IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 9, 2020, accepted August 27, 2020, date of publication September 3, 2020, date of current version September 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3021415

Dynamic Control Method for Tenants' Sensitive Information Flow Based on Virtual Boundary Recognition

XIN LU, LIFENG CAO, AND XUEHUI DU
He'nan Province Key Laboratory of Information Security, Zhengzhou 450001, China
Zhengzhou City Science and Technology Department, Zhengzhou 450001, China

Corresponding author: Lifeng Cao (caolf302@sina.com)

This work was supported in part by the National Natural Science Foundations of China under Grant 61502531 and Grant 61702550, and in part by the National Key Research and Development Plan under Grant 2018YFB0803603 and Grant 2016YFB0501901.

ABSTRACT In the cloud environment, owing to the large-scale sharing of the upper application instance and the underlying virtual machine resources, the tenants' information flow boundary in the shared virtual machine is fuzzy and difficult to identify. In addition, protection of tenant information flow between processes is inadequate, resulting in the leakage of sensitive information of tenants. Therefore, a dynamic control method for tenants' sensitive information flow based on virtual boundary recognition is proposed. By analyzing the behavior and operation log of tenants, the behavior feature vectors of tenants are constructed, and an automatic recognition algorithm of tenant virtual boundary based on the dynamic spiking neural network is designed. This algorithm can realize dynamic identification of the tenant virtual security boundary when the application service demand changes dynamically. Further, combined with the concept of centralized and decentralized information flow control, a dynamic control method of sensitive information flow is established. The security label is formally defined by using the lattice structure, and the control rules of tenants' information flow and the rules of tenant label encryption–declassification are designed. Thus, the independent, dynamic and secure control of tenants' information flow inside and outside the tenant virtual boundary. Finally, the detailed design of a dynamic security control application system for cloud tenants' sensitive information flow is provided. Experiments confirm that the proposed algorithm can identify the security boundary of tenants more accurately and efficiently than the traditional spiking neural network classification methods. Further, the security and effectiveness of the method is verified by the intransitive noninterference theory and the experiment of information flow control.

INDEX TERMS Tenant boundary identification, spiking neural network, information flow control, security label, label encryption and declassification, label tracking.

I. INTRODUCTION

Currently, cloud computing is a major innovation of the information technology service mode, realizing multi-tenant sharing and distribution on demand [1]. However, although the characteristics of cloud computing bring great convenience to tenants, they pose a serious threat to the security of tenants' sensitive data [2]. The cloud platform has the following characteristics [3], [4]:

1. Public infrastructure. This breaks down the barriers between physical resources, rendering the security boundary of tenants in the virtual network environment fuzzy and weak. As a result, it is difficult to effectively identify the virtual security boundary of tenants, leading to challenging security isolation of tenants' data.

2. Cloud management of tenant information. In cloud service outsourcing, the applications and information of tenants are not controlled and managed by tenants themselves but handled by cloud management. This can easily lead to illegal access and disclosure of internal information of virtual machines by untrusted programs in the cloud; thus, effective security of sensitive information cannot be ensured.

3. Large scale, high degree of openness, multi-tenant resource sharing. The relationship between tenants is

The associate editor coordinating the review of this manuscript and approving it for publication was Jing Bi.

complex, and malicious tenants can break the virtual isolation boundary of other tenants and illegally obtain sensitive information.

Given the cloud platform characteristics and the existing security problems, this paper summarizes the following security requirements of cloud tenants: 1. The dynamic upper-level tenants' application behavior, the sharing of physical instance resources, and the decentralized distribution of virtual machines require that the virtual security boundary of the tenants under the software definition can be identified accurately; 2. Tenants cannot fully trust the programs and services provided by the cloud platform and need to protect sensitive information from disclosure or illegal use autonomously; 3. Tenants cannot fully trust other tenants sharing cloud resources with them, so they need to prevent illegal flow of information to other tenants and be able to dynamically control the security sharing of information. To meet the above-mentioned requirements, this study investigates and contributes in the following aspects:

1. We propose a dynamic control method for tenants' sensitive information flow based on virtual boundary recognition. This method combines the automatic learning algorithm of tenant virtual boundary recognition and the dynamic control method of cloud tenants' sensitive information flow to realize security protection of tenant's sensitive information in cloud.

2. We extract the key characteristics of tenants through deep mining of the behavior of tenants and analysis of the operation log. After quantification, normalization, and coding of the key features, we construct the behavior characteristic vectors of tenants. Based on an improved dynamic spiking neural network learning algorithm to train and learn sample data, we perform automatic identification of the tenant operation process in a shared virtual machine instance, which establishes the virtual security boundary between tenants.

3. We propose a dynamic control method of sensitive information flow of cloud tenants on the basis of the identification of the virtual boundary of tenants combined with the concept of centralized and decentralized control mechanisms of information flow. This method can realize self-control of tenant information flow within the boundary as well as dynamic control and security sharing of information flow between tenants.

4. Based on the effective identification of the tenant virtual boundary and the dynamic control method of tenant information flow, we provide a detailed design of the dynamic security control application system of cloud tenants' sensitive information flow.

5. We build a cloud platform through OpenStack, monitor virtual machines on it, analyze tenants' resource information and log information, and obtain sample data. By using the sample data for performing several training and testing experiments, we verify the accuracy and efficiency of the boundary recognition algorithm.

6. In this study, we use the intransitive noninterference theory to confirm the efficiency of the security control application system of cloud tenants' sensitive information flow. With the help of Linux security module (LSM) [5] framework, the security verification experiment of tenant information flow control is carried out.

II. RELATED WORK

''Multi-tenant architecture'' refers to the architecture mode of sharing the same system or program components in a multi-user environment and is one of the most basic features in cloud computing. Multi-tenant architecture requires tenants to ensure the mutual isolation of information among tenants on the premise of sharing physical resources [6], [7]. Therefore, security isolation of tenant data is the key to design multi-tenant architecture and the most important aspect to be considered to ensure security isolation of sensitive information among tenants.

The protection of tenants' sensitive information based on tenant isolation mainly refers to preventing the illegal flow of information among tenants by dividing the tenant security domain and combining with a system isolation method to ensure that tenants' sensitive information and private business are not interfered with by other tenants [8]. Through the analysis of the above definitions, we can see that the key to the implementation of tenants' sensitive information protection is as follows: 1. Realize the effective division of the tenant security domain, and 2. control the legitimate flow of information inside and outside the tenant security domain to prevent leakage of sensitive information.

1. The identification of the tenant system boundary is the basis of division of the tenant security domain; the tenant system boundary can effectively establish the scope of tenant security domain and serve as the basis of tenant information flow security control. Tenant system boundary identification mainly includes two parts: identification of the physical boundary and the virtual boundary. The physical boundary refers to the multi-tenant system-level boundary. Each tenant can use one or more virtual machines to carry application programs and save data. Its identification includes different network addresses such as system IP address. The most typical method for identifying the physical boundary is to divide VLANs [9]; each tenant has a VLAN, but the number of VLANs is limited, so the requirement of large-scale tenants cannot be met. The boundary identification is mainly through manual static identification of IP, with complex configuration and low efficiency. Therefore, overlay network architecture, which is a type of virtualization technology mode superimposed on the network architecture, emerges as a current requirement. Its typical technical implementation includes VXLAN (Virtual eXtensible LAN) and NVGRE (Network Virtualization using Generic Routing Encapsulation) [10, 11]. The VXLAN protocol greatly increases the number of VLANs and realizes cross-regional two-layer interconnection. However, the burden of the VTEP node (i.e., virtual tunnel terminal) is extremely heavy, affecting the overall performance of the network. The NVGRE protocol is mainly encapsulated by using the generic routing encapsulation protocol (GRE). Its maximum number of subnet divisions is the

same as that of VXLAN, and its broadcast mode is more flexible. However, it does not use the standard transmission protocol, resulting in high equipment requirements. These two network types can effectively realize isolation of multi-tenant systems and expand the number of tenant networks. However, the identification of the tenant network boundary mainly depends on the artificial judgment of rental relationship and static identification through the IP address and other ways, so the methods cannot better adapt to the dynamic changes of the boundary.

The virtual boundary mainly refers to the virtual security boundary based on software definition. Greater emphasis is placed on the effective identification of the process level boundary when the tenant application shares the same system instance (such as virtual machines). That is to cut the upper application environment of different tenants at the bottom process level and ensure the mutual isolation of tenant process communication. The software-defined network (SDN) [12], [13] mainly refers to a new network architecture based on network virtualization, which separates the control plane and the data plane of network equipment. By increasing the programmability of the network, it innovates the current partial static and configures a complex network architecture; further, the network can be dynamically constructed according to the application requirements of the upper tenants. Because of its high flexibility and dynamic nature, the tenant security boundary is fuzzy and difficult to be identified on the basis of software definition. The software-as-a-service (SaaS) [14], [15] platform provides software services through a network by deploying the application system on the suppliers' own servers and delivering application services to various tenants based on tenant subscription. Multiple tenants share the same physical instance under the application, because they share the SaaS platform. In addition, because of the complexity of tenant identity and information interaction, the virtual communication security boundary between bottom tenants becomes fuzzy. In addition, most of the existing cloud management technologies [16], [17] mainly focus on improving efficiency, reducing cost, and maximizing revenue, and they do not fully consider the identification of the data security boundary and the security of data flow when tenants share cloud data center resources.

Owing to their learning characteristics and adaptability, neural networks are well suited for tenant boundary classification and recognition [18]. The traditional neural network requires prior knowledge of the number of neurons in the hidden and output layers; in order to improve the reliability of training, such as in the backpropagation neural network and fixed spiking neural network, it is necessary to know the proportion of distribution of all samples in advance. However, in the cloud environment, the number of tenants and virtual machines is constantly changing. In addition, the tenant behavior is complex and the virtual security boundary of tenants changes dynamically, making collection of samples at one time difficult. As a result, the traditional neural network cannot meet the needs of adaptive identification of the tenant boundary.

