EDoS-BARRICADE: A Cloud-Centric Approach to Detect, Segregate and Mitigate EDoS Attacks

S. B. Ribin Jones and N. Kumar

Abstract Cloud computing through realizing the height of virtualization offers service models that can meet dynamic demands through performing auto-scaling of resources [1]. This helps the cloud service providers to broaden the grasp across sectors and the computing service market. Though it follows stretchable and elastic service models, it implements a rigid pay-per-use utility pricing model [Ribin Jones and Kumar in J Adv Res Dyn Control Syst 11(9):541–553, 2019 2]. The idea of dynamically scaling across platform makes it more vulnerable to security threats and makes room for easy exploits [Ribin Jones and Kumar in J Adv Res Dyn Control Syst 11(9):541–553, 2019 2]. Among various security threats, the economic denial-of-service (EDoS) attack presents a serious threat, since it exploits auto-scaling feature to impact the utility pricing model [Ribin Jones and Kumar in IEEE Xplore third international conference on trends in electronics and informatics, pp 1003–1008, 2019 3]. In this paper, a real-time cost incurring EDoS attack is performed against a cloud data center hosted Web page with simple Structured Query Language (SQL) manipulation method for experimental research. The experimental observations are applied to define an effective EDoS-BARRICADE that performs detection, segregation and mitigation specific to EDoS attack. The detection algorithm considers metrics that are associated with the auto-scaling feature to detect a suspicious increase in VM activities. The segregation algorithm implements linear SVM to isolate attack VMs optimally and rapidly. The results show that the developed EDoS-BARRICADE algorithms perform detection and segregation with 100% accuracy.

Keywords Cloud computing · Economic denial of service (EDoS) · DDoS · IDPS · SLA

S. B. Ribin Jones (✉) · N. Kumar
Department of Computer Science and Engineering, VISTAS, Chennai, India
e-mail: ribinjones@gmail.com

V. Bindhu et al. (eds.), *International Conference on Communication, Computing and Electronics Systems*, Lecture Notes in Electrical Engineering 733,
https://doi.org/10.1007/978-981-33-4909-4_44

1 Introduction

Cloud computing through achieving multi-layer virtualization provides resources for provisioning of on-demand access through Internet access [2]. The resources involve, but not limited to programming platforms, software services and hardware capabilities such as storage, processing and networks [3]. The cloud has the inbuilt brokering capabilities, through which the service users negotiate various characteristics termed as QoS parameters to reach a service-level agreement (SLA) with cloud service providers (CSPs) [3]. Once the agreement is reached, the service offered has advanced and unique features such as on-demand access, pay-per-use, dynamic resource scaling and other characteristics that ensure cloud elasticity [3]. This helps to meet the end-users' varying service demands, with minimal resource wastage. In short, the utility-based pricing model and dynamic resource assignment model of cloud ensures that the user is only paying for the resources used [3]. The auto-scaling feature of cloud helps to realize the on-use resource scaling requirements through auto-allocation or de-allocation of resources; it can scale up or down based on the usage [4, 5]. The auto-scaling at VM level applies to the number of processing units, memory, networking components, etc [6]. Most presumably used auto-scaling metrics from the performance standpoint are threshold and duration [7]. These two metrics are considered necessary for triggering auto-scaling based on the need.

The value-added features of the cloud computing that distinguish it from other distributed and parallel computing model open an unfathomable number of possibilities for exploits and security breaches [8]. Among those, the improvised DDoS attack in the form of EDoS can exploit the cloud pay-per-use model to incur a huge financial loss to the customer or service provider [9]. The evolution of EDoS attack and the limitations in detecting and preventing such attacks has been explained in our previous works [2, 3]. The most threatening of EDoS attack involves exploiting the auto-scaling feature to cause scaling up of VMs in such a way to stealthily evade the security systems and the casual observations. The scaling EDoS attack if persists for a period of time runs VMs with junk data that adds to the offered service and impact heavily on service bill. This can damage the credibility of the service provider and can impact the business heavily and yet goes unnoticed. EDoS attack can adapt botnet model or applies any of the traditional DoS and DDoS attack model on paid services and can impact as EDoS attack.

