DOI 10.12694/scpe.v20i4.1569
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 4, pp. 669–685. http://www.scpe.org

THE SLOW HTTP DDOS ATTACKS: DETECTION, MITIGATION AND PREVENTION IN THE CLOUD ENVIRONMENT

A. DHANAPAL* AND P. NITHYANANDAM†

Abstract. Cloud computing is the latest buzzword and cutting-edge technology. The cost-efficiency, easy to operate, on-demand services, availability, makes the cloud so popular. The online web applications based on the internet such as E-Healthcare, E-Commerce are moving to the cloud to reduce the operating investment cost. These applications are vulnerable to slow HTTP Distributed Denial of Service (DDoS) attack in the cloud. This kind of attacks aims to consume the resources of the application as well as the hosting system so that to bring down the services. The various forms of the slow HTTP DDoS are HTTP header attack, HTTP body attack and HTTP read attack. Due to the nature of mimicking the slow network behaviour, this attack is very challenging to detect. This is even more difficult to identify in the cloud environment as it has multiple attack paths. The web applications running in the cloud should have been safeguarded from the slow HTTP DDoS attacks. This paper proposed a novel multi-stage zone-based classification model to identify, mitigate and prevent the slow HTTP DDoS attacks in the cloud environment. The solution is implemented using the OpenStack cloud environment. The open-source slowHTTPTest tool is used to generate different types of slow HTTP DDoS attacks,

Key words: DDoS, Application Layer Attacks, slowloris, RUDY attacks, slow HTTP attacks, OpenStack, Cloud Security, Layer 7 Attacks

AMS subject classifications. 68M14

1. Introduction.

Cloud computing helps the medium and small companies to reduce their initial investments [1] to build their data centres and the infrastructures. National Institute of Standards and Technology (NIST) defines cloud computing [2] is a model for enabling convenient on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications and services that can be rapidly provisioned and released with minimal management effort or service provider interaction. The cloud computing promotes the availability, and it exhibits the following five characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service.

1.1. Cloud computing classifications and threats.

The general broad classifications of cloud computing are:

- Service Delivery based model:
  This is based on the services provided by the cloud. This is further sub-classified as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [3].
- Deployment based model:
  This explains how the cloud is deployed. The finer sub-classification of this category are Private cloud, Public cloud, Hybrid cloud and Community Cloud [4].

Cloud computing is still evolving. The typical issues and threats faced by cloud computing are DDoS, access-control, data integrity, data security [5]. The cloud runs business-critical applications for E-Commerce, E-Healthcare and finance. The availability of these applications is very crucial in their business. The Distributed Denial of Service attack (DDoS) is one of the major threats to cloud applications. The details on the DDoS attacks explained in next sub-section.

1.2. DDoS attacks and types.

The Denial of Services (DoS) attack is a method in which attackers try to make the services or resources unavailable to legitimate users. In Distributed Denial of Service (DDoS) attacks, the intruder launches the attack to victim server or application from compromised machines from multiple locations. The attacker seed the malware into several computers over the internet with the help of the control machine. The compromised machines are acting as bot or army for the DDoS attack. Whenever the attacker decided to attack the victim server or application, he sends an attack command to the bot machines so that the bot machines start attacking the victim. The fundamental idea of the DDoS attack is shown in figure 1.1.

*Research Scholar, School of Computing Science and Engineering, Vellore Institute of Technology, Chennai, India. (Dhanapal.a2013@vit.ac.in, dhanapal.ang@gmail.com).
†Professor, School of Computing Science and Engineering, Vellore Institute of Technology, Chennai, India.

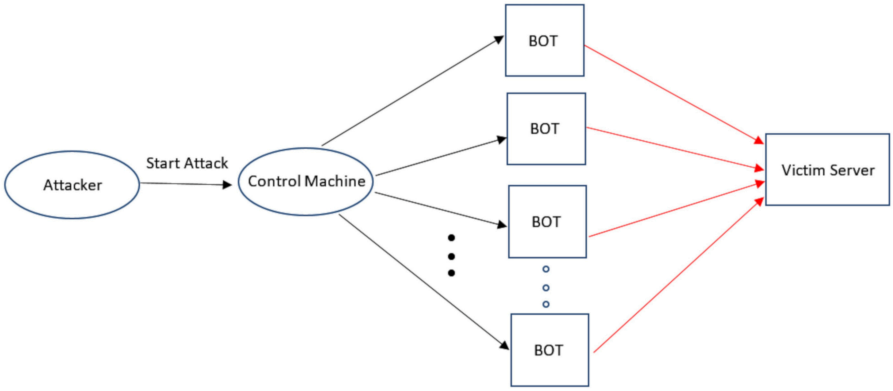

FIG. 1.1. *The basic model of DDoS attack*

The DDoS attacks are categories are [5, 12] :

- Volumetric Attacks :
  The network bandwidth is targeted in this type of attacks.
  Example: UDP flooding attack, ICMP flooding attack.
- Protocol Attacks :
  These attacks aim to exhaust the server resources.
  Example: Ping of death.
- Application Layer Attacks :
  These attacks focused to disturb the availability of the applications.
  Example: slow HTTP DDoS attack.

The DDoS attacks may be carried out due to various reasons like proving the ability, attacking the competition, etc. [6]. This work addressed one of the application layer attacks known as slow HTTP DDoS attacks in the cloud.

2. The HTTP Protocol overview and the slow HTTP DDoS attacks. This section explains the basic design of the HTTP protocol overview, the slow HTTP DDoS attacks and its various types.

2.1. The HTTP Protocol Overview. The Hypertext Transfer Protocol (HTTP) is the very well-known application layer protocol being used over the Internet to access web-based applications or services. The HTTP protocol runs on top of the TCP/IP suite of protocols. The World Wide Web (WWW) uses this HTTP protocol to access any web pages on the Internet. Whenever we type any Uniform Resource Locator (URL) on the browser to access the web pages, TCP 3-way handshaking has been done between web client and web server to establish the valid TCP connection, after successful connection establishment, HTTP protocol sends out one or more HTTP request to the corresponding web server and one or more HTTP responses received from the web server. The HTTP Protocol modelling has been captured in figure 2.1.

2.2. Slow HTTP DDoS attacks. Sslow HTTP DDoS attack is an application layer attack, and it incorporates multiple HTTP incomplete requests to the server. This type of attacks tries to exploit the flaws in the design or implementation of the applications. It is very challenging to detect compared to other DDoS attacks due to the following reasons [7, 8]:

- The attack originates from the real host in a genuine way;
- This type of attack consumes a very low bandwidth as a mimicking low latency network;
- This attack focuses on exhausting the resources of the application in a legitimate manner.

2.3. Slow HTTP DDoS attack types. Slow HTTP DDoS attack further divided into three types:

- Slow HTTP Headers attacks (or) Slowloris attacks [9];
- Slow HTTP Body attacks (or) R-U-Dead-Yet attacsk [10];
- Slow HTTP Read attacks [11].

