Hindawi
Journal of Computer Networks and Communications
Volume 2017, Article ID 7674594, 8 pages
https://doi.org/10.1155/2017/7674594

Research Article

Cloud-Based DDoS HTTP Attack Detection Using Covariance Matrix Approach

Abdulaziz Aborujilah[1] and Shahrulniza Musa[2]

[1]*Malaysian Institute of Information Technology (MIIT), University Kuala Lumpur, Kuala Lumpur, Malaysia*
[2]*University Kuala Lumpur, Kuala Lumpur, Malaysia*

Correspondence should be addressed to Abdulaziz Aborujilah; abdulazizsaleh@unikl.edu.my

Received 20 March 2016; Revised 9 August 2016; Accepted 27 November 2016; Published 24 January 2017

Academic Editor: Gianluigi Ferrari

In this era of technology, cloud computing technology has become essential part of the IT services used the daily life. In this regard, website hosting services are gradually moving to the cloud. This adds new valued feature to the cloud-based websites and at the same time introduces new threats for such services. DDoS attack is one such serious threat. Covariance matrix approach is used in this article to detect such attacks. The results were encouraging, according to confusion matrix and ROC descriptors.

1. Introduction

It has been known that computer networks technology is one of the most important tools to exchange and share data, besides several of our daily tasks done online. Cloud computing offers new online IT services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). However, it suffers from several threats, which target its confidentiality, integrity, and availability. HTTP attack is one of the most critical attacks which compromise the cloud-based web servers availability. Therefore, effective intrusion detection models are needed to secure them. The latest e-crime research survey conducted by the E-Crime Congress 2009 has shown that online consumers were mostly at risk because their websites were under the attacks. Moreover, 63% of the answerers believed that their customers were affected by the poisoned websites, while 40% of the total respondents mentioned that the technical sophistication is increasing because of such attacks [1]. In this study, multivariate correlation analysis-based detection approach (MADM) [2] is used to detect HTTP-flooding attacks in the cloud-based web servers. This approach is preferred as compared with other approaches because it consumes less computing resources. MADM is a statistical based approach which uses second-order statistics to distinguish different kind of flooding attacks based on their behavior. This study is constructed as follows: Section 2 presents the research studies related to IDS approaches used in cloud computing environment. Section 3 presents the research methodology. Section 4 presents MADM implementation in private cloud testbed. In Section 4, the experiments results in cloud environment are shown. Finally, Section 6 presents the conclusion, along with future works suggestions.

2. Related Works

Alsowail et al. [3] proposed a technique to mitigate economic denial of sustainability (EDoS) attacks in cloud computing platform. Their technique relied on comparing the rates of similarity between every two consecutive packets. When two packets requests or more are sharing the same payload information of the requests in one single communication stream, they are marked as attacker packets. Transmitter node is marked as attacker node. Thus, the attacker information is sent to the cluster's nodes. In addition, whole cluster's node is warned to cut off the attacker requests.

In Kumar et al.'s [4] study, they have suggested a technique to mitigate economic denial of sustainability (EDoS) attacks. In this technique, the user legitimacy is verified by using the puzzle method. This technique aimed to supply cloud service to genuine users only and stop unauthorized users from accessing it. In this technique, cloud service has two

modes either normal or suspected modes; based on the end user identification and its puzzle's answer, cloud service mode is switched. In case the end user is legitimate to access, he will be directed to cloud service; otherwise, its request will be transferred to verification procedure.

The study of Mary et al. [5] has proposed DDoS and EDoS-Shield mitigating technique. It uses virtual firewalls (VF) and verifier cloud nodes (Vnodes) to mitigate the EDoS in cloud computing environment. The VF is used to hold the senders IP address for authentication purpose. And V-node verifies the end users requests using tuning tests method and update VF accordingly.

Hybrid-network intrusion detection system (H-NIDS) is suggested by Modi and Patel [6]; this detection method uses Bayesian, associative, and decision tree to detect the network attacks in cloud computing environment. H-NIDS consists of packet capture module that captures the incoming network traffic for auditing purpose. Signature module is used to match the captured packets with the predefined attacks pattern to find any correlation between them. Based on the matching result, an alert message is sent to the score function which collects the messages that may come from different cloud nodes to identify any intrusion in the whole cloud environment.

Gul and Hussain [7] suggested multithreaded NIDS model. It consists of four models which are capture, queue, analysis, processing, and reporting modules. The capture module is responsible for capturing the incoming and outcoming (ICMP, TCP, IP, and UDP) packets. Then, the captured packets are sent to rules based analysis module using shared queue. Based on analysis process, bad and normal packets are identified. Next, a third party monitoring advisory services is used to generate comprehensive expert advisory that reports for cloud users and providers.

