ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse159952020.pdf
https://doi.org/10.30534/ijatcse/2020/159952020

# An Architecture to Mitigate DDoS Attacks in Cloud Web Services

**Narsaiah Putta[1], J Sasi Kiran[2], T Suvarna Kumari[3]**
[1]Asst. Professor,CSE Dept., Vasavi College of Engineering, Hyderabad, INDIA,
p.narsaiah@staff.vce.ac.in
[2]Professor, CSE Dept., Lords Institute of Engineering and Technology, Hyderabad, INDIA,
sasikiranjangala@gmail.com
[3]Asst. Professor, CSE Dept., Chaitanya Bharathi Institute of Technology, Hyderabad, INDIA,
suvarna_t@cbit.ac.in

**ABSTRACT**

Distributed Denial of Service (DDoS) attacks are often overlooked because they represent only a temporary interruption in the normal functioning of a system. With the advent of paradigms like the cloud, the mitigation of this type of threat with the increase of resources for the applications becomes viable, but it causes a problem called economic DDoS. This paper presents an architecture proposal to mitigate DDoS attacks directed at an application hosted in a cloud. Such architecture is based on the instantiation of a replication of the application - simple operation in a cloud - and the redirection of only legitimate requirements for this reply. The proposed architecture does not need to identify attacking customers and, even so, it is able to filter only legitimate traffic without the load and possible errors resulting from the need for identification.

**Key words :** Cloud, DDoS, Security, Response Time.

## 1. INTRODUCTION

Several researches have been developed to address issues of the current Internet, which can spread to the Internet of the Future. Such problems can be broadly categorized in the areas of mobility, quality of service and security, which still move towards acceptable solutions, aggravated by the emergence of new architectures. Today, both data and applications are offered in different and unknown physical locations. Another major change occurred in the way of administering a system, which used to be more local in scope, with its characteristic users and servers, and now these systems are hosted in environments built by the sharing of resources of several autonomous systems (AS) and heterogeneous [1],[18].

Despite many research efforts, Denial of Service (DoS) attacks still pose serious threats to many servers on the Internet and constitute one of the main security challenges currently propagated for Future Internet, which will interconnect many more devices and individuals. A DoS attack is not intended to hack into a computer to obtain confidential information, nor to alter information stored on it. Its objective is the unavailability of a service provided, using the routing of large amounts of traffic to the service host. This issue becomes even more severe when several traffic generators intensify the traffic routing in a distributed manner, featuring a Distributed Denial of Service (DDoS) attack [2]. Although such a load is only a momentary problem, when it comes to applications intended for electronic commerce, a service outage represents major financial losses.

With the new network and application architectures that configure the Internet, complex and robust systems such as clouds have emerged, where the challenge of mitigating DoS attacks becomes even more necessary. Although most solutions commonly offered to mitigate DDoS in the cloud is based on the increased allocation of resources [3], these approaches become inadequate because the premise of the possibility of greater allocation of resources is not always feasible because it is too expensive [4], [5]. This behavior characterizes the economic DDoS (eDDoS) [6],[19].

This work proposes a reactive and fault-tolerant architecture to mitigate DDoS attacks executed against applications hosted in a cloud. Such architecture is based on the instantiation of a replication of the application and the redirection of only legitimate requirements to this reply.

The architecture monitors the traffic of an application and when detecting a possible anomaly, that is, the occurrence of a DDoS attack, it establishes a new instance of this application, ensuring that no malicious traffic can be reached. The differences of this solution for other proposals are that the

hosted application does not need to provide accuracy in filtering legitimate traffic, the use of resources it is not financially burdened, and human intervention is unnecessary. An experimental evaluation considering the response time to customers shows the effectiveness of an architecture implementation in the face of DDoS attacks on a Web service.

## 2. Related Works

Research involving proposals to mitigate DDoS in cloud architectures is still considered incipient and far from convergence. Among the few proposals for these environments, the CluB proactive framework, presented in [7], stands out, which suggests that certain routers be selected, arranged in a distributed way, for the analysis of three affection and consequent prevention of malicious activity. In this framework, every package must be checked to enter, exit or transit the architecture. Each allocated router performs the verification, which is costly due to the overhead caused by the authentication of each packet and the unavoidable need to change the behavior of the routers.