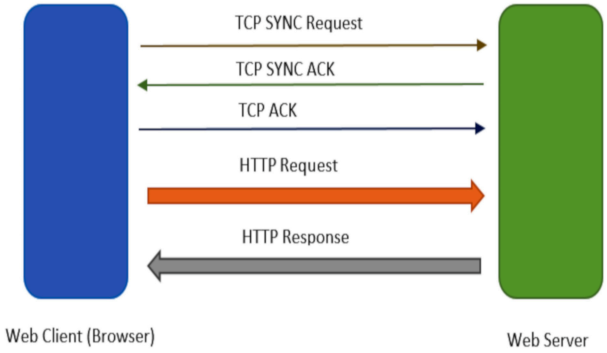

FIG. 2.1. *The HTTP Protocol model*

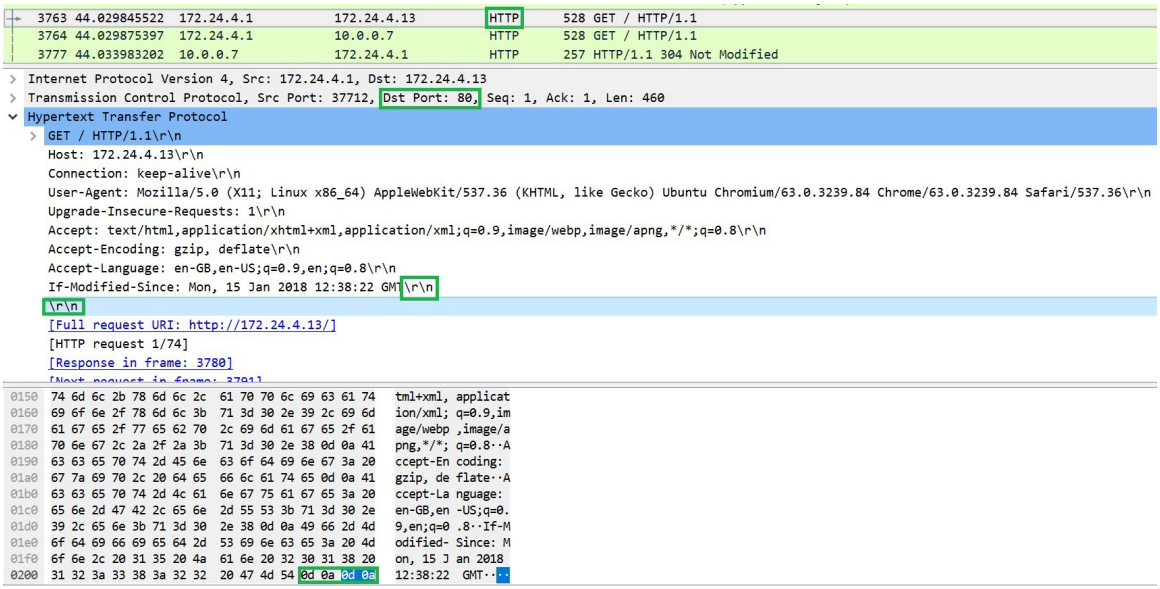

FIG. 2.2. *Normal HTTP request header*

2.3.1. Slow HTTP Headers attacks or Slowloris attacks. In general, the during HTTP GET, the web browser sends HTTP GET requests along with HTTP header. The HTTP protocol designed in such a way that the web server waits until to receive the complete HTTP header for processing the GET requests. Whenever the web server receives an incomplete header, it believes that the requests are being sent from the slow network and keep waiting for the complete header. The slow HTTP header attack exploits this behaviour and sends requests with incomplete HTTP headers. Since the attacker never sends complete header information, the server keeps reserving the allocated resource. This result in the starvation of resources and failure to serve any legitimate requests.

The conventional HTTP GET request header looks as shown in figure 2.2. The header always ends with "CR LF CR LF" (0d 0a 0d 0a). During the slow HTTP header attack scenario, the attacker sends only "CR LF" (0d 0a) to indicate the incomplete header. The same has been depicted in figure 2.3.

2.3.2. Slow HTTP Body attacks or R-U-Dead-Yet attacks. This attack comprises HTTP POST requests with the complete header, but the length field contains a large value so that the server reserves all the allocated resources until to receive the entire message.

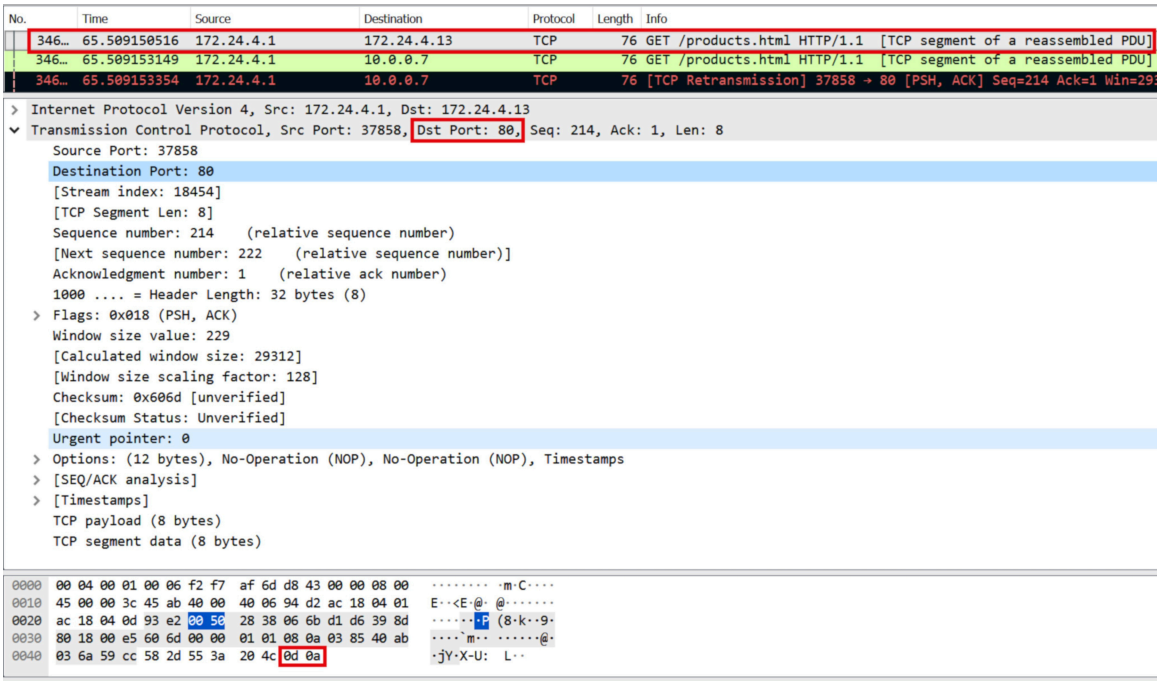

FIG. 2.3. *Malicious HTTP header during the attack scenario*

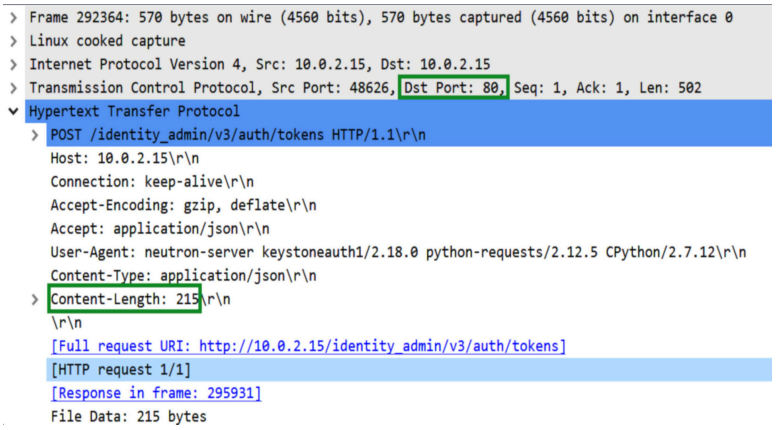

FIG. 2.4. *Normal HTTP Post request*

During this attack, the attacker sends the message with content size as small chunks like one or few bytes in very slow intervals. The attacker also opens multiple such connections to the web server. This way attacker consumes the resources and makes it unavailable to the real users. The regular HTTP POST request depicted in figure 2.4.

The HTTP body attack request captured in figure 2.5, where the length value is notably high.

2.3.3. Slow HTTP Read attacks. Another method of slow HTTP attack where the attackers send the HTTP GET requests with the proper header as well as body to the web server. The attack trick is to read the response in a very slow fashion.

