An Enhanced Intrusion Detection Framework for Securing Network Layer of Cloud Computing

Kamatchi Arjunan
Computer Science and Engineering Department
National Institute of Technology Goa, India
Email: kamatchinitg@gmail.com

Chirag N. Modi
Computer Science and Engineering Department
National Institute of Technology Goa, India
Email: cnmodi@nitgoa.ac.in

***Abstract*—Virtualization is a key enabler of Cloud computing. Due to the numerous vulnerabilities in current implementations of virtualization, security is the major concern of Cloud computing. In this paper, we propose an enhanced security framework to detect intrusions at the virtual network layer of Cloud. It combines signature and anomaly based techniques to detect possible attacks. It uses different classifiers viz; naive bayes, decision tree, random forest, extra trees and linear discriminant analysis for an efficient and effective detection of intrusions. To detect distributed attacks at each cluster and at whole Cloud, it collects intrusion evidences from each region of Cloud and applies Dempster-Shafer theory (DST) for final decision making. We analyze the proposed security framework in terms of Cloud IDS requirements through *offline* simulation using different intrusion datasets.**

keywords: Cloud computing, Virtual network, Security, Intrusion detection system, Classifier, Dempster-Shafer theory

I. Introduction

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. It has mainly two ends viz; *front end* and *back end* (refer Fig. 1). *Front end* offers communication with Cloud and the services. The hardware and software resources process the user's queries and deliver the offered services. A host system supports hypervisor or virtual machine monitor (VMM) to run multiple virtual machines (VMs) on same physical resources. VMM enables the communication between the VMs and the host machine through virtual switch. Current implementations of the VMM are failed to offer strong isolation among VMs due to the vulnerabilities [2].

Due to numerous vulnerabilities in current implementations of VMM, security is the major concern in Cloud computing. At virtualization layer of Cloud computing, virtual network, VM and hypervisor related vulnerabilities can be considered. Different attacks to Cloud can be broadly classified into traditional and virtualization specific attacks. Traditional attacks include flooding, distributed denial of service (DDoS), user to root, port scanning and backdoor channel attacks etc. In this case, an attacker attempts to interrupt the services

978-1-5090-5942-3/17/$31.00

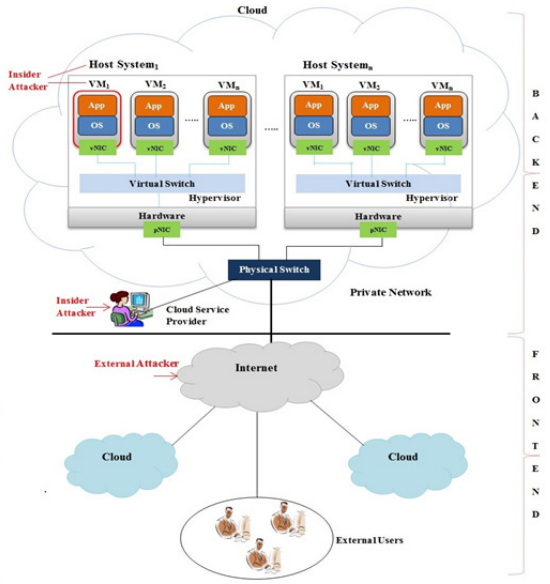

Fig. 1. A common architecture of Cloud computing and threat model [3]

provided to the Cloud users. Virtualization specific attacks can be performed by both external and internal attackers. External attackers (outside the Cloud network) often perform attacks like footprinting, virtual code injection and virtualized botnets on hypervisors or VMs. Insider may be Cloud users, clients at provider side or Cloud provider itself. They can perform attacks like VM escape, side channel, VM to VM, VM monitoring from the host system and communication between VMs. Insiders are considered as more harmful due to their higher privileges on Cloud resources. Thus, detecting insider's is more challenging than the external attacks in Cloud.

To address network security challenges, firewall could be one of the common solutions. However, it has many limitations like protecting only up to a certain perimeter, unable to defend against internal attackers, vulnerable to IP spoofing attacks, etc. An intrusion detection system (IDS) can be another solution and it has been the most widely used to secure traditional networks. However, due to shared and dynamic nature of Cloud, traditional IDS has additional challenges viz; handling large-scale computing systems, detecting a variety of attacks, fast detection, automated and self adaptive capability,

scalability, synchronization of IDS sensors and resistance against compromise. Hence, still there is a room for further improvements in IDS and addressing the newer requirements.

In this paper, we enhance an intrusion detection system to secure virtual network of Cloud. As like in our recent proposal [3], it attempts to detect both known and unknown attacks. Unlike in [3], it can efficiently detect virtual network layer attacks with the improved accuracy and reduced number of false alerts by incorporating the feasible machine learning techniques viz; naive bayes, decision tree, random forest, extra trees and linear discriminant analysis. As these techniques are considered as light weight and require low computation time, it is feasible to incorporate them for anomaly detection. The output from each classifier is applied to Dempster-Shafer theory (DST) [4] for final decision making about intrusions. In addition, intrusive evidences from each layer of Cloud are correlated to detect distributed attacks. For alert correlation and final decision making about the distributed intrusions, it uses the Dempster-Shafer theory (DST) [4]. This framework attempts to fulfill the Cloud IDS requirements. We validate the proposed security framework on virtual network test bed through offline simulation using different intrusion datasets. In this paper, we consider the detection of only network layer attacks at virtual network and traditional network layer of cloud.

The rest of the paper is organized as follows: Section 2 discusses the existing Cloud IDS approaches and identifies the research gaps. Section 3 presents the proposed security framework, while experimental results and validation of the proposed security framework is given in section 4. Finally, section 5 concludes our research work with references at the end.

II. Cloud Intrusion Detection System: Literature Survey

Intrusion detection systems (IDS) is a security system or a device which can detect any unauthorized activity (from external or internal attackers) which affects the confidentiality, integrity and availability of network or systems. It mainly uses two types of techniques viz; signature based and anomaly based detection. Signature based technique identifies known attacks by utilizing the already defined attack signatures in the signature database. The signature database should be frequently updated for the improved performance. However, it detects only known attacks. Anomaly based detection identifies unknown attacks by learning the behavior of different users. However, it suffers from the high false alerts.

