ISSN:0975-9646

Handling DDoS attacks in Cloud

Yesu Jeya Bensh P, Murugan K

College of Engineering, Anna University,

Chennai, India

***Abstract* — Cloud is the fastest growing computing platform. Researches have demonstrated that the essential issues of Distributed Denial of Service (DDoS) attack and defense is the resource competition between the attackers and defenders. A cloud usually have profound resources and has full control and dynamic allocation capabilities. Therefore, cloud offers us the potential to overcome DDoS Attack. We propose a cloud enabled defense mechanism for internet services against DDoS attacks. We propose a dynamic resource allocation strategy and a shuffling mechanism to compute the optimal reassignment strategy for clients on attacked servers, effectively separating benign clients from attacking clients. The proposed shuffling based moving target mechanism enables effective attack containment using fewer resources than attack dilution strategies using server expansion. Using Amazon EC2 [1] instances we demonstrate that we can successfully mitigate large scale DDoS attack in a small number of shuffles.**

***Index Terms* —** *Distributed DoS, Cloud, Mitigation, Shuffling Mechanism, Moving target Defense.*

I. INTRODUCTION

DDoS attacks pose one of the severest security threats to today's internet services. A recent worldwide survey reports both increased DDoS attack intensity and sophistication over the past few years. Popular web services such as e-commerce, financial, banking and blogs are the top targets of the DDoS attacks. However, practical DDoS mitigation still primarily depends on packet filtering using access control lists (ACLs) and traffic signatures [2] despite their functional and operational limitations [3]. Today's attackers uses both volumetric and application layer attacks to execute DDoS [3].

This paper introduces a cloud-enabled and shuffling based moving target defense mechanism that confuse, evade, and isolate attackers from benign clients. Unlike the Moving target solutions [4], [5] that were designed for restricted applications and threat model, this mechanism provides a comprehensive mechanism that protects generic Internet services from both network and computational DDoS attacks alike. By turning the protected servers into moving targets through carefully planned shuffling operations, the proposed architecture not only can evade the naive bots that only target static server location but also able to segregate persistent bots persist their attack to the moving server replicas. To offer protection for Internet services, our defense mechanism takes advantage of resource elasticity and Auto Scaling [6] property in modern cloud computing environments.

When the server is bombarded by a DDoS attack, we perform quick and selective replication of the server which is under attack. Instead of expanding the server in a blind way, we replicate the server instance, which is in attack, in a different location. The clients attached to the server instance was separated as benign clients and bot clients using the separation algorithm and the benign clients are migrated to the new server and to other server which is not under attack. The detected bot clients were redirected to the old server instance. Once all the clients in the server is identified as bots. The attacked servers are taken offline and recycled. The Duplicated server location was unknown to the outside world since the replicated servers are only known to the subset of clients assigned to them. This moving target mechanism automatically evades the naïve bots that target static server locations.

The persistent bots will follow the moved replica servers as benign clients do, and target their attack to the new server location. To segregate those sophisticated and persistent attackers from benign clients, we devised a shuffling mechanism that intelligently assigns clients to the replica servers and keep track of the assignments. By keeping track of the assignments and whether a new replica is also attacked, we can quickly identify the attacker and separate the benign client sessions from the bots. To quickly segregate the persistent bots and mitigate the attack, we aim in isolating the persistent clients and the benign clients through client-to-server shuffling mechanism. By replicating the application servers having the impact of the attackers, we can able to overcome the attacks targeting both network and computational resources.

We designed a dynamic programming algorithm to guide the client-to-server re-assignments throughout the shuffling operations. The dynamic programming algorithm will monitor the activities of the client and reassign based upon the behavior of the client. Our approach is suitable for mitigating DDoS attack in on application layer.

In this paper we made the following contributions:

- We designed and developed a family of dynamic algorithms to monitor the client activities in each application servers and to identify and segregate the benign clients from the bot clients.
- We propose a novel shuffling based mechanism to mitigate large scale DDoS attacks on web services. By duplicating the attacked servers and re-directing the benign clients to the new servers, we can gradually reduce the DDoS attack in the network and restore the quality of service to the benign clients.
- Unlike other mechanisms our proposed mechanism do not require ISP collaboration. With the advantage of resource elasticity and

low cost cloud computing technology, we can deploy our mechanism in the cloud environment.