To date, several studies have suggested using IDS in cloud computing environment. However, most of these studies focused only on using IDS in public cloud computing such as in [3–7]. But very few studies concentrated on testing and evaluating IDS in private cloud environment. So, this research will focus on examining and evaluating MADM performance in private cloud computing environment.

3. Research Methodology

3.1. Overview. This study follows the same methodology as it has been elaborated in [2]. It is divided into training and testing phases. The training phase focuses on constructing the normal network traffic behavior profile. The testing phase test aims to detect the deviation between the normal profile and any other network traffic. In this research, the normal and flooding attacks datasets are captured from our cloud tested. The normal traffic is captured by letting the ordinary end users browse the Internet and capture the traffic, whereas the flooding attack traffic is generated by attacking the virtual web server using PageRebooter tool [8]. Then, the covariance matrices of normal and abnormal traffic are calculated. In training time, the normal profile is made by finding the average and threshold matrices of the normal network traffic. Furthermore, in the testing phase, the covariance matrices of newly network traffic are compared with the expected matrices. In the case of a deviation between the testing covariance matrix and expected matrix more than the threshold matrix, this means that the testing traffic belongs to the flooding attacks; otherwise, it belongs to the normal traffic. More details of the research methodology are in the following sections.

3.2. MADM Algorithms. MADM is implemented in two main algorithms. Training Algorithm 1 and testing Algorithm 2 are as follows.

Algorithm 1 (MADM training).

(1) Start: normal: normal KDD dataset; E(normal): normal dataset expected matrix; D(normal): normal dataset threshold matrix.
(2) Read normal.
(3) Segment normal.
(4) Compute the Covariance matrices of the normal dataset.
(5) Compute the mean of E(normal)
(6) Compute the standard deviation of the normal dataset D(normal).
(7) Compute the normal profile threshold matrix dataset $\sqrt{D(\text{normal})}$.

Algorithm 2 (MADM testing).

(1) Sptart: Test: testing dataset; E(normal): normal dataset expected matrix; $\sqrt{D(\text{normal})}$: normal dataset threshold matrix; R(detection result): detection result matrix.
(2) Read test
(3) Repeat until (test) empty:
(4) Segment test
(5) Compute Covariance matrices of test;
(6) **if** (test) $-$ E(Normal) $\leq$ $3\sqrt{D(\text{normal})}$ or $4\sqrt{D(\text{normal})}$ **then**
(7) Add zero to R(detection result)
(8) **else**
(9) Add one to R(detection result)
(10) **end if**
(11) Output: R(detection result)

3.3. Dataset Preprocessing. In this step, the normal and flooding attack traffics are stored in SQL database that has one table with the following fields: number of packets, number of bytes, number of packets A-B, number of bytes B-A, number of packets B-A, number of bytes A-B, duration, bps A-B, and bps B-A. The description of these features is mentioned in Section 4.2. Next, the dataset tables are segmented or sampled as in Section 3.4.

3.4. Dataset Segmentation. The dataset comprises a vast number of records or samples. Every one of them represents one observation with predefined features $f = \{f1, f2, \ldots, fk\}$.

It also corresponds to one of the predefined classes either normal or attack class. In segmentation step, the whole dataset observations are grouped into matrix with length equal to 10, 50, or 150 for each segment. Then, the covariance matrix of each group or segment is calculated.

3.5. MADM Training. The goal of MADM training is to build the normal behavior baseline which consists of the expected and threshold matrices of the normal covariance matrices. To create the expected training matrix, the average of all the covariances calculated. For example, when Cn equals several covariance matrices $\{c1, c2, \ldots, cn\}$ calculated in the training phase, the expected matrix is calculated to apply the following:

$$E(n) = \frac{\{c1, c2, \ldots, cn\}}{n}. \quad (1)$$

The second phase is calculating the threshold matrix. This matrix is founded by calculating the standard deviation of every two adjacent features in the training covariance matrices as placing the result under power of 3 or 4 roots as in the coming equations $3\sqrt{D(\sigma fl^u, fl^v)}$ (3D) and $4\sqrt{D(\sigma fl^u, fl^v)}$ (4D). Determining these two thresholds is mathematically proofed in [2].

