An Anomalous Behavior Detection Model in Cloud Computing

Xiaoming Ye
the College of Computer Science, Cybersecurity Research Institute, Sichuan University, Chengdu 610065, China.

Xingshu Chen
the College of Computer Science, Cybersecurity Research Institute, Sichuan University, Chengdu 610065, China.

Haizhou Wang
the College of Computer Science, Cybersecurity Research Institute, Sichuan University, Chengdu 610065, China.

Xuemei Zeng
the College of Computer Science, Cybersecurity Research Institute, Sichuan University, Chengdu 610065, China.

Guolin Shao
the College of Computer Science, Cybersecurity Research Institute, Sichuan University, Chengdu 610065, China.

See next page for additional authors
Follow this and additional works at: https://tsinghuauniversitypress.researchcommons.org/tsinghua-science-and-technology

Recommended Citation
Xiaoming Ye, Xingshu Chen, Haizhou Wang et al. An Anomalous Behavior Detection Model in Cloud Computing. Tsinghua Science and Technology 2016, 21(3): 322-332.
An Anomalous Behavior Detection Model in Cloud Computing

Authors
Xiaoming Ye, Xingshu Chen, Haizhou Wang, Xuemei Zeng, Guolin Shao, Xueyuan Yin, and Chun Xu
An Anomalous Behavior Detection Model in Cloud Computing

Xiaoming Ye, Xingshu Chen*, Haizhou Wang, Xuemei Zeng, Guolin Shao, Xueyuan Yin, and Chun Xu

Abstract: This paper proposes an anomalous behavior detection model based on cloud computing. Virtual Machines (VMs) are one of the key components of cloud Infrastructure as a Service (IaaS). The security of such VMs is critical to IaaS security. Many studies have been done on cloud computing security issues, but research into VM security issues, especially regarding VM network traffic anomalous behavior detection, remains inadequate. More and more studies show that communication among internal nodes exhibits complex patterns. Communication among VMs in cloud computing is invisible. Researchers find such issues challenging, and few solutions have been proposed—leaving cloud computing vulnerable to network attacks. This paper proposes a model that uses Software-Defined Networks (SDN) to implement traffic redirection. Our model can capture inter-VM traffic, detect known and unknown anomalous network behaviors, adopt hybrid techniques to analyze VM network behaviors, and control network systems. The experimental results indicate that the effectiveness of our approach is greater than 90%, and prove the feasibility of the model.

Key words: virtual machine; network behavior; anomaly detection; cloud computing

1 Introduction

Cloud computing infrastructure is a hybrid networking system, that integrates hybrid technology, hybrid operating systems, and hybrid hardware. Cloud computing aims to provide on-demand, low-cost, high-performance computing resources, and leverages virtualization technologies to deliver storage, server, network services, CPU, and memory.[1]

Cloud computing has to face traditional security threats and new generations of security threats. Cloud computing vulnerabilities include core technology vulnerabilities (e.g., Web applications and services, virtualization, and cryptography), essential cloud characteristic vulnerabilities (e.g., unauthorized access to management interfaces, Internet protocol vulnerabilities, etc.), and defects in known security controls, and prevalent vulnerabilities (e.g., injection vulnerabilities and weak authentication schemes)[2]. Attackers find vulnerabilities and use them to undertake attacks. There have been many attacks against virtual machines on cloud computing platforms, such as various port scanning attack, attacks on hypervisors, attacks on virtualization, backdoor channel attacks, flooding attacks, user-to-root attacks, and insider attacks (e.g., internal denial-of-service attacks via zombies in the cloud)[3].

Virtualization technology is a core technology in cloud computing. Virtual Machines (VMs) are key components of cloud infrastructure. For example, virtualization technology enables the execution of multiple operating system environments, or VM instances, on a single hardware system. Each VM owns an operating system and applications. A VM executes programs like a physical machine. Cloud computing contains both physical and virtual networks[4]. Virtualization creates blind spots of network traffic, or invisible networks, in the same server infrastructure. Gartner[5] represented six of the
most common virtualization security risks, including noting that “the lack of visibility and controls on internal virtual networks created for VM-to-VM communications blinds existing security policy enforcement mechanisms”. He said that more than 60% of virtual machines in production are less secure than their physical counterparts. VMs are losing their ability to detect and control this communication. Attacks and data can move through the VMs without ever going out to the physical network, which means these attacks will not be detected by traditional tools. To deal with this vulnerability, making all VM communications traffic visible is the first problem that needs to be solved.