1.1 Methodology

Cloud computing nowadays is used extensively on Web servers to enhance the request processing potential. This work therefore intents to perform an EDoS attack through iterating SQL-targeted DoS attack. This component does not compromise computers or host any malicious components in server; this helps to avoid security risks and to make sure that the components function without any security hiccups after the completion of the experiment. Moreover, this helps to understand how a simple attack module can constitute a harmful EDoS attack. The obtained results are then

used to develop the detection and segregation algorithms that work hand-in-hand to monitor, detect and isolate the compromised VMs from genuine VMs. The Snort rules are then configured to filter and block the isolated VMs. This work improvises the segregation accuracy by incorporating simple and quick linear SVM model. The combined effort is then fitted into a framework to offer a wholesome EDoS handling service.

1.2 Organization

The full article is arranged in the following manner. Section 1 describes the introduction of cloud computing and its security implications from EDoS point of view. Section 2 presents a literature survey of EDoS detection, segregation and mitigation-related works. In Sect. 3, an experimental setting to perform a cloud Web server-targeted EDoS attack is narrated. In Sect. 4, the results of the performed experiment have been analyzed. Section 5 presents the proposed algorithms to detect and segregate attacking VMs or instances using linear SVM. Section 6 discusses the obtained results. Section 7 concludes the article by pointing out the future direction for research.

2 Literature Survey

[1] EDoS-ADS uses the threshold and the duration as auto-scaling parameters, an average CPU utilization threshold with two time triggers scaling-up and down are applied to differentiate VM scaling due to attack from normal. The mode switches to the suspicion mode from normal mode when the cloud CPU utilization exceeds the scaling-up utilization thresholds; then if the attack is found, it switches to attack mode if not it switches to flash crowd mode if the increase continues. In NAT run network, it can block the entire range of attacking nodes from its framework. In [10] EDoS-Shield, the main components of the architecture are virtual firewalls (VF) and verifier cloud nodes (V-nodes). The virtual firewalls are VMs with filtering and routing capabilities that work as filter mechanisms based on the white list to allow authenticated sources and blacklists to block unduly sources. The verifier cloud nodes (V-nodes) are a pool of VM that can verify its legitimacy through Turing tests, and the corresponding white or blacklists are updated based on the results of the verification process. In [4] enhanced EDoS-Shield, as an improvement to EDoS-Shield, it presents two algorithms which enhances both from the perspective of TTL field. Algorithm 1 describes the actions to be taken at VF node upon receiving a packet. A packet can be forwarded to the destination if only its source IP address and TTL value match the white list. Algorithm 2 describes the actions to be taken by a V-Node for not forwarded packets. In [11] EDoS ARMOUR, it is a Multilayered defense Architecture. At the first level, it applies the port hiding mechanism to deter

