Defending DDoS Attack using Stochastic Model based Puzzle Controller

Santhosh K M[†] and Elizabeth Isaac[††],

M Tech in Information System Security, IGNOU, India

Summary
Distributed denial of service (DDoS) attack aimed at network is some sort of malicious activity or unusual behavior, which compromise the availability of the server's resources and prevents the legitimate users from using the service. In this paper a preventive mechanism using stochastic model based puzzle controller, to eliminate the possibility of DDoS attack is proposed. A behavior matrix is defined to record the behavior of users in terms of number of requests sent. A puzzle controller based on stochastic model is used to analyze the behavior matrix and compute covariance matrix. The entropy computed from the covariance matrix is compared against the threshold to detect DDoS attack. To defend against DDoS attack, allocate resources only to those clients who solve the puzzle with difficulty level determined based on entropy value.

Key words:
DDoS, stochastic model, entropy, puzzle.

1. Introduction

DDoS attacks [3, 4] are DoS attacks that come simultaneously from many hosts from all over the net. DoS attacks attempt to exhaust the victim's resources. To launch a DDoS attack, malicious users first build a network of computers that they will use to produce the volume of traffic needed to deny services to computer users. To create this attack network, attackers discover vulnerable sites or hosts on the network. Vulnerable hosts are usually those that are either running no antivirus software or out-of-date antivirus software, or those that have not been properly patched. Vulnerable hosts are then exploited by attackers who use their vulnerability to gain access to these hosts. The next step for the intruder is to install slave processes in compromised hosts that allow the attacker to remotely direct the hosts to attack a target. The hosts that are running these attack tools are known as zombies, and they can carry out any attack under the control of the attacker. The attacker installs a master program somewhere on the Internet. Master has a list of all the locations of the zombies. Master waits for instructions. When it is time to strike, the attacker sends a message to the master indicating the target address. The master then sends a message to each of the zombies with the target address. At once, the zombies flood the target with enough traffic to overwhelm it. The traffic from zombies can be sent with spoofed IP source address to make it difficult to trace the actual source.

DDoS be carried at network level which attempts to exhaust network resources. Typical examples are Ping of Death, Smurf Attack etc. In transport layer attacker uses TCP SYN attacks, UDP flooding etc. In application layer attackers use HTTP flooding to overwhelm the server. Popular DDoS tools are Tribe Flood Network (TFN), TFN2K, and Trinoo that generate flooding attacks using a combination of TCP, UDP, and ICMP packets [5].

In Ping of Death attacks, the attacker creates a packet that contains more than 65,536 bytes, which is the maximum IP packet length that the IP protocol defines. This packet can cause different kinds of damage to the machine that receives it, such as crashing and rebooting.
The Smurf IP Attack is named after an application that lets the attacker carry out the attack. In a "smurf" attack, the victim is flooded with Internet Control Message Protocol (ICMP) "echo-reply" packets. The attacker sends numerous ICMP "echo-request" packets to the broadcast address of many subnets. These packets contain the victim's address as the source IP address. Because the ping was sent to a broadcast address, it was received by all other computers on the subnet. They read the source IP address, belonging to the victim, and all of them send ICMP "echo-reply" packets to the victim, overwhelming it with replies. Smurf attacks are very dangerous, because they are strongly distributed attacks.

In a TCP SYN attack, the attacker takes the benefit of vulnerability in TCP/IP implementation. A SYN flood attack occurs during the three-way handshake. In the three-way handshake, a client requests for connection by sending a TCP SYN packet to a server. Then the server sends a SYN/ACK packet back to the client and places the connection request in a queue. Finally, the client acknowledges the SYN/ACK packet. If an attack occurs, however, the attacker sends an abundance of TCP SYN packets to the victim, obliging it both to open a lot of TCP connections and to respond to them. Then the attacker does not execute the third step of the three-way handshake that

Manuscript received April 5, 2013
Manuscript revised April 20, 2013

follows, rendering the victim unable to accept any new incoming connections, because its queue is full of half-open TCP connections.

Another type of Denial of Service attack at the transport layer is the UDP Flood attack which floods the victim with continuous stream of UDP packets. The attacker fires UDP packets at the victim, attempting to overload a service that is listening for UDP packets.

DDoS defense mechanisms are generally classified as preventive mechanisms and reactive Mechanisms.

