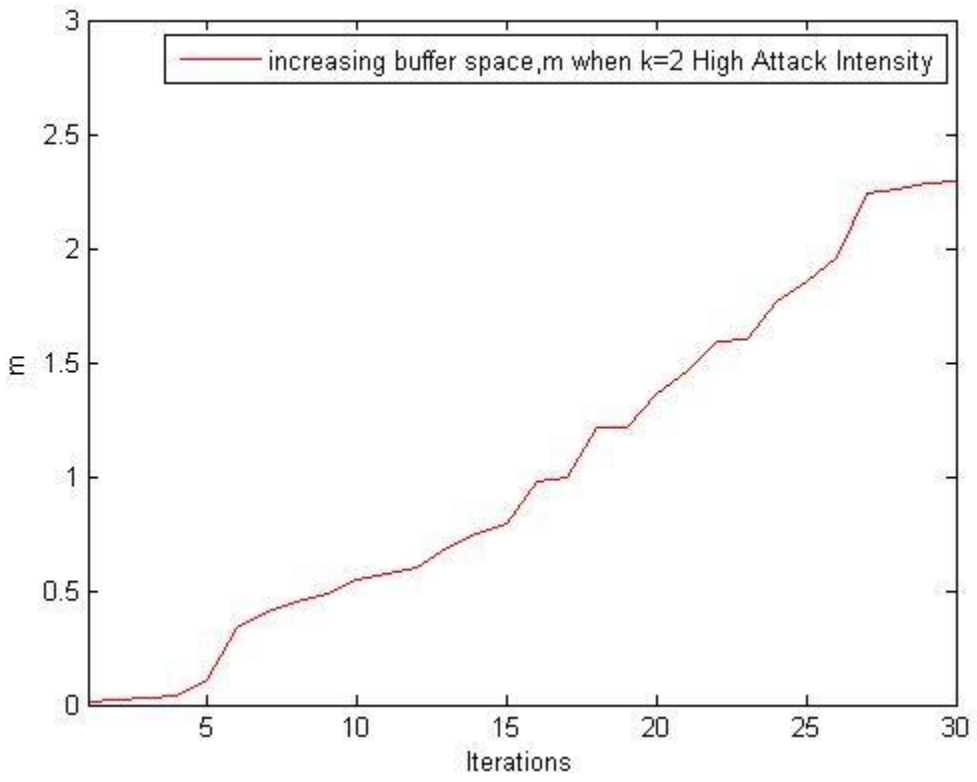

Fig-5.6: Variations of m computed by PSO_SYN in high attack intensity

Here, the intensity is high. So, the capacity of buffer is increasing in a high rate so that of buffer is increasing to make more request to arrive.

5.4.4 Scenario 4: Increase of T and m in variation attack intensity (k = 0.1,1,2)

The figure bellow is for time t in variable attack intensity.