Currently, the challenge is how to establish an effective network behavior detection system for each VM in a cloud computing network, so that it can accurately identify deviations from normal network behavior of the virtual machines, and reduce cloud security risks.

This paper proposes a model to detect anomaly behavior for the VMs in cloud computing. This model is a time-varying system with a number of network traffic features. Here are the main work and contributions of this paper:

- Communications among VMs in cloud are invisible. The model uses Software-Defined Networks (SDN) to build a virtual network, so that the virtual switch network traffic is through the physical network card, then to the node where the deployed system resides.
- The model aims to detect known and unknown anomalous behaviors.
- This paper designs a control model, and adopts hybrid techniques to analyze VM network behaviors and control network systems.

The remainder of this paper is organized as follows. Section 2 introduces state machine definitions and components of the model and methods of state analysis. Section 3 introduces Snort, data processing, application behavior analysis, and decision analysis. The algorithm and technologies used in this paper are also discussed. Experiments were conducted and the results are discussed in Section 4. Conclusions and future work are presented in Section 5.

2 Model Overview

2.1 State definitions

Network behavior has various forms and means of changing characteristics. We cannot describe and identify all the anomalous behaviors of networks but can describe states that characterize a VM under attack.

Before the attack, a malicious user tries to scan VMs and search for vulnerabilities or ports to find the cloud computing infrastructure security “holes”. The attacker then has a planned, purposeful, step-by-step process to undertake the attack, including an attack action plan, tests, and a complete attack process. Normal VM network behavior is a state of dynamic equilibrium. Network attacks will affect this state, which is defined as follows:

**Definition 1** (Homeostasis, $S_1$): Currently, the virtual machine is running properly, the network traffic situation is in dynamic equilibrium. Virtual machines have vulnerabilities and other security threats, but they have not been detected or used.

**Definition 2** (Before imbalances, $S_2$): Suppose anomalous behaviors of network traffic are detected, such as vulnerability scanning. In this state, VM security threats have been detected, but have not yet been utilized by an attacker.

**Definition 3** (Imbalances early, $S_3$): Suppose anomalous network traffic behaviors are detected more than once. An attacker has detected vulnerabilities in the virtual machine, and exploited them.

**Definition 4** (Imbalance, $S_4$): Network traffic anomalies are repeatedly detected. The VM is under continuous cyber-attacks.

Figure 1 depicts the transition of virtual machine states under attack. The sequence starts at state $S_1$. Attack behaviors make VM state $S_1$ activate states $S_2$, $S_3$, and $S_4$. When anomalous behavior has been controlled, the VM state returns to a state of dynamic equilibrium. 

![Fig. 1 VM state transition.](image-url)
Through application behavior analysis, the model determines whether or not application behavior deviates from normal. According to this, the model can be used to describe VM state transitions. The details of its algorithm will be given in Section 2.3.

### 2.2 Components

This paper proposes a cloud computing anomalous behavior detection model. The model can detect known and unknown anomalous behaviors. Hybrid techniques are used to detect anomalies. The model determines whether the network behavior of a virtual machine deviates from normal.

Figure 2 describes the model components and detection processes. This model consists of VM profiles, Snort, data processing, application behavior analysis, state analysis modules, and decision analysis.

- The VM profile module is a dataset used to store and manage VM profiles based on traffic analysis. Application behavior states are used to build a set of VM profiles. The information includes the services, the software version number, open port, IP address, MAC address, and rules. In addition, it also includes rules for communication among virtual machines, and between virtual machines and physical machines. These profiles include VM security rules among other features.

- VM network traffic passes through Snort first. This module is used to detect known anomalous behaviors. Snort uses detection rules based on signature. The model first executes a Snort module, which provides known anomaly detection, improves the detection rate, and reduces the computational cost. Then network traffic flows into the next detection module. The Snort model not only uses the known anomaly behavior rule base, but also reduces the volume of traffic that must be processed in the next module.

- The model then performs application behavior analysis. This module has two parts. In the first part, traffic classification is performed to identify applications. This part manipulates the training examples and produces multiple classifiers to improve the application classification accuracy. In the second part, the application behavior analysis module uses time series to build a baseline for each application. Considering the normal network behavior of VMs, time series analysis is used. For example, people work during the day and rest at night. People work from Monday to Friday and rest on Saturday and Sunday. Other regular behaviors include data backup, “application heartbeat”, and periodic behaviors that are repeated. This module aims to detect unknown anomalous behaviors. So the properties of applications for each VM are stored. The algorithm of this module is given below.