As the third generation of artificial neural networks, spiking neurons are more biomimetic than traditional neurons [19]. However, the number of neurons in the hidden and output layers of a spiking neural network with a fixed structure is determined in advance, which can only be used when the number of classes is known. In view of continuously changing data, a dynamic adaptive spiking neural network [20] is proposed to realize the dynamic increase of output layer neurons and meet the requirements of tenant boundary dynamic identification. However, there are still some problems in the existing research. Thorpe [21] proposed a spiking neural network learning algorithm based on the firing order (Rank order); it was emphasized that the first firing pulse sequence of neurons cannot reflect the pulse information well, and the effect of imprecise time on classification is ignored. Wang [22] proposed a learning algorithm of spiking neural networks based on precise time (only precise). Although it improves the classification accuracy, it increases the classification time, affecting the classification efficiency. Therefore, it is not suitable for large-scale tenant data training in a cloud environment.

2. In view of the security flow of information inside and outside the tenant security domain, it is mainly realized by the way of access control. The traditional access control model [23], [24] includes mandatory access control and autonomous access control. The traditional mandatory access control method strictly regulates the one-way flow of data; it is high in security but low in flexibility and practicability. By contrast, the autonomous access control model cannot guarantee the security of data after they are accessed by using the access control list that controls the access of subjects to objects. Therefore, information flow control technology has become essential. It mainly tracks and controls the flow of data in the system by ''sticking'' the label with policy requirements on the data so as to ensure the data's safe use. The most classical method of information flow control is the lattice [25] model proposed by Denning in 1976. The lattice structure is used to formally describe the information flow strategy, the system state, and the transition relationship among the states. The security policy in the traditional information flow control mode can only be formulated by a security administrator, thus it has poor flexibility. In case of an error, the security or the availability of the system can be easily reduced. In view of the defects of traditional information flow control, Myers [26] proposed a decentralized label model and an extended security-type programming language (Java information flow, JIF), which realize decentralized information flow control. The security strategy is developed by the programmers themselves and considers the problem of declassification, which is not considered in the traditional information flow control method. However, it cannot effectively solve the security problem caused by untrusted nodes. Smalley [27] proposed an information flow control system ''SELinux'', which is essentially a domain-type and

multilevel security-based Mandatory Access Control (MAC) security system. However, it does not support dynamic adjustment of tags and policy. Asbesto [28] proposed a decentralized declassification method, which defines the owner and decryption of data, and defined the receiving and sending tags of processes. However, it does not consider the sharing of system resources among processes, and it cannot realize dynamic adjustment of tags. Based on the research of Asbesto, Histar [29] provided a memory sharing mechanism between processes and solved the problem of covert channels through the method of explicit adjustment of tags. Flume [30] proposed a method to control the information flow of the key resources of the operating system; this method can realize the dynamic adjustment of tags but requires high compatibility between the application software and the system. Weir [31] proposed a decentralized information flow control system; it however has a problem of coarse control granularity and does not support a user-defined information flow control strategy. Wu [32] presented a novel dynamic defense model (DDM) to reduce security risks brought by these suspicious data or codes for the open operating systems. In the model, dynamic label marking, dynamic label tracking, dynamic label modulating, and run-time controlling were given, which provided a good idea for the tracking and control of information flow, but this model was mainly aimed at the security protection of a single operating system, and only the system application on Android was given.

Based on the above research, considering the characteristics of a cloud environment, information flow control technology has been extensively researched. CloudFence [33] realized fine-grained tracking of sensitive data in the cloud based on a pin plug-in platform; this approach effectively guarantees user data isolation and security sharing in the cloud, but it cannot support user-independent policy configuration and information flow control. FlowK [34] and FlowR [35] implemented process-level coarse-grained data propagation control in the cloud tenant operating system layer. Priebe [36] provided a lightweight monitoring framework "CloudSafetyNet" in the cloud environment; it enables tenants to monitor the flow of information between applications and detect the risk of information leakage by using the motivation of cooperation between tenants, but it is used mainly for the security detection of information flow. Wu [37] proposed a two-layer information flow control model for a cloud environment; the model realized the combination of centralized information flow and decentralized information flow control; however, no identity for the virtual security boundary of the tenant exists in the model. Pasquier [38] realized the combination of distributed information flow control technology and trusted platform technology and applied it to cloud data security enhancement; however, this method does not consider the protection mechanism of cloud platform service providers. LV [39] proposed noninterference for cloud architecture in which concurrent access and sequential access coexist, which realized the security control of information flow between security domains when concurrent and sequential actions were executed in the cloud, but the boundary of security domain was not clear, and the availability and security were not verified by experiments. According to the characteristics of cloud computing, Ma [40] formalized the process of the tenant information flow in the cloud computing system, proposed the corresponding separation rules, and verified the security of the separation rules through the noninterference theory. However, the separation rules mainly emphasized the isolation of resources among tenants, but did not design the rules of information flow security control between tenants and between tenants and the cloud platform in a fine-grained way, and also lacked experimental verification.

Although the above-mentioned research has been involved in aspects such as integrity and confidentiality assurance, policy management, label declassification, and mark propagation, it fails to fully consider these capability requirements. Therefore, in the face of the data security needs of cloud tenants and the real-time and dynamic nature of the tenant boundary, it is necessary to study a fine-grained autonomous dynamic control strategy for tenant information flow, which can not only realize the security flow of information within the tenant boundary but also ensure the security of information flow among tenants and between tenants and cloud service programs.

III. DYNAMIC CONTROL METHOD DESIGN OF TENANT SENSITIVE INFORMATION FLOW BASED ON VIRTUAL BOUNDARY RECOGNITION

The identification of the tenant boundary mainly serves the security control of tenant information flow. Based on the accurate identification of the virtual security boundary of tenants, combined with the security control method of the sensitive information flow of tenants, the security of the information flow inside and outside the tenant boundary is guaranteed and leakage of the sensitive information of tenants caused by malicious attack is prevented.

A. AUTOMATIC RECOGNITION OF TENANT VIRTUAL SECURITY BOUNDARY

1) TENANT BOUNDARY IDENTIFICATION PROBLEM DESCRIPTION

Figure 1 shows that one or more shared application instances are deployed to provide customized application services for tenants, allowing different tenants to share the same upper application instance and providing the bottom shared service resources for supporting applications. However, because the tenants must share the underlying virtual machine resources and the information flow between the tenants' processes in the shared virtual machine is transparent for the upper application, the service processes under different tenants may be distributed on the same virtual machine instance. In addition, the number of tenants and the dynamic changes in the upper application requirements are likely to cause frequent migration of tenant application services in the shared virtual machine. This renders the security boundary of tenant data

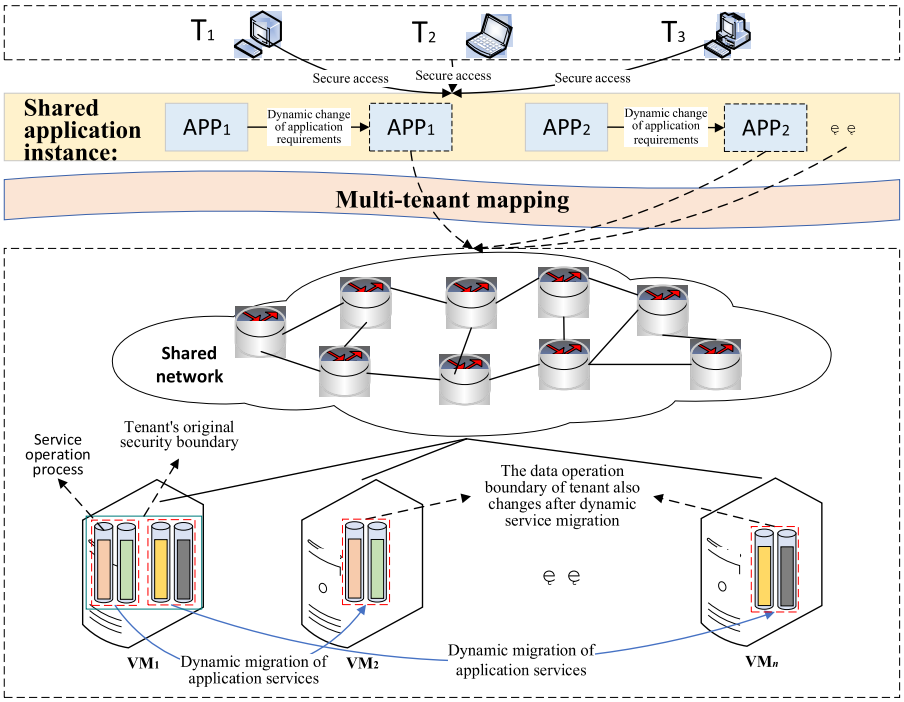

FIGURE 1. Tenant virtual security boundary based on software definition.

in the shared virtual machine fuzzy and difficult to identify dynamically. The traditional artificial static audit method for boundary identification cannot adapt to the dynamically changing virtual security boundary of tenants in real time. Therefore, this paper designs a dynamic control method for tenant sensitive information flow based on virtual boundary recognition (**D_SNNBAR**).

2) TENANT BEHAVIOR FEATURE EXTRACTION AND PROCESSING

In this study, the operation logs of tenants are collected and the key features of tenants are extracted for building the feature vectors for neural network learning. To mine the tenant feature information, first, the tenants and virtual machines in the cloud platform are monitored and the log information is analyzed to obtain tenant information, including the tenant related information obtained through monitoring of the virtual machine. Subsequently, the process of tenant connection is monitored, the key features are extracted together with tenant operation log information, and the feature vector is constructed.