In [8], a proactive scheme that employs Communities of Interest (COIs) is presented to capture data about the collective behavior of remote entities, using them to predict future behavior. Such a scheme assumes that customers who have had legitimate relationships previously have good leads and can be considered legitimate again. However, identifying old customers is not so trivial. In addition to the overhead generated by the scan, IP addresses are usually dynamic and the requirement to login for identification is not possible, given that the DDoS attack may make an identification operation impossible. In [9],[21], attacks are dealt with by creating a new instance of the application. Once a DDoS attack is detected, the proposed mechanism seeks to identify the attackers through PINGs: if a client suspected of being an attacker does not respond to the PING, he is considered as an attacker, not being redirected to the new instance of the application. However, this solution assumes that always and only genuine customers will respond to PINGs, which sometimes does not match reality.

The effectiveness of these mitigation schemes depends directly on the ability to identify or filter legitimate customers. The WebSoS solution [10] offers robust filtering of attacking traffic and blocking unapproved requests, thus forming a secure overlay that mitigates DDoS on web servers. The server uses cryptographic authentication mechanisms and a Turing graphical test [11],[20] to differentiate human clients from attack scripts. These procedures, according to the authors' tests, do not overload the functioning of the service, but require that the routers located in the perimeter of the server be reconfigured, which is not feasible for cloud architectures.

## 3. Architecture for Cloud DDoS Mitigation

This Section describes an architecture to mitigate DDoS attacks on clouds in an autonomous and independent way. The proposed architecture can be used by any web application hosted on a cloud that, when suffering signs of a DDoS attack, filters the legitimate traffic and sends only this to a new instance of the same application.

This architecture, illustrated in Figure 1, is composed of a general module called Traffic Manager (TM), which does not communicate directly with the application. This module has the NAI, BM, TA and TR submodules. In addition, the database instance (DB) is external to the other instances of the cloud, since the database is also in the clouds and can be accessed from any other cloud instance.

The TA sub-module observes the behavior of the incoming traffic for the application in a proactive way. It focuses on estimating the amount of traffic and processing on the server, and takes medication to detect the existence of a possible DDoS attack. If an attack is detected, the NAI submodule is activated. NAI will create a new instance of the application on another server in the cloud, consequently with a different IP address. The TR sub-module handles all incoming traffic, responding with a redirect to the new instance of the application, as seen in Figure 2. The TR starts from the principle that DDoS attackers do not interpret the responses obtained from the server, because if they interpret, their efficiency is reduced. In this way, only legitimate clients (C's in the figure) will, in fact, be redirected to the new application, while non-attackers will not pass through TR.