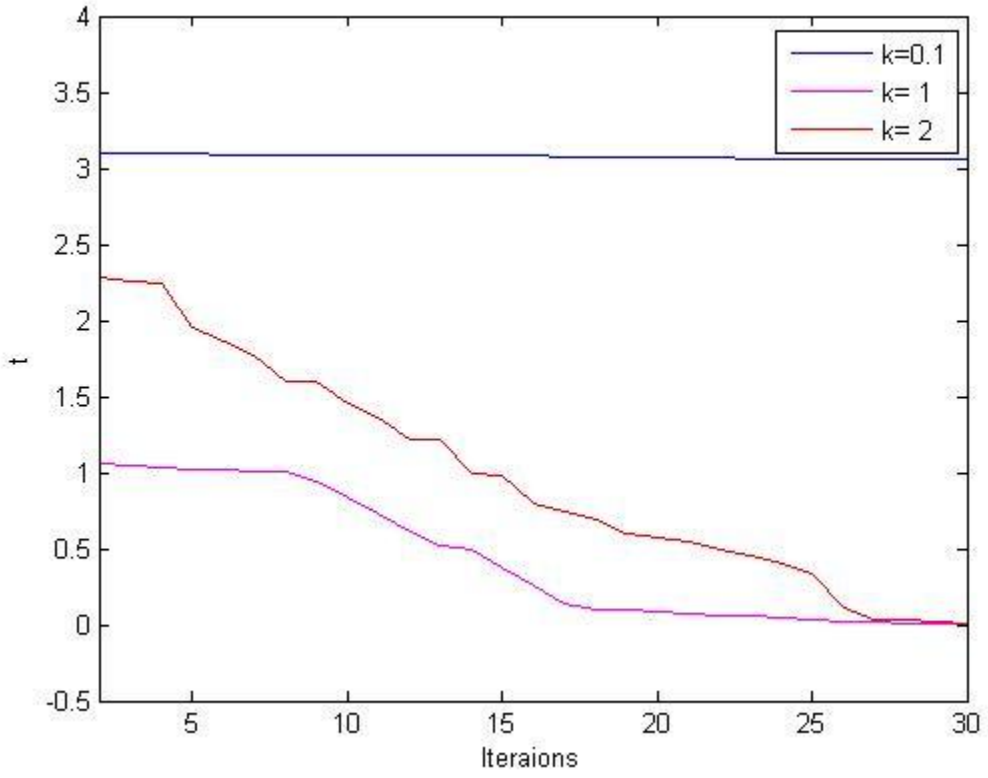

Fig-5.7: Variations of T computed by PSO_SYN in variable attack intensity

Here the figure shows how m changes in several attack intensities. We have used 0.1,1 and 2 as the value of attack intensity respectively in low, medium and high attack.

The figure bellow is for time t in medium attack intensity.

The figure bellow is for time m in variable attack intensity.

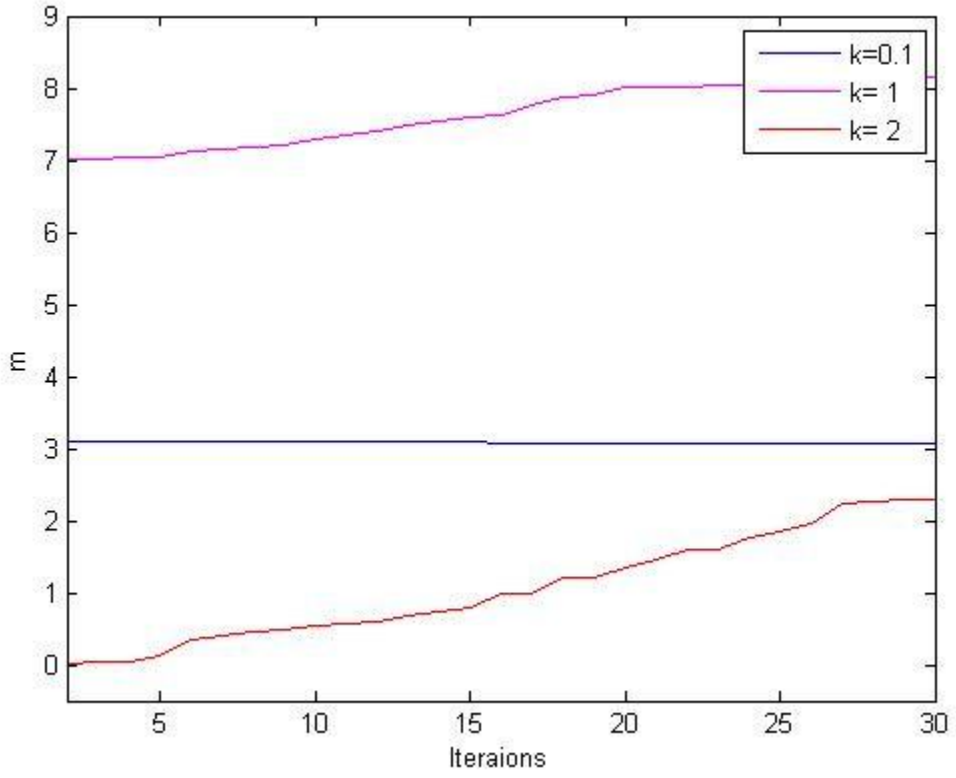

Fig-5.8: Variations of m computed by PSO_SYN in variable attack intensity

Here the figure shows how m changes in several attack intensities. We have used 0.1,1 and 2 as the value of attack intensity respectively in low, medium and high attack.