In Cloud, IDS needs to monitor both physical and virtual network traffic. Due to the dynamic nature of VMs (added/removed dynamically), the security requirements should be relevant to the respective VMs. Hence frequent updates to the signature/behavioral database is required. Cloud IDS should be capable of detecting both traditional and virtualization related attacks.

A. Existing Approaches to Cloud Intrusion Detection System

There are many works till date to detect intrusions in Cloud computing.

Dhage *et al.* [6] have proposed a distributed IDPS for Cloud, which combines both signature and anomaly based detection. Here, Cloud service provider (CSP) manages an IDS controller which in turn controlling multiple instances of IDS. Each IDS controller stores the signature and anomaly data of all users in the specific network. Whenever a new user is added to the network, IDS controller creates a new instance of IDS and assigns it to the user's VM. It monitors the activities of the VM user and updates the IDS controller. Hence, the IDS controller receives the data from all the IDS instances. These user profiles are later used to detect possible intrusions. However, the management of multiple IDS instances is a difficult task in Cloud.

Chung *et al.* [7] have deployed NIDS at both Dom0 and DomU in a XEN based Cloud server. Here, NIDS monitors the systems and generates alerts when malicious traffic is detected. Then the intrusion alerts are passed to the control center. The control center analyzes the attack and computes its severity by using attack graph. Based on the result, it takes the countermeasures. However, it is not considering the detection of virtualization specific attacks. In addition, it is very difficult to come up with scenario based attack graphs for all cases.

Modi *et al.* [8] have combined signature and anomaly based detection techniques. Here, *snort* [9] offers fast detection of known attack detection and minimizes the number of packets to be analyzed by the anomaly detection system. In anomaly module, they have combined the output from three classifiers viz; Bayesian, decision tree and associative by using a weighted average method. Then the final decision on intrusion is taken based on the computed scores. For distributed attack detection, they have applied majority voting method. It minimizes the false alerts with reduced computation time. However, centralized system may lead to single point of failure.

Singh *et al.* [10] have used *snort*, decision tree and support vector machine to design a collaborative IDS framework for Cloud. *Snort* helps in fast detection of known attacks. Fusion of decision tree and support vector machine is used to detect network anomaly. It creates a new attack signature whenever the threshold of an unknown attack frequency is reached and it is updated to the signature database. To monitor all the VMs, NIDS is deployed at the virtual switch of the host machine. Based on the load on the Cloud clusters, the correlation module is deployed at a cluster node to detect distributed attack. However, correlation module is single point of contact which may fail due to heavy traffic in Cloud.

Toumi *et al.* [11] have presented a three phase cooperative hybrid IDS. In the first phase, possible intrusions at each VM are detected. Using these detected intrusions, intrusions at different layers of Cloud are detected in the second and third phase. Here, static agent (SA) monitors the VMs for possible attacks and sends alerts to IDS controller (IDS-C). After

receiving the alerts, IDS-C sends investigative mobile agents (IMA) to collect intrusion evidences from all the VMs under attack. They have used mobile agents to detect intrusions in virtual environment. On the detection of a new attack at local VM, it updates the signature database of the local IDS. Then it updates the signature database of all other IDSs deployed at the Cloud clusters. The use of IMA increases the complexity in system management. It detects only known attacks.

Sangeetha *et al.* [12] have designed a semantic IDS framework to detect application layer attacks. In this model, an IDS consists of a packet sniffer to collect application layer packets. These packets are sent to the protocol analyzer which parses them based on the protocol type and then dispatches to the respective state machine. Each state machine has a message parsing grammar which compares these packets with the semantic classification tree. It classifies the packets as malicious if there is any deviation from the specifications.

In [3], we have tested different machine learning techniques viz; bayesian, decision tree, artifical neural network, support vector machine and genetic algorithms for detecting network anomaly. Here, we theoretical analysis, we observed that the proposed framework can help to fulfill network security needs of cloud. However, there is a need of investigating light weight techniques in machine learning for detecting network anomaly at the reduced computational overhead.

As per literature survey, we observed that most of the approaches are not capable to handle the traffic from both physical and virtual networks. Cloud IDS approaches like in [6][12] are unable to detect both known and unknown attacks. Existing works [3][7][8][11] require fast detection capability. Some approaches like in [6][7][8][10][11][12] do not have self-adaptive ability for dynamic changes in Cloud resources. Approaches in [5][6][10][12] may fail to manage a huge number of systems in Cloud. There is huge amount of data transfer to exchange intrusion evidences in distributed IDS approaches as like in [6][11] which increases the data transfer cost. Existing works such as [7][8][10][12] are not self resistive against attacks in Cloud.

B. Cloud IDS Requirements and Research Scope

From the literature survey, the following requirements and research scope are derived for Cloud IDS.

Handling Large-Scale Computing Systems. Cloud comprises of a large number of interconnected systems to provide powerful computing facilities. Hence, Cloud IDS should be capable of handling high network traffic. It should monitor and manage the systems in real time with less or no human intervention.

Detecting a Variety of Attacks. In addition to the traditional attacks, virtualization specific attacks are also possible in Cloud. Therefore, an IDS should be efficient to detect cloud specific attacks. The anomaly detection module should be capable of self-learning the variety of attacks over a period of time for improved detection accuracy. Since, Cloud is a distributed architecture; the attack evidences from different Cloud layers should be correlated to detect distributed attacks.

Fast Detection. Due to the huge number of interconnected systems in Cloud, there is always heavy network traffic. In addition, the recent advancements in the network are astonishing which enables the systems to receive large number of packets per second. Hence, the IDS should be lightweight and fast enough to handle such heavy traffic to minimize the amount of packet drops.

Self-Adaptive Capability. In Cloud, the VMs are added/removed dynamically as per the user's demand. The signature and the behavioral database consist of the user's profiles for intrusion detection should be relevant to each VM. Due to the dynamic nature, databases should be updated regularly. Therefore, an IDS should be self-adaptive to the changes without human intervention.

Scalability. Due to dynamic nature of Cloud, IDS should be scalable to monitor heavy and dynamic load.

Synchronization of IDS. In cloud, the IDS sensors can be deployed at different layers. These sensors should exchange the intrusion evidences to each other in order to identify correlated or distributed attacks in Cloud. Therefore, IDS sensors should be synchronized to the same clock automatically for accurate detection of collusion attack. The intrusion evidences (which are being exchanged) should be protected from unauthorized access to avoid any alternation. In addition, there should be minimum communication cost.