3.6. MADM Testing. The outcome of covariance matrix-based detection method depends on the dissimilarity function:

$$\text{Dist}\left(M^{\text{obs}}, N; T\right). \quad (2)$$

This function is used to determine the dissimilarity between the normal model profile and testing dataset. M^{obs} is the observed covariance matrix under the testing process, N is the normal baseline matrix, and T is the threshold matrix. In MADM approach, the testing data is classified based on the dissimilarity function applying the following detection rules.

For each covariance matrix sample $\forall M^{\text{obs}}$ in testing dataset,

if $\text{Dist}(M^{\text{obs}}, E(\omega 1) \leq \delta 1) p \times p$, then $M^{\text{obs}} \in \omega 1$,

else if $\text{Dist}(M^{\text{obs}}, \omega 2 \leq \delta 2) p \times p$, then $M^{\text{obs}} \in \omega 2$,

else if $\text{Dist}(M^{\text{obs}}, E(\omega 3) \leq \delta 3) p \times p$, then $M^{\text{obs}} \in \omega 3$,

else $M^{\text{obs}} \in$ unknown attack.

Moreover, each observed covariance matrix M^{obs} in the testing dataset is compared with the normal and flooding attacks classes profiles, if the difference between the observed covariance matrix and the expected matrix of the normal class baseline $E(\omega 1)$ is smaller than or equal to the normal class threshold matrix ($\delta 1$); then, this observed covariance matrix belongs to the normal class, $\omega 1$, or it belongs to one of the flooding attacks classes. The outcome of this detection method is a 0-1 matrix. This matrix represents the degree of the deviation between the normal profile and the testing datasets. In the testing phase, covariance matrices of the newly observed network traffic are compared sequentially with the norm profile by using Dist(·) function. Whenever there is considerable difference from the norm profile, then the flag of flooding attacks will be recorded as value 1, or 0 if there is no difference. The result of this function represented in matrix consists of 0 and 1 values. The value 0 stands for the fact that the observed traffic belongs to the normal class, and 1 stands for one of the predefined attacks. Finally, the detection matrix might be expressed as these examples of matrices (3), (4), and, (5):

$$\text{Dist}\left(M^{\text{obs}}, N; T\right) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad (3)$$

$$\text{Dist}\left(M^{\text{obs}}, N; T\right) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad (4)$$

$$\text{Dist}\left(M^{\text{obs}}, N; T\right) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \quad (5)$$

3.7. Covariance Matrix-Based Detection Method Outcome. The outcome of the covariance matrix-based detection method is a 0-1 matrix. This matrix represents the degree of the deviation between the normal profile and the testing datasets.

On the testing stage, each covariance matrix in the observed network stream is compared with the norm profile by using Dist(·) function; whenever there is a significant difference from the norm profile, 0 values will be stored in the detection result matrix and if there is significant difference, the value 1 will be stored.

The result of this function in each comparison is represented in matrix consisting of 0 and 1 values. The value 0 represents that there is no difference between the observed covariance matrix and the expected matrix; the value 1 shows that there is a difference between the observed covariance matrix and the expected matrix.

Using more testing covariance matrices gives different forms of 0-1 matrices according to their deviation from the normal baseline model. The value of 0 or 1 can be placed in different positions (the coordinates of rows and columns) based on their variation with the normal profile.

For example, by looking at (3) and (4), they both represent the result of comparing two observed covariance matrices with the normal profile N. They are significantly different from the normal profile because they involve several 1 values and at the same time they represent different kinds of attacks. This is because the positions of the 1 values differed from (3) to (4). In (5), the observed matrix belongs to the normal class because it consists of only zero values which means there is no variation between the observed covariance matrix and the normal profile.

The final result of testing phase is determined by calculating the average of all 0-1 matrix. When the values of average matrix is near to zero, that means the testing dataset

Table 1: Confusing matrix parameters.

Actual class	Predicted class Positive	Negative	Total
Positive	TP	FN	Pos = TP+FN
Negative	FP	TN	Neg = FP+TN
Total	TP+FP	TF+TN	Pos+Neg

belongs to the normal class; otherwise, it belongs to one of the flooding attack classes.

3.8. MADM Detection Results Evaluation. In order to evaluate the MADM detection model performance in the private cloud environment, confusing matrix and ROC are used.

3.8.1. Confusing Matrix. The confusing matrix is the matrix used to describe the classification results. It includes TP, FP, TN, and FN values. Moreover, regarding the meaning of these indicators values as in Lutu study [9], the TP value means the number of positive samples that are correctly predicted as positive samples. The value of the FP means the number of negative examples that are incorrectly predicted as positive samples. The value of the TN means the number of negative samples that are correctly predicted as negative samples. The value of the FN means the number of positive examples that are incorrectly predicted as negative samples, as shown in Table 1.

