Institute of Advanced Engineering and Science

International Journal of Cloud Computing and Services Science (IJ-CLOSER)
Vol. 3, No. 2, April 2014, pp. 125 – 132
ISSN: 2089-3337

Detection and Prevention of DDoS Attacks on the Cloud using Double-TCP Mechanism and HMM-based Architecture

Krishna Modi* **and Prof. Abdul Quadir Md.****
*School of Computer Science and Engineering, VIT University, Chennai
**Faculty, School of Computer Science and Engineering, VIT University, Chennai

Article Info

Article history:
Received Feb 19^{th}, 2014
Revised Mar 20^{th}, 2014
Accepted Apr 10^{th}, 2014

Keyword:
Cloud Computing
DDoS
Application-layer DoS
HX-DOS
Double TCP
Hidden Markov Model

ABSTRACT

The rising interest in cloud services has increased the number of security issues in cloud computing. In recent times, several cloud based servers have experienced inevitable DDoS attacks. This papers deals with the denial of service attacks on cloud computing which are on a rising scale. Considering the TCP SYN Flood attack and Application layer DDoS attacks, the proposed architecture for cloud servers can detect them at an early stage and prevent the server from denying services to its legitimate users. The proposed method uses the Double TCP Connection mechanism to ignore the spoofed packets and establish connection only with the legitimate sources and prevent any SYN Flood DDoS attack on the layer 4. For the Slow HTTP Post and related Application layer attacks, a Hidden Markov Model based system called PBMRD has been proposed which gives probability of an incoming request of being malicious or non malicious and handles them accordingly.

Corresponding Author:
First Author
School of Computer Science and Engineering
VIT University
Vandulur-Kelambakkam road
Chennai - 600127
Email : krish512@hotmail.com

1. INTRODUCTION

With the new advancements in virtualization, the use of Cloud and Cloud based products is booming. Many of the private sector companies have shifted to cloud computing as it has many advantages over the traditional computing. Cloud Computing is known to provide Infrastructure (IAAS), Platform (PAAS) and Software (SAAS) as a service to its users. With highly personal data being stored on Cloud servers, it has to maintain an up time of more than 99.9% to ensure the quality of service to be delivered to its customers [1].

With such rise in the use of Cloud Computing, there has been a considerable growth in the issues regarding its security. In recent times, there have been a number of DDoS attacks on the Cloud Computing servers affecting their availability to the users considerably. DoS is Denial of service attack which flood the servers with huge amount of requests such that the server cannot handle them and starts denying service to requests received further. DDoS is Distributed Denial of Service attack which is performed by a group of network of computers together to flood the server which would in return lead it to denial of service.

Cloud Flare, a company which provides security for websites, was successful to handle and mitigate the largest ever DDoS attack in history [2]. Spamhaus.com was being flooded with requests. According to Cloudflare's blog, the peak network traffic was more than 300 Gbps of incoming requests. This is the highest ever recorded network traffic for DDoS attack which occurred in between 19th March, 2013 and 23rd March, 2013.

According to the Cloud Security Alliance's report, the DDoS attack holds fifth rank in the list of top nine most notorious attacks observed on the Cloud in the year 2013 [3]. Computerworld, a leading company dealing with computer and technology analysis mentions that DDoS is Cloud's security Achilles heel [4] which has denied the Cloud to be called a fully reliable source of solutions to our computing problems. With these reviews about DDoS attacks on the cloud, the next part explains the various types of attacks observed on Cloud which this paper deals with.

2. DDOS ATTACKS AND THEIR TYPES

DDoS attacks can be broadly classified in to two major types. First type of DDoS targets the Network Layer, the third layer of the OSI model.