Resistance against Compromise. On successful compromise of an IDS, an attacker can disable the functioning of IDS to prevent it from sending intrusion reports. Therefore, an IDS should be separated from the monitoring system to avoid possible compromise. However, it should be protected against unauthorized access for proper functioning of IDS.

In addition, traditional IDS needs viz; low false alerts, high accuracy, low computational and communication cost are also applicable to Cloud IDS.

III. Proposed Security Framework

A. Design Goals

The main objective is to design a security framework which can efficiently detect virtual network layer attacks, while fulfilling Cloud IDS requirements. The design goals of the proposed security framework are as follows:

The proposed security framework should be capable of monitoring and controlling a huge number of computing systems. Being a hierarchical framework, it should be able to detect known, unknown and distributed attacks. It should provide fast detection for early detection of attacks. It should be scalable enough to handle large number of dynamic systems to minimize the packet dropping rate. IDS sensors deployed at different layers of Cloud are expected to be synchronized to same clock to enable distributed attack detection efficiently. An IDS sensor should be kept outside the monitoring system to avoid possible compromise of the sensor.

B. Background Techniques

We combine signature based and anomaly based techniques in the proposed security framework. Here, we use *snort* (sig-

nature based) as a signature based detection and the different classifiers viz; naive bayes, decision tree, random forest, extra trees and linear discriminant analysis in anomaly detection.

***Snort* [9].** *Snort* is an open source signature based lightweight intrusion detection system. It scans through all the network packets coming across the network. It has built-in rules for more sophisticated attacks which are used to detect known attacks.

Naive Bayes Classifier [13]. This supervised learning method works based on the Bayes rule. Given a class variable c and a set of dependent attributes (features) $a_1, a_2, ..., a_n$, Bayes theorem states the relationship between them as follows:

$$P(c|a_1, a_2, ..., a_n) = \frac{P(c)P(a_1, a_2, ..., a_n|c)}{P(a_1, a_2, ..., a_n)}$$

Using the assumptions, it becomes,

$$P(a_i|c, a_1, a_2, ..., a_{i-1}, a_{i+1}, ..., a_n) = P(a_i|c)$$

For all i, it can be simplified as follows

$$P(c|a_1, a_2, ..., a_n) = \frac{P(c)\Pi_{i=1}^{n} P(a_i|c)}{P(a_1, a_2, ..., a_n)}$$

$P(a_1, a_2, ..., a_n)$ is a constant term for the given input, hence it can be eliminated. Now, the equation for classification rule can be written as:

$$P(c|a_1, a_2, ..., a_n) \alpha P(c)\Pi_{i=1}^{n} P(a_i|c)$$

$$\hat{c} = argmax P(c)\Pi_{i=1}^{n} P(a_i|c)$$

$P(c)$ and $P(a_i|c)$ can be estimated using Maximum A Posteriori (MAP) estimation method. This classifier helps in improving the speed and accuracy of intrusion detection. In addition, it requires only a small amount of training data for estimating the parameters.

Decision Tree Classifier [14]. In the proposed framework, we use C4.5 as a decision tree classifier. It can handle both continuous and discrete data along with missing values. The attributes with different values increases both training time and space complexity. Hence, it uses information gain ratio as a measure to choose the best splitting attributes.

Let C be the total number of classes in the given dataset, $p(A, i)$ gives the proportion of instances in A which are assigned to i^{th} class. The entropy of an attribute A is computed as:

$$Entropy(A) = -\Sigma_{i=1}^{c} p(A, i) log(p(A, i))$$

The gain of a dataset D is computed as:

$$Gain(A, D) = Entropy(A) - \Sigma_{v \in Values(D_A)} \frac{|D_{A,v}| Entropy(A_v)}{|D_A|}$$

where $Values(D_A)$ denotes the set of values of A in D, D_A gives the subset of D induced by A and $D_{A,v}$ is the subset of D in which attribute A has a value of v.

Then, the gain ratio of an attribute A is given by:

$$GainRatio(A, D) = \frac{Gain(A, D)}{SplitInfo(A, D)}$$

whereas $SpiltInfo(A, D)$ is computed as:

$$SplitInfo(A, D) = -\Sigma_{v \in Values(D_A)} \frac{|D_{A,v}|}{|D_A|} log \frac{|D_{A,v}|}{|D_A|}$$

For selecting the best split node, it chooses the attribute with maximum GainRatio. Hence, it leads to the reduced computational complexity.

Random Forest Classifier [15]. It creates multiple classification trees from the given dataset for classifying the unknown data. Each tree in the forest receives the unknown input vector for classification. Then, for the given unknown input, each tree votes for a particular class based on its training. Finally, the forest decides the class with maximum number of votes as the classification label. It can handle large dataset with many features and provides estimation of important variables in classification which makes it suitable for Cloud IDS.

Extra Trees Classifier [16]. This meta classifier is a collection of randomized decision trees created on different subsets of dataset. It mainly takes two parameters viz; number of estimators and maximum number of features. Number of estimators indicates the number of trees in the forest. The larger value leads to more accuracy in class label prediction, but it will increase the computational time. It uses gini index which considers maximum number of features to choose the best split node of a tree. Here, the maximum number of best features are chosen randomly. When this value is set to 1, it constructs the random decision tree. Hence, it allows the classifier to reduce the variance of the model. It controls the over-fitting problem [16].

Linear Discriminant Analysis Classifier [17]. It is a multiclass classifier having linear decision surface. It fits class conditional density (Gaussian density) to each class using Bayes' rule. It assumes that all the classes have the same covariance matrix and the attributes are statistically independent. It predicts an unknown input by choosing a class having high variance value. It finds a linear combination of attributes which best characterizes or distinguishes two or more classes. It attempts to model the difference between the classes explicitly. It uses the following score function $(SF(\beta))$ to capture the separation between the classes.

$$SF(\beta) = \frac{\beta^T \mu_1 - \beta^T \mu_2}{\beta^T Cov \beta}$$

With the given score function (SF), it computes the linear coefficients which maximize the score using following equations.

$$\beta = Cov^{-1}(\mu_1 - \mu_2)$$

$$Cov = \frac{1}{(c_1 + c_2)}(c_1 Cov_1 + c_2 Cov_2)$$

where β represents linear model coefficients, Cov_1, Cov_2 are covariance matrices, μ_1, μ_2 denotes mean vectors and c_1, c_2 are constants.