Any HTTP requests start with creating the TCP session between the browser and the server. The flow of data between the browser and the web server is controlled by TCP window size value. This value indicates that the number of bytes that the browser can receive on the TCP session. The attacker frequently informs to the

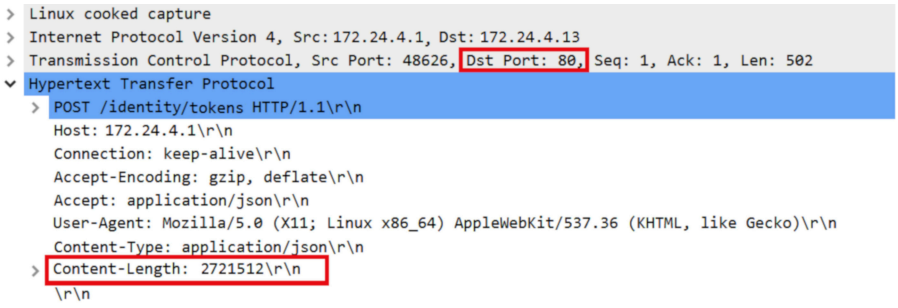

FIG. 2.5. *Slow HTTP Body attacks POST request*

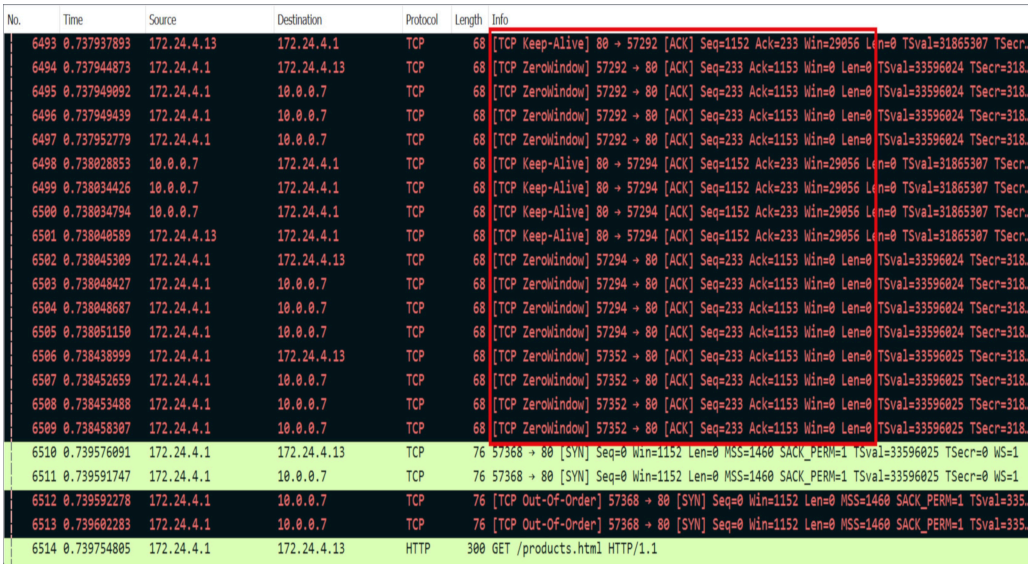

FIG. 2.6. *Slow HTTP Read Attacks*

web server that it can only receive few bytes, or it is not ready to receive any bytes. The attacker opens many such sessions with the server to consume all the resource so that the server will not be able to process any real requests from the legitimate user. The slow HTTP Read attack depicted in figure 2.6.

The rest of the paper organised as sec. 3 covers the literature review of the related works. The proposed architectural model explained in sec. 4. Section 5 captures cloud environmental details. This work is improved version of our earlier work done to detect slow HTTP DDoS attack detection in the cloud environment in [22].

The experimentations, results and discussions are part of sec. 6. The conclusion and future enhancements discussed in sec. 7.

3. Literature Review of the related works. Though the authors reviewed many works related to HTTP DDoS attacks, restricting to add only the most relevant work in this section.

Enrico Cambiaso et al. [13] reviewed the slow DDoS attacks in detail. The authors defined each of the attacks and categorized into different classes. This paper provides an insight into the various slow DDoS attacks.

Thomas Lukaseder et al. [14] used the SDN controller to realise the resolution against the slow HTTP attacks. The solution implemented in three phases. The first phase is the detection, the second phase is the identification, and the final one is the mitigation phase. During the detection phase, the application tries to reach the webserver. The moment the server is not reachable, then it assumes that the server is down and starts the identification phase. In the identification phase, they calculate the low packet rate, packet distance uniformity, etc., to find out the attacks and in the final phase, block those attacks using the SDN controller.

Aanshi Bhardwaj et al. [15] analyzed and evaluated the DDoS attack using the OpenStack cloud platform and captures the attack Wireshark graph. This article studied only the DDoS attack using various tools. This paper has gaps in apprehending their OpenStack details like instances, network topology and providing the solution for detection or mitigation in the cloud environment.

Tetsuya Hirakawa et al. [16] proposed a generic defence mechanism against the slow HTTP DDoS attacks. The attacks identified using the number of connections and their life span to the server. The solution needs to be extended for the cloud environment to detect and mitigate the same.

Kiwon Hong et al. [17] defined the slow HTTP DDoS attack solution using the SDN approach. This method implements the slow HTTP DDoS defence module in the SDN controller. The flow through the switches from the browser to the server and from the server to clients are input to the defence application. This assumes three types of a request such as attacks, slow browser and real clients. The attack flows are identified and blocked. This is a generic solution and has to be enhanced for the cloud environment.

Shunsuke Tayama et al. [18] investigated the slow Read DoS attack to the server, and it considers the bandwidth rate as the essential parameter to identify the attacks. The paper concluded that the bandwidth above 500Kbps and having the connection rate equal to the server processing capability is sufficient to conduct the more efficient slow read attacks.

Suroto [19] discussed the different slow HTTP DDoS attacks. This work explored the various configuration parameters fine-tuning for different web servers like Apache, NGINX to mitigate such slow HTTP attacks.

Junhan Park et al. [20] evaluated the effectiveness of the slow Read in a virtual environment. The conclusion of adding the security measures to the webserver is capable to defend the attacks from a single machine (DoS), but not sufficient for the distributed DoS attacks.

A. Dhanapal et al. [21] proposed an OpenStack cloud testbed model for evaluating the DDoS attacks in the cloud environment. This work explored various attack possibilities for the application running in the cloud environment.

The proposed work is enhanced over our previous works in [21] and [22] to detect, mitigate and prevent slow HTTP DDoS attacks in the cloud environment. The novelty contribution of this works to define the multi-stage zonal classification model integrated into the cloud to detect and prevent the slow HTTP DDoS attack. The multi-stage zonal classification architecture discussed in detail in the upcoming section.

4. The Proposed Architecture Model. The slow HTTP DDoS attacks can happen in multiple ways to the web server in the cloud environment. The attacks in the cloud classified as:

- Slow HTTP attacks within the cloud environment
- Slow HTTP attacks from the outside/internet

4.1. Slow HTTP attacks within the cloud environment. The internal HTTP DDoS attacks originate from the instances running within the cloud environment itself. The target victim web server runs within the same cloud environment. The attacks within the cloud are architecturally represented as in figure 4.1.

4.2. Slow HTTP attacks from outside of the cloud or internet. This type of attack is launched from the internet and targeting to the webserver running in the cloud. The architecture model of the attacks from the external environment represented in figure 4.2.

4.3. Proposed Solution. The offered solution addresses both the internal and external slow HTTP DDoS attacks in the cloud environment. This solution is integrated into cloud infrastructure to safeguard webserver from all possible attack situations. The request from any client to the webserver is categorized into any one of the zone states at any given point in time using the multi-stage zonal model. The multi-stage zonal classification architecture model is shown in figure 4.3.