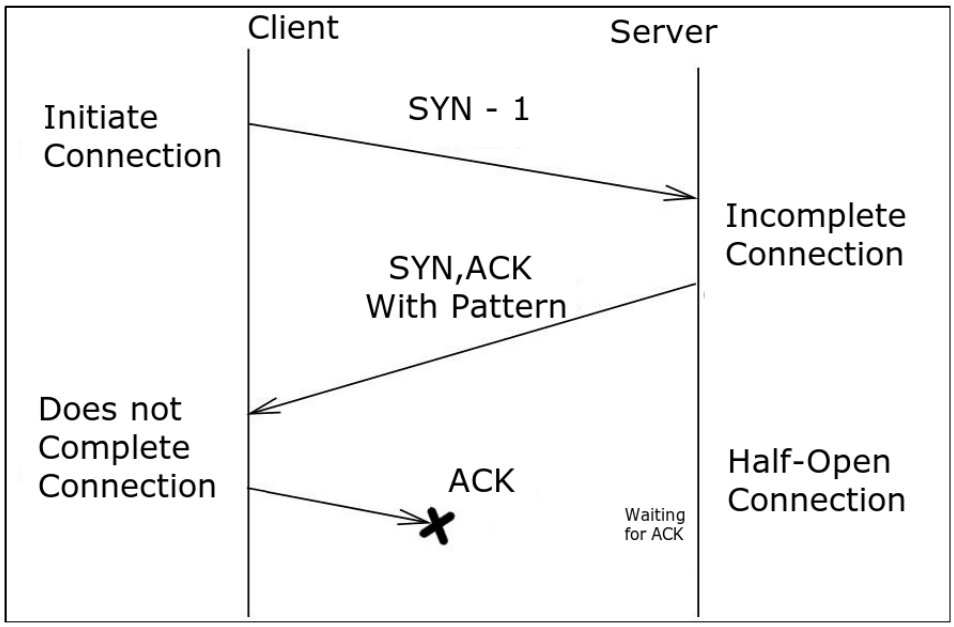

Figure 1. Half-Open TCP Connection

This attack tries to flood the network with connection requests but never completes them. Such kind of connections are known as half-open connection which occupy resources on the server machine but do not make any difference at the client's end. Figure 1 shows a diagrammatic representation of the same.

The second type of DDoS attack targets the Application layer which is the 7th layer of the OSI network model. This is the topmost layer of the OSI model which is closest to the end users as it deals with the applications that directly interact with their users. This is a new type of attack and very hard to detect and prevent. As cloud gives everything as service, it is highly prone to Application layer attacks.

For flooding the application layer minimal bandwidth is required and very few resources are used by the attacker. The main target for flooding in the application layer is not the service provided by the server but by holding all the connections it can handle parallel and not allowing any newer connection to get a free slot for execution.

For receiving requests, every server has a pool of request listeners accepting requests from the clients in parallel. Attacker tries to keep all these listeners busy with one very large request which is sent with the least possible speed and is never ending. This kind of attack is known as Slow HTTP Post attack. Another type of application layer attack targets the applications by sending malicious HTTP and XML requests which may be very time consuming or perform malicious tasks on the cloud server.

The proposed system deals with all these types of attacks and gives a solution better than most of the existing systems. Let us look at the work done by other major researchers in the same field.

3. RELATED WORK

As even the most secured servers were found defenseless against undetectable DDoS attacks and no fully secured mechanism has been found even after numerous efforts, this area of network security and Cloud security has gained much importance and a lot of research is being done on this topic recently with the emergence of Cloud. There are multiple researches available with effective mechanisms for detecting and preventing DDoS attacks. But they still fail to fully protect the system when a newer type of attacking strategy is used against the systems.

For a long time, in most of the Linux distributions, the default mechanism used to prevent system breakdown due to flooding attacks is by implementing SYN-Caches and SYN Cookies as mentioned in [5]. It is the most effective mechanism proposed yet which reduces the impact of the flooding attacks to a considerable extent. SYN caches maintain a lightweight hash of the IP Address, Port Number and secret bits from every incoming connection request packet. It does not allocate huge chunk of TCB for every incoming connection request. This makes the connections

lightweight and need a very high rate of packet flooding to perform DoS attack against the system.

Another mechanism proposed by this paper removes the need of allocating any kind of resources for the incoming connection request until it is completely established. It uses different specific sequence number for packets as per the client and remembers just the sequence number. The drawback of using SYN-Cache is that the options for TCP connection like window size, etc. need to be transmitted again to the server as it does not make note of them until a successful connection is established. The use of SYN-Cookies is also not fault proof as the sequence number of the initial packet can be guessed by the attackers and can hijack TCP connection sessions established by clients.