The preventive mechanisms try to eliminate the chance of DDoS attacks altogether or to enable potential victims to endure the attack and to continue the services to legitimate clients. Examples of system security mechanisms include monitoring access to the computer and applications, and installing security patches, firewall systems, virus scanners, and intrusion detection systems automatically. At the network level, implementing ingress and egress filtering to prevent packets with bogus source addresses from leaving the local network can prevent local machines from participating in DDoS attacks. One method of DDoS prevention is to increase the privileges of users according to their behavior. It tries to verify users' identities so that no threat exists. Another method involves increasing the effective resources to such a degree that DDoS effects are limited, which is costlier and practically impossible.

The reactive mechanisms try to detect the attack and respond to it immediately. Hence, they restrict the impact of the attack on the victim. Again, there is the danger of characterizing a legitimate connection as an attack. For that reason it is necessary for researchers to be very careful. Reactive mechanisms respond to attack after detecting it which may help to reduce the impact of the attack. Some mechanisms react by limiting the accepted traffic rate. This means that legitimate traffic is also blocked. In some cases, techniques try to identify the attacker. If attackers are identified, despite their efforts to spoof their address, then it is easy to filter their traffic. Filtering is efficient only if attackers' detection is correct.

Development of detection and defending tools is very complicated. Designers must think in advance of every possible situation because every weakness can be exploited. One of the difficulties is DDoS attacks flood victims with packets. This means that victims cannot contact anyone else in order to ask for help. Secondly any attempt of filtering the incoming flow means that legitimate traffic will also be rejected. Third difficulty is Attack packets usually have spoofed IP addresses and so it is more difficult to trace back to their source. Last difficulty is Defense mechanisms are applied in systems with differences in software and architecture and hence developers must design a platform independent of all these parameters.

In this paper we propose stochastic model based puzzle controller, a preventive mechanism which concentrates on user behavior. A behavior matrix prepared is used to compute covariance matrix and entropy. Entropy is compared with threshold value to predict the possibility of DDoS attack. The client which solves the puzzle with difficulty level determined by entropy value will be allocated resources.

This paper is organized as follows: Section 2 deals with the related work and Section 3 and 4 deals with DDoS detection and DDoS prevention using puzzle controller. Section 5 deals with results and analysis. Section 6 contains the conclusion.

2. RELATED WORK

In our literature survey, we note that DDoS defense mechanisms are generally classified as preventive mechanisms and reactive Mechanisms. Here we explain two preventive mechanisms- Defending against denial-of-service attacks with puzzle auctions puzzle auction [1] and a puzzle-based defense strategy against flooding attacks using game theory [7]. We will also present a monitoring scheme [2].

Client puzzles helps in defending against DoS attacks. In this approach each client has to solve a cryptographic puzzle for each service request before the server allocates its resources so that it outs a large computational task on adversaries. But, this approach was not used much in practice because of at least two reasons. First, puzzles add to legitimate clients load and in the presence of adversaries with unknown computing power, it may be difficult to approximately tune puzzle difficulty to minimize client cost. Second, very few implementations of client puzzles are available. Xiaofeng Wang; Reiter, M.K. [1] proposed a puzzle mechanism to defend against DoS attack, called puzzle auction. Here each client determined the difficulty of the puzzle it solves and allocates server resources to the client which solved the difficult puzzle first when the server is busy. It gives each client the flexibility to choose service priority against its valuation, i.e., computation paid for the service. They also designed a bidding strategy for clients to increment puzzle difficulty, i.e., bid, gradually via retransmission to just above adversaries capabilities.

Here [1] TCP puzzle auction works as follows. Client first sends a request (SYN packet) to the server without a puzzle solution. After receiving the packet, the server first checks the puzzle difficulty to determine the priority of the request and then adds the request to half open queue if the buffer queue is not full, else the server drops the request with lowest priority and sends back reset packet(RST packet) with server nonce generated according to client's IP address. The receiver of the RST packet uses server nonce to increase its bid (puzzle difficulty). It computes a puzzle and retransmits a new SYN with puzzle solution. If the server again rejects the request, the client will further increase its bid and retransmits again. This process will be continued till either server accepts the request (server sends back SYN-ACK packet) or maximum number of retransmissions set by the protocol exceeds.