Herein, we first use the virtual machine monitor (VMM) to obtain tenant registration and permission information, including tenant category (user group) T_{type}. Then, we extract the virtual machine related information, including the virtual machine identification number V_{id}, and obtain the process number P_{ID} of the virtual machine connected with the tenant. Further, we analyze log information to obtain operation information, including file name F_{NAME}, file path F_{PATH}, operation type F_{OM}, opening time F_{OT}, and closing time F_{CT}.

After collecting the tenant's key feature information, we construct the feature vector $\varphi = (T_{type}, V_{ID}, P_{ID}, F_{NAME}, F_{PATH}, F_{OM}, F_{OT}, F_{CT}) = (\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8)$. According to the different types and units of the above features, the vector is quantized and normalized before neural network learning.

a: QUANTIZING EIGENVALUES

Here, first, tenant categories are mapped; for example, $\mathbf{T_{ype}} = \{$Administrator, Senior, VIP, Normal$\ldots\}$ is mapped to $\varphi_1 = \{1,2,3,4, \ldots\}$. For V_{ID}, the quantization of $\mathbf{P_{ID}}$ can be used directly, that is, $\varphi_2 = V_{ID}$ and $\varphi_3 = P_{ID}$. For the quantization of F_{NAME} and F_{PATH}, we mainly use hash algorithm to map, that is, $\varphi_4 = \mathrm{HASH}(F_{NAME})$ and $\varphi_5 = \mathrm{HASH}(F_{PATH})$. In this paper, considering the hash operation of strings, we adopt the hash operation method based on multiplication. When the multiplier is set to 33, it has a good hash effect on English words. The hash algorithm design is shown in Table 1. For the quantization of $\mathbf{F_{OM}}$, the operation type $\mathbf{F_{OM}} = \{$new, read, write, update, delete, clear, $\ldots\}$ is also mapped to $\varphi_6 = \{1,2,3,4,5, \ldots\}$. For the operation time F_{OT} and F_{CT}, the unified time format is used for counting, with the unit seconds; that is, $\varphi_7 = T(F_{OT})$ and $\varphi_8 = T(F_{CT})$.

b: NORMALIZATION

To reduce the effect of different value range of variables in the feature vector for the neural network, neural network learning is facilitated, and considering each feature variable with the same importance, each feature variable is normalized. We use the method of deviation standardization, that is, x = (x-min)/

TABLE 1. Design of hash algorithm.

```
INPUT: KEY
OUTPUT: HASH(KEY)
1. HASH = 0
2. FOR I=0 TO KEY.LENGTH() DO
3.   HASH = 33*HASH + KEY.CHARAT(I)
4. ENDFOR
```

(max-min), where max and min are the maximum and minimum values of the characteristic variable, respectively; all the characteristic values are controlled within [0,1].

c: INFORMATION ENCODING

In the dynamic spiking neural network, we use the method of Gaussian group coding to transform each eigenvector to be input in a multiple-pulse pattern. This method is based on the Gaussian receptive field, which represents a series of operations related to a Gaussian function. According to this coding method based on the Gaussian hypothesis, suppose that the eigenvalue obeys multiple Gaussian distributions in the data sample space. First, different distributions are obtained by calculating different values of mean and variance of the eigenvalue. Then, the probability corresponding to the eigenvalue is calculated. Finally, multiple pulses corresponding to the eigenvalue are obtained according to the probability and the coding function. The coding process is as follows:

Assuming that the number of Gaussian receptive field is $\boldsymbol{n}$, each eigenvalue will be encoded into $\boldsymbol{n}$ pulses, and the dimension of the eigenvector is $\boldsymbol{m}$, then the encoded eigenvector will become $\boldsymbol{m}\times\boldsymbol{n}$ pulses. The formula of mean $\boldsymbol{u_i^j}$ and standard deviation $\boldsymbol{\delta_i^j}$ of the $\boldsymbol{i}$-th feature in the $\boldsymbol{j}$-th acceptance domain are described as equation 1 and equation 2:

$$u_i^j = \varphi_{min}^i + \frac{(2j-3)(\varphi_{max}^i - \varphi_{min}^i)}{2(n-2)} \tag{1}$$

$$\delta_i = \frac{1}{\beta}\frac{(\varphi_{max}^i - \varphi_{min}^i)}{(n-2)} \tag{2}$$

Among them, $\boldsymbol{\varphi_{max}^i}$ and $\boldsymbol{\varphi_{min}^i}$ are the minimum and maximum values of the $\boldsymbol{i}$-th eigenvalue, respectively; $\boldsymbol{\beta}$ is a parameter that affects the coverage of the Gaussian receptive field by affecting the standard deviation. The above values of mean and variance jointly determine the Gaussian function, which is described as equation 3:

$$\rho_i^j = e^{-\frac{(\varphi_i - u_i^j)^2}{2(\delta_i^j)^2}} \tag{3}$$

According to the results of Gaussian function calculation, the pulse time of each eigenvalue, that is, the pulse time of each input neuron, is calculated as equation 4:

$$t_i^j = \begin{cases} [T(1-\rho_i^j)], & \rho_i^j \neq 0 \\ -0.01, & \rho_i^j = 0 \end{cases} \tag{4}$$

3) ALGORITHM FLOW DESIGN

The process of the tenant virtual security boundary recognition algorithm, as shown in Figure 2, includes network initialization, eigenvector processing and input, information coding (Gaussian group coding), dynamic spiking neural network learning, and tenant boundary review and confirmation.

a: STRUCTURE OF DYNAMIC SPIKING NEURAL NETWORK

The dynamic spiking neural network structure includes an input layer, a coding neuron layer, and an output neuron layer, as shown in Figure 3. The input neuron of the coding layer uses Gaussian coding to transform the input eigenvalue into a series of pulse time.

$\boldsymbol{n}$ in Figure 3 represents the number of Gaussian receptive fields. The neurons in each coding layer generate a pulse time, which is transmitted to the next layer. The coding layer and the output layer are connected in a fully connected way. The number of neurons in the coding layer is determined by the dimension of the eigenvector and the Gaussian receptive field. At the beginning, there is no connection between the coding layer and the output layer of the neural network. When a new sample is input, the output layer will dynamically add a neuron. According to the pulse time of the coding layer, the weights of neuron connections are established. In the learning process, each neuron in the output layer will represent a category label. Finally, the neuron in the output layer will be updated or merged according to the dynamic learning algorithm.

b: LEARNING STRATEGY OF TENANT BOUNDARY AUTOMATIC RECOGNITION

In the process of automatic identification of the tenant boundary, the weight vector of an output layer neuron represents the clustering center of the tenant boundary category. The automatic recognition learning strategy of the tenant boundary includes two parts: initial weight adjustment and dynamic adjustment strategy of output layer neurons.

1) Initial weight adjustment

The weight connection formula between the output layer neuron and the coding layer is as equation 5:

$$w_{ij} = w_0 + \mathrm{rexp}(-\frac{t_i}{\tau}) \tag{5}$$

Among them, w_{ij} is the synaptic weight between coding layer neuron i and output layer neuron j, w_0 is the initial weight, t_i is the pulse time, and τ is the time constant.

2) Dynamic adjustment strategy of neurons

In the training process, the existing information in the neural network is compared with the information presented by the input samples. Learning strategies are then chosen according to the comparison results:

a. Addition of neurons: when the similarity between the existing output neuron category and the input sample data is lower than the threshold, new neurons are added to the output layer to represent the new category.

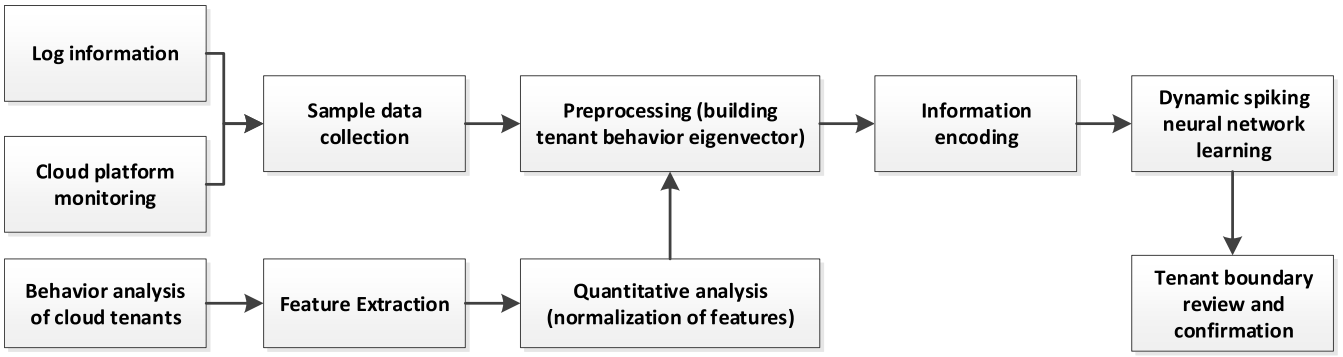

FIGURE 2. Tenant boundary identification process.

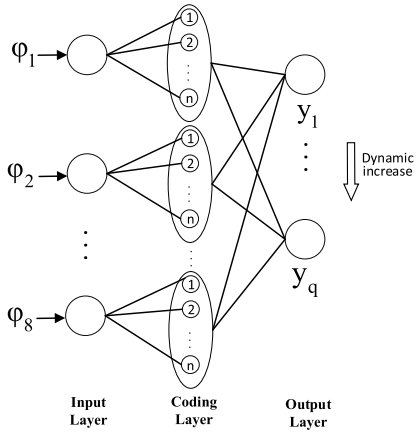

FIGURE 3. Structure of neural network learning.

b. Merge neurons: when the similarity between the input tenant sample data and the existing neuron tenant category exceeds the threshold, new neurons and the most similar neurons are merged and the weight of the merged neurons is updated; that is, the neurons are merged according to the similarity between the output neurons, thus achieving the classification effect.

c. After the classification is determined, the classification is identified. Here, the triple is used to define the classification (security boundary) identification number **Sec_boundary_ID**:

Sec_boundary_ID =(TID, VMID, PID), TID, VMID and PID are respectively tenant ID, virtual machine ID and process ID.

c: DETAILED PROCESS OF ALGORITHM IMPLEMENTATION

After quantizing and normalizing the sample data, the set of eigenvectors to be input is ψ. The algorithm is shown in Table 2.