3.8.2. Confusing Matrix Performance Descriptors. Several detection performance descriptors can be calculated based on the confusing matrix indicator values as in Table 1. In this analysis, the same detection criterion introduced in [2] will be applied to validate the performance of MADM in private cloud computing domain. These descriptors consist of the detection accuracy rate, classification precision rate, false positive rate, false negative rate, and classification precision rate. The detection accuracy rate is the number of the normal samples which have been classified as correctly as normal samples broken down by all the samples classified as normal samples. The classification precision rate is the overall detection model accuracy which can be calculated by finding the number of samples that have been classified correctly and divided by the total number of all the samples. The false positive rate is the percentage of abnormal samples that have been classified as normal samples. The false negative rate indicates the percentage of the normal samples which have been classified as abnormal samples. Classification error rate points out the rate of the misclassified cases over the whole set of samples [10].

3.9. ROC Curves and AUC. Receiver Operating Characteristic (ROC) curves are graphical depictions of the classification model performance [11–13]. They are applied to evaluate the performance of any 2-class classification model. ROC curve measures the classification model performance by drawing the relationship between the FP rate and TP rate under various threshold values.

In order to evaluate the classifiers performance using one single value, AUC is used. AUC value represents the expected performance of one particle classification method [14]. Also, AUC is the segment of the area under the unit square of ROC which always takes a value between 0 and 1. The AUC yields the diagonal line between coordinates $(0, 0)$ and $(1, 1)$ with area 0.5. Thus, no realistic classifier gives AUC values less than 0.5 [15].

4. MADM Implementing in Private Cloud Computing Environment

4.1. Overview. The cloud-based MADM experiments have been conducted in two scenarios, in the internal cloud environment. Since the HTTP-flooding attack is considered as one of the most dangerous attacks in the cloud environment [1], the HTTP-flooding attack has been implemented in this study.

4.2. MADM Features. In this experiment, the network traffic TCP conversation statistic features have been used as follows:

(1) *Number of Packets*. The total number of packets is sent from the source IP address to destination IP address and vice versa.

(2) *Number of Bytes*. The total number of bytes is sent from the source IP address to destination IP address and vice versa.

(3) *Number of Packets A-B*. The total number of packets is sent from the source IP address to destination IP address and vice versa.

(4) *Number of Bytes B-A*. The total number of bytes is sent from the destination IP address to the source IP address.

(5) *Number of Packets B-A*. The total number of packets is sent from the destination IP address to the source IP address.

(6) *Number of Bytes A-B*. The total number of bytes is sent from the source IP address to destination IP address.

(7) *Duration*. It is the duration of the conversation in seconds.

(8) *bps A-B*. It is the average of bits sent between the source IP address to destination IP address.

(9) *bps B-A*. It is the average of bits sent between the destination IP address to the source IP address.

4.3. MADM Implementation. In order to evaluate the applicability and effectiveness of MADM in cloud computing environment, the existing MADM was implemented. Furthermore, the steps of this experiment implementation are as follows: first, the normal traffic dataset is captured during the end user browsing the Internet normally. Second, the flooding attack traffic dataset is captured once the end user runs the attacks by using refresher tool.

Second, this dataset is preprocessed and simplified by converting it into MYSQL database using Microsoft 2007 export data adds-in. Then, in MYSQL database, the flooding

