Hybrid Intrusion Detection System Using Machine Learning Techniques in Cloud Computing Environments

Ibraheem Aljamal
SUNY Polytechnic Institute
Computer Science
ibrahim.aljamal@sunypoly.edu

Ali Tekeoğlu
SUNY Polytechnic Institute
Network Computer Security
ali.tekeoglu@sunypoly.edu

Korkut Bekiroglu
SUNY Polytechnic Institute
Electrical Engineering
korkut.bekiroglu@sunypoly.edu

Saumendra Sengupta
SUNY Polytechnic Institute
Computer Science
sengupta@sunypoly.edu

***Abstract*—Intrusion detection is one essential tool towards building secure and trustworthy Cloud computing environment, given the ubiquitous presence of cyber attacks that proliferate rapidly and morph dynamically. In our current working paradigm of resource, platform and service consolidations, Cloud Computing provides a significant improvement in the cost metrics via dynamic provisioning of IT services. Since almost all cloud computing networks lean on providing their services through Internet, they are prone to experience variety of security issues. Therefore, in cloud environments, it is necessary to deploy an Intrusion Detection System (IDS) to detect new and unknown attacks in addition to signature based known attacks, with high accuracy. In our deliberation we assume that a system or a network *"anomalous"* event is synonymous to an *"intrusion"* event when there is a significant departure in one or more underlying system or network activities. There are couple of recently proposed ideas that aim to develop a *hybrid* detection mechanism, combining advantages of signature-based detection schemes with the ability to detect unknown attacks based on anomalies. In this work, we propose a network based anomaly detection system at the Cloud Hypervisor level that utilizes a hybrid algorithm: a combination of K-means clustering algorithm and SVM classification algorithm, to improve the accuracy of the anomaly detection system. Dataset from UNSW-NB15 study is used to evaluate the proposed approach and results are compared with previous studies. The accuracy for our proposed K-means clustering model is slightly higher than others. However, the accuracy we obtained from the SVM model is still low for supervised techniques.**

***Index Terms*—Cloud Intrusion Detection, Anomaly Detection**

I. Introduction

The exponential growth in modern technology has produced a ubiquitous global networking systems of services and communications along with its attendant problems. Given this state of affair and the cost-saving paradigm of providing resources locally, companies around the globe are increasingly leaning to providing their services and resources for users through Cloud Computing networks, which, in turn pushed the security risks to new heights. Eventually, cybersecurity became a major issue for Cloud Computing. While traditional, signature based security systems could be deployed for cloud based systems, in reality, diverse and swiftly developing malicious threats are challenging this traditional intrusion detection systems, especially in Cloud Computing environment. Today, research efforts towards the development of intrusion detection systems targeted for cloud computing networks, though still at its infancy, have started. To help improve state-of-the-art in intrusion detection in the Cloud, we propose a hybrid system. In brief, the aim of this study is to build a hybrid intrusion detection system that could face the known and novel security attacks in cloud computing networks.

Due to the complexity and size of distributed infrastructure of cloud computing networks, modern and sophisticated technologies are needed to analyze the huge amount of the data generated from the transactions between devices of cloud computing systems. Developing robust intrusion detection systems, requires analyzing the generated complex data in real-time. Therefore, machine learning techniques, such as classification and clustering, are promising for analyzing big data consisting of traffic flows in cloud networks for cues of intrusions, even if they have signatures that are not encountered before.

In this study, simply an adaptive hybrid network-based intrusion detection model to find abnormal network behavior in a cloud environment is proposed. The main contributions of this project are as follows:

- A general survey on the current studies in anomaly-detection problem is performed.
- A hybrid machine learning model for efficient detection of network intrusions to overcome existing shortcomings of intrusion detection systems is proposed.

II. Background & Related Work

Since the early days of cloud computing, intrusion detection systems (IDS) played a crucial role in ensuring security of the network. In the current literature, there are various proposed Network Intrusion Detection Systems (NIDS) for monitoring network traffic flow to identify attacks in such settings. In general, two fundamental approaches are used for IDS; (i) *Signature based* and (ii) *Anomaly based* detection [1], [2].