- Finally, the results of detection from Snort and the application behavior analysis module are saved as anomaly records. In order to improve detection accuracy, the decision analysis module uses the records for in-depth analysis. The algorithm is below.

- After the application behavior analysis, the VM profile information is updated. According to this, the model can describe the states of the VMs in cloud. The formulas are described below.

### 2.3 State analysis

VM profiles have summary information about each VM in the cloud collected from traffic. For each application, detection results from Snort and behavior analysis are added to the VM profiles. Other information includes number of services, open port number, number of flows, number of outgoing connections, number of incoming connections, maximum value of each connection, and duration. In addition, it also includes rules for communication between virtual machines, as well as for communication between virtual machines and physical machines.

\[ A_k \] represents the anomalous performance of the \( k\)-th state of the VM.
th VM in the cloud as discussed in Section 2.1. In this method, the state of each VM is shown in three forms $A^{(1)}, A^{(2)}, \text{ and } A^{(3)}$. Its value is calculated by Eq. (1), where $n(t)$ is the random noise, and $r_1, r_2$, and $r_3$ are parameters. $v_{mk}$ denotes the weight of the $k$-th VM using Eq. (20). $A^{(1)}$ represents the degree of deviation of traffic periodicity of the VM using Eq. (6). $A^{(2)}$ denotes the anomalous status of known applications (app) using Eq. (11). $A^{(3)}$ denotes anomalous status of unknown applications (uapp) using Eq. (12).

The anomalous performance of VMs is:

$$A_k = v_{mk}(r_1A^{(1)} + r_2A^{(2)} + r_3A^{(3)}) + n(t) \quad (1)$$

Here’s how to compute $A^{(1)}$. A VM profile is a time-varying matrix with network traffic features that can describe the state of network traffic. A time series is a sequence of data usually at regular intervals of time during a specific period. The most important feature of this type of data is that neighboring observations are dependent on each other. This paper takes into account history data before time $T$ ($T_1$, $T_2$, and $T_3$ are three adjacent time before detection time $T$), but also last week’s value $WT$, last month’s value $MT$, and last years value $YT$ at each observing time as shown in Fig. 2.

Thus, in Eq. (2), here are six values associated with given time, where $m$ represents the total number of observation characteristics. Create a time matrix $Stvm$ as follows:

$$Stvm = \begin{pmatrix} w_{11} & \cdots & w_{16} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{m6} \end{pmatrix} \quad (2)$$

Build a vector base on each time window $W_i$ at time $t$, where $W_1$ represents $T_1$, $W_2$ represents $T_2$, $W_3$ represents $T_3$, $W_4$ represents $WT$, $W_5$ represents $MT$, and $W_6$ represents $YT$.

$$W_i = (w_{1i}, w_{2i}, ..., w_{mi})^T \quad (3)$$

$$Stvm = (W_1, W_2, W_3, W_4, W_5, W_6) \quad (4)$$

The model then uses Euclidean distance to measure the transformation. It means the likelihood of an anomalous VM state performance can be expressed by the distance spanned by the time window vector. The Euclidean distance can be expressed as

$$\text{dist}(W_i, W_j) = \sqrt{\sum_{k=1}^{m} (w_{ki} - w_{kj})^2} \quad (5)$$

A weight $\beta_i$ is associated with each time window to express its importance in relation to time $T$.

$$A^{(1)} = \frac{1}{6} \sum_{i=1}^{6} (\beta_i \times \text{dist}(W_T, W_i)) \quad (6)$$

Here’s how to compute $A^{(2)}$. In the following equations app$_i$ represents the $i$-th application. The likelihood of anomalous application performance can be expressed in detail by considering factors such as the probability of presence of the application in traffic:

$$F_i = \text{Pr}\{\text{app}_i\} \times \text{Pr}\{\text{anomaly}\mid \text{app}_i\} = \text{Pr}\{\text{app}_i\} \times \{\text{app}_j \text{ is suspicious}\mid \text{app}_i\} \times \text{Pr}\{\text{app}_i \text{ is anomalous}\mid \text{app}_j \text{ is suspicious}\}$$

$$F_{1i} \times F_{2i} \times F_{3i} = \prod_{j=1}^{3} F_{ij} \quad (7)$$

In Eq. (7), $F_i$ denotes the status of the $i$-th application, which consists of three viewpoints $F_{1i}$, $F_{2i}$, and $F_{3i}$. $F_{1i}$ represents the probability of the $i$-th application in traffic, $F_{2i}$ represents the probability of a detected anomaly in Snort or application behavior analysis in the $i$-th application, but not in the results of the decision analysis module. $F_{3i}$ represents the probability of an anomaly being found in the decision analysis module. $F_{1i}$, $F_{2i}$, and $F_{3i}$ can be calculated by Eqs. (8) – (10).