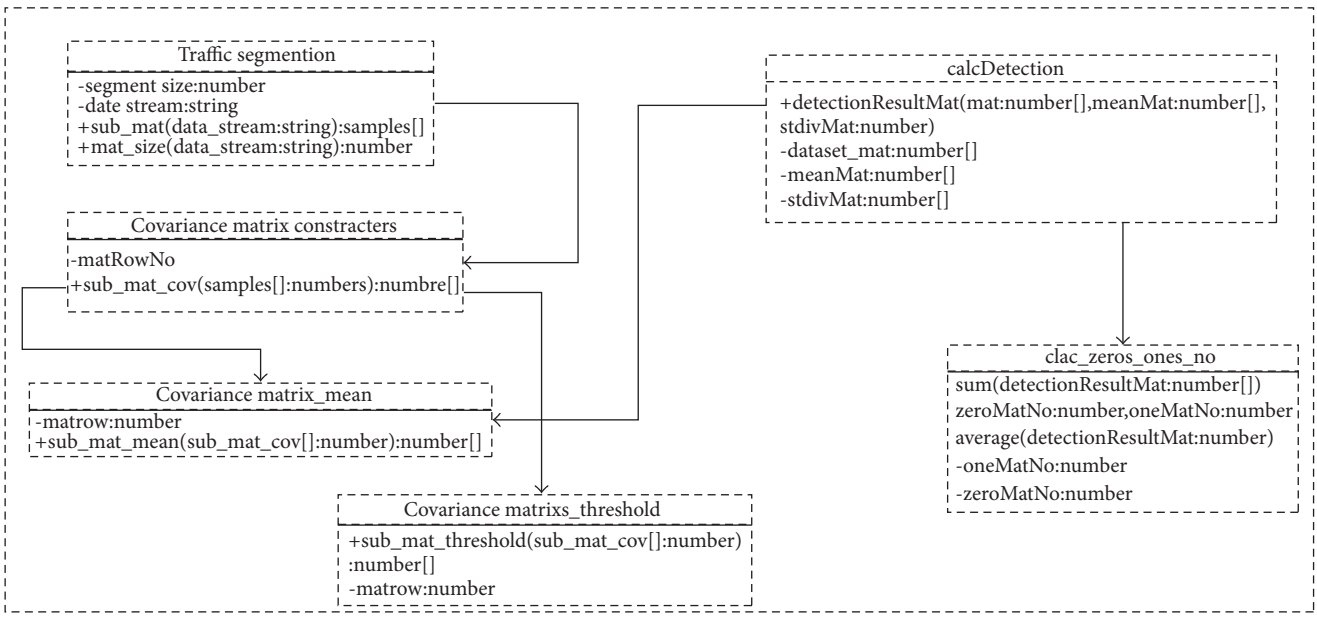

Figure 1: MADM class diagram.

attacks and normal class data are separated into independent tables.

Then, MATLAB R2009a is used to implement MADM experiments. Moreover, the flooding attacks and normal table are exported into MATLAB workspace using Database toolbox in MATLAB.

To train MADM model, four functions are created: the first function is used to segment the exported data into groups of samples with predefined fixed size (10, 50, and 150). This is because of their high stability as it was mathematically proved in [2]. A suitable n can be selected as a relative stable value. In the case studies, we select n as 150. The second function is used to calculate the covariance matrices of each group and then store them in multiple dimensional matrices. The third function is used to calculate the threshold matrix of calculated covariance matrices. And the fourth function is used to calculate the average of training covariance matrix groups.

To test the model, two functions are created. First, detection function is used to compare every covariance matrix in testing dataset with all classes' averages and find the degree of deviation from their threshold matrix. Second, the detection presentation function is used to calculate the average of all the detection result matrices as 0-1 matrix and show the final detection result. See the class diagram in Figure 1.

In order to check out the applicability and capability of MADM in cloud computing environment, the existing MADM was carried out. Furthermore, the steps of this experiment implementation are as follows: first, dataset constricting phase in which the normal cloud traffic is captured in normal Internet browsing case as well as the flooding attack traffic is captured under flooding attacks caused by using refresher tool.

Second, this dataset is preprocessed and simplified by transforming it into MYSQL database using Microsoft 2007 export data adds-in. Then, in MYSQL database, the flooding attacks and normal class data are split into separate tables.

Then, MATLAB R2009a is adopted to implement MADM modeling experiments. Moreover, the flooding attacks and normal table are exported into MATLAB workspace using Database toolbox in MATLAB.

To train MADM model, four functions are made: the first function is used to segment the transported data into groups of fragments with predefined fixed lengths (10, 50, and 150). This is because of their high stability as it is mathematically proved in [2]. We select n as 150 where the corresponding mean and standard deviation. The second function is applied to calculate the covariance matrices of each group and then store them in multiple dimensional matrices. The third function is used to calculate the threshold matrix of calculated covariance matrices. In addition, the fourth function is used to calculate the average of training covariance matrix groups.

To verify the model, two functions are made. First, detection function is used to compare every covariance matrix in testing dataset with all classes' averages and find the degree of deviation from their threshold matrix. Second, a detection presentation function is used to calculate the average of all the detection result matrices as 0-1 matrix and produce the final detection result. See the class diagram in Figure 1.