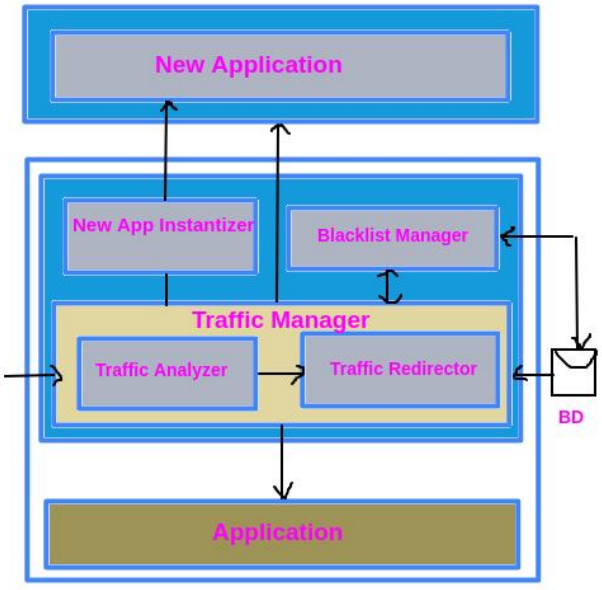


**Figure 1:** DDoS Mitigation Architecture

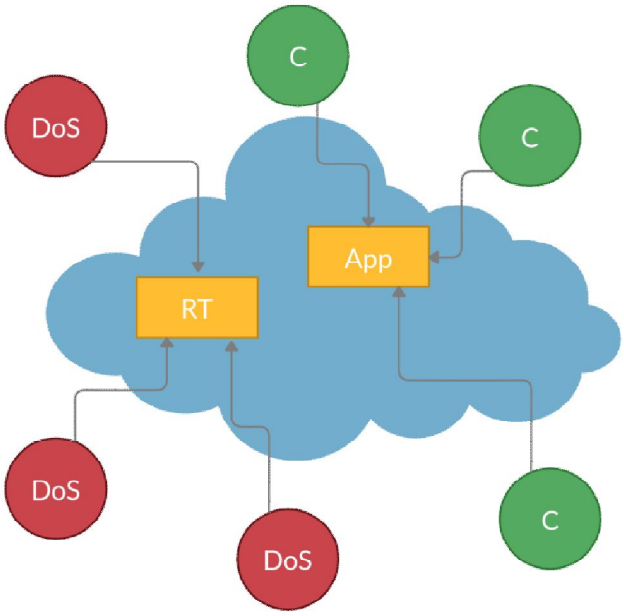


**Figure 2:** Traffic Flow

When trying to redirect customers to the new instance, the customer's address, whether legitimate or not, will be added to a blacklist. Customers on this list have their requirements discarded in order to reduce the cost of processing responses in the server. However, as the legitimate customer is informed of the redirection before his address is blacklisted, he will have access to this new replicated instance and will be able to submit a new request. Records with expiration date are used in this blacklist, since the answers may be lost. The validity time in the list increases exponentially, to further reduce the overhead. BM is responsible for adding and managing the output of customer addresses to the blacklist, as well as the validity of the entry, which increases exponentially.

## 4. IMPLEMENTATION

For the implementation of the architecture, the cloud solution [12] was used. It offers infrastructure as a hosting service, enabling development on Ruby on Rails. The framework architecture [13] is completely based on the Model View Controler (MVC) paradigm, facilitating the organization of our architecture modules. Thus, the structure of the code written in RoR is composed of Model, Vision and Control components. The model components correspond to the data - how they are stored, obtained, correlated. The view part corresponds to the graphic part of the application. Finally, controllers perform data manipulation as a whole, and correspond to the logical and functional part of the code. They also function as a bridge between model and view, so that the data travels in both directions.

Considering RoR, the architecture's traffic analyzer (TA) sub-module corresponds to a controller. Thus, a request to the application will be intercepted by this control component, which will perform the measurement of statistics, and immediately activate the controller that corresponds to the operation of the application itself. It should be noted, however, that the time spent on this controller is minimal. Figure 3 presents a flow chart of the implementation carried out. In other implementations, if time is perceived to affect the functioning of the mitigation mechanism, this processing could still be performed in the background.

When the TA detects the existence of a possible attack, a new cloud instance is created by the NAI submodule and the application is replicated to this instance, stopping the original application, which now responds only as a redirector. The application reinstatement process in the implementation carried out consists of the previous existence of a second application, initially without any resources allocated.

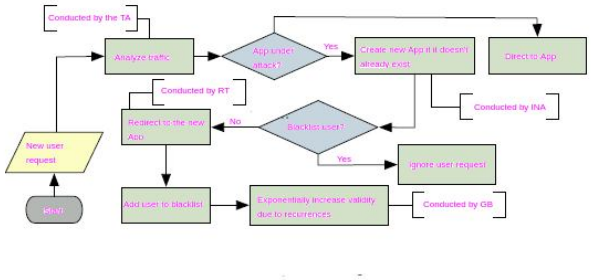


**Figure 3:** Implementation Operations for Mitigation

An interesting feature of the RoR framework is the existence of a route file. The implementation of the traffic redirector (TR) submodule is performed on top of this file, called routes.rb. For displaying any dynamic application page, the route file is inevitably called. In this way, it is used for adding clients to the blacklist and filtering the clients blocked by the blacklist manager (BM). When redirecting traffic to a new instance, an entry will be added, blocking the customer in question for a certain time.

The blacklist itself and the various other control variables are managed by the cloud database [12]. This database is known for its simplicity and efficiency. It basically maps key and data, offering writing and reading times corresponding to hashing. The implementation of the blacklist was done using a client's IP address as the key, and the time this client will remain blocked as data. As it is, indirectly, a hashing mechanism, the search time for a customer will be O (1), which is excellent for a mechanism that will filter all traffic that reaches the application.

## 5. EVALUATION

The evaluation of the proposed architecture consists of an analysis of the server's capacity to meet new requirements, and the service may be only the redirection. If the DDoS attack is properly mitigated, the attackers' requests will be ignored after they are blacklisted. Therefore, the server in the cloud should be able to redirect only legitimate clients to the new instance and guarantee that they will have direct access in the next requisitions.