$$F_{1i} = \frac{\text{Number of connections to app}_i}{\text{Total number of connections}} \quad (8)$$

$$F_{2i} = \frac{\text{Number of anomaly alert app}_i}{\text{Number of connections to app}_i} \quad (9)$$

$$F_{3i} = \frac{\text{Number of anomaly app}_i}{\text{Number of anomaly alert app}_i} \quad (10)$$

A weight $\lambda_i$ is associated with the importance of the app$_i$. $k$ represents the number of the applications. The normalized $A^{(2)}$ from Eqs. (8) – (10) can be given as

$$A^{(2)} = \frac{1}{1 + e^{-\sum_{i=1}^{n} \lambda_i \prod_{j=1}^{k} F_{ij}}} \quad (11)$$

Below is the formula for computing $A^{(3)}$. The likelihood of anomalous behavior in unknown applications (uapp) can be expressed by considering factors such as the probability of presence of the unknown applications in traffic:

$$A^{(3)} = \text{Pr}\{\text{uapp}\} \times \text{Pr}\{\text{anomaly}\mid \text{uapp}\} = N_1 \times N_2 \quad (12)$$

$$N_1 = \frac{\text{Number of connections to uapp}}{\text{Total number of connections}} \quad (13)$$

$$N_2 = \frac{\text{Number of alerts to uapp}}{\text{Total number of alerts}} \quad (14)$$

So the anomalous performance of the $k$-th VM from
Eqs. (6), (11), and (12) can be calculated by Eq. (15).

\[
A_k = \text{vm}_k \left( \frac{r_1}{n} \sum_{i=1}^{n} (\beta_i \times \text{dist}(W_i, W_j)) + \frac{r_2}{1 + e^{-\sum_{i=1}^{k} \lambda_i \prod_{j=1}^{i} F_{ij}}} + r_3 \prod_{i=1}^{2} N_i \right) + n(t)
\]  

Even a single VM is considered important in the cloud if it is connected to many VMs, which multiply the impact of each VM. \text{vm}_k is an impact factor associated with the VM’s importance in the cloud. Now we show how to compute \text{vm}_k.

Figure 3 shows a sample connection graph. Each node represents a VM, where \(V_k\) denotes the \(k\)-th VM, and \(P_j\) denotes the \(j\)-th port of the VM. A connection between \(V_1\) and \(V_3\) exists if a flow record having these addresses is observed. Between nodes \(V_1\) and \(V_3\) there are three edges representing three flow records from IP address \(V_1\) to IP address \(V_3\) with different port numbers.

According to given sample, there are three edges between \(V_1\) and \(V_3\). The vector \(V^{(k)}\) represents the connections of the \(k\)-th VM with other VMs, where \(V^{(1)}\), \(V^{(2)}\), and \(V^{(3)}\) can be expressed by Eqs. (16)–(18). The matrix \(V_{3 \times 3}\) denoting the connections of the three VMs, is expressed by Eq. (19).

\[
\begin{align*}
V^{(1)} &= \begin{pmatrix} 0 & 0 & 3 \end{pmatrix}^T \quad (16) \\
V^{(2)} &= \begin{pmatrix} 0 & 0 & 2 \end{pmatrix}^T \quad (17) \\
V^{(3)} &= \begin{pmatrix} 3 & 2 & 0 \end{pmatrix}^T \quad (18)
\end{align*}
\]

\[
V_{3 \times 3} = (V^{(1)}, V^{(2)}, V^{(3)}) = \begin{pmatrix} 0 & 0 & 3 \\ 0 & 0 & 2 \\ 3 & 2 & 0 \end{pmatrix}
\]  

The normalized \text{vm}_k can be calculated by Eq. (20), where \(u\) represents the total number of VMs.

\[
\text{vm}_k = \frac{\text{sum}(V(:, k))}{\sum_{i=1}^{u} \text{sum}(V(:, i))}
\]  

The method proposed here can be used to describe the anomalous performance of VMs. Estimating the anomalous performance of VMs involves evaluating the situation and trend of the states of the VMs in the cloud.