attackers from knowing the port to perform attacks. At the second level, a learning algorithm is used to monitor user behavior. If inappropriate behavior is spotted, then the corresponding user will get a slower service response. This helps to mitigate application-level DDoS attacks. It also entails a challenge mechanism, admission control, congestion control, user classification and client classification mechanism. Cloud eDDoS mitigation scrubber service in [12] is modeled as an on-demand EDoS-specific security service. Its core functional component implements crypto puzzle to be auto-generated and verified by the users or clients. The users are expected to solve the generated crypto puzzle through brute force to avail the requested service. This helps the CSP to follow the legitimacy of the users. The APART in [8] is a pattern recognition-based EDoS attack mitigation model. The number of packets sent by a user falls in the frequency of 400–800 per second; the APART model gets active and applies pattern recognition to detect EDoS attack. As a deterrence mechanism, it includes time-based and key-sharing post-setup authentication scheme to prevent the replication or replay attacks. It also provides a pre-shared security mechanism to ensure the access of legitimate users on the cloud services. In [9], an enhanced EDoS mitigation mechanism EDOS-EMM has been presented. The design involves three sequential modules, namely (1) data preparation, (2) detection and (3) mitigation modules. Module 1 involves flow monitoring, data collection and processing the flow to segregate it by protocol type and implements the sFlow agent algorithm. Module 2 detects the attack and extracts packet fields such as source, destination IP, port number and no. of packets per second. In this module, Hellinger distance and entropy methods are implemented for anomaly detection to improve accuracy. Module 3 generates alert, initiates rule update process and blocks the attacking IP. In [7] an entropy-based architecture is proposed for the detection of EDoS attacks. The proposed multilayered architecture involves monitoring and aggregation of metrics that affect the cost model, the novelty detection procedures to detect EDoS attack, and the decision-making and action response procedures. It is a predictive mode; it applies entropy variations related with considered metrics such as per-client CPU time when deciding auto-scaling actuations ahead of time for forecasting the EDoS attack possibility.

3 Experimental EDoS Attack

Performing EDoS attack involves components that can corrupt the servers, propagate across the Internet or flood the network or server with traffic with or without the knowledge of the attacker. Hence, performing EDoS attack is not safe in any platform. Therefore, even performing a simple DoS attack over the Internet is prohibited by law. However, without real-time experimentation of EDoS attack, the research cannot be effective. Consequently, an attack which targets the cloud Web server and performs VM scaling, but does not involve compromise, propagation and aftermath exploitation possibility, becomes ideal for our research. However, nowadays, Web servers run in cloud platform to meet any number of request; most of them use pay-per-hit model [13]. Therefore, if the Web servers are forced to process more requests,

then it starts more VMs and ends up as a complicated EDoS attack. This requires writing a Web page and hosting it in a cloud Web server, and to use a minimal number of client computers to start consecutive DoS attacks together, they cause an EDoS attack. This way, single client computer can start any number of VMs in the server that stays and performs EDoS attack on server without compromising the facility.

The objective of this section is to perform a real-time EDoS attack on a Web hosting data center. The experimental setup involves a Windows Server 2016 and 25 machines connected through a LAN. The server runs in a cloud platform. The experimental scenario involves setting up a client Web page and creating db tables in server for Web access. Initially, the database for Web application is loaded in the server with various user names. This helps to ensure that none of the server which runs Web pages is targeted by the attack. The client sends the request; the server uses VMs to create a separate instance to process every request. Usually, once the request is complete processing, the VM is released. However, to perform database-targeted EDoS attack, it involves three steps. Step 1: The blah application is used to insert a user name and password from the client browser. Blah is a database manipulation command which helps to perform SQL injections and DoS attacks [14]. Step 2: To start a DoS attack, a 'blah' shell command is delivered to start a separate VM for every request as well as to performs a half-open TCP SYN-based DoS attack. The command **blah'; exec master.xp_cmdshell 'ping** target Web page **-165000 –t'**; – create a VM in server-side and that VM acts on behalf to reply to the client. It then performs a half-open TCP ping-based DoS attack. Step 3: The attack performs VM scaling by starting five VMs every 5-min interval from five computers through repeating step 2 for 5 times. This avails five VMs every interval. The cost of availing every VM for one minute is set to 0.0011 [8]. The hypervisor access is prohibited to avoid single-point failure because if it fails, the entire cloud facility becomes inaccessible. The VM or workloads of the experiment is monitored by task manager and Wireshark applications.