In the first approach, each previously detected known attack has a unique signature, based on the network flow characteristics, patterns of bytes in the network packets, etc... [3], [4]. The signature for the known attack is compared to the current flow

978-1-7281-0798-1/19/$31.00 ©2019 IEEE
SERA 2019, May 29-31, 2019, Honolulu, Hawaii

for a match, which would result in detection if there is a match. The signature-based approaches have high accuracy and low false report rate. However, difficulty and overhead of building and maintaining an up-to-date malicious signature database for each attack is one of the disadvantages in signature-based approaches. Signature-based approach IDSs are not able to detect unknown attacks. Consequently, the variation of network applications is making this method difficult if not impossible [5].

Machine learning has two main categories of algorithms; supervised learning and unsupervised learning. Supervised Learning algorithms are used to build intrusion detection models with signature-based approach, where previously known attack signatures are used as training (learning) data. Support vector machines, linear regression, logistic regression, Naive Bayes, linear discriminant analysis, decision trees, and neural networks (Multilayer Perception) are the most widely used supervised learning algorithms to build signature-based intrusion detection systems [6]. These supervised machine learning techniques adjust their behavior (learn) according to the known signatures, also called label or event classifier, to achieve the detection. The accuracy of signature-based machine learning IDS depends on the quality of training dataset, and the parameters used for supervised machine learning technique.

The second type of IDS is anomaly-based detection. In this approach, a model formed for normal activities based on the normal behavior of the network, and any deviation from this normal behavior is considered an attack [3]. The main advantage of anomaly-based approach over the signature-based approach is the ability to detect new and unexpected attacks [7]. However, it has some shortages where its accuracy is considered relatively low compared to signature based approach. Also, it has high false alarm rate in cases where lack of a training data that covers all kinds of behaviours, including anomalous as well as normal behavior. More and more research is being published on utilization of unsupervised machine learning techniques to detect anomalous behavior. Clustering (k-means, fuzzy c-means), neural networks(deep neural nets, self-organizing maps) are the most widely used for anomaly-based intrusion detection systems [8] [9]. Unsupervised machine learning techniques build their learned model according to the relation, consistency, or continuity between the events. For the most part, the huge amount of generated data from cloud computing networks makes the labeling method very expensive. In that case, the unsupervised learning approach is a better solution.

III. Hybrid-Detection Model

In this section, we will present details of the proposed hybrid detection model as well as the developed techniques. In Intrusion Detection systems, hybrid approaches perform better than the usual methods. Therefore this study focuses on analyzing the contextual relationship between data flow in the network using hybrid approaches. In more detail, the focus lies more on using the K-means clustering algorithm to produce labels automatically and uses SVM to build learning model