3 Model Methodology

3.1 Snort

Most security concerns have been addressed, and applying traditional security can prevent most intrusions by setting up defenses for each VM\cite{6}. Deploying Intrusion Detection Systems (IDS) on the critical network flow entry is also a feasible solution\cite{7}. Traditional IDS\cite{8,9}, intrusion prevention systems, and firewalls can be used to detect attacks in cloud computing.

Snort\cite{10} is a free and open source Network Intrusion Prevention System (NIPS) and a Network Intrusion Detection System (NIDS). Snort has the ability to analyze traffic in real time and log packets. Based on different configurations, Snort has a sniffer mode, a packet logger mode, and a network intrusion detection system mode\cite{11}.

We propose using a Bayesian classifier and Snort to detect network intrusions in cloud computing environments (see also closely related work in Ref. [12]). This approach has few false positives and affordable computational cost. An OpenFlow and Snort-based Intrusion Prevention System (IPS) is integrated to detect intrusions and deploy countermeasures by reconfiguring cloud computing. Our experimental results demonstrate the feasibility of this approach (see also closely related work in Ref. [13]).

3.2 Data processing

3.2.1 OpenFlow

OpenFlow is an open protocol to program a flow table to deploy new protocols, without changing any networking devices, and it implements programmable networks. It thus makes it possible to experiment on production networks, without danger to operations. McKeown et al.\cite{14} pioneered the control and forwarding separation architecture of OpenFlow. OpenFlow maintains a FlowTable in various switches and routers. The FlowTable includes packet-forwarding...
rules. According to the FlowTable, when a packet arrives at the network device, the rule set determines the packet forwarding. With programmable features, OpenFlow enables networks to reconfigure based on new rules. The paper proposed a new framework that implements network security monitoring using OpenFlow in cloud computing (see also closely related work in Ref. [15]).

3.2.2 Traffic redirection

Internal virtual networks are invisible in cloud computing because their communication traffic does not flow in the same physical machine. Insider threats could increase the chance of malware infection of internal VMs and hosts from unknown neighbor applications. Therefore, a large volume of traffic is out of control. This model employs OpenFlow to build a virtual network, so that the virtual switch network traffic runs through the physical network card, and the network traffic flows to the deployed system with our programs. OpenFlow then allows all the network flows to be inspected.

Figure 4 shows the virtual machine network traffic redirection. The model makes use of OpenFlow technology. OpenFlow can redirect the network traffic of VMs in the same physical machine to the deployed system. This solves the problem that the inter-VM traffic cannot be monitored and managed. And then the model employs OpenFlow to reconfigure control rules to prevent attacks.

3.2.3 Algorithm

We designed Algorithm 1 to get information from flows or packets. The data processing module includes data packet parsing, reorganization of flow session, packet statistics, flow statistics, and a data access interface. NPC is the captured network package collection, which cannot use Snort to detect anomalous network behaviors; $F$ is a flow attribute vector set; and $f_i$ is a property of the flow. $G$ is a data packet attributes vector set and $g_i$ is an attribute of a packet. $M_p$ is a vector of statistical properties of a packet. $M_f$ is a vector of statistical properties of the flow.

A function of the data processing module is to prepare the dataset used by other modules. The system provides a uniform data access interface in order to perform quick and effective behavior detection.

3.3 Application behavior analysis

3.3.1 Application classification

The variety of network applications in cloud computing has dramatically increased along with the growth of users. Accurate application traffic identification and classification is important for anomaly detection. This paper represents four goals of traffic classification, one of which is detecting unknown application or malicious flows[16]. Based on different grained features of network traffic, our research focuses on packet and flow data for traffic classification. At the packet level, the information is collected from packet headers and, optionally, parts of the payload. The
IP quintuple of transport protocol, source IP address, destination IP address, source port, and destination port are common properties of a flow. At the flow level, the information can be collected from flow statistics. Network traffic classification has attracted many researchers over the past few years[17–20]. We focus on behaviors of applications when they deviate from normal behavior. This is a motivation of the work presented in this paper.

The main characteristics of the network traffic used to identify the application are number of packets or bytes per second, number of packets payload (only one byte), number of packets payload (greater than one byte), sequence of number of byte on the first five packets payload, \( D_{\text{stat}} \) of packets payload, \( D_{\text{stat}} \) of packets interval, and \( D_{\text{stat}} \) of TTL. \( D_{\text{stat}} \) represents the statistical value of one characteristic, which contains minimum, maximum, variance, mean, median, and deviation.

