ISSN 2047-3338

Mitigating App-DDoS Attacks on Web Servers

Ms. Manisha M. Patil[1] and Prof. U. L. Kulkarni[2]
[1]Dr. D. Y. Patil College of Engineering and Technology, Kolhapur, (Maharashtra), India
[2]Konkan Gyanpeeth's College of Engineering, Karjat, Dist. Raigad, (Maharashtra), India
manisha_birnale@rediffmail.com, kumeshl@rediffmail.com

***Abstract*—In this paper, a lightweight mechanism is proposed to mitigate session flooding and request flooding app-DDoS attacks on web servers. App-DDoS attack is Application layer Distributed Denial of Service attack. This attack prevents legitimate users from accessing services. Numbers of mechanisms are available and can be installed on routers and firewalls to mitigate network layer DDoS attacks like SYN-flood attack, ping of death attack. But Network layer solution is not applicable because App-DDoS attacks are indistinguishable based on packets and protocols. A lightweight mechanism is proposed which uses *trust* to differentiate legitimate users and attackers. Trust to client is evaluated based on his visiting history and requests are scheduled in decreasing order of *trust*. In this mechanism trust information is stored at client side in the form of cookies. This mitigation mechanism can be implemented as a java package which can run separately and forward valid requests to server. This mechanism also mitigates request flooding attacks by using Client Puzzle Protocol. When server is under request flooding attack source throttling is done by imposing cost on client. Cost is collected in terms of CPU cycles.**

***Index Terms*— DDoS Attacks, App-DDoS and Trust**

I. INTRODUCTION

DISTRIBUTED Denial of Service attack means an attempt to prevent a server from offering services to its legitimate/genuine users. This is accomplished by attackers by sending requests in overwhelming number to exhaust the server's resources, e.g. bandwidth or processing power. Due to such DDoS attacks server slows down its responses to clients or sometimes refuses their accesses. Thus DDoS attack is great threat to internet today.

Now a day many of the businesses like banking, trading, online shopping uses World Wide Web. So it is very essential to protect the web sites from this DDoS attacks.

Traditionally, DDoS attacks were carried out at the network layer, such as SYN flooding, UDP flooding, ping of death attacks, which are called Net-DDoS attacks.

The intent of these attacks is to consume the network bandwidth and deny service to legitimate users of the systems. Many studies has noticed such type of attacks and proposed different mechanisms, solutions to protect the network and equipment from bandwidth attacks. So it is not easy as in the past for attackers to launch the network layer DDoS attacks.

When the simple Net-DDoS attacks fail, attackers are giving their way to more sophisticated Application layer DDoS attacks [2].

Application layer DDoS attack is a DDoS attack that sends out requests following the communication protocol and thus these requests are indistinguishable from legitimate requests in the network layer. Most application layer protocols, for example, HTTP1.0/1.1, FTP and SOAP, are built on TCP and they communicate with users using sessions which consist of one or many requests. As App-DDoS attacks are indistinguishable from legitimate requests based on packets and protocols, network layer solution cannot be used here. Most existing scheme uses packet rate as a metric to identify attackers. But intelligent users can adjust the packet rate based on server's response to evade detection. Even IP address based filtering is not possible as attackers may hide behind proxies or IP addresses can be spoofed.

Application layer DDoS attacks employ legitimate HTTP requests to flood out victim's resources. Attackers attacking victim web servers by HTTP GET requests (HTTP flooding) and pulling large image files from victim server in large numbers. Sometimes attackers can run large number of queries through victim's search engine or database query and bring the server down [6].

Application layer attack may be of one or combination of session flooding attack, request flooding attack and asymmetric attack [1]. Session flooding attack sends session connection requests at higher rates than that of legitimate users. Request flooding attack sends sessions that contain more requests than normal sessions.

Asymmetric attack sends sessions with higher workload requests. The proposed mechanism focuses the session flooding attacks and request flooding attacks.

By considering the bandwidth and processing power of application layer server, threshold for simultaneously connected sessions and maximum number of requests that can be serviced with assurance of Quality of service is decided. Under session flooding attack the proposed mechanism rejects

the attackers and allocates the available sessions to legitimate users. Under request flooding attacks the proposed mechanism sends puzzles to the client and the requests are processed only when client sends result back by solving the puzzles.

The proposed mechanism uses trust to mitigate session flooding attack and Client Puzzle Protocol to mitigate request flooding attack.