- We implement the proposed model in Amazon Web Services by creating a Web application and tested it with generating attacks with many clients. Experiments with the developed system proved that we can quickly mitigate the DDoS attack with minimal latency.

The remainder of this paper is organized as follows. The related Work is discussed in Section II. The mitigation mechanism is discussed in Section III. System modelling is discussed in Section IV. Shuffling based segregation is discussed in Section V. The implementation of the proposed model is discussed in Section VI. The results and the summary is discussed in Section VII & VIII.

II. Related Work

Shui Yu and Song Guo [7] proposed a mitigation algorithm by installing intrusion prevention system (IPS) in front of the application server. In this proposed model the clients should pass through the IPS to reach the application server. When the system is under attack the IPS will be duplicated and the clients filtering will be distributed to the IPS servers.

In [4] Arnob Network researchers had discussed about the different types of DDoS attacks in current market and proved that data-center edge is the best place to perform application layer DDoS detection and mitigation.

Persistent bots uses IP Spoofing to hide their identity while executing DDoS attack. Chen Jin, H Wang, Kang and G. Shin [11] proposed a solution to prevent DDoS attack in Cloud Computing using the hop count filtering mechanism. The system will monitor all the packets arrived to the system and the TTL field in the IP header will be checked to get the Hop-Count value. They desingned and developed a Hop Count Filtering that builds an accurate IP-to-hop-count (IP2HC) table to detect the Spoofed IP packets and filter the spoofed packets.

Quan Jia, Kun Sun and Angelos Stavrou [5] a moving target defense mechanism called MOTOG to secure service access for authenticated clients from DDoS Attack. In MOTOG the application Server IP is concealed from the clients communicating the server. Group of Proxy nodes are installed in the distributed environment. The clients will communicate with the authentication server and the authentication server will assign the proxy nodes to the client. The proxy nodes relay communication between the server and the client. With this architecture it is difficult for the attacker to trace the application server and initiate the attack. The client will be segregated in the authentication server and in the proxy nodes. They proposed a Greedy algorithm for computing client-to-proxy assignment.

J Francois, I Aib and R Boutaba [9] proposed a method to detect and prevent DDoS by implementing Intrusion Prevention Systems (IPS) in Internet Service Provider (ISP) Level. They form multiple Virtual protection rings around the servers to detect and protect the servers from the attack.

Quan Jia , H wang, D Fleck, F Li, A Stavrou and W Powell [8] proposed a Cloud mitigation mechanism to prevent DDoS attack. They implemented the MOTOG architecture in the cloud environment. The mechanism triggers only when the system is under attack. Once the attack is detected, the clients will be re-assigned by client-to-proxy assignment algorithm [5]. They included only the attacked servers in the shuffling process. The clients in the attacked servers will be shuffled between the attacked servers.

P Ferguson and D. Senie [10] proposed a method to defeat DDoS attack by Ingress traffic filtering installed in the routers by restricting the packets originating from downstream network with other IP address.

Most of the discussed defeat mechanisms involved ISP's and Routers. Mitigate DDoS in Application layer in the cloud environment is still in early stage. Most of the mitigations used the IPS's. In this paper we proposed a mechanism without integrating ISP's and IPS's.