4 Experimental Results of EDoS Attack and Observation

The Windows Server on which the EDoS attack is performed is hosting the cloud-based file sharing application termed as 'ownCloud.' ownCloud becomes inaccessible due to the attack from the local client. Since every VM responds individually on behalf of the server, it becomes difficult for the firewalls to detect. The experiment has been conducted in a confined environment only after making sure that the Windows Defender Firewall and Web Shield are turned ON. However, they failed to detect the attack due to two reasons; one is the projected traffic which happens with in the server, that is, VM or frontal instances and Web server runs within the server due to the cloud incorporation. The firewall that looks for anomaly from the boundary of the server, therefore, fails to detect. Another reason is that the VMs send traffic in an individualized manner, so collectively, it will not shoot up at firewall observation point. Therefore, the firewalls fail to register the flood.

The results are observed from the perspective of traffic, processing and memory. According to Figs. 1 and 2, it is evident that the temporary memory usage is increasing gradually, i.e., from 1.9 (48%) to 2.1 (53%). Initiation of application involves starting and allotting space for VMs; therefore, more memory usage is evident in Fig. 1. However, in Fig. 2, due to the similarity of data processed and traffic generated, it did not reflect on memory or processing even after starting additional 20 frontal instances. However, for genuine application, each VM may process unique tasks, so it consumes more processing capability and reflects upon processing and memory.

The Web server due to the cloud implementation allows any number of frontal instances or VMs to be deployed by a single click from the client computer. It also allows every VM to act as the server to the client. At the same time, it manages to receive tremendous TCP hits that are generated from the VMs. In spite of all this, the server functions smoothly except for the subordinate application such as ownCloud.

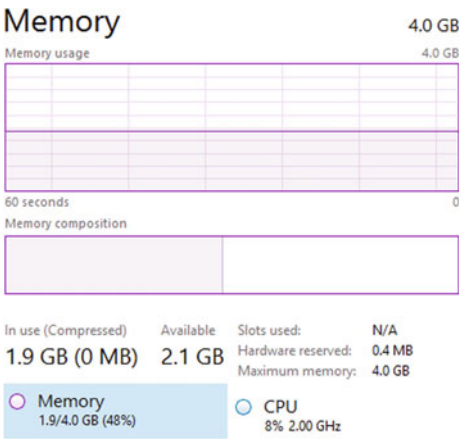

Fig. 1 Memory observation after starting of five VMs

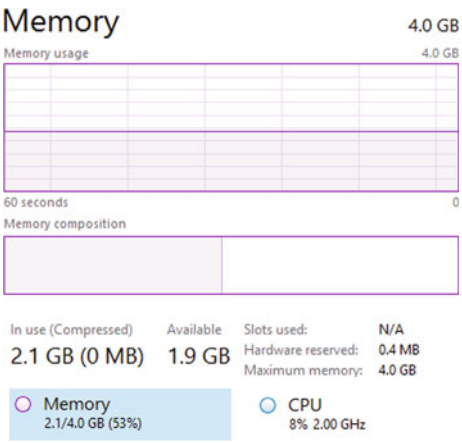

Fig. 2 Memory observation after starting of 25 VMs

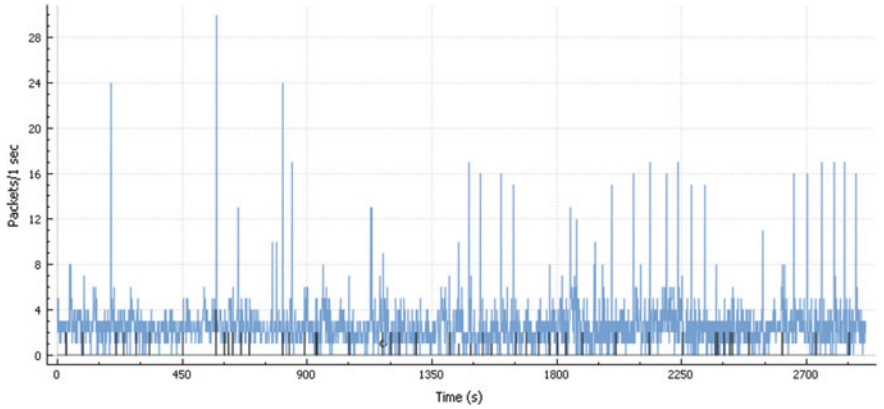

Fig. 3 Horizontal increase of incoming TCP packets from 25 VMs toward the server

This is because the Web service is prioritized over the ownCloud sharing application. The traffic anomaly is however observed with Wireshark, and the observation is presented as the following graph.