Mehran S. Fallah in his paper [6] utilizes game theory to propose a number of puzzle-based defenses against flooding attacks. Preventive mechanisms against flooding attacks can be effectively studied through game theory. This is mainly owing to the several trade-offs existing in a flooding attack defense scenario. For an attacker, there is a trade-off between the severity of his attack and the amount of resources he uses to do so; the more damage an attacker intends to cause, the more amounts of resources he should spend. For a defender, on the other hand, there is a trade-off between the effectiveness of his defense and the quality of service he provides for legitimate users; the more difficult it becomes to exhaust the defender's resources, the more workload, and hence, less quality of service is imposed on legitimate users. A trade-off also exists between the effectiveness of the defense and the amounts of resources a defender expends. Here it is shown that the interactions between an attacker who perpetrates a flooding attack and a defender who counters the attack using a puzzle-based defense can be modeled as a two-player infinitely repeated game with discounting. Then, the solution concepts of this type of games are deployed to find the solutions, i.e., the best strategy a rational defender can adopt in the face of a rational attacker.

Like many puzzle-based defenses, [6] is also based on an assumption that the defender is at least capable of sending reply messages to the origins of incoming requests. This seemingly restricts the applicability of the proposed mechanisms in the case of bandwidth exhaustion attacks in which the attacker sends a huge number of service requests to deplete the victim's bandwidth. However, it can be envisioned that by coordinating multiple routers installed with the defense mechanisms proposed in this paper, one can restrain the attack flows before they converge to the victim. Nevertheless, the game-theoretic approach employed in the current paper is not sufficient for handling such a case. Another assumption made in this paper [6] is the complete rationality of the players. Evidently, the defense strategies proposed in this paper may not be optimal if the attacker has a bounded level of rationality. In other words, the defender can gain payoffs better than the ones attainable by the mechanisms of this paper when his opponent is not completely rational.

In paper [2] a scheme based on document popularity is introduced. An Access Matrix is defined to capture the spatial-temporal patterns of a normal flash crowd. Principal component analysis and independent component analysis are applied to abstract the multidimensional Access Matrix. A novel anomaly detector based on hidden semi-Markov model is proposed to describe the dynamics of Access Matrix and to detect the attacks. The entropy of document popularity fitting to the model is used to detect the potential application-layer DDoS attacks.

3. DDoS DETECTION

In this paper, we propose a stochastic model based puzzle controller to prevent DDoS attack. User behavior is analyzed to detect the anomaly. User behavior is represented by the number of requests to access server resources, particularly the web pages. A behavior matrix of order NxT is computed. Here N is the total number of resources and T is the number of time units considered. Each entry in the matrix is denoted by b_{it} and it is defined as given below.

$$b_{it} = \frac{r_{it}}{\sum_{i=1}^{N} r_{it}}, i \in [1, N], t \in [1, T] \qquad (1)$$

Here b_{it} is the behavior rate of i^{th} resource at t^{th} time unit, r_{it} is the request number of i^{th} resource at t^{th} time unit. Now Behavior matrix B_{NxT} is constructed as follows:

$$B_{N\times T} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1T} \\ b_{21} & b_{22} & \cdots & b_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \cdots & b_{NT} \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & \cdots & b_T \end{bmatrix}$$

Here $b_1, b_2, \cdots, b_T$ are defined as follows:

$$b_1 = \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{N1} \end{bmatrix}, b_2 = \begin{bmatrix} b_{12} \\ b_{22} \\ \vdots \\ b_{N2} \end{bmatrix}, b_T = \begin{bmatrix} b_{1T} \\ b_{2T} \\ \vdots \\ b_{NT} \end{bmatrix}$$

Next we need to compute a covariance matrix which is necessary to fully describe the variation in a distribution. The expected value or mean is defined as follows:

$$\mu_i = E[b_i] = \frac{1}{N}\sum_{j=1}^{N} b_{ji} \qquad (2)$$

$E[b_i]$ or μ_i is expected value of ith resources. Using this value, covariance matrix C_{NxT} can be constructed. Each element of the matrix is a variance represented by $C_{ij.}$ In one dimensional array, the variance σ^2 is

$$\sigma^2 = \mathrm{var}(b) = E[(b-\mu)^2] \qquad (3)$$

In more than one dimension, covariance matrix is

$$C = E[(b-\mu)(b-\mu)^T] \qquad (4)$$

Its entries are the individual covariance and are defined as follows:

$$C_{ij} = E[(b_{ij}-\mu_i)(b_{ij}-\mu_i)^T] \qquad (5)$$

C_{NxT} is constructed as follows:

$$C_{N\times T} = \begin{bmatrix} E[(b_1-\mu_1)(b_1-\mu_1)^T] & \cdots & E[(b_1-\mu_1)(b_T-\mu_T)^T] \\ E[(b_2-\mu_2)(b_1-\mu_1)^T] & \cdots & E[(b_2-\mu_2)(b_T-\mu_T)^T] \\ \vdots & \ddots & \vdots \\ E[(b_T-\mu_T)(b_1-\mu_1)^T] & \cdots & E[(b_T-\mu_T)(b_T-\mu_T)^T] \end{bmatrix}$$

According to information theory, the information entropy is a measure of randomness and uncertainty associated with a random variable. It measures the average information contained in a piece of data. The information entropy (En) of our observations can be calculated from covariance matrix (C_{NxT}) as follows:

$$En = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{T} Cij \log(Cij) \qquad (6)$$

Threshold entropy ($En_{threshold}$) is determined based on entropy distribution. Here we can also consider entropies at the time when DDoS attack (if any) occurred in the past. Now this system can be used to detect anomaly by comparing current entropy with threshold entropy.

4. DDoS PREVENTION WITH PUZZLE CONTROLLER

Here we present the structure of our proposed puzzle controller to prevent DDoS attack. We assume that as the difficulty level of the puzzle to be solved increases, the attacker gives up his attempt to access the service. Only the legitimate user will continue to solve the difficult puzzle and will succeed if he solves it successfully within the required time.

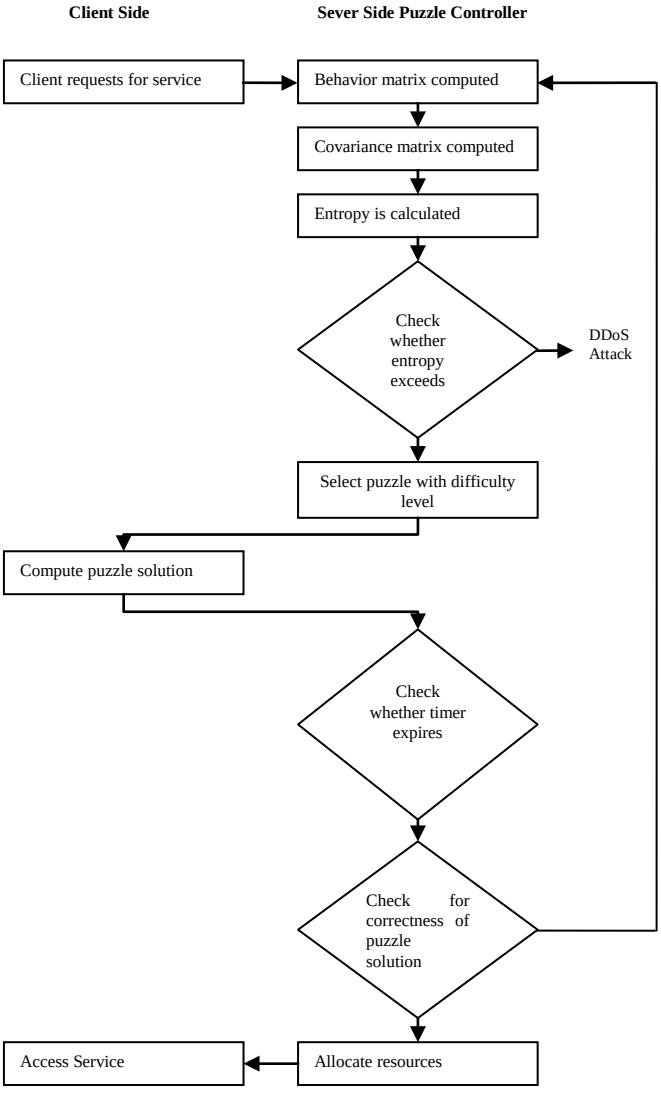

Fig. 1. Puzzle Controller

The structure of puzzle controller (PC) is given in Fig.1. As the first step, client sends his request to server to access the service. Puzzle controller on server computes a behavior matrix to analyze the activity of users. Each entry in this matrix is a measure of average number of requests for a service at a given instance of time.