The SBTA system proposed in [6] is a much simpler and effective approach of defending systems from DDoS attacks. This system depends on the routers for probabilistically sending the path of the client by setting the identity field. It allows an attacker to be traced back using the path generated by the identifications given by the routers. The drawback of this system is that it needs a heavy packet flow from a client to finally detect and trace back its path. It gives result with an unacceptable delay time which might not help to prevent the system from the ongoing attack. Also, packets are robust in nature while travelling from client end to the server end. So the frequent change in path of the packet from the client to the server will lead to failure of the system in detection of the attack and the attackers source.

In the paper [7], the proposed system uses hop count as their measure to detect if the packet is from same source or different. The paper uses the final Time To Live (TTL) value and approximately estimates the initial TTL values. With these values, the Hop count for a packet can be measured. For every source, its hop count is maintained. If the value of Hop count varies considerably, then it is considered to be a spoofed packet meant for attack. But this system may again fail to estimate the Hop count if the attacker does not use the standard values for initial TTL in the packets.

The system proposed in this paper is based on the CSQD system as proposed in [8] for detection and prevention of flooding attacks on the application layer of Cloud servers. CSQD gives an exact architecture to handle every incoming request with different checking mechanisms. It has a self learning mechanism to learn and improve its detection with even newer attacks. The drawback of this system is that for learning every new attack, it needs the system to fail and be unavailable, only then it is considered to be a new unidentified attack and learnt by the system. On receipt of every such attack, the server remains unavailable which is not desired.

After surveying the most effective mechanisms for DDoS prevention, we have proposes a system in the following section which gives an improved performance and removes all the drawbacks that existed in the proposed or existing systems.

4. PROPOSED SYSTEM

The main drawback in detecting a DDoS attack is that it cannot be said for sure that the packet received is from a legitimate source. If we can tell for sure if the connection request is from a legitimate source, then we can surely proceed further in detecting and preventing the attack being carried on.

The proposed system uses a new method called Double TCP for establishing a TCP/IP connection between the server and the client which can tell whether the packet is coming from the source as mentioned in the header or if it is spoofed.

Using Double TCP, the Layer 4 DoS and DDoS attack performed on the server can also be prevented as no Transmission Control Block (TCB) is created until and unless the client replies back to the server with the pattern to authenticate itself.

On connection established, the system needs to check further chances of Layer 7 attacks hidden in HTTP or XML requests. The request is forwarded to the Probability-based Malicious Request Detection (PBMRD) system which handles these requests for a safe execution. The Double TCP connection mechanism and the PBMRD System are explained in the sections below followed by the explanation on the Hidden Markov Model and its use in the proposed system.

4.1. Double TCP Connection

Unlike its name, this method does not use more than one TCP connection to communicate though the ends. Traditional TCP protocol performs a 3-way handshake to establish a connection between the participating members, which means on total only 3 packets are used for connection establishment.

In Double TCP, it uses minimum 5 packets carrying out the 3-way handshake process for 2 times. It also uses SYN Caches for maintaining the log while TCP connection is established. The first 3-way handshake is purposely kept incomplete by the server. The second 3-way handshake for TCP connection overwrites the previous attempt for connection by using same TCP source Port Number and of course source IP address, but with a new identity field.