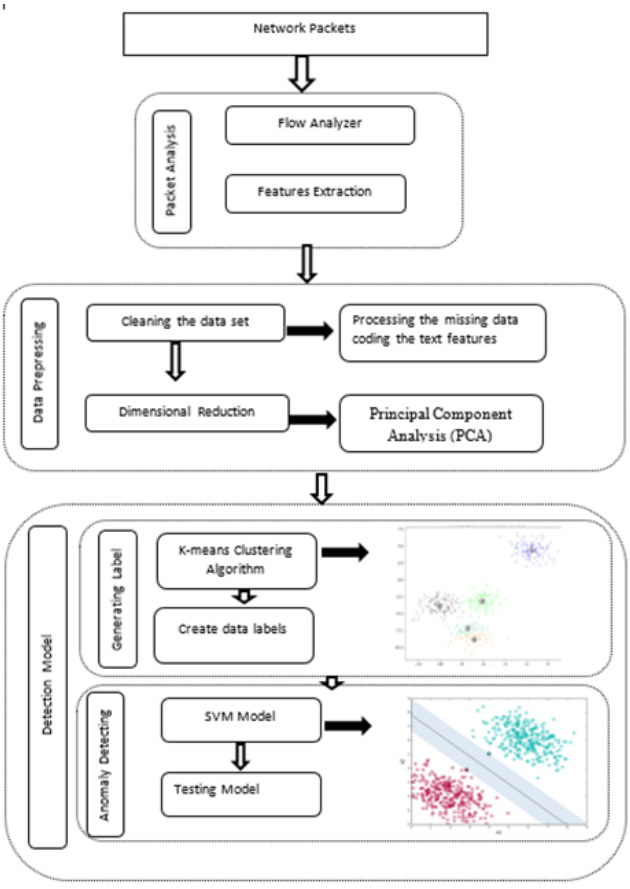

Fig. 1. Proposed Method Framework

that could be used to evaluate the new data questions. This study is diverged from the others by using multiple parameters to improve the outcome of the clustering algorithm and feature extraction methods to select only the more useful features. Figure 1 shows the proposed framework for this study. False positives rates in anomaly detection for intrusion detection systems tend to be high for the following reasons; *(i)* Amount of data samples for training the model are usually not enough to make it accurate for deploying into production environment, *(ii)* Even slightly higher false positive rate in intrusion detection systems can have overwhelmingly negative effects on the system performance in a production environment, *(iii)* Definition of anomaly can be different from organization to organization, which makes it harder to define a global standard for normal or abnormal traffic. *(iv)* Diversity of network applications creates high amount of variation in network traffic. Having these issues in mind, a new hybrid detection system is proposed for network-based anomaly detection in a cloud computing network.

A. Traffic Analysis

The pre-processing task for intrusion detection is necessary to improve accuracy. Thus, traffic analysis contains two crucial concepts; *(i)* network flow analysis, *(ii)* traffic classification.

Network flow is a group of network packets between two end points that are captured at the hypervisor level. To perform

the flow analysis, we use network monitoring and management tools. In this study, we used a free and open source packet analyzer called Wireshark to generate TCP dump files. Dump files are a group of packets that flow through a connection between the nodes in the network. In UNSW-NB15 dataset, there are forty-seven features extracted from TCP dump files. After monitoring the network flow and extracting the features, the output data set will contain some missing data. Then we primarily focused on removing noisy and inconsistent data.

B. Dimensionality Reduction

Some features may not help the detection process and may increase the detection process complexity. So, the removal of features that have a small variance might even lead to better results since these dimensions consist mostly of noise and would, therefore, reduce the weight of features that contain more information [10]. We use principal component analysis (PCA) to remove these features. Therefore, for each instance, only a subset of the components of the feature vector will be used as input to the detection algorithm. Based on exploratory experiments on training data-set from UNSW-NB15, we have decided to retain (as cutoff) the first ten features of sixteen obtained by PCA. In brief, we reduce the number of features from *forty-seven* to *ten* using PCA.

C. Clustering and Labeling the Data-set

The proposed detection module consists of two models. The first model is an unsupervised clustering model to create relevant data classifiers for the whole data-set. Then, the output of the first model will be fed as input for the supervised model, SVM to build our anomaly detection model, as shown in Figure 1.

It is evident that there is no data label in the network packets. We employed a K-Means Clustering algorithm to find any abnormal behavior and label it as an *attack*, while labeling all the normal behavior network packets as *normal*. We use 1 for normal behaviors and -1 for abnormal behaviors.

In this study, K-Means Clustering algorithm is utilized to cluster the data-set. Baek [8] and others build their anomaly detection based on data clustering relying on the following assumption: "Normal data instances belong to large and dense clusters while anomalies either belong to small or sparse clusters" [7]. In this study, we change the first assumption bit where we don't consider the huge sparse cluster normal or abnormal cluster. Instead, we take this cluster and feed it to the clustering algorithm again to create smaller clusters, than apply the previous assumption again. Due to the previous assumption for network anomaly detection, the first step we need to think of are threshold functions that can be used to determine the "*large* or *small*" for size and "*dense* or *sparse*" for density to characterize clusters. In this case, we need three threshold functions that depend on three parameters, *(i)* small or not, *(ii)* large or not, and *(iii)* sparse or not. Next, we will define the three threshold functions used to determine the size and density of the clusters.

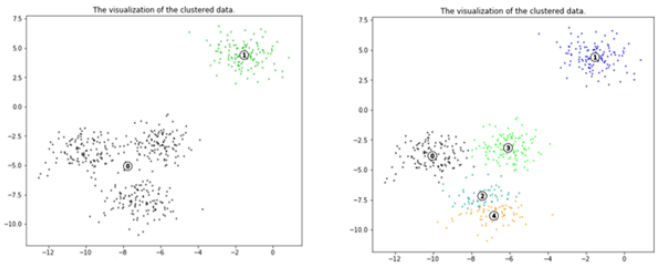

Fig. 2. A large cluster that could be re-clustered to four clusters

Definition 1: **Small clusters threshold function:** Let's assume that we have N data instances in the entire data-set and K clusters in the cluster set $C = \{C_1, C_2, C_3, ..., C_k\}$. The average size of any cluster is calculated as N/K. Thus, the cluster is small if its size is $C_i = \alpha \times n/k$. Where the C_i is the number of data instances in the cluster i and $0 \leq \alpha \leq 1$.