A covariance matrix is computed from the behavior matrix as described in section 3. Each entry in this matrix describes variation in distribution of user behavior. Next step is the calculation of entropy which is a measure of average behavior of users at a given instant of time.

The entropy calculated is compared with a threshold value to detect the possibility of DDoS attack. Threshold value is determined based of entropy distribution and previous values of entropy at which attacks occurred in the past. If the calculated value of entropy is grater than or equal to threshold value, then DDoS attack occurs.

In our proposed system, we implement a method to prevention DDoS attack. Here each client has to solve a puzzle with a required difficulty level. PC determines the difficulty of the puzzle to be solved by the client so as to access the server service or resource. The difficulty level is defined as follows:

$$Level_{Puzzledifficulty} = En \; ? \; N + t \qquad (7)$$

Here En is entropy at current time unit t and N is the total number of resources. PC selects a puzzle with difficulty level determined using above equation and is sent to the client which requests for the service. As the value of entropy tends to close to threshold value, the level of puzzle difficulty also increases to maximum. To defend against DDoS attack, resources will be allocated only to those clients who solve the puzzle successfully within the required time.

Puzzle controller selects a puzzle with the required difficulty level dynamically and is sent to client. Client solves the puzzle within the required time and sends the solution to server. The difficulty level is higher on adversary condition and client has to spend more time and resources to solve the puzzle. We assume that only legitimate user spends to solve puzzle and the attacker gives up his attempt.

The puzzle controller checks for the time taken for solving the puzzle and solution. If both condition satisfied, server allocates the resources to the client.

5. RESULT AND ANALYSIS

In this section we present result and analysis of our implementation. We simulate the proposed puzzle controller in NS2.

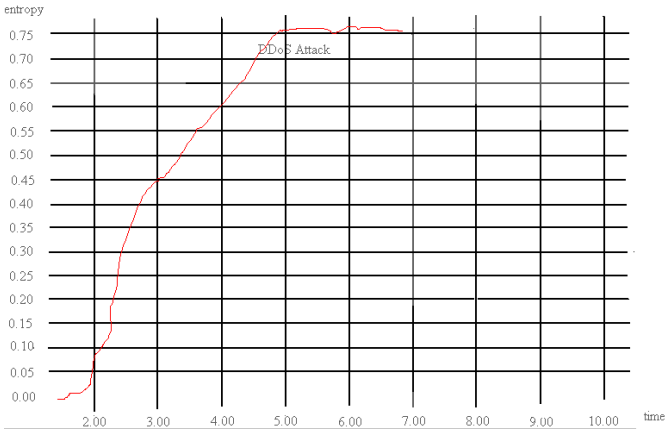

Fig. 2. Entropy Vs time Graph during DDoS Attack

A graph obtained during DDoS attack for the system under experiment is shown in Fig.2. It is the graph obtained when puzzle controller not implanted in the system. Here we noticed that when the system reaches the entropy value 0.75, the system is overwhelmed and further service is denied for the client. The system reaches this threshold entropy during fourth time unit under consideration.

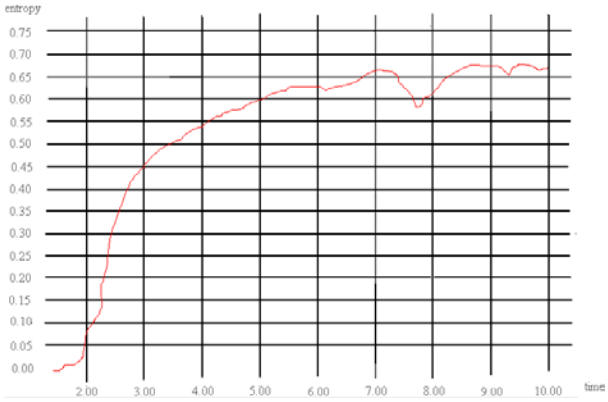

Fig. 3. Entropy Vs time Graph using puzzle controller

Fig.3. shows the graph obtained when puzzle controller is implemented in the system. Here we continuously monitor the system to prevent possibility of DDoS attack. We set the threshold value as 0.75 which is entropy of the system for which DDoS attack is detected with no puzzle controller. When client requests for service, difficulty level of puzzle is calculated based on entropy value and puzzle is selected dynamically. Under our

assumption that as the difficulty level increases the attacker gives up his attempt, we were able to maintain peak entropy between 0.62 and 0.68. With strict monitoring and puzzle at required difficulty level with quality, entropy can never reach a value which is closer to threshold value. Thus we can detect possibility of DDoS attack very early and prevent the attack.