According to Fig. 3, the TCP traffic increases in horizontal manners in specific to the VMs, which is uncustomary for a flood attack. Moreover, occasional hits from the database (DB) or Web server were also visible from the graph, denoted as black bars. Consequently, the results show the requirement for individualized monitoring of VMs both from the perspective of the following auto-scaling metrics and behavior of VMs. Based on the experimental results, the following detection and segregation algorithms were designed.

5 Proposed EDoS-Barricade

The complications of cloud security arise from its unique features. The proposed EDoS-Barricade is therefore designed to look into the cloud features closely to eliminate the most threatening form of cloud attack, i.e., EDoS. Evidently after close examination of the experimental results, for initial EDoS detection, following the number of VMs with respect to a predetermined interval becomes necessary to deter the auto-scaling-based attack from exploiting the cloud pay-per-use model. The detection algorithm, therefore, requires to follow the VM to server communication and capturing to-and-fro packets. However, to achieve that, the Snort firewall has been chosen. The Snort is a widely used firewall for Linux, and it suits both research and commercial purposes and helps to build OS kernel-level solution for security implementation. However, Snort is tedious to work with, after careful tuning of Snort SO rules, pre-processor rules, etc.; it still lacks the proper command to detect VM to server traffic. After careful inspection, the following command is tailored to detect VM to server communication; so far, there is no reference to do that .

Table 1 Snort VM to server detection procedure

Innovative Snort command to detect in-server VM to server hits
alert tcp $HOME_NET any any -> $HOME_NET any flow:to_server; Seq: N; ack: A;
win: W;length: l; Options:O; classtype:attempted-user;

The above command helps to detect VMs with its IDs, as well as to follow all sorts of in-house traffic between VM and server communication; it can also follow server to VM communications.

5.1 Auto-Scaling-Based EDoS Detection

The 15 number of VMs is identified as the reasonable number for presetting detection metric from our experimentation standpoint. The chosen interval is 5 min. This number can be deduced by looking into cloud history. For instance, the average number of VMs running with respect to the time interval for a period of history, such as for a month and the year, can be computed to set this parameter. If the number of VMs increases for three consecutive intervals, then this algorithm raises the detection alert as well as switches mode to suspicious and calls the EDoS-BARRICADE Segregation () algorithm.

5.2 Support Vector Machine (SVM)

The EDoS segregation requires classifying data into two binary sets {EDoS, normal}. However, to achieve that, the SVM classifier is an ideal choice. SVM offers both linear and nonlinear mapping techniques to project and classify the dataset or input into 2-D or 3-D plane. This projection helps to deduce a hyperplane that can segregate data into its corresponding classes. The linear SVM can offer quick and accurate classification compared to other classifiers. Moreover, SVM can stretch to big data processing since it can handle outlier and overfitting effectively. The dataset during the preprocessing is usually depicted as follows;

$$(x_1,\ y_1),\ (x_2,\ y_2), \ldots (x_i,\ y_i), \ldots (x_n,\ y_n) \tag{1}$$

where x_i denotes the metrics that can help to classify the given data; from our observation, they are incoming and outgoing packets in specific to VMs, n denotes the maximum number of values, and i denotes the iteration of values from 1 to n.

$$\text{Class}x_i \in \{\text{Incoming\_pkts},\ \text{Outgoing\_pkts}\} \tag{2}$$

The class y_i serves as a label and helps to separate the VM data into its corresponding class.