Distributed Denial of Service attacks have been increasing in the recent times. Most of the well known sites are affected by these kinds of attacks. Commercial sites are more vulnerable during the business time as there will be many genuine users accessing it, and attacker needs only a little effort to launch DDoS attack. It is difficult to prevent such attacks from happening and the attackers may continue their damage using new and innovative approaches. Proposed mechanism is a way to handle the situation without any change at the user end and very little change at the server end.

The idea is to assign trust value to each client according to his visiting history and allocate available number of sessions to users according to their decreasing order of trust values. To improve the server performance under request flooding DDoS attacks, attacker enforced to pay the CPU stamp fee, hence making the attacker also to use his resources more or less equally [4]. When a client is making legitimate requests, this cost is negligible but when the client becomes malicious the costs grow huge there by imposing a limit on the number of requests that the client can send.

To clarify the idea, we can design a small hypothetical website which will handle 500 requests per second. The distributed attack is launched against the website using web stress tool and it will start sending 1000 requests per second. Then performance of website is measured without mitigation mechanism and with mitigation mechanism.

II. RELATED WORK

S. Ranjan *et al.* proposed a counter-mechanism by building legitimate user model for each service and detecting suspicious requests based on the contents of the requests [2]. To protect servers from application layer DDoS attacks, they proposed a counter-mechanism that consist of a suspicion assignment mechanism and DDoS resilient scheduler DDoS shield. The suspicion mechanism assigns continuous value as opposed to a binary measure to each client session, and scheduler utilizes these values to determine if and when to schedule a session's requests.

M. Srivatsa *et el.* performed admission control to limit the number of concurrent clients served by the online service [3]. Admission control is based on port hiding that renders the online service invisible to unauthenticated clients by hiding the port number on which the service accepts incoming requests. The mechanism needs a challenge server which can be the new target of DDoS attack.

J. Yu, Z. Li, H. Chen, and X. Chen proposed a mechanism named DOW (Defense an Offence Wall), which defends against layer-7 attacks using combination of detection technology and currency technology [5]. An anomaly detection method based on K-means clustering is introduced to detect and filter request flooding attacks and asymmetric attacks. But this mechanism requires large amount of training data.

Yi Xie and Shun-Zheng Yu introduced a scheme to capture the spatial-temporal patterns of a normal flash crowd event and to implement the App-DDoS attacks detection [9]. Since the traffic characteristics of low layers are not enough to distinguish the App-DDoS attacks from the normal flash crowd event, the objective of their work is to find an effective method to identify whether the surge in traffic is caused by App-DDoS attackers or by normal Web surfers. Web user behavior is mainly influenced by the structure of Website (e.g., the Web documents and hyperlink) and the way users access web pages. In this paper, the monitoring scheme considers the App-DDoS attack as anomaly browsing behavior.

Our literature survey has noted that many mechanisms are developed to service legitimate users only. Abnormalities are identified and denied. But large amount of training data is required. Sometimes mitigation mechanism can itself becomes target of DDoS attack.

The need is felt to design and develop a new lightweight mechanism that can mitigate both session flooding and requests flooding Application layer DDoS attacks with small amount of training data. It will service all users if and only if resource is available and use bandwidth effectively. It will identify the abnormalities and serve them with different priorities.

III. LEGITIMATE USER & ATTAKER MODEL

We can build legitimate user model and attacker model with several attack strategies of different complexities. We can make few assumptions about web server.

Assumption 1: Under session flooding attacks, the bottleneck is maximal number of simultaneously connected sessions called *MaxConnector*. It depends on banwidth and processing power of the server.

Assumption 2: Without attacks, the total number of session connections of server should be much small than *MaxConnector*.

Assumption 3: Under request flooding attacks, the bottleneck is maximal number of requests in one session that can be processed with assured quality of service.

Legitimate User Model:

Legitimate users are people who request services for their benefit from the content of the services. So, the inter-arrival time of requests from a legitimate user would form a certain density distribution *density (t)*. Here *t* is inter-arrival time and *density (t)* is the probability a legitimate user will revisit the website after time *t*. The traces collected at AT&T Labs Research and Digital Equipment Corporation by F. Douglis *et el.* [8] is used to build model *density (t).*

Attacker Model:

The goal of session flooding DDoS attack is to keep the number of simultaneous session connections of the server as large as possible to stop new connection requests from legitimate users being accepted. Attacker may consider using following strategies when he controls lots of zombie machines.