6. CONCLUSION

Seven/twenty-four (7 × 24) operations have become the norm in many of today's businesses and systems must be continuously online. If an outage occurs, the company stands to lose tens of thousands of dollars an hour. In today's gloomy economy, stockholders don't want to hear that their favorite investment is having system availability problems. So it is the primary responsibility of all concerned to make the system available and hence the resources and services to all the legitimate users. An attacker can make all resources unavailable by initiating a DDoS attack. So we need to detect the possibility of such attack and it is required to prevent the same.

In this paper we proposed a method to detect and prevent the DDoS attack. Our system is a stochastic model based puzzle controller. It forces each user to solve a puzzle of required difficulty level determined based on entropy. A legitimate user spends time and other resources to solve the puzzle correctly and gets the service. Our simulation results show that it is effective mechanism in preventing the DDoS attacks.

REFERENCES

[1] Xiaofeng Wang, Michael K. Reiter, "Defending against denial-of-service attacks with puzzle auctions", Symposium on Security and Privacy, 2003.

[2] Yi Xie, Shun-Zheng Yu, "Monitoring the Application-Layer DDoS Attacks for Popular Websites ", IEEE/ACM Transactions on Networking Vol 17 No 1, Feb. 2009, pp. 15-25.

[3] Charalampos Patrikakis, Michalis Masikos, and Olga Zouraraki , "Distributed Denial of Service Attacks " The Internet Protocol Journal - Volume 7, Number 4

[4] T. Peng, C. Leckie, and K. Ramamohanarao, "Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring," The University of Melbourne, Australia, 2003.

[5] Alefiya Hussain, John Heidemann, and Christos Papadopoulo, "A Framework for Classifying Denial of Service Attacks," 25 February 2003.

[6] Mehran S. Fallah, "a puzzle-based defense strategy against flooding attacks using game theory" IEEE TRANSACTIONS on dependable and secure computing, vol. 7, no. 1, January-March 2010

[7] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, Darrell Kindred, " Statistical Approaches to DDoS Attack Detection and Response" Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03)

[8] N. Jeyanthi1, N.Ch. Sriman, Narayana Iyengar"MAC Based Routing Table Approach to Detect and Prevent DDoS Attacks and Flash Crowds in VoIP Networks" CYBERNETICS AND INFORMATION TECHNOLOGIES -Volume 11, No 4

[9] Yuexiang Yang, Hailong Wang and Xicheng Lu, Entropy-"Based Classification of Large-Scale Network Traffic Anomalies", ComputerEngineering & Science, Vol.29 ,No.2 ,2007, 40-43

[10] Akoi M., "Optimization of Stochastic Systems-Topics in Discrete-Time Systems", Academic Press, New York, 1967

[11] Hiromitsu Hama, Pyke Tin, Thi Thi Zin and Takashi Toriu, " A Stochastic Model for Popularity Measures in Web Dynamics" , IEEE Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing,2010

[12] Shuyuan Jin, Daniel S. Yeung, "A Covariance Analysis Model for DDoS Attack Detection" IEEE Communications Society, 2004

[13] S. Noh, C. Lee, K. Choi, and G. Jung, "Detecting Distributed Denial of Service (DDoS) attacks through inductive learning," Lecture Notesin Computer Science, vol. 2690, pp. 286–295, 2003.

[14] W. Yen and M.-F. Lee, "Defending application DDoS with constraint random request attacks," in Proc. Asia-Pacific Conf. Commun., Perth, Western Australia, Oct. 3–5, 2005, pp. 620–624.

[15] J. J. Yuan and K. Mills, "Monitoring the macroscopic effect of DDoS flooding attacks," IEEE Trans. Dependable and Secure Computing, vol. 2, no. 4, pp. 324–335, Oct.-Dec. 2005

[16] Yingjie Zhou Guangmin Hu Weisong He, "Using Graph to Detect Network Traffic Anomaly", IEEE Communications Society, 2009