$$\text{class} y_i \in \{\text{EDoS,normal}\} \tag{3}$$

Given the datasets x_i and y_i, the SVM classification engine naturally creates a hyperplane as follows

$$w^{\mathrm{T}} * x + b = 0 \tag{4}$$

The hyperplane is a separator that determines the maximum margin between two classes, where w is a weight vector and b is a bias. The SVM automatically creates weight vector and bias to group the data using a hyperplane. The grouping is done as follows.

$$\text{if} \quad w^{\mathrm{T}} x + b \geq +1 \text{ then normal} \tag{5}$$

$$\text{if} \quad w^{\mathrm{T}} x + b \leq -1 \text{ then attack} \tag{6}$$

However, the closest point called a support vector from the respective classes can only establish the optimally separating hyperplane. Support vector from the given dataset determines the hyperplane by using:

$$\min(\tau) = \left\{ {}^{1}\!/_{2} ||w||^2 \right\} \tag{7}$$

It has been shown that the optimal separating hyperplane can be found by minimizing (7), with respective to

$$y_i (w,\ x_i + b) \quad \text{for all} \quad i = 1, \ldots, n \tag{8}$$

This is a convex optimization problem; the SVM classifier solves it with Lagrange multipliers

Algorithm:1 EDoS-BARRICADE Detection ()

Step 1: Monitor increase or decrease in workloads or VM scaling
Step 2: If no. of VM increases above the default limit then initiate EDoS detection
Step 3: If it continues to increase for consecutive intervals, raise detection alert and spontaneously call the segregation procedure

(continued)

(continued)

Algorithm:1 EDoS-BARRICADE Detection ()

Input: {List of VMs with unique identifier, Interval}
Output: {List of VMs, No. of, Interval, Total No. of incoming & Outgoing pkts}
Define VM_Number; \\ Number of workloads
Define interval = 5 min; \\ Time Interval
Define i = 0; \\Increment Variable
For each application;
if VM_number > 15 then Monitor;
For each interval;
if current_VM_No > previous_VM_No;
{ i +1
if i > 3 then raise detection alert
Switch Mode to Suspicious
Call EDoS-BARRICADE Segregation () }

5.2.1 Linear SVM

The dataset contains the non-separable case of data; however, by introducing a cost for violating constraint (8) such as a positive slack variable ξ, a hyperplane can be realized for optimal separation.

$$y_i.(x_i.w + b) \geq 1 - \xi_i. \quad \text{For all } i \tag{9}$$

It can be furthered for erroneous cases with cost parameter C, and such a scenario is not required for this research.

5.2.2 Nonlinear SVM

If the data is not mapable using linear decision function, the data will be mapped into higher dimensional feature space through a nonlinear transformation which incorporates kernel functions. Vastly used kernel functions for binary classification problems are as follows;

$K(x_i,x_j) = x_i x_j$: *linear SVM*
$K(x_i,x_j) = (x_i x_j + 1)^p$: *polynomial of degree p*
$K(x_i,x_j) = \tanh(a.x_i x_j + b)$ *and* $a > b$: *multi-layer perceptron* (*MLP*) *classifier*
$K(x_i,x_j) = \exp\{||x_i - x_j|^2/\sigma^2\}$: *radial basis function* (*RBF*) *classifier.*

5.3 Behavior Heuristics-Based EDoS Segregation

The training dataset has been generated during the monitoring with respect to VMs. The data is provided to an SVM-based EDoS-BARRICADE segregation algorithm to differentiate malicious VMs from normal VMs.

Algorithm:2 EDoS-BARRICADE Segregation ()

Step 1: Upon invocation from the detection algorithm; Gather the data.
Step 3: Perform anomaly-based segregation by applying relevant SVM.

Input : {List of VMs, No. of, Interval, Total No. of incoming & Outgoing pkts}
Output: {Scatter plot, Attacking VM list for snort to filter and block }
Call SVM Classifier ()
For each VMs;
{ if $wT\,x + b \geq +1$
then project as normal
if $wT\,x + b \leq -1$
then project as EDoS }
For all VMs;
{if linearly separable
Deduce hyperplane $yi.(xi.w + b) \geq 1 - \xi i$
else apply kernel}
project result in scatter plot
Generate Attack VMs list
Call Snort()
Perform Filtering and Blocking

The above algorithm presents the result for input dataset instantaneously; it can be plotted into two classes separated with a hyperplane. Moreover, the output presents a list of attacking VMs which can be blocked and filtered out. The following sections examine the result of EDoS-BARRICADE through varying the input.