Definition 2: **Large clusters threshold function:** Only difference between previous definition is, the cluster is larger if its size is $C_i = \beta \times n/k$ and $0 \leq \beta \leq 1$.

In Figure 2, it can be seen that the cluster 0, located on the lower left corner, is a very large cluster. After re-clustering, this cluster is divided into four small clusters.

Definition 3: **The density threshold function of a cluster:** This function is used to determine whether a cluster is sparse or not. Because of using k-means clustering algorithm, each cluster has a centroid position. Let's assume that $\mathcal{O}$ is the centroid position, d_i is a data point in the cluster, and M is the number of data points in that cluster. The average sum of squares (E) is defined as, $E = 1/M \sum_{i=0}^{M} (\mathcal{O} - d_i)^2$. As a result, if the value of E is small that means the density is high, and vice-versa. At this point, we can define another parameter γ to define the sparsity level of a cluster. So, if $E_i > \gamma \times median(E)$, we assume that cluster i is sparse [8].

After defining the three threshold functions, we use k-means algorithm to create the clusters, then apply the threshold function on resulting clusters. Consequently, we add a new feature called *label*. We go through the small and sparse clusters and assign -1 to label for normal behavior, and the rest of data instances with label value 1 for abnormal behavior.

D. Detection Model

Since supervised learning techniques are faster in the testing mode than the unsupervised ones, we chose support vector machine (SVM), one of the supervised techniques, to be our detection model. We feed our data-set with the new label to SVM with the polynomial kernel to create a learned model. Then, the learned model will be used for any further anomaly detection with the new network traffic.

IV. Evaluation of Hybrid-Detection Model

In this section, evaluation results of our proposed intrusion detection model are presented. First, we explain the techniques used to evaluate the results from the proposed anomaly detection model, and then we compare the results of the model developed in this work to several other anomaly detection

models in the literature. We conduct the experiments in two phases. In the first phase, we focus on the clustering module and provide a thorough analysis of the clustering module using UNSW-NB15 [1] data set. In the second phase, we use the output data from the previous phase to evaluate the classification module using SVM and provide a comprehensive analysis of the SVM module.

A. Data Normalization and Features Selection

To evaluate the proposed model, UNSW-NB15 data-set was used with some of the normalization methods that help to increase the overall accuracy. As it was mentioned before, the UNSW-NB15 data-set has a number of files but we used two of them in this study, *(i)* training and *(ii)* testing data files. Each of these files has forty-nine features. Two of them are label feature. The training data set is a CSV file containing 82332 instances. And testing file, contains 175341 instances.

1) Data Set Normalization: The testing (CSV) file, contains 175341 instances. In both files, types of features are different, their values are sparse, and there are some missing values. Therefore, we suggest removing the non-numerical features. Four features were removed; *(i)* transaction protocol, *(ii)* state, *(iii)* service, and *(iv)* attack type. We normalize the rest of the features using the following formula, called *decimal scale normalization*:

$$\forall X_i^{`} = \frac{X_i}{10^m}$$

Where m is the number of digits of the maximum value in a vector, and X_i is each data point's value. In simple words, we divide the value of each instance in the data set, by its feature's mean value.

2) Dimensional Reduction: The output of the last step is the normalized data set. However, we still have a data set with forty-four features. Some of these features contribute nothing to the detection model. We suggest, using principal component analysis (PCA) to perform the dimensional reduction. In order to reduce data set from `N-dimensions` to `K-dimensions`, we need to calculate the co-variance matrix by using the following formula:

$$COV = \frac{1}{m-1}\sum_{i=0}^{n-1}(X^i)(X^i)^T$$

Where X^i is a matrix from the original data set, $(X^i)^T$ is its transpose. Then, we calculate the eigenvector for the covariance matrix by using the following formula:

$$[U\ S\ V] = SVD(COV)$$

Where SVD is singular value decomposition function that returns three different matrices; U, S, and V. Then we use U matrix, PCA base.