Initially, all the incoming connection requests monitored for their activity in the monitoring state zone. The legitimate behaviour clients moved into the green state zone. In the green state zone, all the client requests forwarded to the webserver.

In case, the client activities are suspicious like opening higher number of connections and sending frequent GET requests with the incomplete header or POST requests with a smaller number of bytes with longer interval duration to its turn-around time are moved to orange state zone and continue to be monitored.

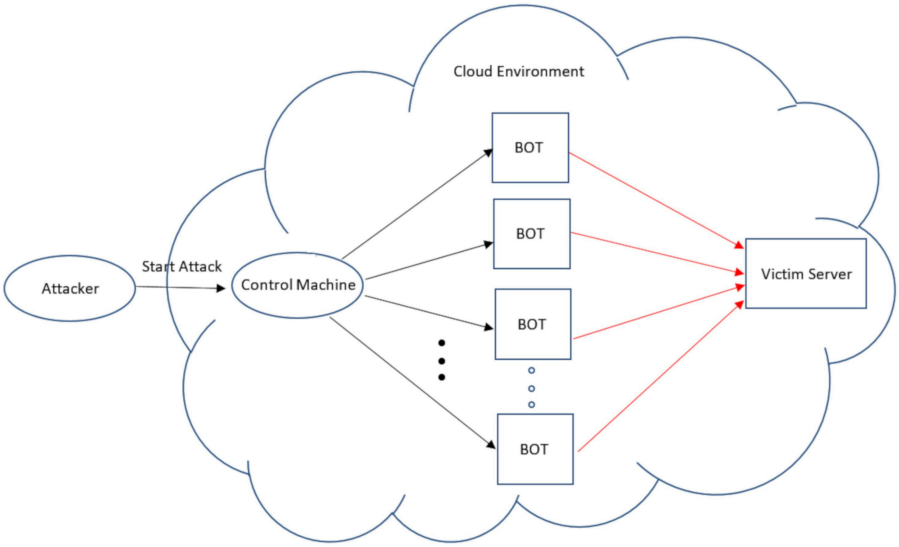

FIG. 4.1. *The DDoS attacks architecture within the cloud environment*

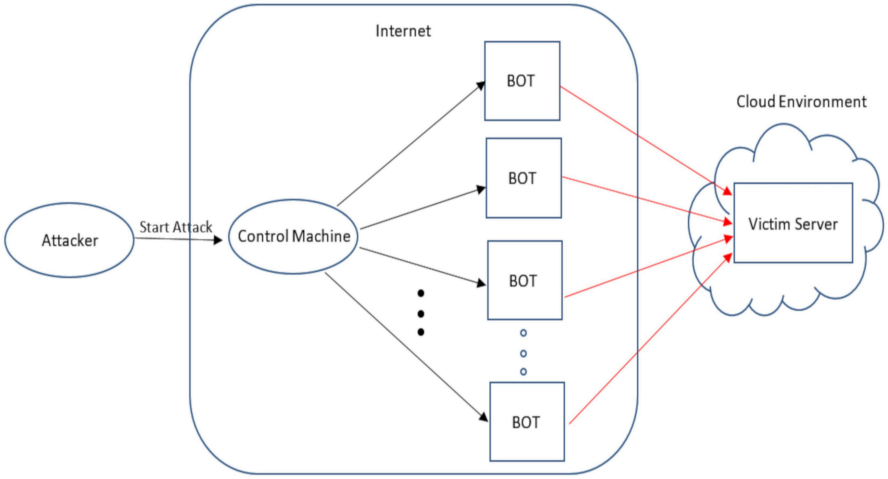

FIG. 4.2. *Architecture Model of external DDoS attacks to the cloud environment*

The clients in the monitor state zone or green state zone or orange state zone activity looks abnormal such as opening very high number of connection requests than the allowed maximum number of connections and every request received with partial header or post request with few bytes or request to server with very high interval like more than 80% of the defined keep alive interval when compared to its turn-around time are treated as DDoS attacks and those clients are put into the red state zone. Any request coming from the clients in the red state zone is blocked.

The clients in the green state zone or orange state zone or red state zone with no activity or no active connection request are moved back to the monitor state zone.

The green state zone client exhibits the suspicious activities persistently is moved into the orange state zone, and similarly, the client shows normal activities after entering orange state zone consistently for every request is placed into the green state zone.

The multi-state zonal model has a provision to limit the allowed number of connections per client and the number of ping requests to calculate the average real network latency value. The configuration file Zone.conf

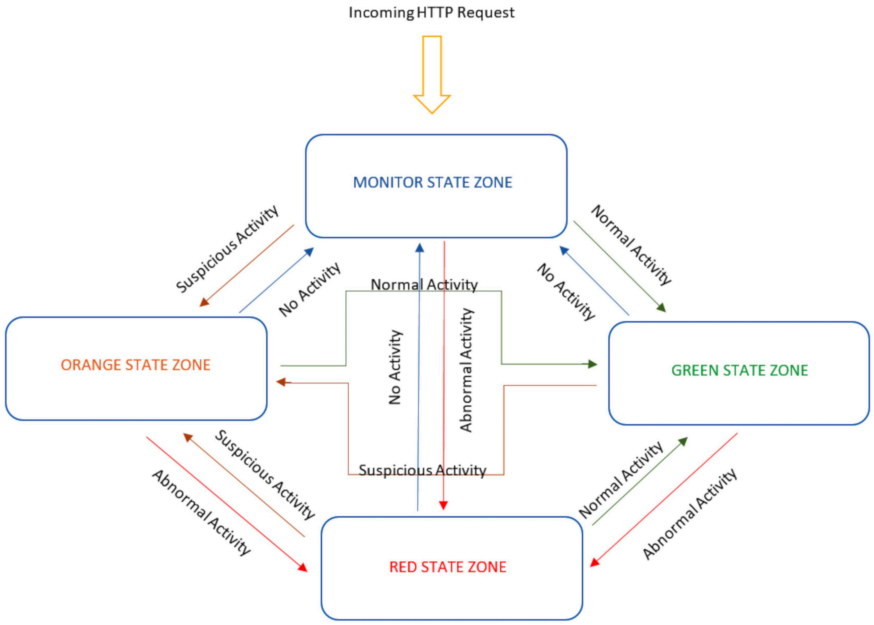

FIG. 4.3. *The multi-stage zonal classification architecture*

FIG. 4.4. *Configuration file for the multi-stage zonal classifier model*

used for the same. The default value of the web server configuration utilised if the configuration value or file is not present. The Zone.conf looks like as shown in figure 4.4.

The zonal model uses the maximum number of connections allowed per client, the average real network latency (ARNL) of the client reachability are parameters to decide the slow HTTP DDoS attack to the webserver.

The clients try to open more than 70% of the allowed connections are closely monitored. The proposed model sends 5 ping requests to the client. The response to the ping used to calculate the average network latency.

The ARNL value is compared with the clients average interval request time. The attackers mimicking the slow network and the client with the real slow network are identified correctly using this technique. The clients with slow requests exceed ten times of its ARNL is considered as an attack and moved into the red state zone. The value ten is chosen by considering the application processing time.

The clients with 61 to 70% of the allowed connection requests and the average request time delay between 6 to 9 times of its ARNL moved into the orange state zone. The clients in this state remain monitored and classified accordingly based on analyzing behaviour continuously.

The clients with the connection rate between 40 to 60% and the average delay between requests are between 4 to 6 times of the average network latency are kept in the green state zone.

A newly incoming client is kept into the monitor state zone to identify the activity pattern and classified into any one of the states based on its behaviour defined above.