Finally, an unknown point is projected onto the maximally separating direction for class label prediction. It is classified as $class_1$ if:

$$\beta^T(Y - (\frac{\mu_1 + \mu_2}{2})) > log\frac{p(class_1)}{p(class_2)}$$

where β^T is the coefficient vector, Y represents the data vector and $p(class_1)$, $p(class_2)$ are class probabilities.

It is especially suitable for datasets having large number of features. It helps in minimizing the computation time.

We have chosen above classifiers for the proposed framework due to their significant performance on different intrusion datasets. The performance comparison of different classifiers with cross validation on NSL-KDD dataset [19] is shown in Table I.

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT CLASSIFIERS ON NSL-KDD DATASET [19]

Classifier	True Detection (%)	False Detection (%)	Accuracy (%)	ET (sec)
Naive Bayes	**89.94**	**10.6**	**94.73**	**0.105**
Decision Tree	**99.82**	**0.18**	**99.81**	**0.345**
Random Forest	**99.9**	**0.1**	**99.9**	**1.127**
Extra Trees	**99.84**	**0.16**	**99.85**	**0.457**
LDA	**92.7**	**7.3**	**94.19**	**0.273**
QDA	89.18	10.82	94.2	0.426
Adaboost with DT	97.38	2.62	97.67	2.221
SGD	38.83	61.17	20.69	0.193
Logistic Regression	83.67	16.33	90.09	1.949
SVM	98.73	1.27	83.34	17.55
KNN	99.7	0.3	99.61	3.372

ET - Execution Time, sec - seconds

C. Design of the Proposed Security Framework

The deployment of the proposed security framework in cloud is shown in Fig. 2. It consists of *management module (MM)*, *correlation module (CM)* and NIDS. In hierarchical architecture of Cloud, based on the geography, Cloud is divided into multiple groups. The evidences of an attack from lower level are being passed to a higher level. The evidences includes the intrusive connection profile with connection information like source ip, source port, destination ip, destionation port and protocol. The connection profile generation is discussed later. The bottom most layer has only detection component and all other layers have both detection and correlation components. Expect lower layer, intrusion detection and correlation is performed at each layer of Cloud and thus reducing the attack risks at any layer. The modules of the proposed security framework are deployed at different layers viz; *Cloud layer, Cloud cluster layer, Host layer and VM layer*. *VM Layer:*

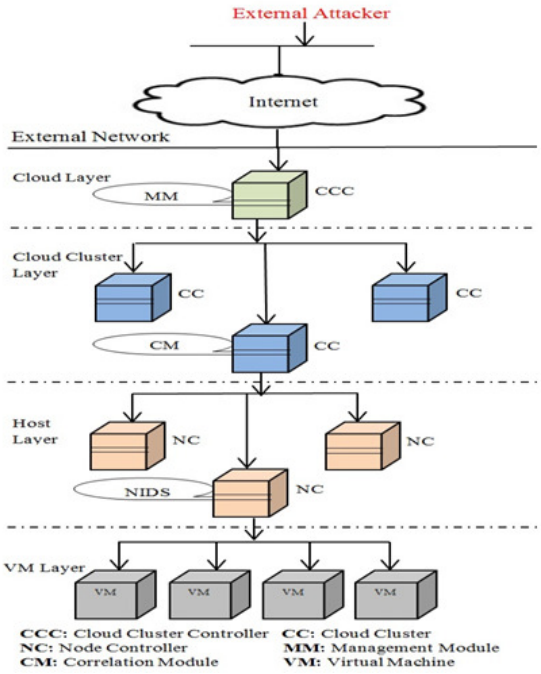

Fig. 2. Deploying the proposed security framework in Cloud

It is the bottom most layer of the Cloud, which forms the core component of the Cloud network. It consists of the VMs allocated to the Cloud users, from where intrusions should be detected.

Host Layer with NIDS: It comprises multiple *node controllers (NC)* or host systems. Each host system can have multiple VMs running on it through hypervisor. Here, NIDS is deployed at the virtual switch to monitor the virtual network traffic from all the VMs. Such deployment helps in enabling single instance of an NIDS to monitor multiple VMs and to secure the virtual network from both insider and external network attacks. It passes the intrusive alerts to the *correlation module (CM)* of the corresponding cluster node.

Cloud Cluster Layer with Correlation Module: It forms the next level of hierarchy consisting of *Cloud Clusters (CC)* connected in Cloud. Each *CC* manages and controls all the underlying host systems. It has both correlation and detection components. *Correlation module (CM)* deployed at *CC* receives the intrusion evidences from NIDSs. It applies Dempster-Shafer Theory (DST) [4] on these evidences to identify distributed attack at Cluster level. Using access control, only privileged hosts are allowed to send intrusion alerts to *CC*.

Cloud Layer with Management Module: It forms the top most level of hierarchy consisting of *Cloud Cluster Controller (CCC)*. It manages and controls all the underlying *Cloud Clusters (CC)*. It has both correlation and detection components. Distributed attack evidences from *CC* are being passed to the *Management module (MM)* deployed at *CCC*. *MM* uses these

evidences to detect distributed attack at Cloud level using DST combination rule. Here, an access control is also applied to allow alerts only from the authorized Clusters.

Fig. 3 shows the detailed view of the proposed security framework. Here, we use firewall with the proposed security framework.