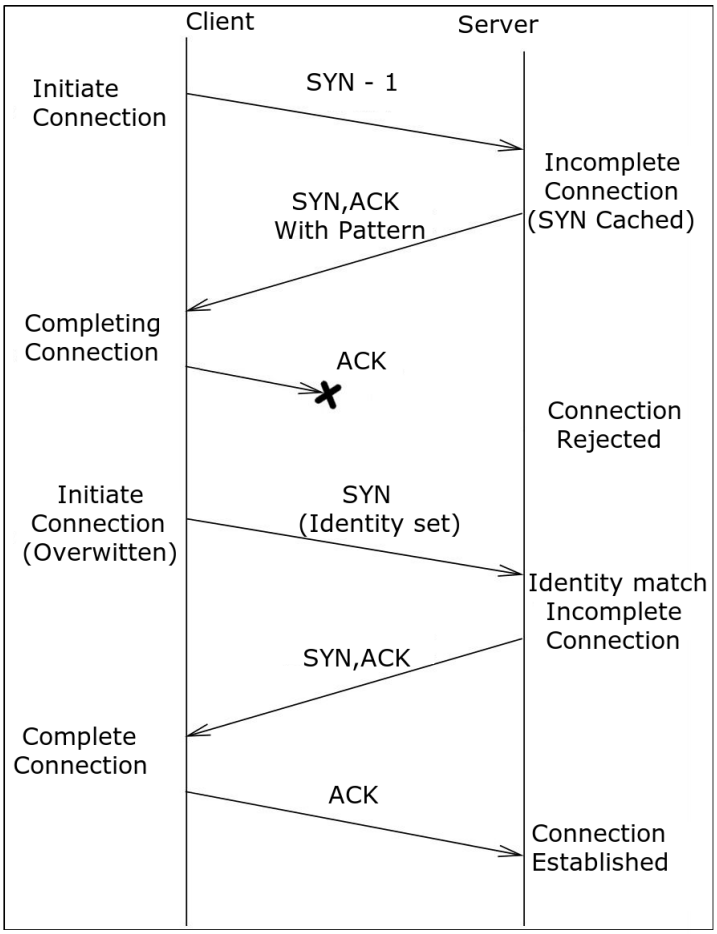

Figure 2. Double TCP Connection mechanism

This Double TCP Connection method is compatible to work with currently existing TCP protocols with no modification required. Double TCP works like the usual TCP connection for the clients. Only the server needs to handle the mechanism in special way. Figure 2 depicts the process in detail. Following is its step by step explanation.

Firstly, the client who intends to establish a connection with the server sends a TCP packet with the SYN flag set. Assume that the source IP address for this packet is ”a.b.c.d” and the source port number is P.

On receipt of SYN packet from the source, the server uses the concept of SYN Caches to store the 32 bit IP address, 16 bit Port number and a 16 bit pattern together as 64 bit hash entry in a table. There is no Transmission Control Block (TCB) allocated yet which is usually takes more than 280 bytes [9]. Just a 64 bit hash (8 bytes) is entered in the connection request table. The server replies to the SYN packet by sending a SYN ACK packet with the 16 bit pattern stored in the 16 bit Identity field of the packet to the source address ”a.b.c.d” at the port P.

The client, if source address was not spoofed, receives the packet. The client may or may not send the final ACK packet which is ignored by the server.

The client now needs to start connection establishing process again from sending the SYN Packet to the server, but this time with the Identity field containing the 16 bit pattern as received in the previous SYN-ACK packet from the server.

The server extracts the source IP address, source Port number and the Identity field value from the received packet containing some value in the Identity field. Then the server compares this value with the values in the connection request table. If the entry exists then the source is sent SYN-ACK else the packet is dropped.

The client has to send the final ACK on receipt of SYN-ACK from the server and the connection is established successfully.

With this method, there can be no half open connections created as the client would fail to reciprocate with the pattern. The SYN Cache uses very less amount of memory and has a long buffer length which cannot be filled up easily. Every entry in the connection request table expires after 511 seconds [5] and is removed.

If the client further tries to carry out Application layer (Layer 7) attack, then the source can be easily traced down from its source IP Address.

The SYN Cache mechanism could not store the extra information like windows size, options, etc in a packet while establishing a connection. The client had to re-transmit those fields after connection was established [5]. The main advantage of Double TCP over the usual SYN cache method is that it can use all the fields during connection establishment process.

On successful establishment of connection, the connection is handled by the PBMRD system to check for malicious requests intended for Application Layer attacks.

4.2. Probability-based Malicious Request Detection (PBMRD)

If the client further tries to carry out Application layer (Layer 7) attack, then the source can be easily traced down from its source IP Address. Probability-based Malicious Request Detector (PBMRD) identifies the request and classifies it to be malicious or non-malicious for the further measures to be taken accordingly.

PBMRD uses Hidden Markov Model (HMM) to detect the probability of a given request to be malicious. Using this probability, three different cases can be formed. For a very high probability, the request is termed to be malicious and added to the blacklist database. For an average valued probability, the request is sent to the XML Vulnerability Detector for further analysis of the request. If the HMM returns lower probability, then the request is handled further by the Request Scheduler. Figure 3 shows the complete flowchart of the proposed system.