In the result, `U` is reduced matrix of our data set with reduced features. So, if we need to use k-dimensions, we just took the features from 0 to K from the matrix U. In this work, we use `KNIME` platform to perform the data pre-processing.

The output of the PCA is 36 reduced features. We chose 25 of 36 features for the output of PCA because we built a supervised model to evaluate the features. The model was built using *random forest* learning model and predictor. We made 25% of the data as testing data and train the model with 75% of the data. First, we trained the model with 5 features, and then we increase the number of features by 5 in each succeeding training session.

Number of Features	5	10	15	20	25	30	36
Accuracy	0.871	0.940	0.948	0.95	0.965	0.967	0.968

TABLE I
ACCURACY RESULTS BY NUMBER OF FEATURES

We calculated the accuracy of the model using the testing data set for each evaluation session. The experiment result could be seen in the Table-I. The first 25 features got almost the same result as the evaluation with all features, as seen in Table-I. Thus, in further processes, we use the first twenty-five features to build our detection model. After we finished the data pre-processing, the new data set is ready for the first phase of our proposed model. It will be stored in two files. The first file contains the *training* dataset with dimensional reduced features. The second contains the *testing* data set with dimensional reduced features.

B. Clustering Module

The purpose of this phase of evaluation is to measure the impact of applying the k-means clustering result that is used to create relevant labels on the data set. Before we start measuring the result of the clustering algorithm, we describe the most critical parameters that we adjust in our experiments. (α, λ, β) are the parameters used in our detection model to create the relevant data set labels. The values of these parameters should be between 0 and 1. Specifically, the value of the parameter α should be close to 0 to select the small clusters in the result of the k-mean. Also, the value of the parameter λ should be close to 1 to select the large clusters in the result of the k-mean. The last parameter β should be close to 1 to select the sparsest clusters in the result of the k-mean. The number of the clusters is the generic parameter for the k-mean that we need to adjust to evaluate our model. For the experiments, we apply different values for each of these parameters; $0.1 \leq \alpha \leq 0.2$, $0.85 \leq \lambda \leq 0.99$ and $0.85 \leq \beta \leq 0.99$. We use these values in the same model in different nodes and compare the result of each model node trying to choose the best values to these parameters.

We choose the range of these values based on some previous studies and our analysis. For instance, the value of the parameter β is taken for the study proposed by Baek in [8]. Potential values (10, 16, 32, 45, 64) for numbers of clusters were chosen as follows. Since UNSW-NB15 dataset contains nine different attacks and normal activities, we suggest clustering the data set to 10 clusters hoping that algorithm will put each different attack in separate clusters and normal activities in one cluster. Study done by Baek [8] used 16, 32, and 64 as number of

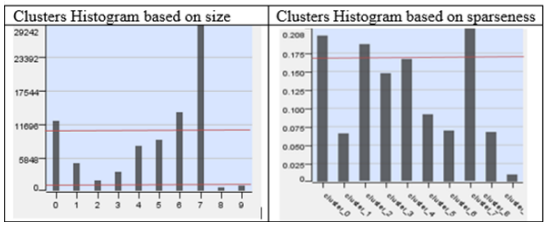

Fig. 3. Model 1 histogram

clusters. The value 45 was chosen based on the number of the instances in the data set and how many instances were taken per second.

We create five different models using the K-means algorithm. The difference between these models is the number of the cluster while the other parameters values are the same in all the models. The first model clusters the data into 10 clusters, the second model clusters the data set into 16 clusters, the third model clusters the data set into 32 clusters, the fourth model clusters the data set into 45 clusters, and the last model clusters the data set into 64 clusters. Inside each model, there is a k-means algorithm with two histogram nodes to visualize the model cluster; one based on the number of the instances in each cluster and the other histogram based on the sparseness of the clusters.

We use KNIME, a Java platform for machine learning applications, and Python scripting language to implement our model. The KNIME provides a visualization of our model while python provides code to implement the model. We have two Java Snippet nodes in each model in KNIME. The first one is for extracting the clusters numbers from the feature Cluster. The second one where we put our python code decides whether the cluster is consisting of normal or abnormal activities. Scorer node computes the accuracy of the model. Each model has two outputs; one for the confusion matrix and accuracy, another output is for the data set with the new label. We run all the different models and compare the result with an actual label that came with the data set and report the results next.