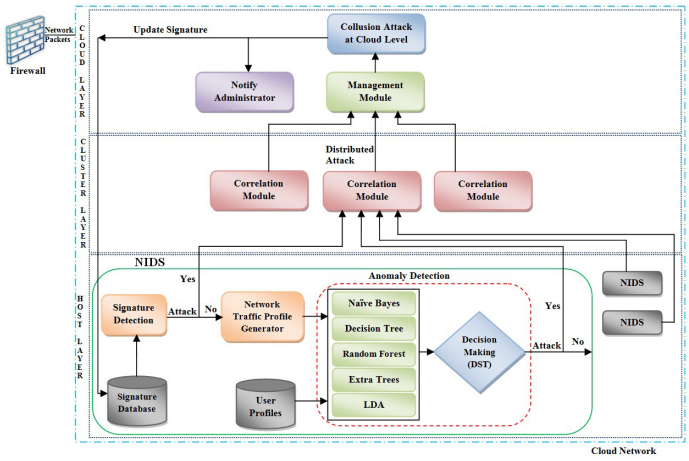

Fig. 3. Detailed view of the proposed framework

a) ***Firewall.****:* It is a software or hardware system to prevent unauthorized access to/from a network [5]. It acts as a first line of defense in the network. It allows only authorized IP addresses of the users. Thus, it helps in eliminating the monitoring of external network traffic, and thus NIDS has to monitor only authorized network traffic and to detect possible attacks.

b) ***NIDS.****:* It is deployed at the virtual network of each host to monitor the network traffic related to the VMs. It monitors the network traffic to/from VMs and VM to VM. This, it helps in detecting the virtual network layer attacks. It consists of signature based detection, network traffic profile generator, anomaly detection and decision making (refer Fig. 4). *Signature-based Detection.* It uses signature database

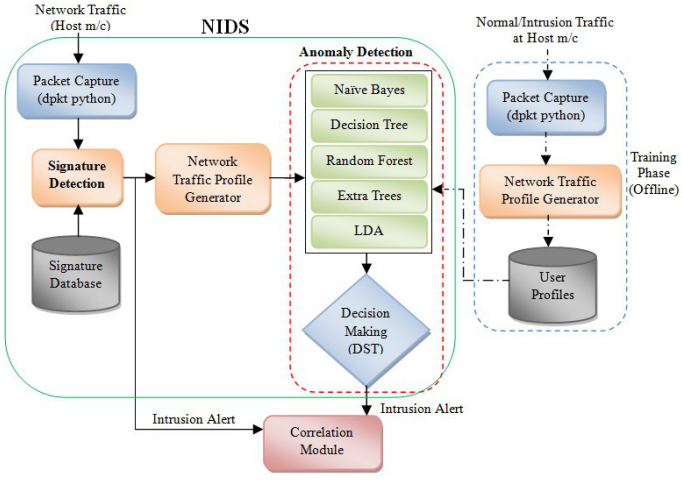

Fig. 4. Design of the proposed NIDS

which stores the already defined set of rules or signatures to identify the known attacks. In the proposed NIDS, first network packets are passed to signature detection module. This module matches the captured network packets with the signatures in signature database for any correlation. If any correlation is found, then those network packets will be considered as intrusive packets. The report of these intrusive packets is sent to the *correlation module* at the cluster node for further analysis. The packets which are considered as normal are passed to network traffic profile generator module. For signature based detection, we use *snort* [9].

Network Traffic Profile Generator. It generates network traffic profile from the captured network packets (refer Fig. 5). It maintains two lists viz; ongoing connection list and closed connection list. Initially, all the packets are added to the ongoing connection list. Once an ongoing connection gets completed, it is moved to the closed connection list and the network traffic profile is generated. Each profile consists of basic features, traffic features related to same host and traffic features related to same service. The traffic features are extracted from the previously closed connection(s) within the given time frame (e.g., 2 seconds). Each profile is then applied to *anomaly detection module*. *Anomaly Detection.* It

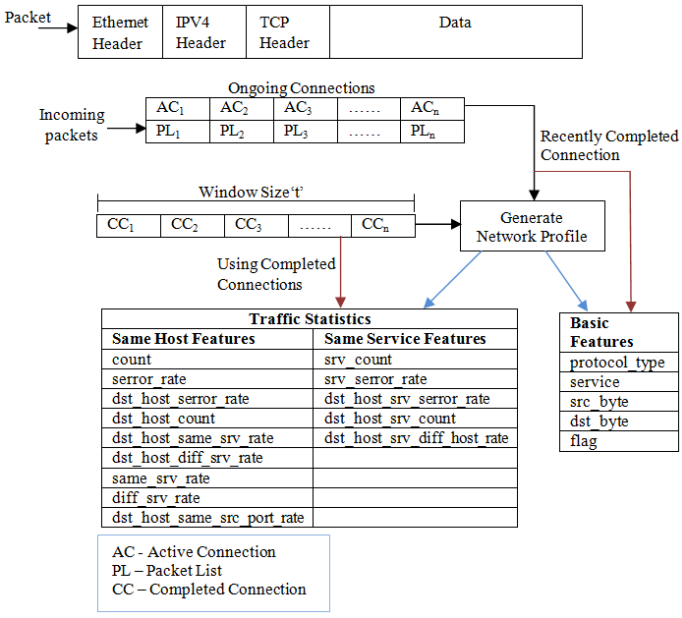

Fig. 5. Network traffic profile generator

takes network traffic profiles and identifies the possibility of any intrusion with the help of different classifiers viz; naive bayes, C4.5 (decision tree), random forest, extra trees and linear discriminant analysis (LDA) as presented earlier. These classifiers are trained (*offline*) using the previously observed network traffic profiles. Thus, these trained classifiers are used to predict the possibility of attacks from the derived network traffic profile in *real time* which enable the NIDS in detecting unknown attacks in future. Here, each classifier classifies the given network traffic profile as normal or intrusion. The dif-

ferent outcomes for a given traffic profile are applied decision making module for final decision.

Decision Making Module. It applies the Dempster-Shafer Theory (DST) combination rule to the collected evidences (intrusion or normal) from each classifier to make the final decision about intrusion. The background of the DST combination rule is as follows:

Dempster-Shafer Theory [21]. DST is being used quantitatively in fusion for solving the problem of analyzing the uncertainty. Here, uncertainty is represented using belief functions. It helps to decide three solutions: Intrusion, Normal, or Unknown (Intrusion or Normal). The last option "Unknown" allows ignorance which makes it suitable for intrusion detection. A value associated with ignorance makes it suitable for anomaly detection. Here, it takes intrusive evidences from five different feasible classifiers, and thus it helps in improving the detection accuracy.

Basic Probability Assignment (bpa) [21]. It is a mass function which assigns belief values (denoted as $m(A)$). In DST, assignments are done to all the possible elements of the subsets. The value of bpa indicates the degree of supporting or refuting the evidence. It ranges between 0 to 1 such that *bpa* for *null* set ϕ is m(ϕ) = 0 and the sum $m(A_1) + ... + m(A_n) = 1$.