Each output neuron represents a tenant boundary. Lines 3–12 calculate the weight vector of the output neuron connected with the coding layer; lines 14–18 classify the unclassified processes; lines 19–30 dynamically update the classified processes. The algorithm process indicates that the number of feature vectors and the number of Gaussian acceptance regions are determined, and the two values are relatively small, so the time complexity of the algorithm is O(n). In addition, the training process of the recognition algorithm does not need iteration, and only uses the key information and dynamic adjustment mechanism underlying the spiking neural network; thus, the efficiency of data classification is greatly improved.

B. DYNAMIC CONTROL METHOD FOR SENSITIVE INFORMATION FLOW OF CLOUD TENANTS

1) SECURITY ISSUE DESCRIPTION OF CLOUD TENANTS' SENSITIVE INFORMATION FLOW

Tenant sensitive information refers to private tenant data with a sensitivity level in the virtual machine. The sensitivity level mainly reflects the importance of the tenant's private data, such as tenant account information, tenant calculation data, and data information of different departments or users of the tenant, which requires fine-grained security control. The importance and use of tenant data information of different sensitivity levels differ; therefore, tenant sensitive information should be handled by specified service levels to avoid cross operation and flow, thereby preventing leakage.

In the process of renting cloud services, tenants upload their own data to the cloud platform for processing and lose the direct control of their own sensitive information to a certain extent, which seriously threatens the security of tenant data. If the security protection of tenant data depends only on the identification of the data security boundary and lacks an effective information flow control method, the data will be vulnerable to the cross-border attack of untrusted programs in the virtual machine or other tenants, resulting in the disclosure of sensitive information of tenants.

Due to the large scale of sharing of virtual machine resources among different tenant applications in the upper layer and the weakening of tenants' control over their own data, there is a possibility of illegal flow of information between processes inside and outside the tenant virtual boundary. Figures 4-① ② ③ depict the different security threats faced by the tenant information flow: ① A malicious

TABLE 2. Algorithm flow of D_SNNBAR.

Input: ψ
Output: Classification results

1. R=NULL, Y=0 /* Initialize the output neuron Library R; Y is the number of output neurons */
2. **for** i=1,z=0 to $|\psi|$ by 1 **do** /* Each feature vector is coded by Gaussian group coding */
3. **for** j=1 to 8 by 1 **do** /* Calculate the pulse time of each eigenvalue in the eigenvector */
4. $\delta[j] \leftarrow \frac{1}{\beta}\frac{(\varphi^j_{max}-\varphi^j_{min})}{(n-2)}$/* Calculate the standard deviation of the eigenvalue */
5. **for** k=1 to n by 1 **do** /* the number of Gaussian receptive fields is n*/
6. $u[j][k] \leftarrow \varphi^j_{min} + \frac{(2k-3)(\varphi^j_{max}-\varphi^j_{min})}{2(n-2)}$/* Calculate the mean value of the k-th acceptance region of the j-th eigenvalue */
7. $\rho[j][k] \leftarrow e^{-\frac{(\varphi_j-u[j][k])^2}{2(\delta[j])^2}}$/* Calculate the Gaussian function */
8. $t[j][k] \leftarrow \begin{cases}[T(1-\rho[j][k])], \rho[j][k] \neq 0 \\ -0.01, \rho[j][k] = 0\end{cases}$/*计算脉冲时间*/
9. $\boldsymbol{w}[i][z] \leftarrow w_0 + \text{rexp}(-\frac{t[j][k]}{\tau})$/* Generate the initial connection weight vector w[i] between the coding layer and the output layer*/
10. z←z+1
11. **endfor**
12. **endfor**
13. Traverse all output neurons and determine whether the new input sample data has been classified.
14. **if** (Flag==0) **then** /*Not classified yet, then create a new category c_p*/
15. $Y \leftarrow Y+1$ /* Add an output layer neuron*/
16. $\boldsymbol{w_p} \leftarrow \boldsymbol{w[i]}$ /* Set the weight vector */
17. $N_p \leftarrow 1$ /* The number of initial training samples on the output neuron is recorded */
18. **endif**
19. **else** (Flag==1) **then** /* Input sample data has been classified, and it belongs to C_n */
20. $p \leftarrow argmax\left(\frac{1}{(w-w_j)^2}\right), j \in [1,Y]$ /* Find the most similar neuron p */
21. **if** $(c_p = c_n)$&&(S(p)>Th_{sim}) **then** /* S(p) is the similarity, and Th_{sim} is the similarity threshold */
22. Merging output neurons c_p *and* c_n.
23. $\boldsymbol{w_p} \leftarrow \frac{w_p * N_p + w[i]}{N_p+1}$/* Update the weight vector*/
24. $N_p \leftarrow N_p+1$ /* Add a training sample */
25. **else if** $(c_p \neq c_n) \| ((c_p = c_n)$&&(S($p$)<$\text{Th}_{\text{sim}}$)) **then**/* Create a new category*/
26. $Y \leftarrow Y+1$
27. $\boldsymbol{w_p} \leftarrow \boldsymbol{w[i]}$
28. $N_p \leftarrow 1$
29. **endif**
30. **endif**
31. **endfor**

process of other tenants under the same virtual machine illegally accesses the information in the authorized process of a tenant, resulting in the outflow of sensitive information. ② Among different virtual machines, an untrusted process outside the boundary steals the authorized process within the boundary, resulting in the leakage of tenant sensitive information. ③ Information is illegally shared among tenants, for example, through virtual machine escape attack or DDoS attack.

For the security of the information flow within the boundary of a cloud tenant, the security strength of the data and the permission of the application in the virtual machine are determined by the cloud tenant itself, aiming to realize the centralized control of the information within the boundary by the tenant. The control strategy of information flow among cloud tenants is jointly formulated by participating tenants, and cloud tenants can only formulate their own information flow or data sharing security strategy with other cloud tenants for achieving distributed dynamic control of information flow among tenants.

2) DESIGN OF SECURITY LABEL

Definition 1: Security label L represents a set of security policies, each of which represents a tenant's security requirements for information, including confidentiality and integrity security requirements.

L is composed of the policy subject owner (i.e., the information owner or policy maker, identified by the boundary *ID* of the tenant) and value domain R (a collection of policy executors, determined by the owner). It is formally expressed as $L = (ID : R)$, and it includes two types: confidentiality label L_c and integrity label L_i. $L_c = (ID \rightarrow R)$ indicates that the owner of the information marked by the confidentiality label only allows the information to flow to the subjects in R; for example, $L_c = (ID_1 \rightarrow r_1, r_2)$ indicates that $r_1$andr_2 are allowed to read the information with the confidentiality label L_c. $L_i = (ID \leftarrow R)$ indicates that the owner of the information allows the subjects in R to write the information. In addition, for the data marked by the tag, the tag will follow the data in the whole system, and the object derived from the data will also inherit the original tag. Data, data owner and data operator are identified by the boundary identification number (**Sec_boundary_ID**).

Definition 2: Confidentiality label lattice G_c means that the confidentiality label system is abstracted by a lattice; that is, G_c=(L_c, $\wedge$, Δ_c, ∇_c) is used for the confidentiality protection of tenant data, where L_c represents the set of confidentiality labels, and for any label value, L_c.R belongs to the value domain of L_c. "$\wedge$" represents the intersection operator, and the result is the union "$\cup$" of the label set, which satisfies the following characteristics:

① Idempotence: $L_c.R_x \wedge L_c.R_x = L_c.R_x$;
② Exchangeability: $L_c.R_x \wedge L_c.R_y = L_c.R_y \wedge L_c.R_x$;
③ Associativity: $L_c.R_x \wedge (L_c.R_y \wedge L_c.R_z) = (L_c.R_x \wedge L_c.R_y) \wedge L_c.R_z$.

"$\wedge$" specifies the partial order relation "$\preccurlyeq$" on the value domain of the label, which satisfies reflexivity, antisymmetry, and transitivity. If $L_{c_1} L_{c_2}$ and $L_{c_2} L_{c_1}$, then $L_{c_1} = L_{c_2}$. For example: if L_{c_1}=($ID \rightarrow r_1, r_2$) and L_{c_2}=($ID \rightarrow r_1$), then $L_{c_1} L_{c_2}$, which indicates that L_{c_2} requires higher confidentiality. "Δ_c" represents the maximum upper bound of the value domain of the confidentiality label, indicating the maximum range of reading the data; "∇_c" represents the minimum lower bound of the value domain of the confidentiality label, indicating the minimum range of reading the data.

Definition 3: Integrity label lattice G_i means the integrity label system is abstracted by lattice; that is, G_i=(L_i, $\wedge$, Δ_i, ∇_i) is used for the integrity protection of tenant data.