For experimentation, like our attackers, eight machines from the research laboratory were used to process the attacks, observing a network latency in the range of 3ms to 7ms. Each of these machines operated with 25, 50, 75 or 100 instances of an attacking script, which uses the curl command to bombard the server with HTTP GET requests. Such experiments were carried out several times, obtaining results of similar behavior. As for hosting, a dyno was used for each application. For the Heroku cloud, a dyno is an isolated virtual server instance with 512MBs of RAM and 4 Intel Xeon X5550 @ 2.67GHz colors.

### 5.1 RESULTS

To explore and validate our proposal, experiments were carried out according to the metrics specified in [8]. First, the impact of DDoS mitigation on response time was assessed and then on the response rate and overhead. As seen in Figure 4 and Figure 5, with a 95% confidence interval, the use of the proposed architecture, Figure 4 reduced the response time to legitimate requirements compared to the Figure 5 which shows the time spent to meet the same number of requests originating without our mechanism. Such behavior occurs because with the use of architecture, the blacklist will prevent an application from responding to the same client multiple times, still guaranteeing that the legitimate client is able to reach the new instance.

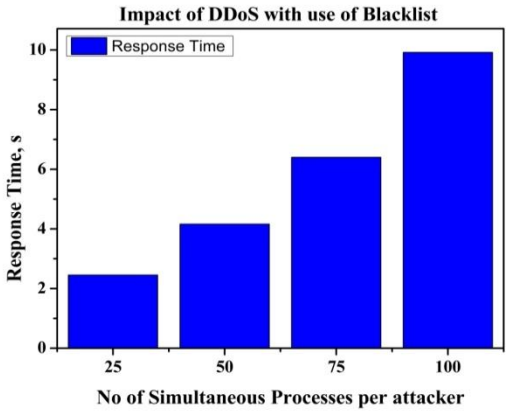


**Figure 4:** Response time for legitimate customers with use of Blacklist

Another metric evaluated is the rate of requested pages received successfully, shown in Figure 6 and Figure 7. There is a drop in the number of responses only when blacklist filtering and the consequent ones were not used redirect. In these cases, the application sends a response to the attacker, who discards this response immediately, continuing the attack. Note that the delivery rate without the use of the blacklist is affected by the occurrence of unresolved HTTP request timeouts, as the application remains busy with attacking requests.