1. Send session connection requests at a fixed rate, without considering response or the service ability of victim.

2. Send session connection requests at a random rate, without considering response or the service ability of victim.

3. Send session connection requests at a random rate and consider the service ability of victim by adjusting requests at a rate according to the proportion of accepted session connection requests by server.

4. First send session connection requests at a rate similar to legitimate users to gain trust from server, then start attacking with one of the above strategies.

5. Sends sessions containing large number of requests than that of the legitimate user session.

IV. ASSIGNING THE TRUST VALUE

For every established connection four aspects of trusts are recorded. They are short term trust, long term trust, negative trust and misusing trust [1]. To evaluate visiting history of clients, trust value is used. The client who behaves better in history gets higher value of trust. Four aspects of trust are used for calculating overall trust value of the client.

1) Short term trust: It estimates recent value of trust. It is used to identify those clients who send session connection requests at a high rate when server is under session flooding attack.

2) Long term trust: It estimates long term behavior of client. It is used to distinguish clients with normal visiting history from clients with abnormal visiting history.

3) Negative trust: It is calculated by cumulating the distrust to the client, each time clients overall trust falls below initial trust value.

4) Misusing trust: It is calculated by cumulating the suspicious behavior of the client who misuses his cumulated trust.

Every time client makes session connection request, new trust value is calculated. The calculated trust value is stored at client side using cookies.

V. TRUST VALUE COMPUTATION

Every time when new session connection request is made by client, new value of short term trust and long term trust is first calculated. Short term trust relies on the interval of the latest two accesses of the client. Long term trust is calculated using the negative trust, average access interval and total number of accesses. Using long term trust, short term trust just calculated and misusing trust provided in the license, new value of overall trust is computed.

Negative trust is computed by cumulating difference of newly computed trust to the initial trust value each time new trust value is smaller than initial value. The misusing trust is computed by cumulating the difference in trust value if new trust value is smaller than previous value.

VI. TRUST BASED SHEDULER

The session connection request first reaches to the mitigation mechanism. Then new trust value is calculated. If it is below the minimum value then request is directly rejected. If it is above the minimum value then the scheduler decides whether to redirect it to the server based upon its trust value. If total number of ongoing sessions and number of waiting sessions is less than the threshold value of server then all requests are redirected to server. Otherwise requests up to threshold value are redirected to server in decreasing order of trust value.

This mechanism can be implemented as a package, which can run separately and redirect scheduled requests to web servers and thus mitigate session flooding attack.

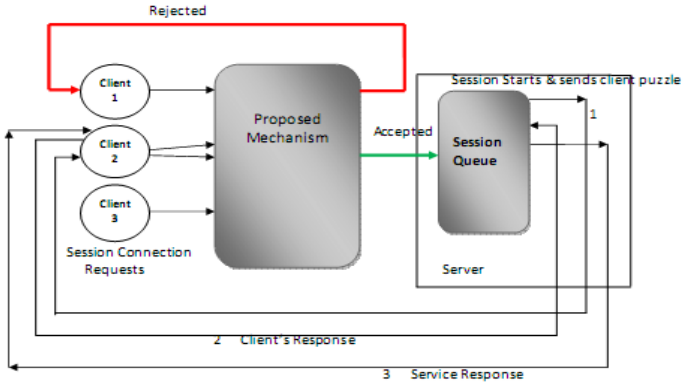


Fig. 1. Proposed Mechanism

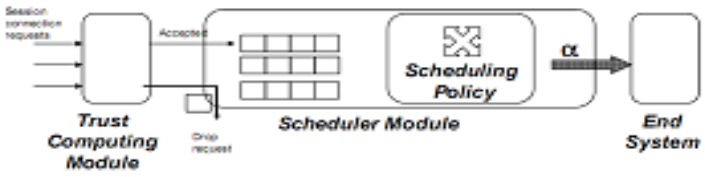


Fig. 2. Module Structures

VII. HANDLING REQUEST FLOOING ATTACKS

Once the mitigation mechanism for session flooding attacks redirects requests to web server, session is started. Request flooding attacks are those that send sessions with large number of requests than that of legitimate users. So here numbers of request are compared with predefined threshold and if it is less than threshold then all requests are processed in normal way. Otherwise some cost is imposed to the web client to make each such request [4].