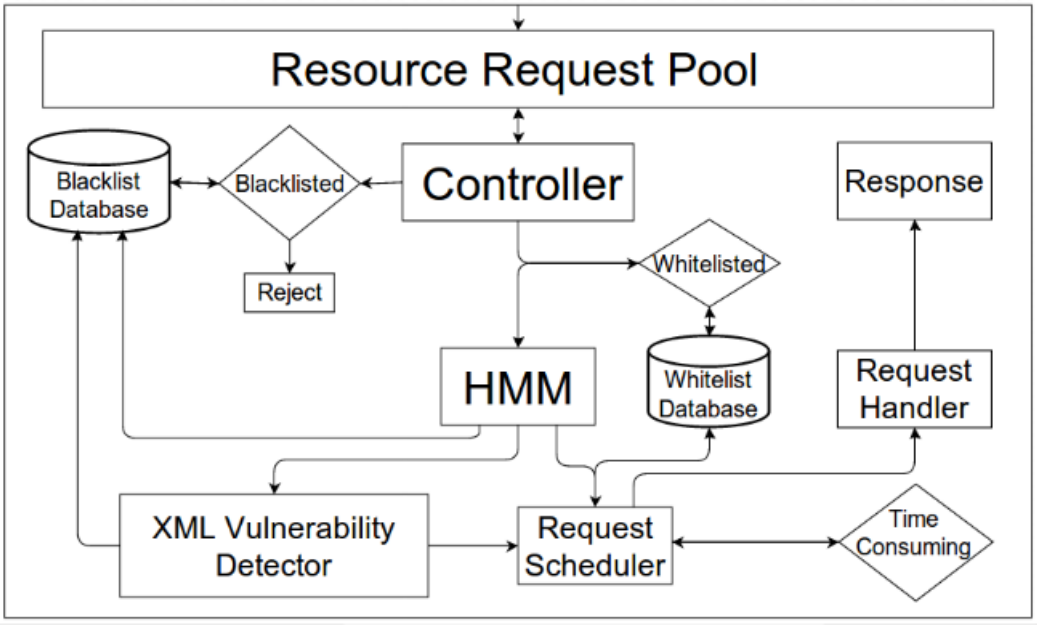

Figure 3. Probability-based Malicious Request Detector (PBMRD)

Resource Request Pool is a pool of processes listening to requests from the client side. This request pool is susceptible to the Slow HTTP post type of DoS attacks. To prevent it, PBMRD has the Controller which keeps a check on this Resource Request Pool.

A client needs to send just 1 byte of data after every 110 seconds to keep the connection alive. To defend the system, after 20 seconds of every request's arrival, the controller check if there is any data from the request available. If no data has been received yet, then the request is discarded.

On receipt of a request from the client, it is simultaneously checked against the Blacklist Database for known malicious requests and the Whitelist Database for known safe requests. An identified malicious request is straight away discarded. An identified safe request is sent directly for execution depending upon its probability. The safe request is again checked by the HMM model after its every product of safety probability with 100th occurrence. If the request does not match with any known attack sequences, it is sent forward to the HMM block.

PBMRD uses Hidden Markov Model (HMM) to detect the pattern in the incoming requests and prevent any malicious request from affecting the system. HMM gives a probability of the request being a malicious one. If the probability is too high, the request pattern is added to the Blacklist database. If the probability is average valued, it is

sent to the XML Vulnerability Detector for further detection mechanism. And for lower probabilities, the request is sent to the Request Scheduler for executing the request.

XML Vulnerability Detector uses the request header and content to check it for any possible irregularity. It checks the size of the request, header parameters, content of the request and matches them all for any possible DoS attack pattern. On detection, the request is transferred to the Blacklist database.

Request Scheduler sends the request to Request Handler to process the request and get the response message for the client. If the request processing takes longer than usual, then it is sent to wait state and scheduled with other request process by the Request Scheduler.

4.3. Hidden Markov Model (HMM)

Hidden Markov Model is a Markovian model which is based on the principle given by the Russian mathematician Andrey Markov, which states that, the future state of a process in dependent only on the current state of that process and not on its previous past states. In Hidden Markov Model, the states of the process are not visible but their outputs are visible. With these outputs, it can be probabilistically estimated which state the process is in using the formula of HMM's Forward-Backward algorithm as given below,

$$P(X_t \mid Y_{1:t}) = \frac{P(Y_t \mid X_t) \cdot P(X_t \mid Y_{1:t-1})}{\sum(P(Y_t \mid X_t) \cdot P(X_t \mid Y_{1:t-1}))} \quad (1)$$

With this formula, the probability of the current state can be estimated using the data from the previous state. This is the model used by PBMRD for the computation of probability of an HTTP request being malicious or non-malicious. HMM is widely used for temporal pattern recognition for applications like speech and handwriting recognition [10].