6 EDoS-Barricade Result Observation and Verification

The experimental data along with normal VM dataset obtained from GitHub has been fed into RapidMiner. The linear SVM has been simple enough to classify the normal flow from the attack flow as follows.

According to Fig. 4, attacking VMs distinguishes itself with the following heuristics, i.e., for attacks, 'Accumulated incoming packet > Accumulated normal Incoming packets' and 'Accumulated Outgoing packet < Accumulated normal Incoming packets.' Moreover, to intensify the testing, data migration dataset obtained from GitHub has been incorporated into the dataset, and then, testing is performed. For that, the SVM is required to transform data into a higher dimensional plane using a kernel as shown in Fig. 5.

According to Fig. 5, the attack instances become as blue dots becomes classifiable using SVM kernel when the data migration dataset is involved. However, this is an

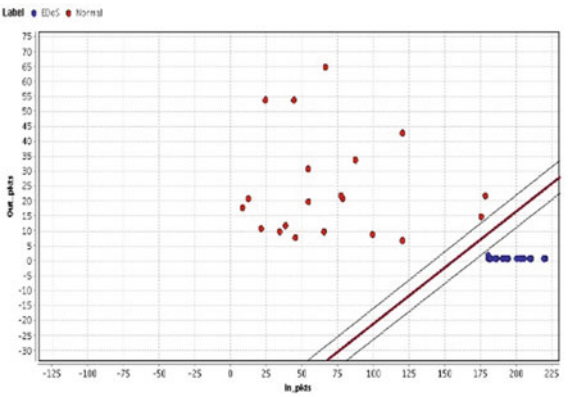

Fig. 4 Linear SVM segregation result after the third interval

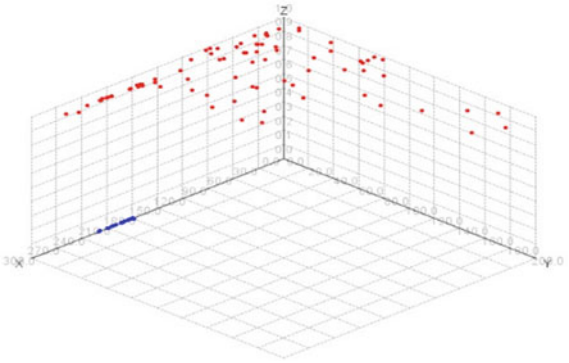

Fig. 5 Nonlinear SVM segregation result for a dataset with data migration instances

unusual and rare happening; for other cases, simply linear SVM proves sufficient to detect EDoS attack. The accuracy achieved for EDoS detection after the third interval is 100%. The obtained SVM-based EDoS-BARRICADE result is then interpreted as the list of attacking VMs and forwarded to the Snort firewall for filtering and blocking.

7 Conclusion and Future Scope

The EDoS attack is performed on any cloud Platform through exploiting the auto-scaling feature to increase the number of VMs, which in reality processes junk data and floods the server with junk requests. This phenomenon manages to exploit the cloud usage-based pricing model and impact heavily on customer bills. However, to have a real-time perspective, a Web server-based EDoS attack is performed, and the impact of the attack is observed. The result shows that the EDoS attack is detectable by following the auto-scaling and segregate-able by following incoming

and outgoing packets. In accordance with the results and through making necessary improvement to the Snort, a quick auto-scale-based EDoS-BARRICADE detection algorithm has been proposed. The tedious part involves isolating attack VMs from the normal ones without false detections. An EDoS-BARRICADE segregation algorithm which involves linear SVM as its base classifier is proposed and tested with datasets involving normal and attack data. The linear SVM performs classification with 100% accuracy since the normal data has more number of incoming packets and less number of outgoing packets than the attack data. However, more rigorous testing has been performed by including data migration samples to the dataset. In such cases, the algorithm switches to nonlinear kernel-based SVM classifier to isolate the attack class in higher dimensional plane. In the future, the presented work will be extended into a framework with more added characteristics.