The cost can be collected in terms of CPU cycles. Here server will send a puzzle to the client and wait for reply from that client before the request is processed. If client does not send reply, request will not be processed. Thus automatically rate of requests will be decreased as client's processer has to spend some time to solve the puzzle. When number of requests is less then this cost is very negligible but as number of requests grows it will be significant. It will cause source throttling effect. If requests are sent by compromised hosts then they might not be able to send reply of puzzle. JavaScript is used to implement this. When number of requests is more than threshold, java script is invoked to send the number '*n*' which is the product of two 4 digit prime numbers, to the client making the request. Then client has to compute two prime factors of '*n*' and send back the result. When the client sends answer, then and then only request is processed. Here processing power of attacker's CPU is used. This will achieve attacker source throttling effect. Source throttling module will calculate the value of '*n*' by taking two prime numbers '*p*' and '*q*' from primes array and multiplying them.

Algorithms to generate 'p' and 'q' values dynamically are as follows:

Algorithm 1: Generate p

```
GenerateP(NP,primes,st)
{
  pMapValue=(st) mod NP
  p=primes[pMapValue]
  return p
}
```

In the above algorithm the *st* represents the server's current time in milliseconds. As *st* differs for every millisecond the '*p*' value generated will be unique for each request.

Algorithm 2: Generate q

```
GenerateQ(NP,primes,cip)
{
  cip="A.B.C.D"
  ipMapValue=224*A+216*B+28*C+D
  qMapValue=(ipMapValue) mod NP
  q=primes[qMapValue]
  return q
}
```

In the above algorithm the *cip* represents the clients IP address and it is in the form of A.B.C.D. *ipMapValue* is the value that is generated from the client IP address and this value is unique for each client. So the '*q*' value generated for each client will be unique. The '*NP*' in the above algorithm represents the number of primes in '*Primes*' array.

VIII. RESULT AND ANALYSIS

Fig. 3 shows the change of overall trusts of attackers. Fig. 3(a) shows trusts of legitimate user. All requests are accepted as trust is above the threshold 0.1. It shows that the trusts of legitimate users quickly increase from 0.1 to 0.3 in first few sessions.

For Fig. 3(b), attacker use strategy 1. He sends session connection requests with fixed rate at one request per 30 seconds. The trust of attacker fluctuates and decreases below the threshold after few sessions.

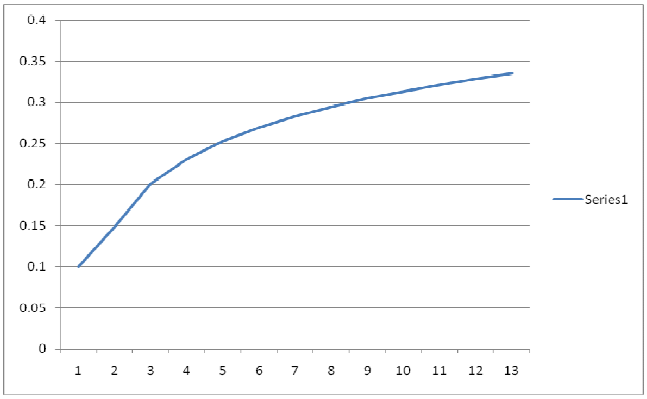
For Fig. 3(c), attacker uses strategy 2. He sends session connection requests at random rate. The randomness of attack rate causes fluctuation of the trust values as shown in figure.

For Fig. 3(d), attacker use strategy 3. He adjusts sending rate according to the rate of accepted requests by the server. The attack strategy increases fluctuation of trusts and most of the times trust value goes below the threshold and session is rejected.

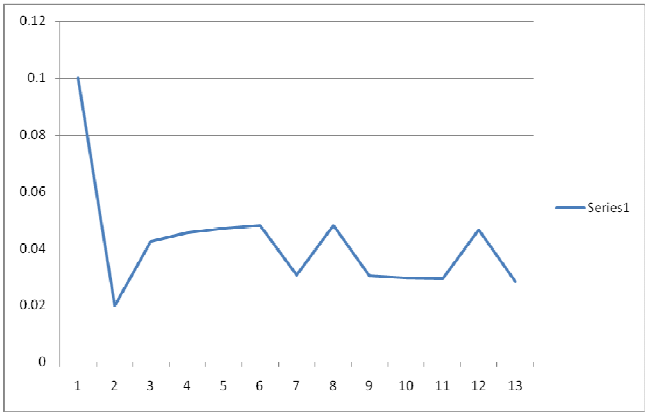
For Fig. 3(e), attacker use strategy 4. First he sends session connection requests like a legitimate user, so the trust value increases for first few sessions. But as he starts attacking by using strategy 2, misusing trust starts increasing and so within next few sessions trust decreases below the threshold and sessions are rejected.