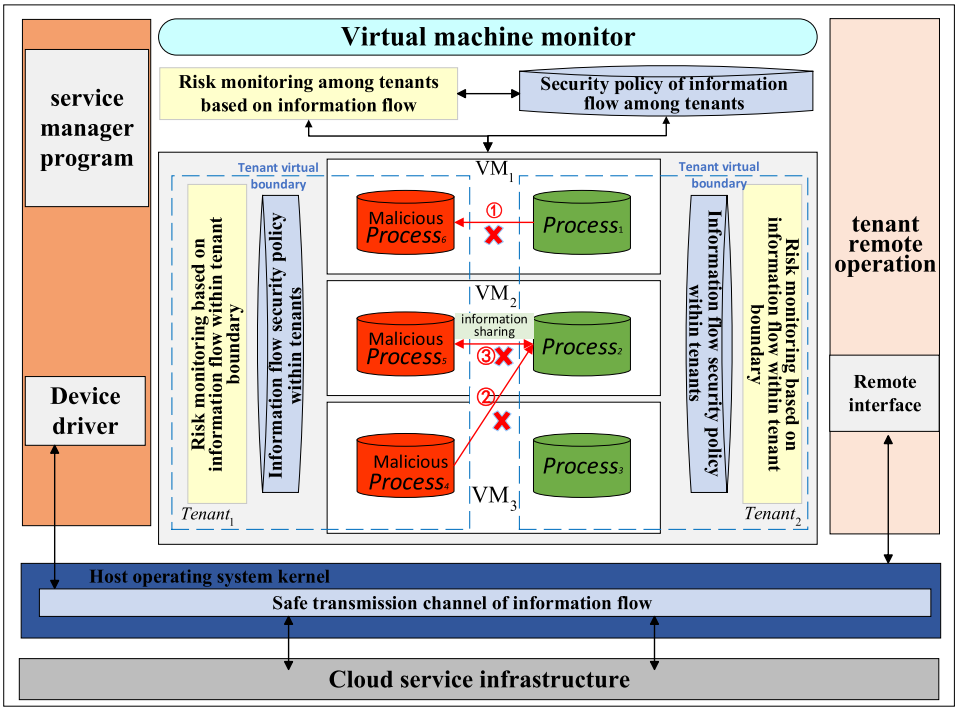

FIGURE 4. Cloud tenant information flow control architecture.

If the partial order relation is satisfied in the confidentiality domain, the opposite partial order relation is satisfied in the integrity domain. For example, if $L_{i_1} = (ID \longleftarrow w_1)$ and $L_{i_2} = (ID \longleftarrow w_1, w_2)$, then $L_{i_2} L_{i_1}$, which indicates that L_{i_1} requires higher integrity. "Δ_i" represents the maximum upper bound of the value range of the integrity label. From the dual relationship between data confidentiality and integrity, it can be seen that $\Delta_i = \nabla_c$. "∇_i" represents the minimum lower bound of the value range of the integrity label. Similarly, $\nabla_i = \Delta_c$.

Definition 4: Partial order of label lattice $L = (L_c, L_i)$, that is, the set of $L_c \times L_i$. L means to meet the partial order of the confidentiality label and integrity label simultaneously, which is described as equation 6:

$$\left(L_{c_1} \times L_{i_1}\right) \preccurlyeq \left(L_{c_2} \times L_{i_2}\right) = (L_{c_1} \preccurlyeq L_{c_2}) \bigwedge (L_{i_2} \preccurlyeq L_{i_1}) \quad (6)$$

In order to better understand the information flow control strategy, the symbol definitions are given in detail in Table 3.

3) INFORMATION FLOW SECURITY LABEL CONTROL STRATEGY

(1) Rule 1. Label value field minimization

If data is marked by labels $L_1 = (ID_1 : R_1)$ and $L_2 = (ID_2 : R_2)$, the security label of the data is the union of the two labels, that is, the intersection of the label value field, which is described as equation 7:

$$\text{if } L_{min} = L_1 \cup L_2 \text{then}\{L_{min}.R = L_1.R_1 \cap L_2.R_2;\} \quad (7)$$

Rule 1 indicates that operations on data satisfy the minimum privilege principle, and data flows only to subjects that satisfy all label policies. This rule is the security foundation of the data label propagation rule and information flow control among tenants.

(2)**Rule 2.** "and or" of the label value field

Rule 2.1: The "and" of the label value field means that data needs to be operated by multiple entities simultaneously, and a single entity cannot read or write data. The label is formalized as $L = (ID : R_1 \text{and} R_2)$, which expresses the principle of separation of authority and duty.

Rule 2.2: The "or" of the label value field refers to the priority of the main body's operation on data. The label is formalized as $L = (ID : R_1 or R_2)$. This rule specifies the operation order of R_1 and R_2, which cannot be operated simultaneously.

(3) **Rule 3**. Tenant information flow control rules

Assuming that the confidentiality label and integrity label of any two data, $data_1$ and $data_2$, are L_{c_1}, L_{i_1} and L_{c_2},L_{i_2}, respectively, the protection rules of data flow from $data_1$ to $data_2$ are as equation 8 and equation 9:

$$data_1 \longrightarrow data_2 \text{ if } (L_{c_1} \preccurlyeq L_{c_2}) \wedge (L_{i_2} \preccurlyeq L_{i_1}) \quad (8)$$

$$data_1 \leftrightarrow data_2 \text{ if } (L_{c_1} = L_{c_2}) \wedge (L_{i_2} = L_{i_1}) \quad (9)$$

Rule 3 indicates that the necessary condition for data flow is to meet the partial order relationship of the confidentiality label and integrity label of data simultaneously. The confidentiality label of data requires that tenant data can only flow

TABLE 3. Symbol definitions of information control strategy.

No.	SYMBOL	NOTION	DESCRIPTIONS
1	L_c	CONFIDENTIALITY LABEL	$L_c = (ID \rightarrow R)$, "ID" IS THE TENANT IDENTIFICATION (THE OWNER OF THE DATA), "R" REPRESENTS THE OPERATION VALUE FIELD (THE READER OF THE DATA);
2	L_i	INTEGRITY LABEL	$L_i = (ID \leftarrow R)$, "ID" IS THE TENANT IDENTIFICATION (THE OWNER OF THE DATA), "R" REPRESENTS THE OPERATION VALUE FIELD (THE WRITER OF THE DATA);
3	G_c	CONFIDENTIALITY LABEL LATTICE	$G_c = (L_c, \wedge, \Delta_c, \nabla_c)$, " L_c " REPRESENTS THE CONFIDENTIALITY LABEL, " $\wedge$ " REPRESENTS THE INTERSECTION OPERATOR, "Δ_c" REPRESENTS THE MAXIMUM UPPER BOUND OF THE CONFIDENTIALITY RANGE, "∇_c" REPRESENTS THE MINIMUM LOWER BOUND OF THE CONFIDENTIALITY RANGE;
4	G_i	INTEGRITY LABEL LATTICE	$G_i = (L_i, \wedge, \Delta_i, \nabla_i)$, " L_i " STANDS FOR THE INTEGRITY LABEL, " $\wedge$ " STANDS FOR INTERSECTION OPERATOR, " Δ_i " STANDS FOR THE MAXIMUM UPPER BOUND OF THE INTEGRITY RANGE, " ∇_i " REPRESENTS THE MINIMUM LOWER BOUND OF THE INTEGRITY RANGE;
5	$L_{\preccurlyeq}$	PARTIAL ORDER OF LABEL LATTICE	$L_{\preccurlyeq} = (L_c, L_i)$, "$L_{\preccurlyeq}$" MEANS TO MEET THE PARTIAL ORDER OF THE CONFIDENTIALITY LABEL AND INTEGRITY LABEL SIMULTANEOUSLY.
6	S_c	THE SET OF POLICY ADJUSTMENT	THE SET OF CONFIDENTIALITY LABEL POLICY ADJUSTMENT.
7	S_i	THE SET OF POLICY ADJUSTMENT	THE SET OF INTEGRITY LABEL POLICY ADJUSTMENT.
8	S_C^+, S_C^-	POLICY SET	"S_C^+": THE SET OF ADDITIVE CONFIDENTIALITY LABEL POLICIES, "S_C^+": THE SET OF REMOVABLE CONFIDENTIALITY LABEL POLICIES.
9	S_i^+, S_i^-	POLICY SET	"S_i^+": THE SET OF ADDITIVE INTEGRITY LABEL POLICIES, "S_i^-": THE SET OF REMOVABLE INTEGRITY LABEL POLICIES.

from a label with a weak constraint to that with a strong constraint in order to prevent data leakage. The integrity label of data requires that data flow only from high integrity to low integrity in order to prevent data from being polluted.

Based on the flow control rules of information flow, the control rules of sending and receiving processes of information flow in virtual machines are given here:

Rule 3.1: Set sending process P_1, sending data D_1, receiving process P_2, and receiving data D_2 according to the equation 10:

$$P_1 \rightarrow P_2 \text{ if } (P_1 \in L_{c_{D1}}.R)\&\&((L_{c_{D1}} \preccurlyeq L_{c_{D2}}) \wedge (L_{i_{D2}} \preccurlyeq L_{i_{D1}})) \&\&(P_2 \in L_{i_{D2}}.R) \quad (10)$$

The necessary condition for process P_1 to be able to send data D_1 to P_2 is that process P_1 belongs to the value domain of the confidentiality label of D_1, that is, flow from data D_1 to data D_2 must meet rule 3, and P_2 must be in the value domain of the integrity label of D_2.

(4) **Rule 4.** Propagation rules of label

Suppose that the labels flow with data_1 to data_2, and the security labels of data_2 needs to be updated. The updated security labels are $L_{c_{new2}}$ and $L_{i_{new2}}$, and the rules are as equation 11:

$$\text{if data1} \rightarrow \text{data2 then}\{L_{c_{new2}} = L_{c_1} \cup L_{c_2}; L_{i_{new2}} = L_{i_1} \cup L_{i_2};\} \quad (11)$$

This rule indicates that the label should be stricter after data flow, so the intersection operation of labels is followed, that is, the union of labels. The propagation of tags can be divided into two situations: ① In process operation, the propagation of internal tags; for example, assignment operation X = Y:, where the information in Y flows to X, and the label of X is updated to $L_X = L_X \cup L_Y$. ② Data transfer between processes; for example, process P transfers D_1 to process Q and stores it with D_2. Here, the label of data D_2 is updated to $L_X = L_X \cup L_Y$.

4) TENANT LABEL ADJUSTMENT STRATEGY

To complete the tenant's independent and dynamic control of data, the tenant's ability to adjust its own data security label is designed; this process is divided into label encryption and label declassification rules. To better realize the tenant's adjustment of labels, we introduce the set of confidentiality label policy adjustment "S_c" and the set of integrity label policy adjustment "S_i". S_c^+ represents the set of additive confidentiality label policies, S_c^- represents the set of removable confidentiality label policies, S_i^+ represents the set of additive integrity label policies, and S_i^- represents the set of removable integrity label policies.