Application behavior analysis consists of two steps. The first step aims to identify applications. This module manipulates the training examples and produces multiple classifiers to improve the application classification accuracy. The second step aims to detect anomalous behaviors of the application. This paper adopts the AdaBoost algorithm given in Ref. [21]. AdaBoost produces a sequence of \( k \) classifiers, such as \( K \)-Means, Support Vector Machines (SVM), etc. The weight for all training examples is equal at beginning. In each iteration, the error of the previous classifier is calculated. If it is too large, delete the iteration and exit. Training examples that are incorrectly classified by the previous classifiers are given higher weights for the next classifier[22]. The iteration stops until the error rate reaches a predetermined value.

Figure 5 shows the process of application classification. A application classifier is learned from the labelled training samples during the training phase and then the class label of every application is obtained from the trained classifier in the classification phase. Traffic samples that contain various applications (such as HTTP, QQ, PPLIVE, DNS, SSH, MSN, POP3, etc.) are collected. The module then uses time series technology to analyze applications. As mentioned previously, each module will get information from the data processing module. After identifying applications, this module gets various applications as input and then we use time series analysis method to detect anomalies based on application behaviors.

Tsinghua Science and Technology, June 2016, 21(3): 322–332

3.3.2 Time series analysis

The characteristics of networking behaviors are also closely correlated with history data \( (T_1, T_2, T_3, \text{WT}, \text{MT}, \text{YT}) \) as mentioned in Section 2.3. The time series is defined as in Ref. [23].

\[
\text{TS} = \{T_1, T_2, T_3, \text{WT, MT, YT}\}
\]  

\[
C = (C_1, C_2, \ldots, C_m)
\]

where \( m \) is the total number of the application characteristics, and \( C_i \) represents the value of the \( i \)-th feature can be any characteristic of a network application (such as byte counts, packet counts, number of connection requests, source mask bits, destination mask bits, incoming and outgoing traffic, duration, average connection duration, protocol, packet rate, maximum or average packet, etc.). \( \hat{C} \) is the predicted value at time \( T \). \( \theta \) determines whether the application behavior deviates from normal. This means that some deviation between the forecast values and the values can be observed. This deviation is given by Eq. (22).

\[
\theta(T) = C(T) - \hat{C}(T) = (\theta_1, \theta_2, \ldots, \theta_m)
\]  

Here’s how to compute \( \hat{C} \) using Eq. (23). \(|\text{TS}|\) denotes the size of the set TS.

\[
\hat{C}_i = \frac{1}{|\text{TS}|} \sum (C_i(t)) = \frac{1}{|\text{TS}|} (C_i(T_1) + C_i(T_2) + C_i(T_3) + C_i(\text{MT}) + C_i(\text{YT}))
\]  

However, if the detection time is too short, you cannot show a regularity; if the time is too long you will have a lot of historical data as a basis, which is the next key issue to be resolved, along with determining threshold \( \theta \).

3.4 Decision analysis

In order to improve detection accuracy, the decision analysis module uses the anomaly records for in-depth
analysis, where the various computation processes are described in Algorithm 2. This module uses a self-training algorithm, which is an incremental algorithm. The known and unknown records of anomalous behaviors are used to construct a sample library. In this module, a Naive Bayesian classifier[22] is trained with the labeled set, which is applied to classify the unlabeled set. Then, the highest-confidence samples are added to the labeled samples. This process iterates until all the unlabeled samples are added to the labeled samples.

The features $A = (A_1, A_2, ..., A_m)$ are extracted from anomaly records, which are used to construct a sample library. If a number of labeled samples meet the condition, the system will get results through self-learning. The number of labeled samples will affect the final result, which is the next problem to be solved. Naive Bayesian is a classifier $F$. The task of classification can be regarded as estimating the class posterior probabilities. In this module, there are two classes. One is anomalous behaviors, the other is normal behaviors.

Each sample is assigned to its most probable class. The self-training algorithm[22] is given. $Z$ is labeled samples, $(a_1, a_2, ..., a_m)$ represents observed attributes. Using label samples $Z$, train Naive Bayesian classifiers $F$. This classification is then used to classify the unlabeled sample $Q$: then the highest confidence samples are added to the labeled samples. This process iterates until all the unlabeled data have been given class labels. This module aims to find out which applications have anomalous behaviors.