The Double TCP Connection mechanism handles the layer 4 DDoS attacks and the PBMRD system manages all the Application Layer attempts of DDoS attack in the form of HTTP and XML requests.

5. EXPERIMENTAL RESULTS

To test the proposed system, we have used two Linux powered machines of which one runs the server program and other runs the client program. Wireshark being the most popular network packet analyzer tool available, we have used it to analyze and test all the communication between the client and the server using the TCP protocol over the wireless LAN network.

The packets transmitted during this communication were captured and studied for further implementation. A similar TCP communication was implemented using hping3 tool in Linux bash to frame custom packets and transmit them to the client on request of a connection.

We have used the proposed Double-TCP mechanism to establish a connection between the client and the server using the custom TCP packets sent from the server-side to the client. A successful connection could be easily established.

On experiment with a variety of options set for the TCP connection between the client and the server with and without Double-TCP connection mechanism, we observed that on an average it takes 0.7367 times more time for Double-TCP based connections as compared to the regular TCP connections. This extra time required for TCP connection using the proposed mechanism can be considered feasible and acceptable as in the protocol HTTP/1.1, an established TCP connection can be reused for fetching a number of files rather than making a separate connection for each file to be fetched from the server [11]. Hence, the extra time taken for establishing Double-TCP connection will hardly affect the performance of the system.

Figure 4 shows a graph representing the performance of Double-TCP connection mechanism in comparison to regular TCP connections as a function of time taken for TCP connection with the number of simultaneous connection attempts.

To implement the PBMRD architecture for checking the HTTP requests for malicious traits, we have used a python program implementation of a web server. This python web server runs on port 8080 and accepts all the requests from the web browser of the client transferred to the server's port.

Using the given HMM model, every expected incoming request is given its initial probability distribution. After setting up all the initial, emission and transition probabilities, the server completes its initialization and starts accepting the HTTP requests from the client. For every incoming HTTP GET request, the server computes the probability according to the HMM model and takes suitable action as per the computed values.

Over the time, the observation was that the probability values get stable and take value with minimum variance. Whenever an attacking sequence of HTTP requests gets initiated, there is a change in the probability distribution

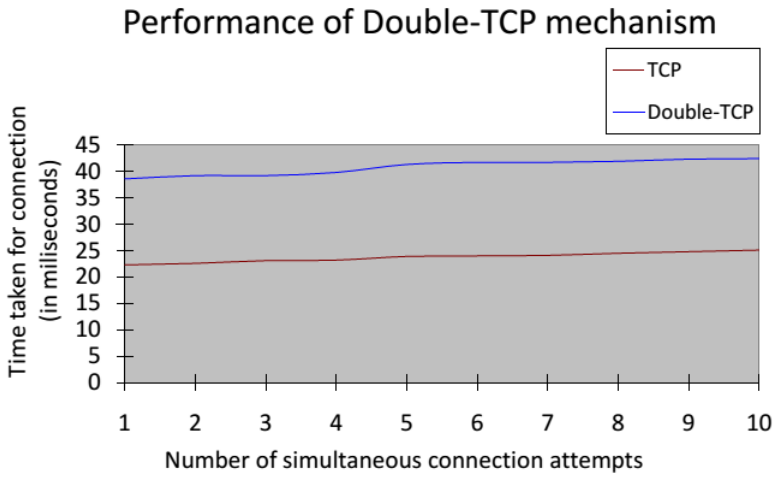

Figure 4. Performance of Double-TCP Connection Mechanism

which denotes the high probability of an attack on the system.

For various different combinations of attacking sequences, on an average we observed that the proposed system with the HMM model fairs well and gives a positive attack detection rate of 92%. This observed rate of positive attack detection is considerable high and serves the purpose of the proposed system. The implementation of the blacklist database and whitelist database give a reduced computation rate resulting in higher speed of execution.