(1) Rule 5. Label encryption rule

Suppose tenant T has confidentiality label set $L_c = (L_{c_1}, L_{c_2}, \ldots, L_{c_n}), L_{c_i} = (ID \longrightarrow R_2)$, and the encryption set of confidentiality label corresponding to L_c is S_c. In addition, suppose tenant T has integrity label set $L_i = (L_{i_1}, L_{i_2}, \ldots, L_{i_n}), L_{i_j} = (ID \longrightarrow R_2)$, and the integrity label encryption set corresponding to L_i is S_i. The authorization rules are as follows:

Rule 5.1: Confidentiality label encryption is described as equation 12:

$$L_c = L_c \cup L_{add-c} \text{ only if } (L_{add-c} \subset S_c^+) \quad (12)$$

The necessary condition for tenants to add the confidentiality label L_{add-c} to the original label is that L_{add-c} is included in S_c^+.

Rule 5.2: Integrity label encryption is described as equation 13:

$$L_i = L_i \cup L_{add-i} \text{ only if } (L_{add-i} \subset S_i^+) \quad (13)$$

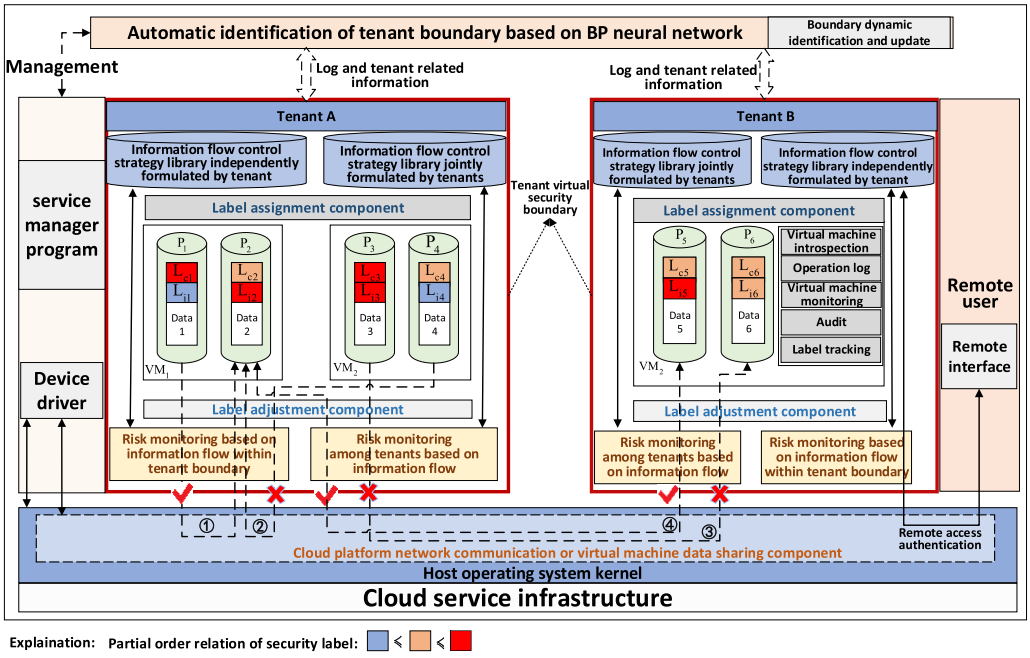

FIGURE 5. Architecture design of cloud tenants' sensitive information flow application system.

The necessary condition for tenants to add the integrity label L_{add-i} to the original label is that L_{add-i} is included in S_i^+.

In addition, according to rule 5 and the minimum lower bound of label, the complete encryption formula of data is described as equation 14:

$$(L_i.\nabla_i)\ \&\&\ (L_c.\nabla_c) \tag{14}$$

(2) Rule 6. Label declassification rule

Rule 6.1: Confidentiality label declassification is described as equation 15:

$$\begin{aligned}&\left((L_c = L_c - L_{sub-c})\ only\ if\ \left(L_{\text{sub}-c} \subset S_c^-\right)\right)\ ||\\ &\quad((L_{c_i}.R_i = L_{c_i}.R_i \cup R_{add})\ only\ if\ (\{R_{add}\} \subseteq \{L_{\text{sub}_i-c}.R\})\\ &\quad \&\&L_{\text{sub}_i-c} \subseteq S_c^-))\end{aligned} \tag{15}$$

There are two situations in the declassification rule of confidentiality label: ① remove the confidentiality label directly, and ② add the subject to the value domain of the label. The necessary condition for the tenant to reduce the confidentiality label constraint is that L_{sub-c} is included in S_c^- or R_{add} belongs to the value domain of a security label in set S_c^-.

Rule 6.2: Integrity label declassification is described as equation 16:

$$\begin{aligned}&((L_i = L_i - L_{sub-i})\ only\ if\ (L_{\text{sub}-i} \subset S_i^-))||\\ &\quad((L_{ij}.R_j = L_{ij}.R_j \cup R_{add})\ only\ if\ (\{R_{add}\} \subseteq \{L_{\text{sub}_j-i}.R\})\\ &\quad \&\&L_{\text{sub}_j-i} \subseteq S_i^-))\end{aligned} \tag{16}$$

There are two situations in the declassification rule of integrity label: ① remove the integrity label directly, and ② add the subject to the value domain of the label. The necessary condition for the tenant to reduce the integrity label constraint is that L_{sub-i} is included in S_i^- or R_{add} belongs to the value domain of a security label in set S_i^-.

In addition, according to rule 6 and the maximum upper bound of the label, the complete decryption formula of the data is described as equation 17:

$$(L_i.\Delta_i)\ \&\&\ (L_c.\Delta_c) \tag{17}$$

IV. DYNAMIC SECURITY CONTROL APPLICATION SYSTEM FOR CLOUD TENANTS' SENSITIVE INFORMATION FLOW (DSCLoud)

Based on the aforementioned methods, we present the design of a dynamic security control system for sensitive information flow of cloud tenants, which aims to accurately identify the virtual boundaries of tenants and realize the security control and sharing of sensitive information flow of tenants in the cloud. The overall architecture design of the system is shown in Figure 5.

The system consists of three parts: automatic identification of tenants' virtual security boundaries, centralized and autonomous control of information flow within the boundary of tenants, and decentralized dynamic control and security sharing of information flow among tenants. All programs of cloud tenants run in virtual machines with the operating system installed. In the virtual machine, modules such as a fine-grained label tracking module, an instant virtual machine introspection module, a virtual machine monitoring module, an audit module, and a user interface module, are included. The tenant boundary also includes an information flow control strategy library, a label dynamic tag component, a label dynamic adjustment component, and a risk monitoring module of tenant information flow, among others. The flow of tenant information inside and outside the boundary is

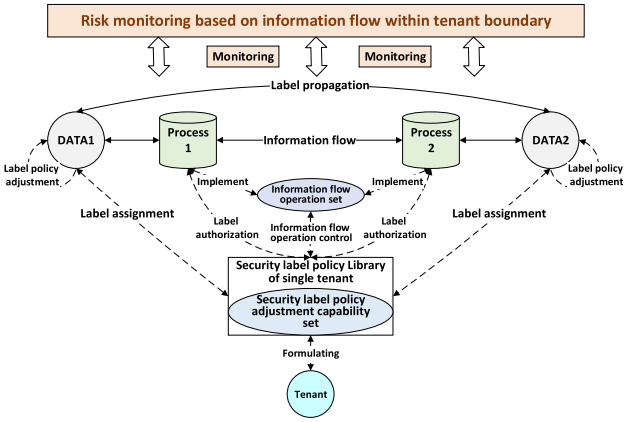

FIGURE 6. Autonomous control strategy for tenant information flow.

completed by cloud platform network communication or the virtual machine data sharing component.

According to the security requirements of sensitive information flow control of cloud tenants, the system design is as follows:

(1) In the system, the module of automatic identification of tenant boundaries based on the neural network is designed; this module can automatically determine the virtual security boundary of tenants and provide the basis for tenant information flow tracking and control.

(2) In the interior of the cloud tenant boundary, the method of centralized formulation of the information flow control strategy (the method implementation is shown in Figure 6) is adopted, and the control method is independently developed by the tenant. This approach can realize the information tracking and control of the virtual machine at the process level and the network communication at the byte level within the tenant boundary to prevent the leakage of the tenant's sensitive information. The examples of information flow within the tenant boundary are shown in Figures 5-① and ②:

① shows that the information flow between processes in the same virtual machine within the tenant boundary is monitored by the information flow risk monitoring module within the tenant. When the information flow control rules are met, the information flow from P_1 to P_2 is allowed.

② shows that the information flow between processes in different virtual machines within the tenant boundary is monitored by the information flow risk monitoring module within the tenant. Because the integrity of $data_4$ is less than that of $data_2$, the information flow from P_4 to P_2 is not allowed.

(3) Among the cloud tenants, the decentralized information flow control method is adopted, and the tenants jointly formulate the information flow control strategy (the method implementation is shown in Figure 7). The cloud tenants can formulate the corresponding information flow control strategy and view the information flow audit information through the program interface. For example, tenant A can participate in the formulation of the information flow control strategy between tenant B and tenant A, but not between tenant B and tenant C. According to the information flow control policies formulated by each tenant, the distributed policy control set among tenants is formed. This can realize the tracking and control of virtual machines at the process level and network communication at the byte level between different tenant boundaries to prevent malicious tenants from illegally obtaining sensitive information from other tenants. An example of information flow between tenants is shown in Figures 5-③ and ④:

③ shows that the illegal flow of information when tenants share the same virtual machine is monitored by the information flow risk monitoring module between tenants.

④ shows that the legitimate sharing of information between tenants is also monitored by the information flow risk monitoring module between tenants.