The goal of request flooding attack is to send so many requests in one session that server remains busy in handling those requests and it cannot accept other legitimate user's requests.

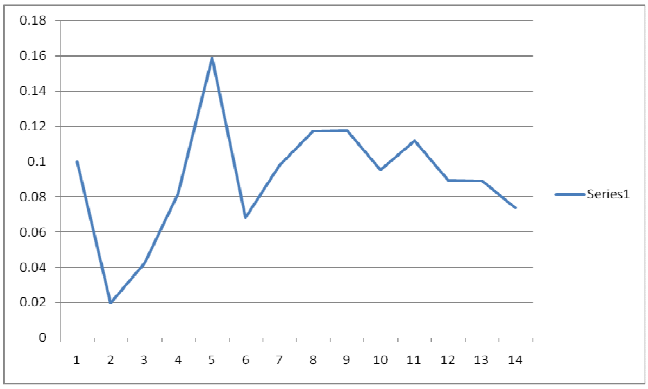
Here source throttling module is invoked to send puzzle to client, when number of requests in one session goes beyond the threshold. Thus for each next request cost is imposed on the client in terms of CPU cycles.



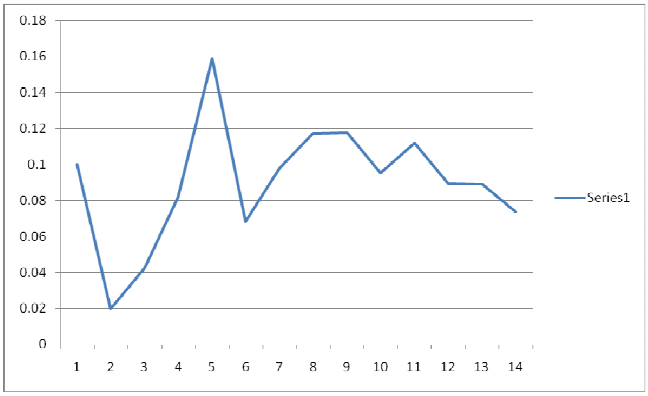
(a) No attack



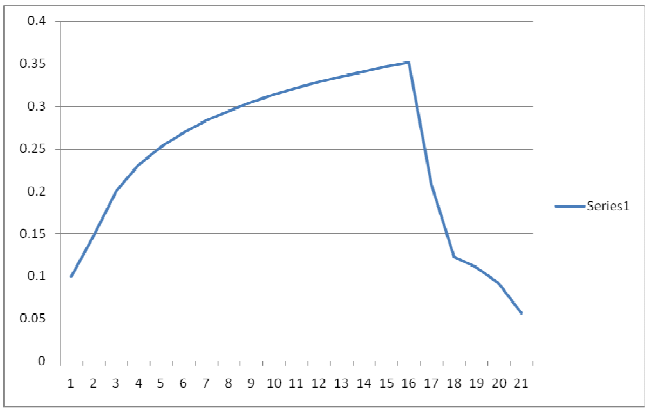
(b) Attack with Strategy 1



(c) Attack with Strategy 2



(d) Attack with Strategy 3



(e) Attack with Strategy 4

Fig. 3. Trusts Over the Number of Sessions

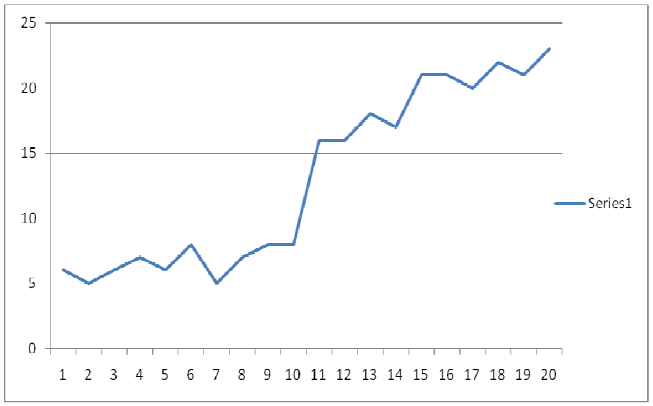


Fig. 4. Client's CPU Utilization Over the Number of Requests in a Session

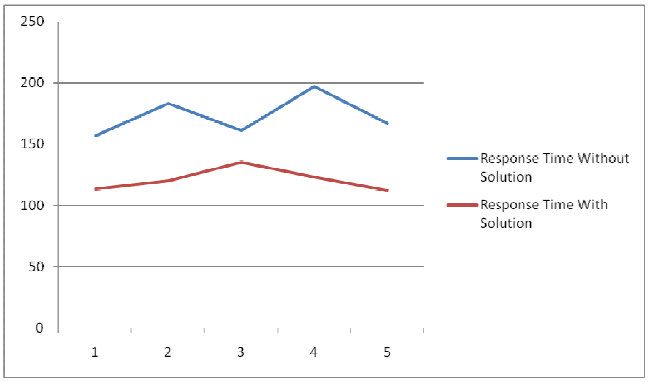


Fig. 5. Client's Response Time (in milliseconds) With Solution and Without Solution

Fig. 4 shows client's CPU utilization against the number of requests. When number of requests goes beyond the threshold, client's CPU utilization also increases due to source throttling module.

Fig. 5 shows graph of Response time of genuine user with and without solution. The graph shows that response time of genuine user decreases if proposed solution is used.

IX. CONLUSION

To defend against application layer DDoS attack is pressing problem of the Internet. Motivated by the fact that it is more important for service provider to accommodate good users when there is scarcity in resources, we have used lightweight mechanism to mitigate session flooding attack using trust evaluated from user's visiting history. The request flooding attack is also handled by throttling client's CPU. Due to this mechanism genuine user's response time decreases and attacks are mitigated. In future, work can be extended to mitigate other types of application layer DDoS attacks like asymmetric attack.

REFERENCES

[1] Jie Yu, Chengfang Fang, Liming Lu, Zhoujun Li. Lightweight Mechanism to mitigate Application layer DDoS attacks. In 4th International ICST conference, INFOSCALE 2009