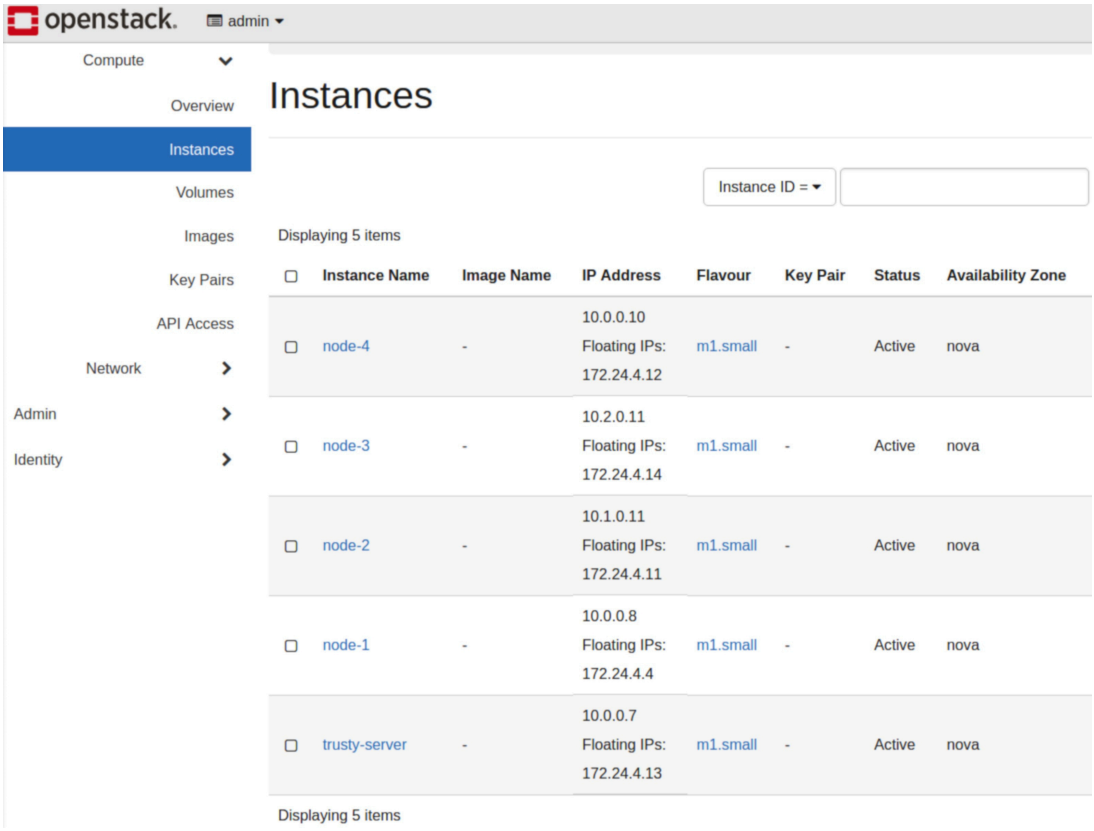

FIG. 4.5. *The OpenStack instance details*

The abnormal activity is defined as the client trying to open high number of connection than the maximum allowed number of connections and every request received with partial header or post request with few bytes or request to server with very high interval like more than 80% of the defined keep alive interval when compared to its real turn-around time.

The suspicious activity is defined as the clients trying to open 61 to 70% of the allowed connection requests and keep-alive as close to 71 to 80% of the predefined value or with the average request time delay between 6 to 9 times of its real the average network latency.

4.4. The Cloud Environmental Details . The proposed model implemented using the OpenStack cloud environment. The OpenStack is open- source software available freely over the internet and providing Infrastructure as a Service cloud service model. The cloud environment runs multiple virtual instances for different customers to implement multitenancy of the cloud. The experiment environment has four networks namely private, green-private, orange-private and public network. Each of the networks represents different customers. The OpenStack instance details depicted in figure 4.5.

The OpenStack network topology in the experiment showed in figure 4.6. Every OpenStack instance associated with a local/internal IP address for communication within the cloud environment and one public/floating IP address for connecting with the external world via the internet. The internal IP address is in the range of 10.xx.xx.xx network and public network IP starts with 172.24.xx.xx.

The customer in the green-private network runs the NGINX webserver in the instance trusty-server. The internal IP address of the webserver is 10.0.0.7, and the public IP address is 172.24.4.13.

4.5. The Experiments, Results and Discussion . The slowHTTPtest tool used for generating different type attacks like slow HTTP header, slow HTTP body and slow HTTP read DDoS attack.

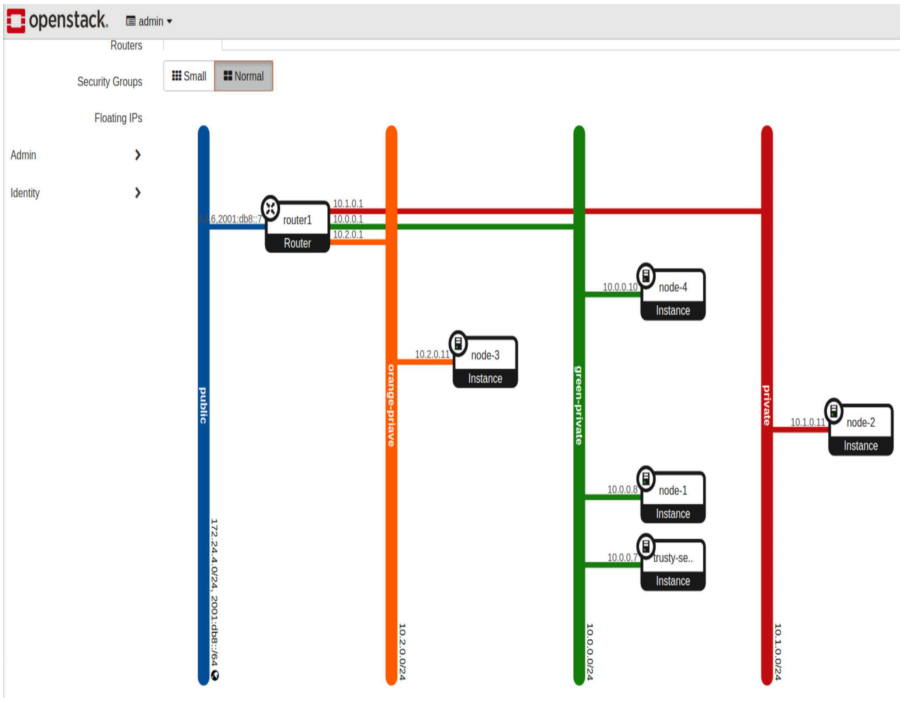

FIG. 4.6. *The OpenStack network topology*

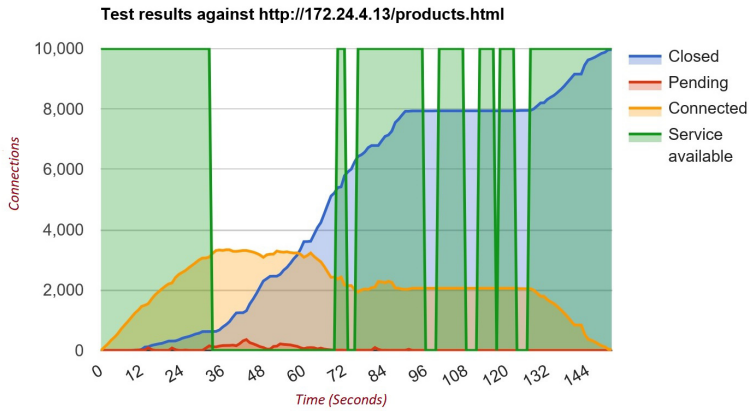

FIG. 4.7. *The result of slow HTTP header attack with NGINX default configuration*

Initially, the authors launched the slow HTTP attacks to the webserver without any changes in the webserver configuration. This experiment did not have the slow DDoS protection layer implementation as well.