In the application of the information flow control strategy, the security restriction of the entire information flow process is realized by the transfer rule of security label. By introducing the minimization rule of the value domain of labels, the flow of information conforms to the minimum privilege principle. By introducing "and or" of the tag value field, the principle of separation of authority and duty in data operation is realized. Through the introduction of label encryption and declassification rules, the tenants can dynamically control the flow of their own information flow, and thus jointly formulate information flow policies and the share security of information flow among tenants in a convenient manner.

V. EXPERIMENTAL VERIFICATION AND SYSTEM SECURITY ANALYSIS

A. ACCURACY VERIFICATION OF D_SNNBAR ALGORITHM

In this study, OpenStack-Ocata was used to build a cloud platform. Three MSI GT63 physical machines were used for conducting the experiments. The processor was Intel (R) core (TM) i7-8750h @ 2.2GHz, six cores/twelve threads, memory was 32 GB, and hard disk capacity was 512GB solid state + 1TB mechanical. An ubuntu 12.04 virtual machine was selected for the cloud environment, and the minimum instance of a single machine was used for deployment. The experimental configuration of each virtual machine was 1 CPU core, 512 MB memory, and 20 GB hard disk capacity.

We created a control node controller and three calculation nodes, namely nova1, nova2, and nova3, for the test platform.

1) PLATFORM CONSTRUCTION

To better reflect the identification of tenant boundaries after the number of tenants and the number of virtual machines change dynamically, the platform is constructed in the following two stages:

① Initialization: create five tenants and two users for each tenant. On average, create three virtual machine instances initially for each tenant to collect operation data.

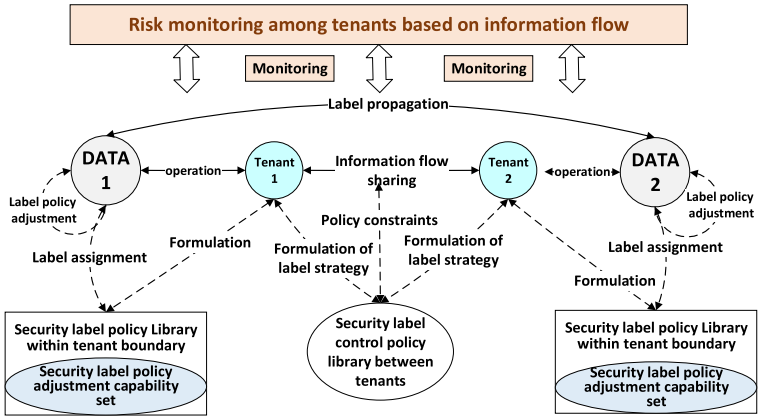

FIGURE 7. Method for tenants to jointly formulate information flow control strategy.

② Change the number of tenants and virtual machines dynamically: collect sample data multiple times by increasing the number of tenants and virtual machines. The dynamic change situation is as follows: add two tenants, add two virtual machine instances for each tenant, and collect sample data.

2) SAMPLE DATA COLLECTION

Through continuous monitoring of virtual machines in the cloud platform, the resource statistics information of tenants (e.g., by the command Nova usage list) and the detailed information of virtual machines (the command Nova show ID or name) can be obtained in OpenStack. In the computing node, the log information of starting and running of virtual machines can be obtained (e.g., by the command Nova compute. Log).

Through the virtual machine monitoring platform, every T = 5 minutes was recorded (including the log information of all virtual machines), and eight consecutive records were taken as a group of sample data for neural network learning. The same items in the records were combined to get the sample data, of which 80% in each experiment were training data and 20% were test data.

3) EXPERIMENTAL VERIFICATION

The verification experiments of the D_SNNBAR algorithm are divided into four parts: setting of the similarity threshold "TH_{sim}" and the number of Gaussian receptive fields "n", verification of the algorithm's recognition accuracy, verification of the algorithm's dynamic boundary recognition accuracy, and verification of the algorithm's efficiency.

① Setting of the similarity threshold "TH_{sim}" and the number of Gaussian receptive fields *"n"*.

A set of 1148 sample data was collected by the initial platform. By calculating the accuracy rate (AR) and error rate (ER), the similarity threshold and the number of the Gaussian receptive field are determined. Under a certain similarity threshold and Gaussian receptive field size, "T" is the result recognized by the algorithm and "N" is the classification result under the standard condition. The calculation formulas of the two rates are as follows:

AR $= |T \cap N| / |N|$; that is, the ratio of the number of data intersected by T and N to the total number of data in the standard case;

ER $= |T - (T \cap N)| / |N|$; that is, the proportion of data with incorrect identification.

The experimental process is as follows:

a. Similarity threshold "TH_{sim}": If the value of "TH_{sim}" is too large, the classification will be excessive and the recognition accuracy will reduce. By contrast, if the value of "TH_{sim}" is too small, the incorrect classification and therefore the error rate will increase. It can be seen that the size of "TH_{sim}" directly affects the accuracy of algorithm classification. Therefore, in this experiment, the threshold value was increased from 0.6 to 1 consecutively at an increment of 0.01.

b. Gaussian receptive field "n": it represents the pulse time range covered by the input neuron. If it is set too large, too many pulses will be covered, which are not enough to excite neurons; this reduces the accuracy of recognition. By contrast, if n is too small, the neurons cannot be accurately represented by pulses, thus increasing the recognition error rate. Therefore, to determine the optimal size of the Gaussian receptive field, "n" was increased from 1-16 consecutively at an increment of 1 in the experiment.

At the value of each Gaussian receptive field, each threshold was tested. The recognition accuracy and error rate vary as shown in Figure 8.

Figure 8 and 9 indicate that when $TH_{sim} = 0.83$ and $n = 8$, the accuracy rate reaches 0.983 and the error rate reaches 0.007, and the algorithm recognition effect is the best. Therefore, "n" and "TH_{sim}" for the subsequent experiments were taken as 8 and 0.83, respectively.

② Verification of the algorithm's recognition accuracy.

On the basis of the sample data in ①, two groups of sample data, consisting of 1079 and 1247 data, were collected. These three groups of sample data are tested by "rank order" [21], the pulse neural network learning algorithm based on the precise time "only precise" [22], and the

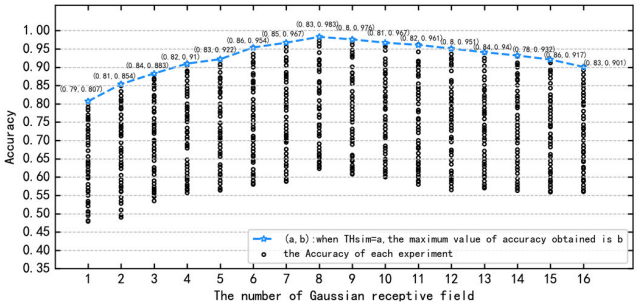

FIGURE 8. Accuracy rate of recognition under different number of Gaussian receptive fields and similarity threshold values.

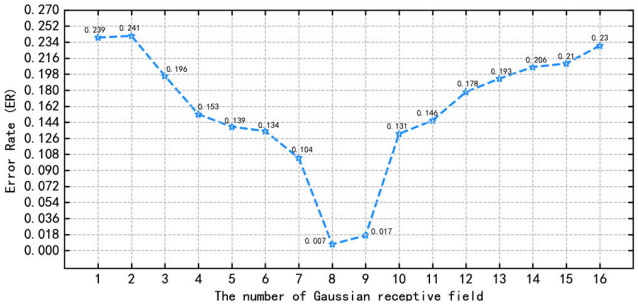

FIGURE 9. Error rate of recognition under different number of Gaussian receptive fields and similarity threshold values.

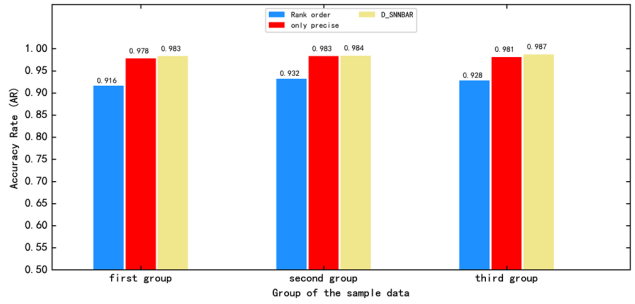

FIGURE 10. Comparison of recognition accuracy rates of different algorithms.

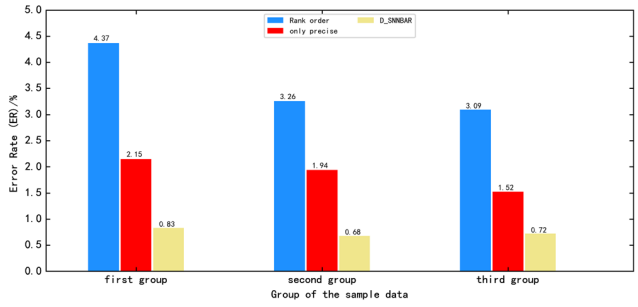

FIGURE 11. Comparison of recognition error rates of different algorithms.

proposed algorithm in this paper. Their recognition accuracy rate and error rate are compared, and the results are shown in Figures 10 and 11.

Figure 10 indicates that the recognition accuracy rate of the D_SNNBAR algorithm for tenant boundaries is slightly higher than that of the ''only precise'' algorithm and far higher than that of the ''rank order'' algorithm. Figure 11

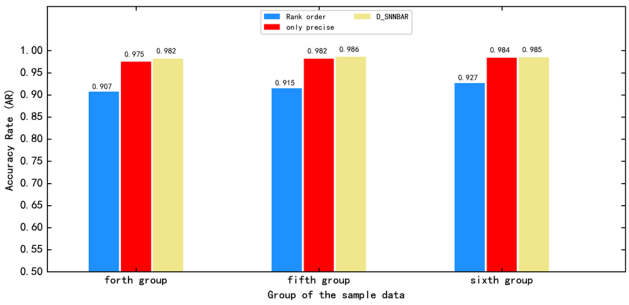

FIGURE 12. Comparison of dynamic recognition accuracy rate of different algorithms.