DST Combination Rule [4]. It is used to combine two or more independent observers for decision making. It utilizes the orthogonal sum of basic probability assignments from independent observers. For example, if there are two independent observers having *bpa* as $m_1(A)$ and $m_2(A)$, then Dempster's combination rule is as follows:

$$m(B) = m_1(B) \bigoplus m_2(B) = \frac{\sum_{i,j:A_i \cap A_j = B} m_1(A_i) m_2(A_j)}{\sum_{i,j:A_i \cap A_j = \phi} m_1(A_i) m_2(A_j)}$$

In the proposed NIDS, we combine decisions from different classifiers. We have assigned the trustworthy score to different classifiers based on their accuracy. The modified DST combination rule along with trustworthy scores (ts_1, ts_2) is derived as follows.

$$m(B) = m_1(B) \bigoplus m_2(B)$$

$$m_1(B) \bigoplus m_2(B) = \frac{\sum_{i,j:A_i \cap A_j = B} [m_1(A_i)]^{ts_1} [m_2(A_j)]^{ts_2}}{\sum_{i,j:A_i \cap A_j = \phi} [m_1(A_i)]^{ts_1} [m_2(A_j)]^{ts_2}}$$

Thus, this module identifies unknown attacks accurately and sends evidences to the correlation module at cluster layer of Cloud. If it says no attack, then further analysis is not required.

*c) **Correlation Module (CM).**:* It is deployed at each cluster node to detect distributed attacks at the respective cluster level (refer Fig. 6). It receives the intrusion evidences from the NIDS deployed at host systems. For compromising NIDS at host layer, root level privileges are required and thus, it is difficult to compromise the NIDS at host layer. This modules allows only the host level NIDSs to send intrusion evidences by applying access control. These evidences are passed to the decision making module which uses DST combination rule to identify distributed attacks. Here, basic probability is assigned based on the accuracy of classifiers. Then the intrusive evidences which are considered as distributed attack at cluster level are passed to *management module*.

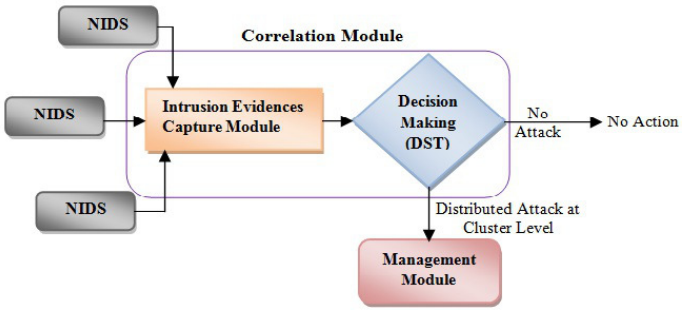

Fig. 6. Correlation module

*d) **Management Module (MM).**:* It is deployed at the Cloud layer and hence it forms the centralized system (refer Fig. 7). It gathers the distributed attack evidences from cluster nodes. Using access control, it accepts the evidences only from the clusters. These evidences are passed to the decision making module for the detection of collusion attack at Cloud level. Here, it uses the DST combination rule for combining the decisions from multiple clusters and to identify collusion attack. The cross validation accuracy is used as basic probability assignments for making decision about collusion attack. On detection of collusion attack at Cloud level, it notifies the administrator and the attack signature is updated to all the clusters. If there is no collusion attack, then the attack signature is updated only to host layer NIDSs of the reported clusters.

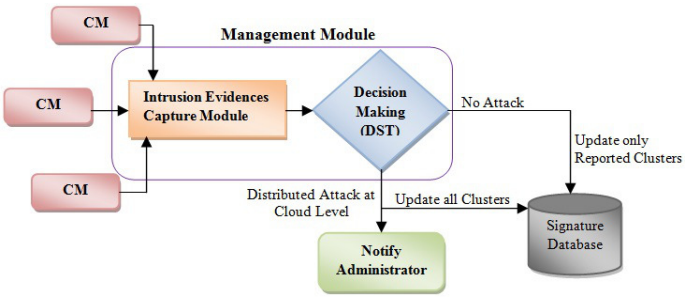

Fig. 7. Management module

In the proposed framework, NIDS monitors both virtual and physical network traffic since it is deployed at the virtual network. It handles the traffic from all the underlying VMs. Thus, it helps in analyzing the insider network traffic in order to detect insider attacks. The proposed signature cum anomaly-based NIDS detects both known and unknown attacks. Retraining of the proposed NIDS according to dynamic changes in Cloud makes the system self-adaptive. Synchronization of various IDS sensors at different layers of Cloud

enables the detection of distributed attack. NIDS, correlation and management modules deployed at privileged domains prevent them against possible compromise. Minimum number of intrusive evidences are exchanged, hence communication cost is reduced.

IV. Performance Evaluation and Validation

A. Experimental Setup

We have validated the proposed framework on virtual network test bed at NIT Goa. The experimental setup consists of five host machines and different virtual machines with different operating systems viz; Ubuntu 14.02, Windows 7 and Ubuntu Server Edition 14.04. These VMs are running using kernel-based virtual machine (KVM) [22], a hypervisor. A virtual bridge is created using bridged networking [23] mode to allow the external systems to directly access the underlying VMs.

For experiments, we have used different datasets viz; KDD intrusion dataset [18], NSLKDD intrusion dataset [19], Booter DoS Attack dataset [20] and Botnet dataset [24]. Some of these datasets are old and contain only traditional attacks such as DoS, buffer overflow, etc. However, we have used these datasets to validate the performance of proposed NIDS in terms of Cloud IDS requirements against traditional attacks.

Table II shows the list of tests conducted on different datasets. For each test, we have considered three cases: Case1 represents the proposed NIDS model with three classifiers viz; decision tree, random forest and extra trees with DST. Case2 includes four classifiers viz; decision tree, random forest, extra trees and linear discriminant analysis (LDA) with DST. Finally, Case3 includes five classifiers (as in the proposed NIDS) with DST. The aim behind motivation behind considering different cases is to prove the feasibility of the classifiers fusion to detect intrusions. The performance of the classifiers is evaluated with the help of different parameters viz, true positive rate, false positive rate, true negative rate, false negative rate, accuracy and execution time [25]. In our experiments, we consider intrusion as positive and normal as negative.