The study of the results are as follows:

> The slow HTTP header attack launched against the webserver with
> URL: http://172.24.4.13/products.html.

The parameters used for these attacks and results captured in figure 4.7 and different values used for this testing are:

- The total number of connections: 10000;
- The interval between follow-up data : 10 seconds;
- The number of connection requests: 500 request/seconds.

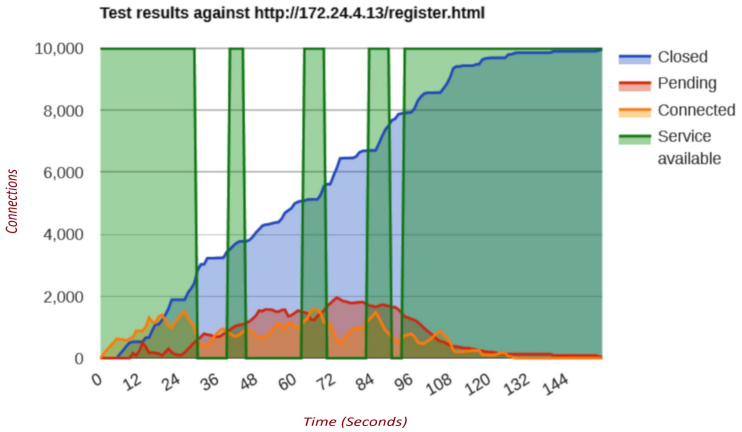

FIG. 4.8. *The result of slow HTTP body attack with NGINX default configuration*

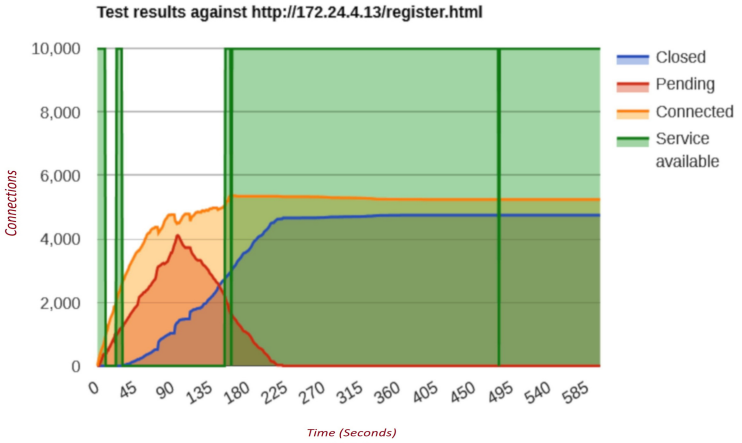

FIG. 4.9. *The result of slow HTTP read attack with NGINX default configuration*

The results of this experiment show that a slow header attack with the total number of connections above 3000 can bring down the service completely. The webserver is will not be available to process any of the legitimate requests.

The number of connections closed during this point of time is 500. The pending requests are 150. Once the service becomes unavailable, the connection requests are dropped by the server. This can be seen with the line indicating connected in figure 4.7.

The slow HTTP body attack test parameter values used are same as the slow HTTP Header attacks.

The slow HTTP body attack carried on the webserver against
URL: http://172.24.4.13/register.html.

The result along with parameters are showed in figure 4.8.

From figure 4.8, the service unavailability is frequently seen at multiple intervals of time.

The connection closed is increased in a continuous manner as shown by closed line in figure 4.8. The connected request is toggling and this is seen from connected line of figure 4.8.

The slow HTTP read attack launched against
URL: http://172.24.4.13/register.html.

The test results depicted in figure 4.9.

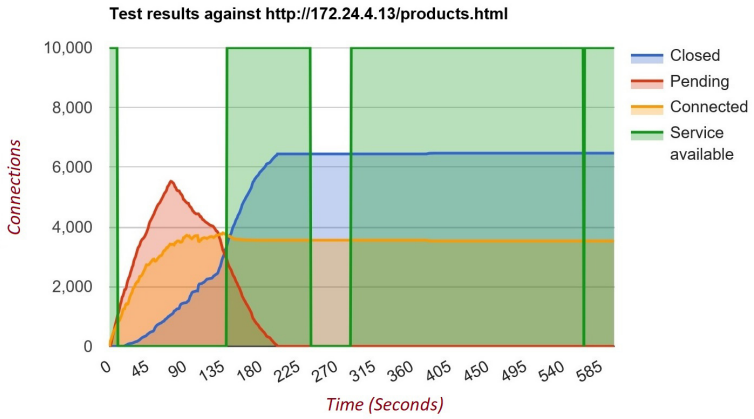

FIG. 4.10. *The slow HTTP header attack results with server fine-tuned configuration*

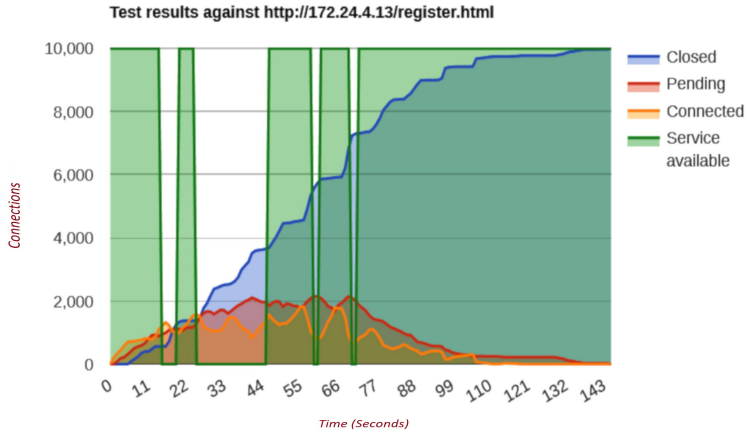

FIG. 4.11. *The slow HTTP body attack results with server fine-tuned configuration*

The testing parameters used are 10000 number of connections made with the rate of 500 connection request per second. The rate at which request processed are 5 bytes per second. From figure 4.9, the server can withstand till around 2000 connections, beyond which failed to serve any requests. The number of pending requests are increased very sharply as depicted in pending line of figure 4.9.

There is a solution discussed as part of [19] to tighten the web server configuration to mitigate the slow HTTP DDoS attacks. The suggested configuration has been done to the webserver to understand the results.

The same above-mentioned experiments are carried out with recommended web server configurations. The result shows that the configuration helps some extent to mitigate the attack and still there is an excellent scope for improvements.

The service failures detected when the connection number reached 1000 for the slow HTTP header attack shown in figure 4.10.

The slow HTTP body attack also shows the server failure in two intervals of the period when the connection reached above 1500. Figure 4.11 captures the observation of this attack. The different values used for this testing are:

- The total number of connections: 10000
- The interval between follow-up data : 10 seconds
- The number of connection requests: 500 request/seconds

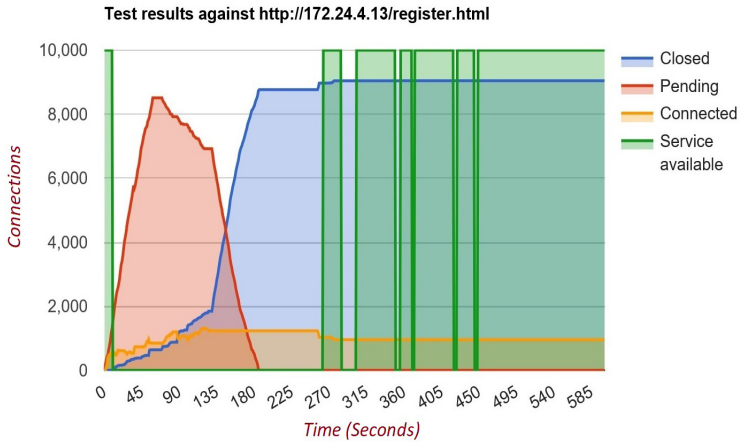

FIG. 4.12. *The slow HTTP read attack results with server fine-tuned configuration*

The slow HTTP read attack with the updated configuration depicted in figure 4.12. The web server brought down and the service unavailable for multiple intervals. The failure started when the connection reached 500.