### 4 Experiments

The approach is able to establish a behavioral baseline of normal network activity for each service, and then when network activity deviates from a baseline, anomalous activity will be detected. Zhao et al.[24] proposed detection botnets for classifying network traffic behavior, and that it is possible to identify the presence of existing and unknown botnet activity with high accuracy. Lin et al.[25] proposed a behavior-based approach that can detect known and even unknown malware. Koch et al.[26] used behavior-based techniques to detect intrusions in encrypted environments. Behavior profiles of each VM and service are used to detect cooperative anomalous behavior in our approach.

In order to detect anomalous network behaviors in cloud computing, we propose the model presented in Fig. 2. For illustration purposes, a cloud environment with several nodes is set up and we have used this platform to develop the security architecture for IaaS[27]. We deploy an experimental cloud computing platform based on a QEMU emulator v2.0.0 (Debian 2.0.0+dfsg-2ubuntu1), OpenStack IceHouse, and OpenFlow v1.3.

We use the KDD-99 dataset as training data, which is used for the Third International Knowledge Discovery and Data Mining Tools Competition[28]. It contains 4,898,431 network connections with 41 network traffic features. There are seven discrete-valued features, and others are continuous-valued features. KDD-99 is well-known and widely used for network attack detection[29–31]. The system will first preprocess some text features into numeric features. As shown in Table 1, the service type “UDP” is mapped to 2. Then the system transforms continuous-valued features into discrete-valued features.

KDD-99 is partitioned into ten equal-size disjoint subsets as training data, including six services in Table 1. For testing purposes, our system focuses on the same types of application traffic. Table 2 shows data distribution of connection records on six services, and

```
Algorithm 2: Decision Analysis

Input Data: Features of Network Traffic
Output Data: $y = \{\text{yes, no}\}$
1: While classifier $F$ use labeled samples $Z$
2: unlabelled samples is not null
3: $F(Q)$;
4: for $r \leftarrow 1$ to $|Q|$ do
5: Compute per class $Pr(C = c_j)$
6: Compute per feature $Pr(A_i = a_i | C = c_j)$
7: $c_1 = Pr(c = \text{yes}) \times \prod_{i=1}^{A} Pr(A = a_i | c = \text{yes})$
8: $c_2 = Pr(c = \text{no}) \times \prod_{i=1}^{A} Pr(A = a_i | c = \text{no})$
9: if $(c_1 > c_2$ and $c_1 > \epsilon)$ then
10: $q.y = \text{yes}$;
11: end if
12: if $(c_2 > c_1$ and $c_2 > \epsilon)$ then
13: $q.y = \text{no}$;
14: end if
15: Add $q.y$ to $Z$
16: remove $q$ from $Q$
17: end for
18: end while
```
Table 1 Data transformation.

| Types | Class | Value |
|-------|-------|-------|
| Protocol | TCP | 1 |
| | UDP | 2 |
| | ICMP | 3 |
| Service | login | 1 |
| | http | 2 |
| | shell | 3 |
| | smtp | 4 |
| | ssh | 5 |
| | telnet | 6 |

average accuracy of classification.

In this experiment, the dataset is partitioned into ten equal-size disjoint subsets. The 10-fold cross-validation method is used. As shown in Fig. 6, this approach is able to classify almost one hundred percent of normal traffic. Detection of attack traffic decreases by approximate 3%-8% when the dataset is unbalanced for each class, which is left for future work. The results show that the proposed algorithms are able to classify a majority of the attack traffic. The experimental results indicate that the effectiveness of our approach is more than 90%, and the model can detect attacks accurately.

A Receiver Operating Characteristic (ROC) curve is used to evaluate classification results. We aggregate the classification results, and demonstrate the effectiveness of this model. Figure 7 shows ROC curves for six services. Considering some acceptable behaviors can be classified as unacceptable, we plan to further evaluate the proposed approach using false negative analysis in the future. In terms of the per-service attack sample rate, “login”, “shell”, and “ssh” have the best classification performance across all services, due to the existence of large and long-duration attack flows in the training data. The effectiveness of the algorithms are evaluated in terms of its ability to distinguish attack traffic from normal traffic. We focus on the six services in this work and leave other types of services for future work. The experimental results show the feasibility and accuracy of our proposed approach.