TABLE II
Experiments conducted in *offline* simulation of NIDS

Test	Training Dataset	Testing Dataset
Test-1	KDD99(10%) training dataset	KDD99(10%) test dataset
Test-2	NSLKDD training dataset	NSLKDD test dataset
Test-3	Booter1 training dataset	Booter1 test dataset
Test-4	Botnet training dataset	Botnet test dataset

B. Results

The tables (Table III, Table IV, Table V and Table VI) show the performance results viz; true positive rate (TPR), false positive rate (FPR), true negative rate (TNR), false negative rate (FNR), accuracy and execution time of the proposed NIDS. From these tables, it is observed that the proposed NIDS exhibits good performance results when compared other combinations of classifiers. It is observed that the detection accuracy is improved with the help of weights to the classifiers in DST computation. It justifies the feasibility of the selected classifier in the proposed NIDS.

TABLE III
Performance of the proposed NIDS in Test-1

Case	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	Execution Time (Seconds)
Case1	90.87	9.13	99.52	0.48	92.55	1.246
Case2	90.97	9.03	99.49	0.51	92.63	2.14
Case3	93.65	6.35	99.35	0.65	94.75	3.539

TABLE IV
Performance of the proposed NIDS in Test-2

Case	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	Execution Time (Seconds)
Case1	57.18	42.82	97.42	2.58	74.51	0.231
Case2	59.86	40.14	97.23	2.77	75.96	0.329
Case3	72.27	27.72	96.79	3.21	82.83	0.7366

TABLE V
Performance of the proposed NIDS in Test-3

Case	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	Execution Time (Seconds)
Case1	89.61	10.39	99.9	0.1	90.73	0.093
Case2	99.45	0.35	62.88	37.12	95.51	0.167
Case3	99.99	0.01	99.83	0.17	99.97	0.171

TABLE VI
Performance of the proposed NIDS in Test-4

Case	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	Execution Time (Seconds)
Case1	99	1	80	20	89.95	0.0244
Case2	99	1	83	17	91.46	0.0274
Case3	99.6	0.4	89	11	94.47	0.0303

The graphical representation of the performance results is as follows. Fig. 8 and Fig. 9 show the true positive rate and false positive rate respectively for individual classifiers with the proposed NIDS on different tests. It shows that there is improvement in the true positive rate ($> 99\%$) and the proposed NIDS produces less number of false positives when compared to individual classifiers.

The true negative rate ($> 97\%$) of the proposed NIDS is higher than individual classifier, while false negative rate ($<$

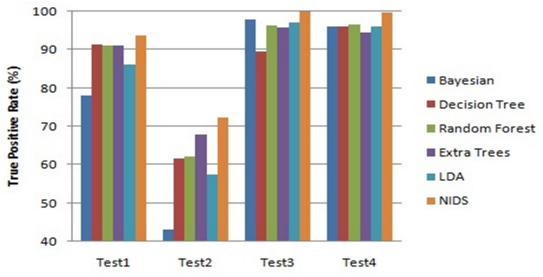

Fig. 8. True positive rate of the proposed NIDS

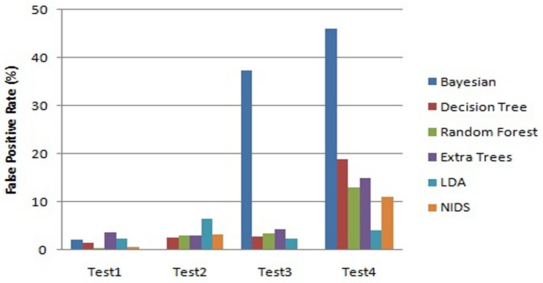

Fig. 9. false positive rate of the proposed NIDS

3%) of the proposed NIDS is lower than individual classifier, as shown in Fig. 10 and Fig. 11. The accuracy of the proposed NIDS is higher than individual classifier and the execution time is slightly more since it combines multiple classifiers (refer Fig. 12 and Fig. 13).

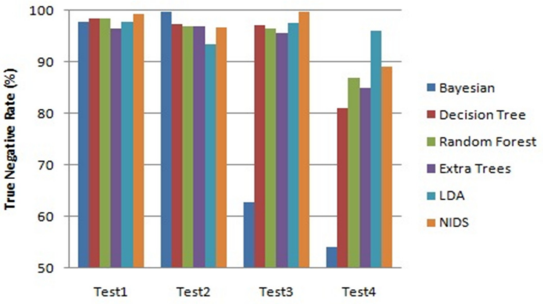

Fig. 10. True negative rate of the proposed NIDS

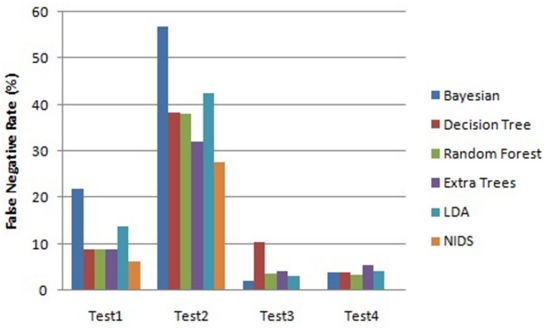

Fig. 11. False negative rate of the proposed NIDS

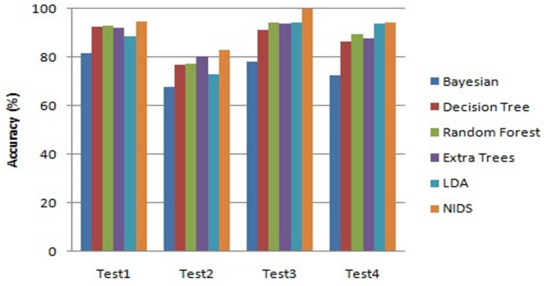

Fig. 12. Accuracy of the proposed NIDS

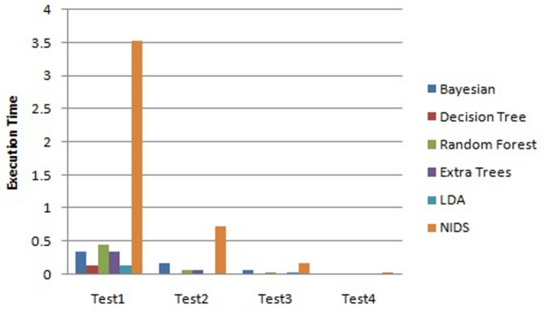

Fig. 13. Execution time (in seconds) of the proposed NIDS

C. Analysis of the Cloud IDS Requirements

Dealing with Large-Scale Computing System. As NIDS is deployed at the virtual network layer, it can monitor both virtual network traffic and physical network traffic to the host systems. It helps in handling large-scale VMs at each host system. In addition, only distributed attack evidences are passed to correlation and management modules, and thus minimizing the network packets drops at Cluster and Cloud layer.