The receive window size used in the range of 1 to 512 bytes and the read rate buffer size is 5 bytes per second.

All the above experiments conducted it is evident that once the service is back an available as shown by service available line, it is still seen by the attacker for further attacks. The attacker is not identified and blocked in by the cloud. This makes the server vulnerable once it is restored by the administrator.

The same experiment has been done with the implementation of the proposed solution. The results exhibit that with the implemented solution, the webserver can identify the slow HTTP DDoS attacks effectively and prevent such attacks. Whenever the suspicious clients recognised, the multi-stage zone model classifies them to the orange state zone. Upon further observation and understanding of the clients continuous malicious activity, classifying them as attacks and those clients moved to the red state zone. Any further request from the client in the red state zone is blocked in the system. Since none of the newer requests serviced, the attacker thinks that the webserver is down, and attack is successful. In reality, those attackers are classified by the proposed solution and blocked to reach the webserver to make sure that the server is available to the legitimate requests.

The experiment result of the various attacks with the proposed solution is displayed below:

The slow HTTP header attack against the webserver with
URL: http://172.24.4.13/products.html with the implemented solution as seen in figure 4.13.

The total number of connections request used in this experiment is 10000. The follow-up interval of data sent with 10 second interval and the number of requests per second is 500.

The service is seen as unavailable by the attacker when the connection reached 400. At this point of time the attacker is identified and blocked by the implemented solution. Since the attacker is blocked by the solution and none of the requests are further serviced. This makes the attacker to think that server is down. The same has been understood from service available line is plotted against to zero figure 4.13.

The slow HTTP body attack has been launched against the web page http://172.24.4.13/register.html. The total number of connection request made are 10000 and the number of connections per second is 500. The follow-up data send with the interval of 10 seconds to make the connections alive.

The result is shown in figure 4.14. The implemented protection layer is detected the body attack at the early stage when the number of connections reached 1000 by the attacker. The attacker has been successfully blocked by the system. This results in server unavailable to the attacker and continued to unavailable as the service available line is plotted as zero in figure 4.14. But, the attack is identified, and attacker is blocked by the system.

The slow HTTP read attack is launched for the webpage http://172.24.4.13/register.html. The number of

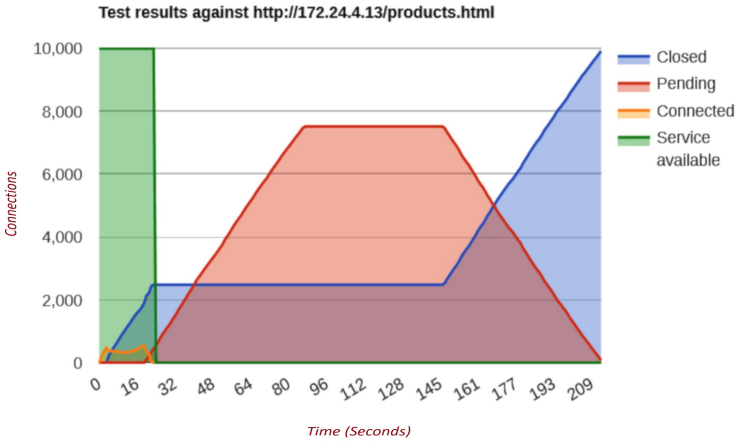

FIG. 4.13. *The slow HTTP header attack results with the proposed solution*

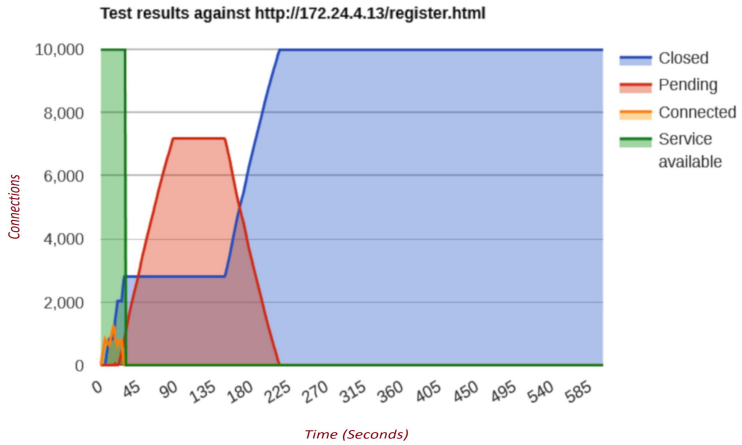

FIG. 4.14. *The slow HTTP body attack results with the proposed solution*

bytes processed per second is 5 bytes in the read window. Thus, makes the read attack to the webserver to consume the resources.

The service is unavailable as soon as system detects the attack during when the number of connections request reached 1200 as shown in figure 4.15. Upon detected, the attacker is moved into red zone and further request are not serviced. This is the reason for service available line is plotted against zero by the attacker. The webserver is available and continue to service the legitimate requests.

The several experiments carried with default web server configurations and the suggested fine-tuned configurations are not adequate to identify and mitigate the slow HTTP DDoS attacks. The proposed solution can identify and prevent those attacks efficiently.

4.6. Comparison of related works with the proposed work. Table 4.1 compares the various work done so far in the space of slow HTTP DDoS attacks detection, mitigation and prevention. This research work covers all the aspects and the gaps identified with the existing works.

5. Conclusion and Future Enhancements. In this paper, we have discussed slow HTTP DDoS attacks and several forms of such attacks in details. The slow HTTP attack scenarios in the cloud are discussed. The solution is proposed to detect and prevent such attacks in the cloud environment. The offered solution is implemented using the open-source OpenStack Infrastructure as a Service model. Each type of the slow HTTP

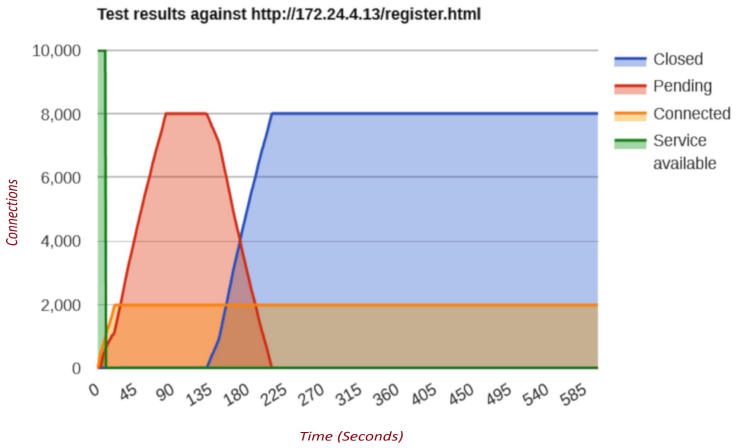

FIG. 4.15. *The slow HTTP read attack results with the proposed solution*

attacks carried out using slowHTTPTest tool against NGINX webserver running in the OpenStack cloud. The experiments with the default NGINX settings and the fine-tuned configurations are done to understand the results. The observation is that the web server fine-tuning is not enough to detect and mitigate attacks. The experiments performed with the recommended solution exhibits excellent results as it identifies the slow HTTP attack and prevents effectively. The corresponding experiments details and results are discussed.

The future enhancement is to generate slow HTTP attacks using various available tools in the cloud environment to study the result of the offered solution and improve the rate of success.

REFERENCES

[1] SALESFORCE, *https://www.salesforce.com/uk/blog/2015/11/why-move-to-the-cloud-10-benefits-of-cloud-computing.html*, 09-Jan-2019.