III. DDoS mitigation in Clouds

In this section, we propose a mechanism to mitigate the DDoS attack in clouds by implementing the moving target defense mechanism to isolate the benign clients from the attackers. We achieve this by dynamically allocating extra resources by cloning the attacked Application servers. The architecture consists of Load Balancers, a Coordination Server and Application Servers. The Load Balancer will allocate the servers to the clients by re-directing the requests to the application servers. All the incoming requests to the Application servers will be monitored. The server will be monitored under normal conditions and the average requests will be calculated with the records. The Application servers communicate with the coordination servers by sending the status of the server, client details, number of requests processed and the request received from individual clients. We assume the occurrence of DDoS Attack is detected by abnormal increase in the traffic. When the attack is detected, it send the message to the coordination server about the attack. The coordination server will duplicate the application server. Coordination server starts the segregation algorithm and the shuffling algorithm. The Clients will be segregated as bot clients and benign clients with the records send by the application server by the segregation algorithm. After segregating the clients the shuffling algorithm will start re-assign the server to the benign clients. All the benign clients will be redirected to the servers which are not under attack and the details of the bot clients will be informed to the application servers to block the traffic. If the bots are identified as benign clients and re-directed to new servers, it will identified easily by the abnormal traffic in the new server and the client will be blocked. Once all the clients in the application servers are identified as bots, the server will be taken into offline and recycled. By shuffling the benign clients to the non-attacked servers, the Quality of Service (QoS) will be maintained.

IV. SYSTEM MODELLING

In this section we discuss about the system architecture, approximations made in this proposed system and the components involved in the system to enable the shuffling based moving target defense mechanism. The Architecture and the components of the proposed model is given in fig 1. The Figure does not include the DNS lookup activities. It has only the Cloud domain components. When the Clients enters the Web Service/Application address, the DNS will redirect the users to the Load balancer IP which then allocates the Server to the clients. One or more Load Balancers are installed Depend on the Application request. The Load balancer keeps records of the Number of servers in the cloud and the list of Clients associated to the servers. The Coordination server is the central controller and maintains the server allocation. Next, we describe each components in the proposed system in detail.

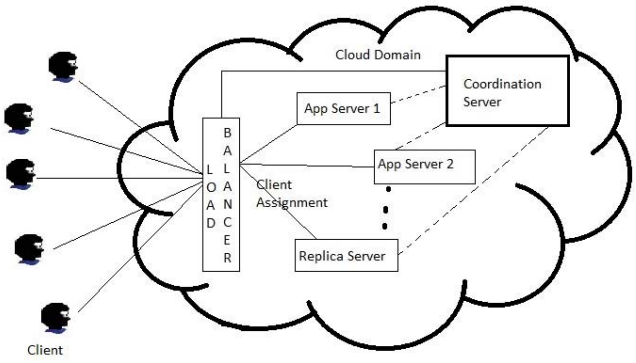

Fig 1. Architecture and Components

A. Load Balancer

A load balancer automatically distributes the incoming traffic across multiple servers in the cloud. It assigns an active server replica to the new clients. Clients are identified by their IP address. Each Client IP is assigned to only one server and one server may assigned to multiple Clients. While During the client-server re-assignment process, load balancer will inform the server and the client about the client-to-server re-assignment. A minimum of one load balancer will be installed in the cloud domain to keep track of all the servers in the cloud and the client-server assignments.

Multiple cloud domains can be installed for the service and multiple datacenters can be deployed across the geo locations by the same cloud service provider or by different providers. Fault tolerance can be improved by installing multiple load balancers per cloud domain. Load balancers maintain a up-to-date list of application servers running in the cloud domain. Any kind of load balancing algorithms can be implemented to assign clients to the server. Amazon AWS EC2 uses Round-Robin algorithm and least outstanding request routing algorithm to assign clients. It is reported that the throughput of dedicated Amazon load balancers can reach 40Gbps. To avoid the bottleneck multiple load balancers can be deployed at each cloud domain to increase resiliency against huge flooding attacks.

B. Replica Servers

Replica Servers or Applications servers are more vulnerable to DDoS Attack. Since we re-direct the client to the replica server, the attack packets will be routed to replica servers. Under normal condition only few servers will maintained and the traffic will be monitored. The traffic report will be send to the coordination server on every period of time. Application server will save the details of the traffic and calculate the average traffic for every minute. Whenever the servers reaches the traffic level much higher than the average, it will trigger an alert to the coordination server. When the DDoS attack is detected and informed to the coordination server, coordination server will duplicate the server across different geo locations in the cloud. The clients served by the attacked application server will be segregated as bot clients and benign clients by the coordination server and the benign clients will be re-assigned to other replica servers across the cloud. Our solution is able to isolate both the computational and network resources because we use individual replica servers to compartmentalize the impact of attackers on both network level and computation level.