4.4. Flooding Attacks Implementation in Private Cloud Environment. To simulate the DDoS attack in the cloud environment, refreshthis website [8] has been adopted as in [1]. Moreover, three virtual machines have been employed to carry out the flooding attacks; in every virtual machine, 20 pages with 20 tapes in each were opened. In each tap, the PageRebooter (2009) is used to refresh the victim cloud-based web server 200 times per minute. The normal and

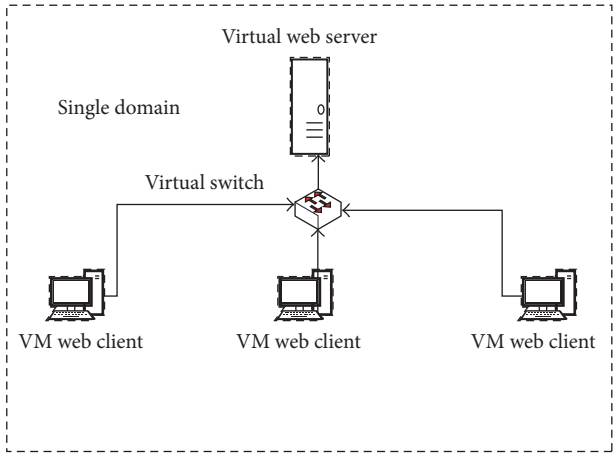

Figure 2: Cloud internal flooding attacks network architecture.

attacks traffic have been collected by using Wireshark tool [16].

4.5. Flooding Attacks Seniors in Private Cloud Environment. Flooding attacks in cloud environment have two forms: the attacker can be true cloud user which means the attacker and victim virtual machines are in the same subnet (internal attack) as it was presented in Figure 2. Or the attacker can be illegitimate cloud user which means the attacker and victim virtual machines are in different subnet (external attack), as it was shown in Figure 3.

4.6. Cloud Experiments Assumptions. In these experiments, it is presumably that the attacker that has a control of one or more than one VMs. And the attacker and the victim are at the same cloud network. Moreover, the attacker is able to find the location of the victim machine. And both of them can be hosted in the same network. In addition, attacker also is able to access to the private cloud from any location in Internet.

5. MADM Performance Results in Cloud Environment

5.1. Overview. In order to evaluate MADM performance, the confusing matrix and ROC measurements were used; more details are in Sections 3.8.1 and 3.9.

5.2. Dataset Description. In these experiments, the summed up statistical features of the TCP/IP conversion statistics are used as it has been pointed out in Section 4.2.

The normal and flooding attack data are captured using the Wireshark tools and web refresher is used to generate the flooding attack traffic. In addition, the number of the captured packets which are used in the training and testing phases is nearly the same as 10% of KDD cup 99 dataset samples number [17]. Furthermore, the sequence lengths are 10, 50, and 150. And cov. len. 10 means covariance matrix of 10 records, cov. len. 50 equals 50 records, and cov. len. 150 equals 150 records. See Tables 2, 3, 4, and 5.

Table 2: Cloud dataset description.

Class type	Total	Training	Testing
Normal	97278 packets	58366 packets	38911 packets
HTTP attacks	107201 packets	64321 packets	42880 packets

Table 3: Number of samples in cov. len. 10.

	Cov. len. 10	
	Training	Testing
Normal	5836 packets	3891 packets
HTTP attack	6432 packets	4288 packets

Table 4: Number of samples in cov. len. 50.

	Cov. len. 50	
	Training	Testing
Normal	1167 packets	778 packets
HTTP attack	1286 packets	4288 packets

Table 5: Number of samples in cov. len. 150.

	Cov. len. 150	
	Training	Testing
Normal	389 packets	259 packets
HTTP attack	428 packets	285 packets

5.3. Experiments Result

5.3.1. MADM Performance Results Summaries and Comparisons in Cloud Environment

(1) MADM Performance Results in Cloud Environment Using 3D Threshold. By looking at Table 6 and Figure 4 which compares MADM performances in the internal and external cloud topologies with covariance matrices categories 10, 50, and 150 using 3D threshold, it can be concluded that MADM performance in internal cloud topologies is better than its performance in external cloud topology. In the internal topology, MADM performance results were as follows.

The detection rate equals 84.25%, classification accuracy rate equals 86.85%, the false positive rate equals 10.99, false negative rate equals 15.75, classification error rate equals 13.15, and AUC equals 84.44. In the external topology, MADM performance results were as follows: The detection rate equals 77.77%, classification precising rate equals 79.85%, the false positive rate equals 19.26, false negative rate equals 21.88, classification error rate equals 20.15, and AUC equals 82.43.

(2) MADM Performance Results in Cloud Environment Using 4D Threshold. By looking at Table 7 and Figure 5 which compares MADM performances in the internal and external cloud topologies with covariance matrices categories 10, 50, and 150 using 3D threshold, it can be concluded that MADM performance in internal cloud topologies is better than its performance in external cloud topology.