5 Conclusion

This paper presents an anomalous behavior detection model in cloud computing that takes into account hybrid data sources and hybrid approaches. Our proposed detection model can deal with both discrete and continuous attributes. Experimental results show that it has high precision values and low recall values. The model uses SDN programmable technology to solve the inter-VM network traffic that cannot be monitored. The VM states are analyzed to propose efficient countermeasures to fuse several analysis approaches for preventing and handling the anomalous traffic of VMs.

A good direction for future work would be to study weights of samples and optimizing parameters of the proposed algorithm. We also hope to combine a deep learning algorithm and genetic algorithms to improve the accuracy of the model.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61272447) and the National Natural Science Foundation of China.
Fig. 7 ROC curves of six services.
Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z. Zhang, A modular machine learning system for flow-level traffic classification in large networks, *ACM Transactions on Knowledge Discovery From Data* (TKDD), vol. 6, no. 1, p. 4, 2012.

A. Tongaonkar, R. Torres, M. Illofotou, R. Keralapura, and A. Nucci, Towards self adaptive network traffic classification, *Computer Communications*, vol. 56, no. 1, pp. 35–46, 2015.

Y. Freund and R. E. Schapire, Experiments with a new boosting algorithm, in *Int’l Conf. Machine Learning (ICML)*, 1996, pp. 148–156.

B. Liu, M. J. Carey, and S. Ceri, *Web Data Mining*. Springer, 2011.

G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2008.

D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, Botnet detection based on traffic behavior analysis and flow intervals, *Computers & Security*, vol. 39, pp. 2–16, 2013.

Y. D. Lin, Y. C. Lai, C. N. Lu, P. K. Hsu, and C. Y. Lee, Three-phase behavior-based detection and classification of known and unknown malware, *Security and Communication Networks*, vol. 8, no. 11, pp. 2004–2015, 2015.

R. Koch, M. Golling, and G. D. Rodosek, Behavior-based intrusion detection in encrypted environments, *Communications Magazine*, vol. 52, no. 7, pp. 124–131, 2014.

L. Chen, X. S. Chen, J. F. Jiang, X. Y. Yin, and G. L. Shao, Research and practice of dynamic network security architecture for IaaS platforms, *Tsinghua Science and Technology*, vol. 19, no. 5, pp. 496–507, 2014.

KDD Cup 1999 Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 2015.

P. A. R. Kumar and S. Selvakumar, Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems, *Computer Communications*, vol. 36, no. 3, pp. 303–319, 2013.

S. S. Sathy, R. G. Ramani, and K. Sivaselvi, Discriminant analysis based feature selection in kdd intrusion dataset, *International Journal of Computer Applications*, vol. 31, no. 11, pp. 1–7, 2011.

P. Casas, J. Mazel, and P. Owczarski, Unsupervised network intrusion detection systems: Detecting the unknown without knowledge, *Computer Communications*, vol. 35, no. 7, pp. 772–783, 2011.

---

**Xiaoming Ye** is a PhD candidate at College of Computer Science of Sichuan University. She got the BE degree from College of Information Engineering of Jiangnan University in 2005 and MS degree from College of Computer Science of Sichuan University in 2008. Her research interests include cyber security and big data analytics.

**Xingshu Chen** received the PhD degree from Sichuan University in 2004. She is now a professor of the College of Computer Science and Cybersecurity Research Institute of Sichuan University. She is the member of China Information Security Standardization Technical Committee. Her research interests include cloud computing, cloud security, distributed file system, big data processing, network protocol analysis, and new media supervision.

**Haizhou Wang** received the BE degree and PhD degree from College of Computer Science, Sichuan University, China, in 2008 and 2014, respectively. From 2013 to 2014, he visited University of Toronto. He is currently a lecturer in the College of Computer Science, Sichuan University, China. His research interests include peer-to-peer streaming system, information security, and network measurement.

**Xuemei Zeng** is a PhD candidate at College of Computer Science of Sichuan University. She received the MS degree from Computer Science College of Sichuan University in 2004. Her current research interests include computer and network security, big data, and cloud computing security.

**Guolin Shao** is a PhD candidate of College of Computer Science of Sichuan University. He got the BE degree from Sichuan University in 2013. His general research interests lie in cyber security.

**Xueyuan Yin** is a PhD candidate at College of Computer Science of Sichuan University. He got the BE degree from Sichuan University in 2008. His research interests mainly focus on computer network and information security.

**Chun Xu** received the PhD degree from Sichuan University in 2008. He is now an associate professor of the College of Cybersecurity Research Institute of Sichuan University. His research interests mainly focus on computer network and information security.