For a longstanding attack, the botnets have to chase the moving replica servers continually. With the Assumption that the botnets are incapable of overwhelming the cloud infrastructure, they have to rely on persistent bots to chase the moving replica clients. This will be discussed in the forthcoming section while our shuffling mechanism can identify and isolate the benign clients and persistent bots to save the benign clients from the bots.

C. Coordination Server

The coordination server maintains global state of the system. Coordination server will maintain the states of the application servers run across the cloud. It tracks the number of clients attached to the servers and the number of requests are processed for the clients at a particular amount of time. The replica servers will send the report about their client details and the numbers of requests received by them at an interval of time. When a server is attacked, it informs to the coordination server. The coordination server will run the segregation algorithm with the client records send by the server over the period of time and start to separate the benign clients and the bot clients. Based on the number of servers attacked in the cloud and the strength of the attack, the coordination server computes an optimal shuffling plan that optimizes the probability of attack segregation. The coordination will separate the bot clients and the benign clients and re-assign the benign clients to other replica servers in the cloud. The detected bot client IP's will be send to the load balancer and other application servers to block the request/traffic from those clients. The coordination sever is not open to the world. Only the servers within the cloud domain can access the coordination server. Hence attacking coordination server is not possible and the coordination server will communicate the replica servers via a dedicated command and control channel which is not accessible by any of the clients.

V. SHUFFLING-BASED SEGREGATION

This section presents the in-depth analysis of development of the segregation algorithm and the shuffling mechanism. The ultimate goal is to separate the benign clients from the persistent bots that trace the moving replica servers to perform the DDoS attack. It is hard to identify the persistent bots since it behaves like the benign clients. With our solution, new replica servers are installed dynamically when a servers is bombarded by DDoS attack. The segregation algorithm will separate the benign clients and the bot clients by the client records. A record will be maintained by the coordination server about their behavior in the system. After segregating the benign clients the shuffling algorithm will re-assign the servers to the clients. Most of the servers will participate in the shuffling mechanism. Even though the bot client is identified as benign client, it will be identified soon after the sudden traffic change in the re-assigned server and the service to the client will be blocked. The identified bots will be re-assigned to the server under attack. Once all the clients are identified as bots by repeated computation, the server will be taken into offline and the list of bot clients will be broadcasted to all the servers in the cloud and the load balancer to drop the request from the clients.

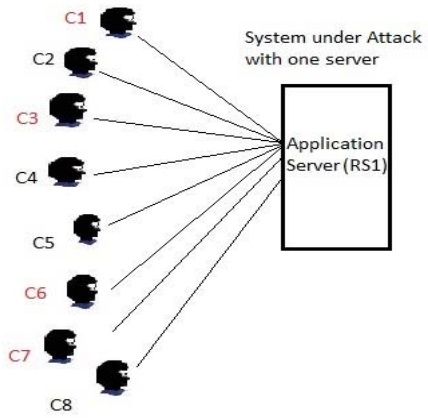

Fig 2. System under Attack with one Server

To illustrate the concept of client-to-server assignment and the segregation, we show the steps for one round of shuffling through the example in fig 2 and 3. The initial state of the system under attack with one server is shown in fig 2 which contains eight clients {C1, C2, … , C8}. Among these eight clients C1, C3, C6 and C7 are bots that performs the DDoS attack on the server. The Server will inform the coordination server about the attack and send the clients details. It will immediately replicate the server and identify the benign clients and bots with the past activities of the client details provided by the server. After a round of segregation, the clients identified as benign are re-directed to the replicated server. The new server details were not informed to the client performing attack and the clients will continue with the old server. In the given example three clients were identified as benign clients in the first round of segregation and swapped to new server. The IP address of the bots will also be informed to new servers to drop the packets from the clients. The coordination server needs to be informed periodically by the servers in the cloud to take decision with optimal solution. The coordination server will run multiple rounds of segregation algorithm to segregate benign clients from the persistent bots.