Identifying Variety of Attacks. The proposed NIDS consists of both signature and anomaly based techniques. Thus it is capable of detecting a variety of attacks (both known and unknown) at virtual network layer. From the results, it is observed that it can detect various attacks (e.g., DoS, buffer overflow, etc.) including distributed attacks.

Fast Detection. Known attack detection is fast due to only signature matching. We have applied signature detection prior to anomaly detection, and therefore it reduces number of network connections to be analyzed by anomaly detection. The results show that the classifiers for anomaly detection require less time, and thus this framework is feasible for real time environment.

Automated Self-Adaptive Capability. The network traffic profiles of the VMs are subjected to change over a period of time. Hence, retraining of the classifiers with the updated network traffic profiles are being carried out by NIDS at regular interval in offline mode to make the system self-adaptive to these changes.

Scalability. As per results, the proposed NIDS is capable of handling the network traffic from all the VMs. The correlation

and management modules are also scalable since the number of network profiles passed to them is minimized.

Synchronization of IDS Sensors. NIDS sensors deployed at each host are synchronized to the same clock for efficient detection of distributed attack.

Resistance to Compromise. In the proposed security framework, an access control is applied to CM and MM which helps in achieving resistance to compromise. It is difficult to compromise CM and MM, as these modules are running at administrator level, where root privileges are required to disable the functioning of CM and MM. In addition, the successful compromise of one NIDS does not affect the performance of other NIDSs in the network. In addition, such compromise can be recovered.

V. Conclusions

Due to numerous vulnerabilities in the current implementations of virtualization, security is inherently the major concern in Cloud computing. To detect attacks at the virtual network layer of Cloud, we have designed a security framework which combines signature based and anomaly detection. We have applied signature based detection prior to anomaly detection, and thus improving the detection efficiency. For anomaly detection, we applied different suitable classifiers and final decision about intrusion is derived with the help of Dempster-Shafer Theory (DST), and thus improving the detection accuracy. For identifying the distributed attacks at cluster or Cloud layer, intrusion evidences from the host layer are collected and then final decision about distributed attack is derived using DST. We have validated the feasibility of the classifiers for intrusion detection in Cloud through *offline* simulation using different intrusion datasets. The experimental results are very encouraging and it seems that the proposed security framework fits very well for virtual network security in Cloud computing.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg*, 2011.

[2] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey on security issues and solutions at different layers of Cloud computing," *The Journal of Supercomputing*, Springer, vol. 63, no. 2, pp. 561-592, 2013.

[3] Kamatchi A. and C. Modi, "An Efficient Security Framework to Detect Intrusions at Virtual Network Layer of Cloud Computing," *19th conference on Innovations in Clouds, Internet and Networks (ICIN)*, 2016.

[4] T. M. Chen and V. Venkataramanan, "Dempster-shafer theory for intrusion detection in ad hoc networks," *IEEE Internet Computing*, vol. 9, no. 6, pp. 35–41, 2005.

[5] C. N. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in Cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42-57, 2012.

[6] S. Dhage, B. Meshram, R. Rawat, S. Padawe, M. Paingaokar, and A. Misra, "Intrusion detection system in cloud computing environment," *International Conference & Workshop on Emerging Trends in Technology*, 2011, pp. 235–239.

[7] C.J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems," *IEEE Transactions Dependable and Secure Computing*, vol. 10, no. 4, pp. 198-211, 2013.

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

[9] Snort. [Online]. Available: https://www.snort.org/

[10] D. Singh, D. Patel, B. Bhavesh, and C. Modi, "Collaborative IDS Framework for Cloud," *International Journal of Network Security*, vol. 18, no. 4, pp. 699-709, 2015.

[11] H. Toumi, A. Talea, B. Marzak, A. Eddaoui, and M. Talea, "Cooperative Trust Framework for Cloud Computing Based on Mobile Agents," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 7, no. 2, 2015.

[12] S. Sangeetha, R. Ramya, M.K. Dharani, and P. Sathya, "Signature Based Semantic Intrusion Detection System on Cloud," *In Information Systems Design and Intelligent Applications*, 2015, pp. 657-666.

[13] Naive Bayes Classifier. [Online]. Available: http://scikit-learn.org/stable/modules/naive_bayes.html

[14] X. Niuniu and L. Yuxun, "Review of decision trees," *International Conference on Computer science and information technology* (ICCSIT), 2010, pp. 105-109.

[15] R.M. Elbasiony, E.A. Sallam, T.E. Eltobely, and M.M. Fahmy, "A hybrid network intrusion detection framework based on random forests and weighted k-means," *Ain Shams Engineering Journal*, vol. 4, no. 4, pp. 753-762, 2013.

[16] Extra Trees Classifier. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn-.ensemble.ExtraTreesClassifier.html

[17] Linear Discriminant Analysis. [Online]. Available: http://www.saedsayad.com/lda.htm

[18] KDD cup 1999 data. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[19] M. Tavallaee, E. Bagheri, W. Lu, and A.A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *In Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications*, 2009.

[20] J.J. Santanna, R.V. Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Zambenedetti Granville, and A. Pras, "BootersAn analysis of DDoS-as-a-service attacks," *IEEE International Symposium on Integrated Network Management* (IM), 2015, pp. 243-251.

[21] W. Hu, J. Li, Q. Gao, "Intrusion detection engine based on Dempster-Shafer's theory of evidence," *International Conference on Communications, Circuits and Systems Proceedings*, vol. 3, pp. 1627-1631, 2006.

[22] KVM Installation. [Online]. Available: https://help.ubuntu.com/community/KVM/Installation

[23] KVM Networking. [Online]. Available: https://help.ubuntu.com/community/KVM/Networking

[24] CTU-Malware-Capture-Botnet-4. [Online]. Available: http://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-4/

[25] Performance Measures for Machine Learning. [Online]. Available: http://www.cs.cornell.edu/courses/cs578/2003fa/performance_measures.pdf