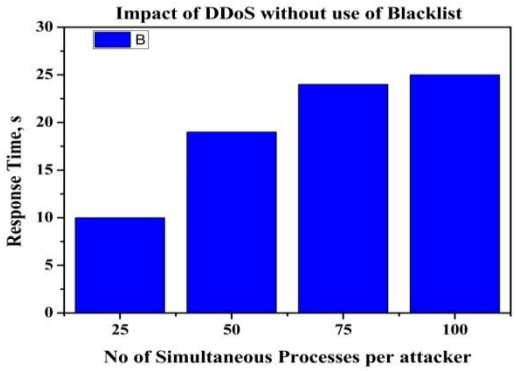


**Figure 5:** Response time for legitimate customers without use of Blaclkist

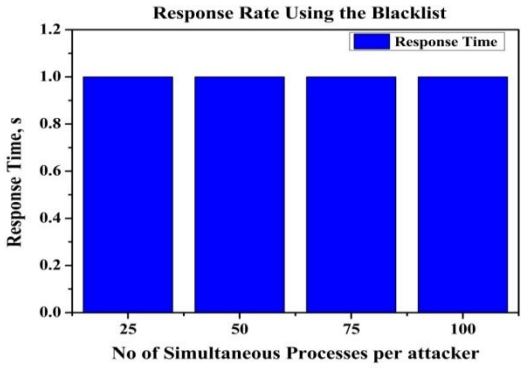


**Figure 6:** Rate of server responses to legitimate clients using Blacklist

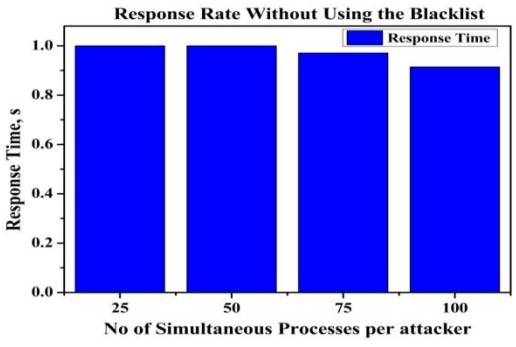


**Figure 7:** Rate of server responses to legitimate clients without using Blacklist

## 6. CONCLUSION

This work presented a mitigation architecture for DDoS attacks directed at web applications hosted in the cloud. The architecture is dependent only on the existence of the environment in the cloud and allows free access to legitimate customers. The use of a blacklist is efficient for filtering due to the validity of the records, which in the case of attackers, will have an exponential increase for recurrences. The results achieved in the experiments demonstrate the validity of the proposed solution, since they manage to direct legitimate traffic in a satisfactory way, making it impossible for attackers to access the newly created instance. More robust mechanisms for checking the blacklist at lower and optimized levels will be developed as future works, complementing the current solution. In addition, experiments on larger scales both in the DDoS attack and in the resources allocated to the application instances will be carried out.

## ACKNOWLEDGEMENT

Authors would like to express gratitude towards Vasavi College of Engineering, Hyderabad, India for their constant supervision as well as for providing necessary information regarding this research and also their support in completing this endeavor.

## REFERENCES

1. Jerome et al.,"**FireCol: A Collaborative Protection Network for the Detection of Flooding DDoS Attacks**", IEEE 2012.
2. Igor Anastasov et al., "**SIEM Implementation for Global and Distributed Environments**", IEEE 2014
3. Tingting et al.,"**A Comprehensive Security Model for Networking Applications**", IEEE 2012
4. Zhong-Hua Pang and Guo-Ping Liu, "**Design and Implementation of Secure Networked Predictive Control Systems Under Deception Attacks**", IEEE 2012
5. Radwane Saad, Farid Naït-Abdesselam and Ahmed Serhrouchni", "**A Collaborative Peer-to-Peer Architecture to Defend Against DDoS Attacks**", IEEE 2012
6. J. Yu et al.,"**Mitigating application layer distributed denial of service attacks via effective trust management**", IEEE 2012
7. N. Shipilov et al.,"**Simulation ofDDoS-attacks and Protection Mechanisms Against Them**", IEEE 2015
8. Ting Wang et al.,"**Improving the Efficiency of Server-centric Data Center Network Architectures**" IEEE 2014
9. S Meliopoulos et al.,"**Cyber Security and Operational Reliability**", IEEE 2015
10. Cornel Barna et al.,"**Model-Based Adaptive DoS Attack Mitigation**", IEEE 2012
11. P Gasti et al.,"**DoS & DDoS in Named Data Networking**", IEEE 2013
12. Service, A.W. 2017. Available online: https://aws.amazon.com/compliance/data-privacy-faq/ (2017).
13. **Hewlett-Packard. Security Overview of the Integrity Virtual Machines Architecture**. http://h20564.www2.hpe.com/hpsc/doc/public/display?docId=emr_na-c02018861&DocLang=en&docLocale=en_US ( August 2017).
14. Masdari eta al., "**A survey and taxonomy of DoS attacks in cloud computing**". In Secure Communication Networks, 2016
15. Osanaiye et al., "**Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework**", In Journal of Network Computer Applications, 2016
16. Somani G et al.," **DDoS attacks in cloud computing: Collateral damage to non-targets**", In Journal of Computer Networks,2016
17. Krebson Security. "**DDoS on Dyn Impacts Twitter, Spotify, Reddit**". https://krebsonsecurity.com/2016/10/ddos-on-dyn-impacts-twitter-spotify-reddit/ (August 2017).
18. Ghourabi, Abdallah, "**Experimental Evaluation of a Hybrid Intrusion Detection System for Cloud Computing**", in International Journal of Advanced Trends in Computer Science and Engineering, VVol.8, No. 5, ISSN 2278-3091, Page 3065 - 3073,2019.
19. Brijesh Pandey, D. K. Yadav et al., "**Application Portability Pertaining TOSCA in Cloud Computing Environment** ", In an International Journal of Advanced Trends in Computer Science and Engineering, Vol.8, No. 5, ISSN 2278-3091, Page 2252 – 2259, 2019
20. Mishra, Pragya, "**Revocable Identity Based Signature Scheme with Outsourced Cloud Revocation Authority**", In an International Journal of Advanced Trends in Computer Science and Engineering. 1537-1544. 10.30534/ijatcse/2019/7684, 2019.
21. Karimunnisa, Syed et al.," **Cloud Computing: Review on Recent Research Progress and Issues**", In an International Journal of Advanced Trends in Computer Science and Engineering. 8. 216-223. 10.30534/ijatcse/2019/1882, 2019.