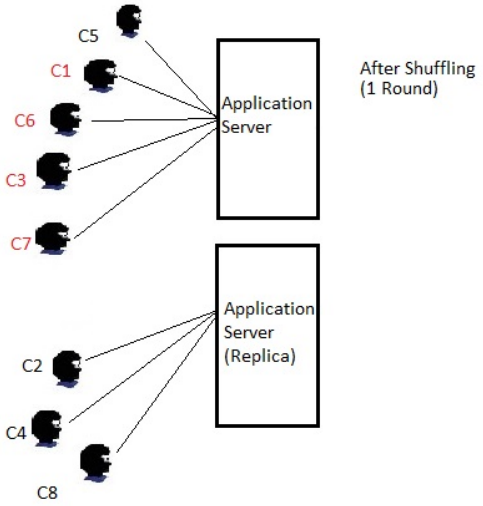

Fig 3. After first round of shuffling

The optimized shuffling process will enable us to mitigate a strong DDoS attack in a few rounds of shuffling.

A. Traffic Average Calculation Algorithm

This algorithm will run the application servers. In this algorithm we first observe the arrival patterns of the requests in non-attack case. This will be running as a demon process in the servers. It will monitor the incoming traffic in the particular server and keeps record of the traffic. With the recorded data it will calculate the average traffic flow of the server. The algorithm to calculate the traffic average is given in Algorithm 1. If the server receives a huge traffic than the average traffic, it is considered as an Attack and triggers a message to the coordination server along with the client details to segregate and shuffle the clients to separate the benign clients from the persistent bots.

Algorithm 1: *Traffic Average Calculation*

```
Initialize the traffic flow as 0
for each minute
  get the number of received requests
    if first minute
      set as average
    else
      compare with average
       if  received requests ≈ average
         calculate average
         Send client details and request count to
coordination server
        if received requests > ( ≈25) average
         Trigger attack message and send it to
coordination server.
```

B. Segregation Algorithm

This algorithm runs in the coordinator server. When any of the application server sends the attack message, the coordination server starts to execute this algorithm with the previous traffic data's send by the application server. This algorithm executes till all the clients in the server is identified as bots and all the benign clients are re-assigned to new servers.

Algorithm 2: *Segregation Algorithm*

```
Initialize N ← No of Clients
Initialize M,S ← 0; bots and benign clients
While ( N > M) do
 for i ← 0, N do
  Initialize X ← No of Records of client i
  for j ← 1,X do
  Compare request packets with the average packets.
   if request count ≤ average (all)
      Set i as benign client
      Increment S
      call shuffling()
      Remove client details from list
   Else
     Set i as bot client
     Increment M
```

C. Shuffling Algorithm

After completion of each round of segregation, shuffling algorithm starts to re-assign the clients of the affected servers. This algorithm will re-assign the servers to the clients identified as benign clients by the segregation algorithm.

Algorithm 1: *Shuffling Algorithm*

```
Initialize P ← replica servers not under attack
Initialize S ← list of benign clients.
if P > S
  for j ← 1,S do
   for i ← 1, S do
      Assign P[i] ← S[j]
else
 while S ≠ 0
  for i ← 1,P do
   get client count and the max load capacity
   if ( max load capacity – client count ) < benign  clients
     assign server to client S[i]
```

VI. IMPLEMENTATION

In this section, discuss about the implementation of the proposed model and the observations. We implemented our proposed method in Amazon web services EC2[1]. We installed one applications server(AS1) and one coordination server(CS). A simple web server will run in the application server which displays a static web page fetched from the server storage. Launch configuration rule was defined in auto scaling[6] to dynamically launch new instances when the system is under attack to provide uninterrupted service to the benign clients and to re-direct the benign clients from the attacked server. The load balancer was added with the auto scaling group. The load balancer will assign serve to the clients and re-direct the clients to replica servers when a server is in attack.

To stimulate the attack, 50 instances were in the cloud in different location and send the web page request to the application server AS1. In the initial stage, only few clients were programmed to send the requests in slow rate for few days for the application server to study the system. It it for the application server to generate the average packet flow and to save the details of the clients. After few days, remaining instances were programmed to initiate the attack by sending continuous requests to the application server. Once the server was bombarded with the requests, it sends the attack message to the CS and the client details. The CS cloned the server and then consolidated all the saved records send by the application server and started the segregation algorithm to segregate the benign clients from the attacker clients. After each round of segregation, identified benign clients are re-directed to the new replica server. Requests from the new clients will be handled by the load balancer to assign the server. The new requests will be re-directed to the AS1. Existing clients was served by the AS1.