References

1. Shawahna A, Abu-Amara M, Mahmoud ASH, Osais Y (2018) EDoS-ADS: an enhanced mitigation technique against economic denial of sustainability EDoS attacks. IEEE Trans Cloud Comput (Feb)
2. Ribin Jones SB, Kumar N (2019) Unraveling the security pitfalls that stem from core cloud benefits through analyzing various DoS attacks, detection and prevention. J Adv Res Dyn Control Syst 11(09):541–553 (Aug)
3. Ribin Jones SB, Kumar N (2019) Precursory study on varieties of DDoS attacks and its implications in Cloud Systems. In: IEEE Xplore third international conference on trends in electronics and informatics, pp 1003–1008, Apr 2019
4. Sqalli MH, Al-Haidari F, Salah K (2012) Enhanced EDOS shield for mitigating EDoS attacks originating from spoofed IP addresses. In: IEEE eleventh international conference on trust, security and privacy in computing and communications
5. Kumar D (2019) Review on task scheduling in ubiquitous clouds. J ISMAC 1(01):72–80
6. Bulla S, Basaveswara Rao B, Gangadhara Rao K, Chandan K (2018) An experimental evaluation of the impact of the EDoS attacks against cloud computing services using AWS. Int J Eng Technol 7(1.5):202–208
7. Monge MAS, Vidal JM, Villalba LJG (2017) Entropy-based economic denial of sustainability detection. MDPI Entropy 19(12) (Nov)
8. Thaper R, Verma A (2015) Adaptive pattern attack recognition technique (APART) against EDoS attacks in cloud computing. In: IEEE third international conference on image information processing
9. Singh P, Rehman SU, Manickam S (2017) Enhanced mechanism to detect and mitigate economic denial of sustainability (EDoS) attack in cloud computing environments. Int J Adv Comput Sci Appl 8(9)
10. Sqalli MH, Al-Haidari F, Salah K (2011) EDOS shield—a two-step mitigation technique against EDoS attacks in cloud computing. In: IEEE fourth international conference on utility and cloud computing
11. Masood M, Anwar Z, Raza SA, Hur MA (2013) EDoS Armor: a cost effective economic denial of sustainability attack mitigation framework for e-commerce applications in cloud environments. In: IEEE sixteenth international multitopic conference
12. Kumar MN, Sujatha P, Kalva V, Nagori R, Katukojwala AK, Kumar M (2012) Mitigating economic denial of sustainability (EDoS) in cloud computing using in-cloud scrubber service. In: Fourth international conference on computational intelligence and communications networks

13. Abusitta A, Bellaiche M, Dagenais M (2018) An SVM-based framework for detecting DoS attacks in virtualized clouds under changing environment. Springer J Cloud Comput: Adv Syst Appl 7(9):1–18 (April)
14. Mazrekaj A, Shabani I, Sejdiu B (2016) Pricing schemes in cloud computing: an overview. Int J Adv Comput Sci Appl 7(2):80–86
15. Duraipandian M, Vinothkanna R (2019) Cloud based internet of things for smart connected objects. J ISMAC 1(02):111–119
16. Udhayan J, Hamsapriya T, Vasanthi NA (2012) DDoS attack detection through flow analysis and traffic modeling. In: SPIT 2011, LNICST 62, pp 89–94
17. Baig ZA, Sait SM, Binbeshr F (2016) Controlled access to cloud resources for mitigating economic denial of sustainability (EDoS) attacks. Elsevier Comput Netw 97:31–47 (Mar)