In the internal topology, MADM performance results were as follows.

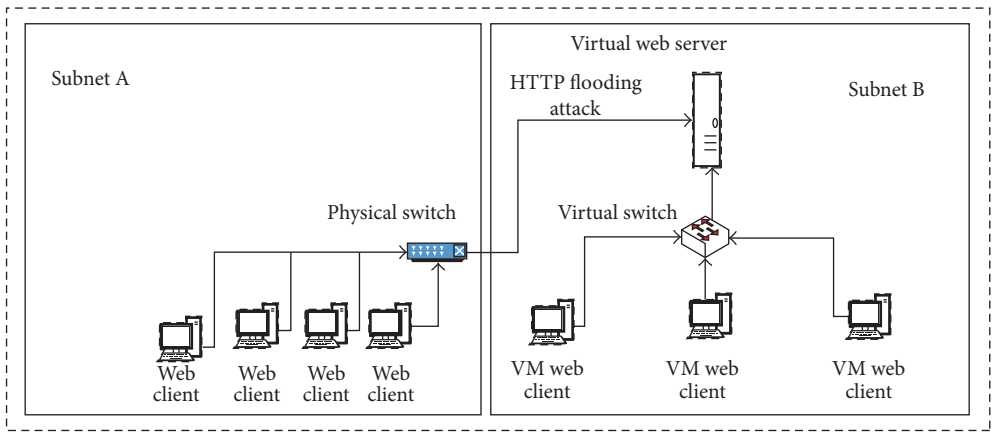

Figure 3: Cloud external flooding attacks network architecture.

Table 6: MADM performance results in cloud environment using 3D threshold.

	Internal cloud topology	External cloud topology
Detection rate	84.25%	77.77%
Classification accuracy rate	86.85%	79.85%
False positive rate	10.99%	19.26%
False negative rate	15.75%	21.88%
Classification error rate	13.15%	20.15%
AUC	84.44%	82.43%

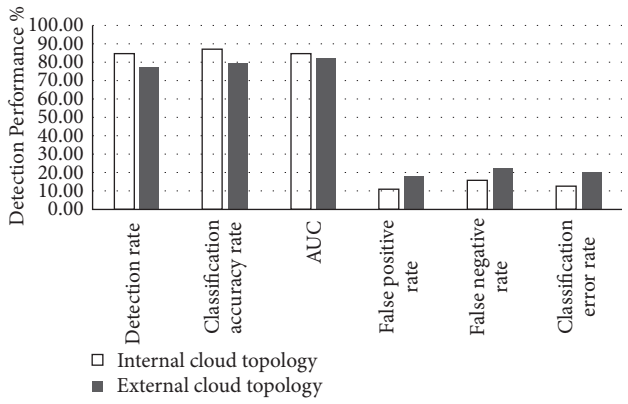

Figure 4: MADM performance results in cloud environment using 3D threshold.

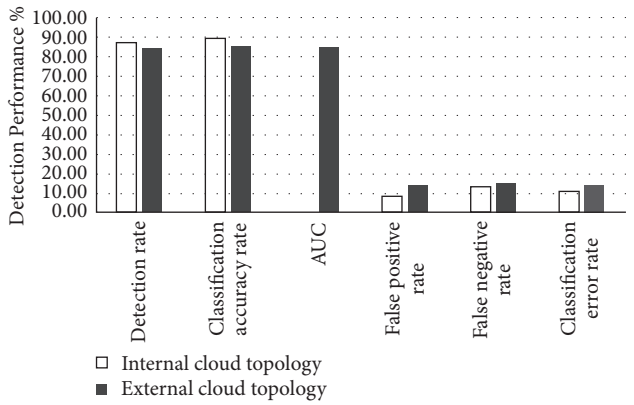

Figure 5: MADM performance results in cloud environment using 4D threshold.

The detection rate equals 86.77%, classification precising rate equals 89.23%, false positive rate equals 8.72, false negative rate equals 13.23, classification error rate equals 10.77, and AUC equals 86.32.

In the external topology, MADM performance results were as follows: The detection rate equals 84.52%, classification accuracy rate equals 85.63%, false positive rate equals 14.06, false negative rate equals 15.48, classification error rate equals 14.37, and AUC equals 84.63.