The clients were re-directed using the iptables routing policy. The POSTROUTING and PREROUTING rules were defined in the old servers and SNAT was defined. Hence the new server will send data to the client directly without forwarding the packets to the old application server. Hence the Clients will send the request to the new server IP instead of old server IP.

The Simulation Result for the Shuffling Algorithm is shown in Figure 4. The simulation was carried with different numbers of servers under attack with same number of clients (1000) including benign clients and bots. The ultimate goal is to separate benign clients from bots.

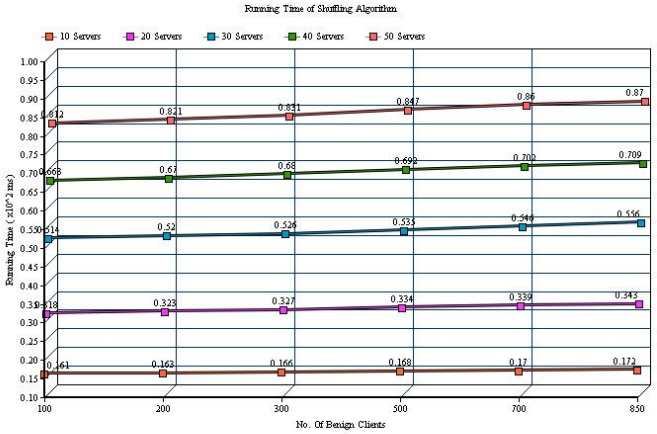

Figure 4. Running Time of Shuffling Algorithm with 1000 clients in each server

The benign clients are identified by the average no of packets send by the client over a period of time matches with the average count of the value calculated by the co-ordination server and the application server. Then it get the details of the servers which are not under attack and checks the no of clients it can hold. Then it will start assigning the clients to the servers. The Algorithm performs a linearity time while separating the benign clients. The simulation was repeated for 25 times and the average time was provided. The algorithm takes few milliseconds to separate benign clients from the bots.

Server re-direction operation is always started by the attacked server. The co-ordination server will send the list of IP's to be re-directed and to which server the request should forward. We use the iptables routing technique to re-direct the clients. The clients were re-directed using the PREROUTING and POSTROUTING rules and SNAT will not be defined so that the new server will send the requested page directly to the client instead of forwarding the page to the old server. Since the client receives the requested page directly from the new application server, the client starts its communication with the new server. The time taken to redirect the clients to new replica server was given in Figure 5.

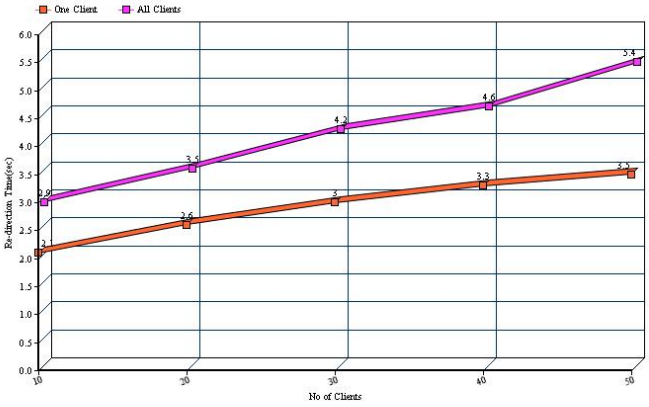

Figure 5. Client Re-direction time between two Application Servers

VII. Discussion

In this section we analyse how our defense mechanism will respond to different attack strategy and discuss about the deployment and the effectiveness of the shuffling algorithm.

Naive bots: The Shuffling algorithm is able to evade naïve bots by dynamically installing the application servers in the cloud environment in different geo-locations for the server bombarded by the DDoS attack. Since the naive bots are unable to trace the new address of the server, it will send packets to the old server.