[2] Supranamaya Ranjan, Ram Swaminathan, Mustafa Uysal, Edward Knightly. DDoS Shield: DDoS-Resilient Scheduling to Counter Application Layer DDoS Attacks. In IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 17, NO. 1, 2009.

[3] M. Srivatsa, A. Iyengar, J. Yin, and L. Liu. Mitigating application-level denial of service attacks on Web servers: A client-transparent approach. ACM Transactions on the Web, 2008.

[4] Saraiah gujjunoori, Taqi Ali Syed, Madhu Babu J, Avinash D, Radhesh Mohandas, Alwyn R Pais. Throttling DDoS Attacks. In Proceedings of International Conference on Security and cryptography (SECRYPT 2009), Milan, Italy, July 7-10, 2009.

[5] J. Yu, Z. Li, H. Chen, and X. Chen. A Detection and Offense Mechanism to Defend Against Application Layer DDoS Attacks. In Proceedings of ICNS'07, 2007.

[6] P. Niranjan Reddy, K. Praveen Kumar, M. Preethi. Optimising The Application-layer DDoS Attacks for Networks. In IJCSIS Vol. 8 No. 3, June 2010

[7] Y. Xie and S. Yu. A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors. IEEE/ACM Transactions on Networking, 2009.

[8] F.Douglis, A. Feldmanz, and B.Krishnamurty. Rate of change and other metrics: a live study of the World Wide Web. In Proceedings of USENIX Symposium on Internetworking Technologies and Systems, 1997.

[9] Yi Xie and Shun-Zheng Yu. Monitoring the Application-Layer DDoS Attacks for Popular Websites. In IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 17, NO. 1, 2009.

AUTHORS PROFILE

Ms. Manisha Mohan Patil has achieved B.E. (Computer Science and Engineering) degree from Walchand College of Engineering, Sangli in 2002. She is now pursuing M. E. (Computer Science and Engineering) degree from Dr. D. Y. Patil College of Engineering & Technology, Kolhapur, Maharashtra.

Prof. U. L. Kulkarni has completed M.E. (Computer Science and Engineering) degree from Walchand College of Engineering, Sangli. He is working as a Assistance Professor at Konkan Gyanpeeth's College of Engineering, Karjat, Dist.-Raigad, (Maharashtra) India. He has 11years of teaching experience. His research areas are Artificial Neural Network, Image Processing, and Network Security.