1) Model One with Ten Clusters: This model clusters the dataset into ten different clusters expecting that the model will cluster the data set into nine attack clusters (since there are nine different attacks in the data set used to evaluate our model) and one cluster for the normal behaviors in the network. The accuracy of this model is 0.861, which is a little bit low accuracy compared to other models. The values of the parameters are 0.12 for α, 0.98 for λ, and 2 for β.

Figure 3 shows two histograms for the data set based on the clusters size and how far are the cluster points from the center point. We see two red lines that point out the range for normal activities. In other words, in the first histogram, any cluster under or above the red line will be considered abnormal and clusters between the two lines will be considered as normal. The value of the first is 1029 which means any cluster with 1029 or less points will be under this line. The second line value is at 8068 which means any cluster that has more than 8068 will be considered above the line. In the second histogram, if the average of the distances between the cluster points and center cluster point above the red line, it will be considered an abnormal cluster. Otherwise, it will be considered normal.

2) Model Two with 16 Clusters: In this model, cluster size is set to sixteen; the model will cluster the data set into some small and big attack clusters and some clusters for the normal action in the network. The accuracy of this model is 0.855, which is a lower than first model. The values of the parameters are 0.12 for α, 0.98 for λ, and 2 for β.

3) Model Three with 32 Clusters: The third model clusters the data into thirty-two different clusters expecting that the model will cluster the data set into smaller clusters and multiple clusters for the normal action in the network. The accuracy of this model is 0.801, which is a very low accuracy compared to two previous models. The values of the parameters are 0.12 for α, 0.98 for λ, and 2 for β.

4) Model Four with 45 Clusters: Fourth model clusters the data into forty-five different clusters expecting that the model will discover some relations between the data set instances based on the period of time. However, it creates smaller clusters and larger cluster, but the value of the boundaries will change too. The greatest accuracy for this model is 0.87, which is still lower than the accuracy results in [8]. The values of the parameters are 0.12 for α, 0.98 for λ, and 2 for β.

5) Model Five with 64 Clusters: Final model clusters the data into sixty-four different clusters trying to separate the data set even more. From the previous model, while the cluster number is increased, the accuracy is getting better. Thus, it creates smaller clusters and keeps some larger clusters. However, the value of the boundaries will be smaller than before. The greatest accuracy for this model is 0.886, which is the highest accuracy we got, and it is a little bit higher than the result in [8]. The values of the parameters are 0.12 for α, 0.98 for λ, and 2 for β.

The results show that model that clusters data set into 64 clusters has highest accuracy. Table II shows the overall accuracy for all the models. However, if we compare the highest result to others such as [8], we observe that we got a little higher.

Model	1	2	3	4	5
Accuracy	0.86	0.85	0.802	0.87	0.886

TABLE II
ACCURACY RESULTS FOR THE FIVE MODELS

At this point, we export the new labeled data set to a new CSV file and use it as a training data set for `SVM` model, which is our detection model in the final stage.

C. Anomaly Detection Model

After getting the labeled data set from the previous phase, we trained the SVM model using this labeled data set. Then, we use the trained model as our detection model for any new coming data. As mentioned, the SVM algorithm need a data set that has events and a classifier for these events. The classifier, label, is usually binary classifier. In our case, it is binary classifier with two values, `1` for normal activities and

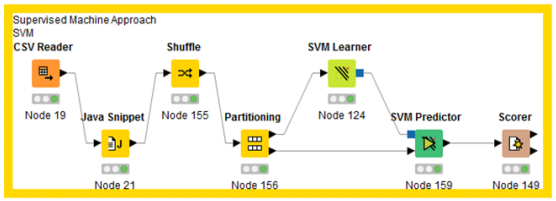

Fig. 4. Detection Model, SVM Model nodes

-1 for abnormal activities. Figure 4 shows the SVM model nodes. The accuracy of the SVM outcome depends on the data and the kernel that is used to train the model. We tried different models aiming to choose the best for our data based on the final accuracy. We loaded the labeled data set, created by the clustering model, using the CVS reader node. Then, to test our model, we need to have a testing data set. Thus, we use 85% of our data set as training set and 15% as the testing set. The accuracy we got from our detection model is 0.867. It is little bit low, but we believe that it could do better in the real-life systems because the considerable variance between the data instance in the online data sets is not as same as in the real-life systems.