Persistent bots: For persistent bots, our system will take few rounds of shuffling to identify and block the bot. whenever the client traces and bombard the server with the requests, with the increase of traffic in the new server the server will alert the co-ordination server to segregate the bots from the benign clients. The clients in the new server will undergo segregation and the clients were identified and hence the persistent bots will be segregated. The requests from the IP's identified as bots will not be accepted by any servers in the system. By carefully optimizing the shuffling plan, our mechanism can mitigate the strong DDoS attack in a medium scale Internet service in approximately 40 - 50 shuffles. Our shuffling algorithm will be executed immediately when the attack is noticed in the application servers. Therefore, the system is attack free for a long period of time.

Resiliency against other Attacks: Some persistent bots may stop their attack on the server once they discover the shuffling operation (by identifying the change in the ip address of the servers). They may once again enter into the system as a new client via load balancer. We maintain the record of client re-assignment and the black listed clients identified as bots even after the client leaves the system. The records will be available for a considerable time. The record will be deleted after some time, since many of the clients uses the dynamic IP which changes from time to time. If the bot re-enters the system before the records were deleted, the clients will be re-directed to the same recorded application server. The client will be identified as new client only when the persistent bot enters into the system with the new IP. The system will defeat IP Spoofing attacks. If the bots did not used the original IP address, they will not receive the re-direction packets send by the servers, hence will be left behind in our application servers.

Deployment: Our DDoS defense mechanism using shuffling algorithm is designed to be deployed as an end system in the cloud. As we implemented the system in the cloud, it can be implemented and managed independently by a non-ISP organization.

VIII. Summary

In this paper, we proposed a cloud enabled, shuffling based, moving target mechanism to mitigate DDoS Attack against the web services. When under attack, our approach will dynamically install the servers in the cloud in different geo-locations and intelligently re-maps the client sessions to the new replica servers. This defense strategy not only helps to evade the naive bots but also enables the client to server shuffling to separate the benign clients from the bots. Our simulation results show that the proposed shuffling mechanism can effectively mitigate strong DDoS attacks by saving 80% of the benign clients in few shuffles each takes only few seconds to complete. In conclusion, our defense mechanism is easy to implement and scalable against strong DDoS attacks. Therefore, our approach offers a defense solution to the web applications implemented in the cloud.

References

[1] https://aws.amazon.com/ec2/

[2] D. Anstee and D. Bussiere, "Worldwide infrastructuresecurity report viii," 2012. [Online]. Available: http://www.arbornetworks.com/report.

[3] "Layered intelligent ddos mitigation systems," 2011. [Online]. Available: http://www.arbornetworks.com/ddos/Layered%20Intelligent%20DDoS%20Mitigation%20Systems.pdf

[4] A. Stavrou, A. D. Keromytis, J. Nieh, V. Misra, and D. Rubenstein,"Move: An end-to-end solution to network denial of service," in Proceedings of NDSS. The Internet Society,2005.

[5] Quan Jia , Kun Sun and Angelos Stavrou , "Motag: Moving target defense against internet denial of service attacks," in Proceedings of 22nd International Conference on Computer Communications and Networks(ICCCN). IEEE, 2013.

[6] http://aws.amazon.com/autoscaling/

[7] Can We Beat DDoS Attacks in Clouds?, IEEE Transactions on parallel and distributed systems, Vol. 25, No. 9, September 2014, pg 2245-2254

[8] Quan Jia, Huangxin Wang, Dan Fleck, Fei Li, Angelos Stavrou, Walter Powell "Catch Me if You Can: A Cloud-Enabled DDoS Defense", 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks pg 264-275

[9] Jérôme François, Issam Aib, Member, IEEE, and Raouf Boutaba, Fellow, IEEE "FireCol: A Collaborative Protection Network" IEEE/ACM Transactions on networking Vol. 20, NO. 6, DECEMBER 2012 , pg 1828-1841

[10] RFC 2827 P. Ferguson , D. Senie, Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing

[11] Cheng Jin Haining Wang Kang G. Shin "Hop-Count Filtering: An Effective Defense Against Spoofed Traffic", Available Online at http://www.eecs.umich.edu/techreports/cse/2003/CSE-TR-473-03.pdf