6. CONCLUSION

With the results observed from the experiments performed on the implemented proposed system, it is evident that the Double-TCP connection mechanism is practically feasible and removes the drawbacks of the currently existing SYN-cache protocols which are implemented in most of the Linux-based systems as default.

Also, another prominent result obtained was a high detection rate given by the proposed HMM-based PBMRD model. As compared to the performance of the CSQD system proposed in [8], the PBMRD removes the drawbacks where the CSQD could detect new attacks only on failure of the system. In PBMRD, as apparently visible, the system learns new attacks without causing any failure in the system and carrying on with the remaining legitimate requests without any disturbance because of the interfering malicious requests with the help of the Blacklist database. Also, PBMRD has considerable reduced number of computations as it uses white list database to avoid useless overhead in needless computation for known safe requests.

On account of the efficient output obtained from the experimental results performed on the proposed system, we can conclude inferring that the proposed system is practically feasible model and gives a much improved result with better performance and higher efficiency as compared to the other existing and proposed systems with least possible computation overhead.

ACKNOWLEDGEMENT

We thank our Program Chair, Dr. Rajesh Kanna for helping us to complete our research successfully with his expertise in Computer networks and packet analysis. We are grateful to him for suggesting such a challenging and innovating field for research. We sincerely thank him for his supervision and constant motivation to help us write this paper in a more versatile manner. We also thank the dean, Dr. L. Jagannathan for encouraging us for research work and providing us with all the facilities required for the field study and experiments.

REFERENCES

[1] R. K. Yadav, Daya Gupta, Devendra Dadoriya, "Prevention Of DOS & DDOS Attack Using Count Based Filtering Method In Cloud Computing", *International Journal of Engineering Research & Technology (IJERT)*, vol. 2 issue 6, June 2013

[2] Matthew Prince, "The DDoS That Almost Broke the Internet", *Cloudflare*, 17 March, 2013

[3] Top Threats Working Group, "The Notorious Nine: Cloud Computing Top Threats in 2013", *Cloud Security Alliance*, February 2013
[4] Tim Lohman, "DDoS is Cloud's security Achilles heel", *Computerworld.com*, 16 September, 2011
[5] Jonathan Lemon, "Resisting SYN flood DoS attacks with a SYN cache," *FreeBSD Project*.
[6] Lanjuan Yang, Tao Zhang, Jinyu Song, et al "Defense of DDoS Attack for Cloud Computing", *IEEE Computer Society*, 2012
[7] Vikas Chouhan, Sateesh Kumar Peddoju, "Packet Monitoring Approach to Prevent DDoS Attack in Cloud Computing", *International Journal of Computer Science and Electrical Engineering (IJCSEE)*, ISSN No. 2315-4209, Vol-1 Iss-1, 2012
[8] Reza Manouchehri Sarhadi, Vahid Ghafori, "New Approach to Mitigate XML-DOS and HTTP-DOS Attacks for Cloud Computing", *International Journal of Computer Applications*, Volume 72 No.16, June 2013
[9] Wesley M. Eddy, Verizon Federal Network Systems, "Defenses Against TCP SYN Flooding Attacks", in *The Internet Protocol Journal*, Volume 9, Number 4.
[10] Sreeja Rajesh, "Protection from Application Layer DDoS Attacks for Popular Websites", *International Journal of Computer and Electrical Engineering*, Vol. 5, No. 6, December 2013
[11] Internet RFC 2616, "Hypertext Transfer Protocol – HTTP/1.1", *IETF Tools*.

BIOGRAPHY OF AUTHORS

Krishna Modi is a post graduate student pursuing for Master of Technology in Computer Science and Engineering with specialization in Cloud Computing at VIT University, Chennai, India (2014). He obtained Bachelor Degree in Computing Engineering from NMIMS's Mukesh Patel School of Technolog Management and Engineering, Mumbai (India) in 2013. His researches are in fields of cloud security, network security, image processing, and storage systems. He is a certified Cloud Infrastructure and Services Associate by EMC. He works as a freelancer and has developed a number of Cloud-based SaaS over Google's AppEngine platform. Besides, he is also involved in reearch projects for companies, student associations, and managing his own company Khandesh BPO Services Pvt. Ltd. Further info on his homepage: http://www.krish512.com