D. Overall Performance

By multiplying the clustering model accuracy and the SVM model accuracy, we got the overall accuracy of the hybrid detection system, which is 0.77. It is a low accuracy comparing to others. But when using the test data set, which contains 175341 data instances, to test our model, the final accuracy for SVM model was 0.843 which is excellent accuracy for a hybrid model. In the final analysis, the false alarm rate is still very high for such a system like could computing network. For the reason that if an attack goes undetected, that may affect the whole system. Thus, the result of our model still not practical in a procustion system.

E. Comparison

We compare the result of this study with a previous study, detailed in [8]. We were not able to reproduce the result of the compared study because the features that used to build the model is not determined in the study. So, we used their result from their report. However, we change our hyperparameter values to the values used in the compared study and trained our model again. The value is $\alpha = 0.125$, $\gamma = 0.5$, $\beta = 1.0$. The number of the clusters are 16, 32, and 64. Table III shows the accuracy for the two studies.

Cluster#	Accuracy of compared study	Accuracy of this study
16	0.88	0.846
32	0.80	0.84
64	0.78	0.847

TABLE III
ACCURACY RESULTS FOR COMPARISON

When we look at the result in table III, we see that the accuracy in our study slightly changed when a compared study is a big jump. This has happened because, in their study, they classified all point in a cluster that does not pass the threshold function as abnormal cluster while we study each point in the clusters that do not pass the threshold function. The result of their model is better than us with these parameter values. However, we achieved a higher result with out hyper parameter values which are $\alpha = 0.12$, $\gamma = 0.98$, $\beta = 2.0$ which is 0.886.

V. CONCLUSION

A hybrid intrusion detection system, a combination of unsupervised and supervised machine learning techniques to detect unknown and known attacks, in cloud computing environments are presented in this paper. Our hybrid intrusion detection system is a network-based approach that clusters the network flows using K-means algorithm into sixty-four clusters. Then it classifies these clusters into two categories; (i) normal or (ii) abnormal events. The clustering phase allows us to build relevant classifiers for network events. The second phase of this approach takes the network events; at this point, it is called data instances and trains a supervised detection model based on the SVM algorithm. The SVM trained model is our detection model that is used to detect anomalies in new events. UNSW-NB15 is a publicly available, labeled data set, that is used to evaluate our detection system. It is a rich, labeled data set, containing nine different types of attacks. The overall performance for the proposed approach is observed in every phase of our system. The clustering model reaches *0.886* which is slightly higher than other studies, by 0.06. Highest accuracy for the SVM model was *0.847* which is considered an acceptable for a hybrid model. KNIME platform, Python and JavaScript are the components used for the implementation of proposed hybrid intrusion detection system.

REFERENCES

[1] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, Nov 2015, pp. 1–6.

[2] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating computer intrusion detection systems: A survey of common practices," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, p. 12, 2015.

[3] D. M. S. Zekrifa, "Hybrid Intrusion Detection System," Theses, School of Information Technology & Mathematical Sciences, Jun. 2014. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01584217

[4] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, vol. 25, pp. 152–160, 2018.

[5] A. Hushyar, "Network traffic clustering and geographic visualization," *SAN JOS STATE UNIVERSITY*, 2009.

[6] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in non-stationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[8] S. Baek, D. Kwon, J. Kim, S. C. Suh, H. Kim, and I. Kim, "Unsupervised Labeling for Supervised Anomaly Detection in Enterprise and Cloud Networks," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, June 2017, pp. 205–210.

[9] P. S. Rao, R. Mutukuri, and G. P. S. Varma, "Classification of Network Violation Detection Using Machine Learning," *International Journal of Advanced Research in Computer Science*, vol. 9, no. 1, pp. 4 – 9, 2018.

[10] A. Aldribi, I. Traore, and B. Moa, "Hypervisor-Based Cloud Intrusion Detection through Online Multivariate Statistical Change Tracking," *IEEE Transactions on Information Forensics and Security*, 2018.