Chapter 6

CONCLUSION

6.1 CONCLUSION

This paper proposed a novel approach to defend against SYN-flooding DoS attacks. For this purpose, it formulated the defense problem as an optimization problem, which tries to minimize the number of attacking requests and to maximize the number of regular request by maximizing the buffer capacity and minimizing the time for each request to stay in the buffer queue . Then it used PSO technique to solve this optimization problem. This solution led to a self-securing server that continually monitors some performance metrics and then tries to enhance the security degree by dynamically setting of some parameters. Theoretical analysis show that the proposed solution definitely converges to an optimal point and by an extensive simulative study in matlab environment it was shown that the proposed defense mechanism remarkably improves performance of the under-attack server. This research show that t and m are effective control points to protect the TCP servers against SYN flooding attacks. As directions for future works the following issues can be examined.

6.2 FUTURE PLAN

As directions for future works the following issues can be examined.

– Can other parameters affect SYN flooding attacks?
– How can we improve our defense performance by modifications over buffer allocation strategies of TCP? For example, can we assign buffer space only to full established connections and not half-open connections?
– Can other meta-heuristic algorithms such as ant colony optimization and genetic algorithm be employed to design defense mechanisms against SYN flooding attacks?
–Can a similar approach be employed in the network routers?

Chapter 7

APPENDIX

Code for PSO algorithm

```
%% Initialization
% Parameters
clear
clc
iterations = 30;
inertia = 1.0;
correction_factor = 2.0;
swarm_size = 49;

% ---- initial swarm position -----
index = 1;
for i = 1 : 7
   for j = 1 : 7
      swarm(index, 1, 1) = i;
      swarm(index, 1, 2) = j;
      index = index + 1;
   end
end

swarm(:, 4, 1) =10;          % best value so far
swarm(:, 2, :) = 0;           % initial velocity

%% Iterations
for iter = 1 : iterations

   %-- evaluating position & quality ---
   for i = 1 : swarm_size
      swarm(i, 1, 1) = swarm(i, 1, 1) + swarm(i, 2, 1)/1.3;     %update x position
      swarm(i, 1, 2) = swarm(i, 1, 2) + swarm(i, 2, 2)/1.3;     %update y position
      x = swarm(i, 1, 1);
      y = swarm(i, 1, 2);
      lambda1= 10;
      r1= poissrnd(lambda1);

      lambda2= 20;                          %k=attack_intensity= lanbda2/lambda1=0.1
      r2= poissrnd(lambda2);

      val = ((r1*x)/(r2*x))*(y+abs(y-(r1*x+r2*x)));                    % objective function;

      if val > swarm(i, 4, 1)              % if new position is better
         swarm(i, 3, 1) = swarm(i, 1, 1);    % update best x,
         swarm(i, 3, 2) = swarm(i, 1, 2);    % best y postions
         swarm(i, 4, 1) = val;             % and best value
      end
   end

   [temp, gbest] = min(swarm(:, 4, 1));       % global best position

   %--- updating velocity vectors
```

```
    for i = 1 : swarm_size
        swarm(i, 2, 1) = rand*inertia*swarm(i, 2, 1) + correction_factor*rand*(swarm(i, 3, 1) - swarm(i, 1, 1)) +
correction_factor*rand*(swarm(gbest, 3, 1) - swarm(i, 1, 1));   %x velocity component
        swarm(i, 2, 2) = rand*inertia*swarm(i, 2, 2) + correction_factor*rand*(swarm(i, 3, 2) - swarm(i, 1, 2)) +
correction_factor*rand*(swarm(gbest, 3, 2) - swarm(i, 1, 2));   %y velocity component


    end

end
```