5.4. MADM Performance Result Analysis and Discussion. In this section, the applicability of using MADM in private cloud computing environment is discussed. MADM has archived an acceptable performance in this environment as shown in Tables 6 and 7. And the performance of MADM in the internal cloud topology was better than the external one because the traffic captured in internal cloud topology belongs to the same LAN, which means that there is no WAN traffic involved. But the traffic captured in external cloud topology belongs to the same network and the other WAN networks. Moreover, the threshold matrix played the main role in differentiation between several kinds of attacks. Therefore, maximizing the MADM threshold values gives better performance as compared with low values. By looking at the MADM performance results using 3D and 4D, it can be concluded that, by using 4D, MADM performance was

Table 7: MADM performance results in cloud environment using 4D threshold.

	Internal cloud topology	External cloud topology
Detection rate	86.77%	84.52%
Classification accuracy rate	89.23%	85.63%
False positive rate	8.72%	14.06%
False negative rate	13.23%	15.48%
Classification error rate	10.77%	14.37%
AUC	86.32%	84.63%

high as compared with 3D. These results further support the results mentioned in [2]. Another important finding was that MADM performance in private cloud computing environment is lower than its performance by using KDD dataset [16]. It seems possible that these results are because KDD dataset was obtained under high controlled circumstances but this study experiments were not.

6. Conclusion

This research presents a new application of MADM in cloud computing environment. The methodology of applying MADM in the cloud is described above. The experiments were conducted using real private testbed. The result of this study has shown high performance of MADM in detecting the HTTP-flooding attacks in the cloud environment based on the confusing matrices and AUC results. And it has been concluded that MADM performance using 4 thresholds is higher as compared with using 3 thresholds. This is because the threshold matrix plays the main role in distinguishing several kinds of attacks. So, maximizing the MADM threshold values gives better performance as compared with low threshold values. From this research, it can be pointed out that MADM approach is a powerful detection method. And it can be implemented in the cloud environment whereby it gives encouraging detection results.

Competing Interests

The authors declare that they have no competing interests.

References

[1] A. Chonka, Y. Xiang, W. Zhou, and A. Bonti, "Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1097–1107, 2011.

[2] D. S. Yeung, S. Jin, and X. Wang, "Covariance-matrix modeling and detecting various flooding attacks," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 37, no. 2, pp. 157–169, 2007.

[3] S. Alsowail, M. H. Sqalli, M. Abu-Amara, Z. Baig, and K. Salah, "An experimental evaluation of the EDoS-shield mitigation technique for securing the cloud," *Arabian Journal for Science and Engineering*, vol. 41, no. 12, pp. 5037–5047, 2016.

[4] M. N. Kumar, P. Sujatha, V. Kalva, R. Nagori, A. K. Katukojwala, and M. Kumar, "Mitigating economic denial of sustainability (EDoS) in cloud computing using in-cloud scrubber service," in *Proceedings of the 4th International Conference on Computational Intelligence and Communication Networks (CICN '12)*, pp. 535–539, Uttar Pradesh, India, November 2012.

[5] I. M. Mary, P. Kavitha, M. Priyadharshini, and V. S. Ramana, "Secure cloud computing environment against ddos and edos attacks," 2014.

[6] C. N. Modi and D. Patel, "A novel hybrid-network intrusion detection system (H-NIDS) in cloud computing," in *Proceedings of the IEEE Symposium on Computational Intelligence in Cyber Security (CICS '13)*, pp. 23–30, April 2013.

[7] I. Gul and M. Hussain, "Distributed cloud intrusion detection model," *International Journal of Advanced Science and Technology*, vol. 34, no. 38, p. 135, 2011.

[8] Refreshthis, http://www.refreshthis.com/.

[9] P. E. N. Lutu, *Dataset selection for aggregate model implementation in predictive data mining [Ph.D. thesis]*, University of Pretoria, 2010.

[10] F. Provost and T. Fawcett, "Robust classification for imprecise environments," *Machine Learning*, vol. 42, no. 3, pp. 203–231, 2001.

[11] J. Li and J. P. Fine, "ROC analysis with multiple classes and multiple tests: methodology and its application in microarray studies," *Biostatistics*, vol. 9, no. 3, pp. 566–576, 2008.

[12] C. Ferri, J. Hernández-Orallo, and M. A. Salido, "Volume under the ROC surface for multi-class problems," in *Proceedings of the 14th European Conference on Machine Learning*, pp. 108–120, Dubrovnik, Croatia, September 2003.

[13] Y.-J. Wu and C.-T. Chiang, *Roc Representation for the Discriminability of Multi-Classifcation Markers*, Department of Mathematics, National Taiwan University, 2011.

[14] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[15] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[16] wireshark. wireshark, https://www.wireshark.org/.

[17] K. Cup, 2007, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.