[2] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST), *https://www.nist.gov/programs-projects/nist-cloud-computing-program-nccp*, 09-Jan-2019.

[3] WHATISCLOUD.COM, *http://whatiscloud.com/cloud_delivery_models/index*, 09-Jan-2019.

[4] WHATISCLOUD.COM, *http://whatiscloud.com/cloud_deployment_models/index*, 09-Jan-2019.

[5] IMPERVA INC., *https://www.incapsula.com/blog/top-10-cloud-security-concerns.html*, 09-Jan-2019.

[6] SIMPLICABLE, *http://arch.simplicable.com/arch/new/the-5-motives-for-DDoS-attack*, 09-Jan-2019.

[7] ACUNETIX, *https://www.acunetix.com/blog/articles/need-pay-attention-slow-http-attack/*, 09-Jan-2019.

[8] QUALYS,*https://blog.qualys.com/securitylabs/2011/07/07/identifying-slow-http-attack-vulnerabilities-on-web-applications*, 09-Jan-2019.

[9] CLOUDFARE, LU-*https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/*, 09-Jan-2019.

[10] CLOUDFARE, *https://www.cloudflare.com/learning/ddos/ddos-low-and-slow-attack/* , 09-Jan-2019.

[11] CLOUDFARE, *https://www.cloudflare.com/learning/ddos/ddos-attack-tools/r-u-dead-yet-rudy/*, 09-Jan-2019.

[12] RADWARE, *https://security.radware.com/ddos-knowledge-center/ddospedia/dos-attack/*, 09-Jan-2019.

[13] ENRICO CAMBIASO, GIANLUCA PAPALEO, GIOVANNI CHIOLA, AND MAURIZIO AIELLO, *PSlow DoS attacks: definition and categorisation*, Int. J. Trust Management in Computing and Communications, Vol. 1, Nos. 3/4, pp. 300–319, (2011).

[14] THOMAS LUKASEDER, LISA MAILE, BENJAMIN ERB, AND FRANK KARGL, *SDN-Assisted Network- Based Mitigation of Slow DDoS Attacks*, SecureComm18, arXiv:1804.06750 [cs.CR], (2018).

[15] AANSHI BHARDWAJ, ATUL SHARMA, VEENU MANGAT, KRISHAN KUMAR AND RENU VIG , *Experimental Analysis of DDoS Attacks on OpenStack Cloud Platform* , Proceedings of 2nd International Conference on Communication, Computing and Networking, Lecture Notes in Networks and Systems 46, (2019).

[16] TETSUYA HIRAKAWA, KANAYO OGURA, BHED BAHADUR BISTA, TOYOO TAKATA, *A Defense Method against Distributed Slow HTTP DoS Attack*, 2016 19th International Conference on Network-Based Information Systems, DOI: 10.1109/NBiS.2016.58, (2016).

[17] KIWON HONG, YOUNJUN KIM, HYUNGOO CHOI, AND JINWOO PARK , *SDN-Assisted Slow HTTP DDoS Attack Defense Method* , IEEE Communications Letters, Vol. 22, Iss. 4 , DOI: 10.1109/LCOMM.2017.2766636, pp. 688–691, (2018).

[18] SHUNSUKE TAYAMA, AND HIDEMA TANAKA , *Analysis of Slow Read DoS Attack and Communication Environment*, Mobile and Wireless Technologies 2017, Lecture Notes in Electrical Engineering 425, DOI 10.1007/978-981-10-5281-1_38, pp. 1718–1724, (2018)

TABLE 4.1
Comparison of related works with proposed work

Related Works	**Cloud Characteristics Considered**	**Cloud Based Solution**	**Given Solution**	**Gaps Identified**
Enrico Cambiaso et al.	No	No	Definition and Categorization of the slow DDoS Attacks.	This paper gives insight into slow DDoS attacks, but the solution is not defined.
Thomas Lukaseder et al.	No	No	The SDN Based solution.	This is specific to SDN based environment and has to be enhanced for the cloud
Aanshi Bhardwaj et al.	Yes	Yes	This is OpenStack Based slow DDoS attack study using various tools.	The OpenStack cloud instance, network details are not discussed. The detection, mitigation and prevention are not covered.
Tetsuya Hirakawa et al.	No	No	The generic solution defined for slow HTTP DDoS attacks.	The solution needs to be enhanced for cloud environment.
Kiwon Hong et al.	No	No	The SDN based solution for slow HTTP DDoS attacks.	The cloud-specific characteristics and attack paths to be covered to fit into the cloud environment.
Shunsuke Tayama et al.	No	No	Investigated the slow Read DDoS attacks based on Bandwidth.	Only slow Read attacks considered. Slow Header and Body attacks are not covered.
Suroto	No	No	Configuration tuning-based detection and mitigation of slow HTTP DDoS attacks.	The configuration tuning is alone not enough to mitigate and prevent attacks. The same configuration tuning is explored to confirmed further opportunities available to betterment the results.
Junhan Park et al.	Yes	No	This work confirmed adding security measure to the webserver can defend DoS attacks, but not DDoS attacks.	This work needs to be enhanced for DDoS attacks in the cloud environment.
Proposed Work	**Yes**	**Yes**	**Alow HTTP DDoS attack detection, mitigation and prevention in the cloud.**	**This research work focused on all types of slow HTTP DDoS attacks in the cloud environment. Covered all possible attacks paths to safeguard webserver from slow HTTP DDoS attacks.**

[19] SUROTO, *A Review of Defense Against Slow HTTP Attack*, , Int. J. on Informatics Visualization, Vol. 1, No. 4, pp. 127–134, (2017).

[20] JUNHAN PARK, KEISUKE IWAI, HIDEMA TANAKA, AND TAKAKAZU KUROKAWA, *Analysis of Slow Read DoS Attack and Countermeasures on Web servers*, Int. J. of Cyber-Security and Digital Forensics, pp. 339–353. (2015).

[21] A.DHANAPAL, AND P. NITHYANANDAM , *An OpenStack based cloud testbed framework for evaluating HTTP flooding attacks*, Wireless Networks - Springer, DOI: 10.1007/s11276-019-01937-4, pp. 570–575, (2019).

[22] A.Dhanapal, and P. Nithyanandam , *The slow HTTP Distributed Denial of Service Attack Detection in Cloud*, Scalable Computing, DOI: https://doi.org/10.12694/scpe.v20i2.1501,Volume 20, Number 2, pp. 285–297, (2019).

[23] Clifford Kemp, Chad Calvert, Taghi M. Khoshgoftaar, *Utilizing Netflow Data to Detect Slow Read Attacks*, 2018 IEEE International Conference on Information Reuse and Integration (IRI), DOI: http://doi.ieeecomputersociety.org/10.1109/IRI.2018.00023, pp. 108–116, (2018).

[24] Tanishka Shorey, Deepthi Subbaiah, Ashwin Goyal, Anuraag Sakxena, Alekha Kumar Mishra, *Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools*, 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), DOI: 10.1109/ICACCI.2018.8554590, (2018).

[25] Ghafar A. Jaafar, Shahidan M. Abdullah, and Saifuladli Ismail, *Review of Recent Detection Methods for HTTP DDoS Attack*, Journal of Computer Networks and Communications - Hindawi, DOI: https://doi.org/10.1155/2019/1283472, Volume 2019, Article ID 1283472, 10 pages, (2019).

[26] Trung V. Phan, and Minho Park, *Efficient Distributed Denial-of-Service Attack Defense in SDN-Based Cloud*, IEEE Access, DOI: 10.1109/ACCESS.2019.2896783, (2019)

Edited by: Dana Petcu
Received: Jun 3, 2019
Accepted: Nov 30, 2019