Plotting Code in MATLAB

```
t=[
    3.0957
    3.0942
    3.0939
    3.0932
    3.0922
    3.0912
    3.0902
    3.0901
    3.0891
    3.0881
    3.0873
    3.0867
    3.0858
    3.0845
    3.0842
    3.0797
    3.0761
    3.0753
    3.0744
    3.0699
    3.0682
    3.0671
    3.0656
    3.0638
    3.0622
    3.0615
    3.0609
    3.0607
    3.0599
    3.0595
    ];
```

```
iter = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30];
plot(iter, t, 'blue')
hold on;
t = [
    1.0993
    1.0616
    1.0477
    1.0362
    1.0259
    1.0150
    1.0048
    1.0007
    0.9401
    0.8396
    0.7281
    0.6180
    0.5171
    0.4968
    0.3755
    0.2547
    0.1340
    0.1008
    0.0993
    0.0898
    0.0770
    0.0652
    0.0534
    0.0429
    0.0314
    0.0205
    0.0198
    0.0091
    0.0070
    0.0036
    ];


iter = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30];
plot(iter, t, 'magenta')
hold on;

t = [2.2941
   2.2856
   2.2568
   2.2439
   1.9568
   1.8567
   1.7681
   1.6012
   1.5955
   1.4586
   1.3598
   1.2129
   1.2181
   0.9947
```

```
0.9804
0.7916
0.7468
0.6854
0.6018
0.5715
0.5521
0.4891
0.4552
0.4037
0.3412
0.1112
0.0385
0.0284
0.0198
0.0112
]; % my dependent vector of interest


iter = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30];
plot(iter, t, 'r')
hold on;
xlabel ('Iteraions');
ylabel('t');
axis([2 30 -0.5 4]);

legend('k=0.1','k= 1','k= 2');
```

BIBLIOGRAPHY

[1]SeungJae Won, "TCP SYN Flood - Denial of Service", web2.uwindsor.ca/courses/cs/aggarwal/cs60564/Assignment1/Won.pdf

[2] Deepak Singh Rana, "A Study and Detection of TCP SYN Flood Attacks with IP spoofing and its Mitigations"
www.ijcta.com/documents/volumes/vol3issue4/ijcta2012030428.pdf

[3] J. Lemon, "Resisting SYN Flooding DoS Attacks with a SYN Cache", *Proceedings of USENIX BSDCon'2002*, February, 2002.

[4] D. J. Bernstein and Eric Schenk, "Linux Kernel SYN Cookies Firewall Project", http://www.bronzesoft.org/projects/scfw.

[5] 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.

[6] S. Gavaskar, R. Surendiran and DR. E. Ramaraj , "Three Counter Defense Mechanism for TCP SYN flooding attack".

[7] S. M. Bellovin, "ICMP Traceback Messages", *Internet Draft: draftbellovin-itrace-00.txt*, March 2000.

[8] K. Park and H. Lee, "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack", *Proceedings of IEEE INFOCOM 2001*, March 2001.

[9] S. Savage, D. Wetherall, A. Karlin and T. Anderson, "Practical Network Support for IP Traceback", *Proceedings of ACM SIGCOMM'2000*, August 2000.

[10] D. Song and A. Perrig "Advanced and Authenticated Marking Schemes for IP Traceback", *Proceedings of IEEE INFOCOM'2001*, March 2001.

[11] A. C. Snoren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent and W. T. Strayer, "Hash-Based IP Traceback", *Proceedings of ACM SIGCOMM'2001*, August 2001.

[12] S. F. Wu, L. Zhang, D. Massey, and A. Mankin, "Intention-driven ICMP traceback", *Internet Draft: draft-wu-itrace-intention-00.txt*, February 2001.

[13] T. Darmohray and R. Oliver, "Hot Spares for DoS attacks", *;login*, 25(7), July 2000.

[14].F. Karray and C. W. de Silva, Soft Computing and Intelligent Systems Design, Addison Wesley, New York, NY, USA, 2004.

[15].A. S. Cherry and R. P. Jones, “Fuzzy logic control of an automotive suspension system,” IEE Proceedings: Control Theory and Applications, vol. 142, no. 2, pp. 149–160, 1995.

[16].G. Slaski and M. Maciejewski, “Skyhook and fuzzy logic controller of a semi active vehicle suspension,”Transport, vol. 78, pp. 97–111, 2011.

[17].D. P. Rini, S. M. Shamsuddin, and S. S. Yuhaniz, “Particle swarm optimization: technique, system and challenges,” International Journal of Computer Applications, vol. 14, no. 1, pp. 19–26, 2011.

[18].MathWorks, 2012, http://www.mathworks.com/help