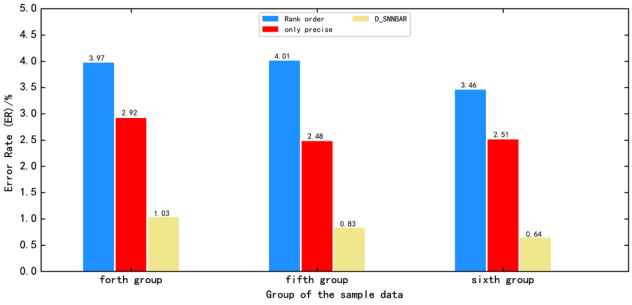

FIGURE 13. Comparison of dynamic recognition error rates of different algorithms.

indicates that the recognition error rate of the D_SNNBAR algorithm is lower than those of the other two algorithms. Thus, the proposed algorithm is confirmed to have high recognition accuracy rate and low recognition error rate, indicating its ability to accurately identify tenant boundaries.

③ Verification of the algorithm's dynamic boundary recognition accuracy.

After varying the number of tenants and virtual machines dynamically, the experimental data were collected continuously and input into the network one by one. The continuous recognition results of three groups of sample data were counted. The accuracy rate and error rate comparison results of dynamic recognition are shown in Figures 12 and 13.

Figures 12 and 13 indicate that the D_SNNBAR algorithm has the ability to dynamically identify the tenant boundary in the case of dynamic changes in the number of tenants and virtual machines and can more accurately realize the dynamic update of the tenant boundary.

④ Verification of the algorithm's efficiency.

To verify the efficiency of the D_SNNBAR algorithm, the execution time of the three algorithms in ② is compared. The comparison results are shown in Figure 14.

Figure 14 indicates that the D_SNNBAR algorithm has superior efficiency compared to the other two algorithms, and the processing time of data is far less than the time of data collection. Thus, the algorithm can realize real-time updating of tenant boundary identification.

Table 4 indicates the comparison of accuracy and time consumption between the D_SNNBAR algorithm and the other two algorithms in the above experiments.

TABLE 4. Comparison of recognition performance of different algorithms.

	Rank order			Only precise			D_SNNBARR		
	AR	ER	Time/min	AR	ER	Time/min	AR	ER	Time/min
1st group	0.916		1.19	0.978		6.89	0.983		0.26
2nd group	0.932		1.10	0.983		6.74	0.984		0.24
3rd group	0.928		1.32	0.981		7.47	0.987		0.31
4th group	0.907		2.41	0.975		13.73	0.980		0.58
5th group	0.915		2.05	0.982		13.06	0.986		0.46
6th group	0.927		2.27	0.979		12.54	0.984		0.55

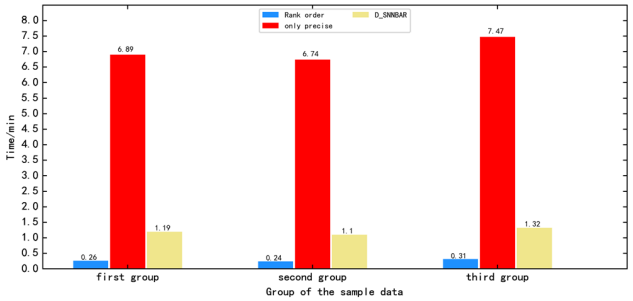

FIGURE 14. Comparison of recognition efficiency of different algorithms.

4) CLASSIFICATION ACCURACY VERIFICATION UNDER STANDARD DATASETS

To further verify the performance and accuracy of the algorithm, this study conducted experiments using two standard datasets from the UCI machine learning library [41]: the "IRIS" dataset and "Yeast" dataset.

First, the size of the Gaussian receptive field and the similarity threshold in the two datasets were tested. The results show that the classification effect is the best when the Gaussian receptive field and similarity threshold are set to 6 and 0.85, respectively, in the "IRIS" dataset, and 12 and 0.86, respectively, in the "Yeast" dataset.

On the basis of the above parameters, the classification accuracy, error rate, and simulation time of the algorithm for the two datasets are compared as shown in Table 5.

Table 5 indicates that the recognition accuracy of the D_SNNBARR algorithm is better than the only precise and rank order algorithms for the two standard datasets and has high recognition accuracy. In addition, the running time of the proposed algorithm is considerably less than that of the other two algorithms. Figures 15-(a), (b), (c) intuitively show the recognition effect of the three algorithms for the two datasets.

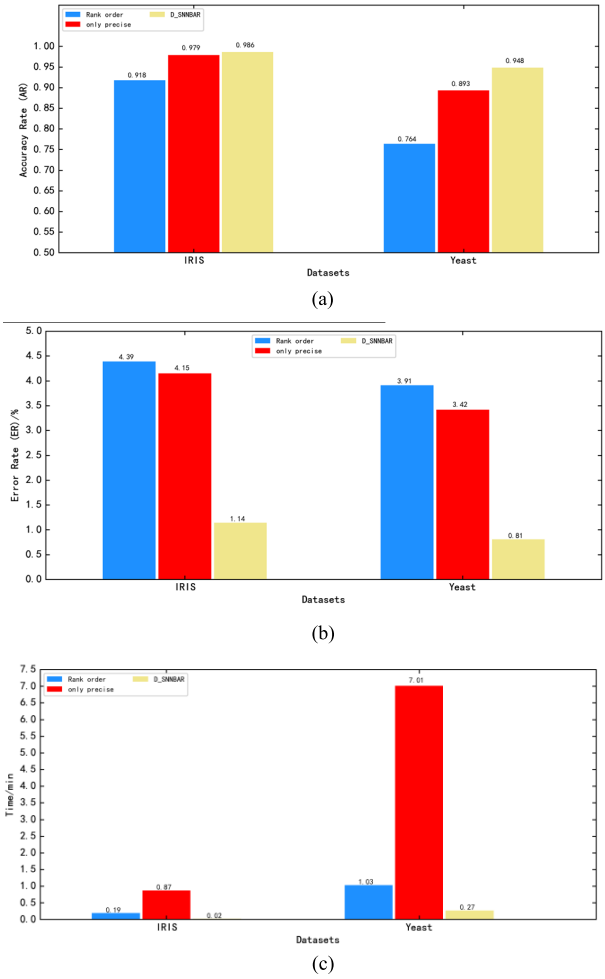

FIGURE 15. (a). Comparison of recognition accuracy rates of different algorithms for the standard datasets. (b). Comparison of recognition error rates of different algorithms for the standard datasets. (c). Comparison of recognition efficiency of different algorithms for the standard datasets.

B. SECURITY ANALYSIS AND VERIFICATION OF THE METHOD

1) SECURITY ANALYSIS

According to the basic concept of the noninterference theory [42], secure information flow can be regarded as noninterference of information flow between tenants and cloud service programs and between tenants' security domains; that is, there are two domains in the system. If the information flow in one domain does not destroy the system output observed in the other domain, it is an indication of no interference. The traditional noninterference theory [43] can be applied to only the security policy environment with the nature of transmission, which obviously does not meet the security requirement of information flow between joint tenants. Therefore, in this study, we choose the intransitive noninterference theory based on the TA-safety model [44] to prove the security of the system.

To confirm the non-interference of the system, we first analyze the security of confidentiality and integrity of tenant information flow, as follows:

TABLE 5. Comparison of experimental results on standard datasets.

	RANK ORDER		ONLY PRECISE		D_SNNBARR	
	ACCURACY	TIME/S	ACCURACY	TIME/S	ACCURACY	TIME/S
IRIS	0.918	0.19	0.979	0.87	0.986	0.009
YEAST	0.764	1.03	0.893	7.01	0.948	0.27

TABLE 6. Information flow control matrix.

M_{ij}	L_{c_L} ´ L_{i_L}	L_{c_L} ´ L_{i_H}	L_{c_H} ´ L_{i_L}	L_{c_H} ´ L_{i_H}
L_{c_L} ´ L_{i_L}	rw	R	w	No
L_{c_L} ´ L_{i_H}	w	Rw	w	w
L_{c_H} ´ L_{i_L}	r	R	rw	r
L_{c_H} ´ L_{i_H}	No	R	w	rw

To analyze the security of tenant data flow, first, the security features of the Bell–La Padula (BLP) and Biba models [23], [24], are introduced.

Security features of the BLP model: (a) simple security feature: the necessary condition for s to read o is $L_{c_o} \preccurlyeq L_{c_s}$; (b) "$*$" feature: the necessary condition for s to write o is $L_{c_s} \preccurlyeq L_{c_o}$. The BLP model requires information to flow only at the same level or from the low level to the high level.

Security features of the Biba model: (a) simple integrity feature: the necessary condition for s to read o is $L_{i_s} \preccurlyeq L_{i_o}$; (b) "$*$ -" feature: the necessary condition for s to write o is $L_{i_o} \preccurlyeq L_{i_s}$. The Biba model mainly considers the integrity of information, contrary to the BLP model.

Based on the above two models, we can get the data flow security characteristics under the constraint of rule 2 according to the definition of the label lattice partial order: (a) simple security feature: the necessary condition for s to read o is $(L_{c_s} \preccurlyeq L_{c_o}) \wedge (L_{i_s} \preccurlyeq L_{i_o})$; (b) "$*$ -" feature: the necessary condition for s to write o is $(L_{c_s} \preccurlyeq L_{c_o}) \wedge (L_{i_o} \preccurlyeq L_{i_s})$. The information flow control matrix based on the label lattice partial order is given, as shown in Table 6: ("r" stands for reading, "w" for writing, "rw" for reading and writing, and "No" for not executing; "L_{c_H}" stands for high level confidentiality label, "L_{c_L}" stands for low level confidentiality label, "L_{i_H}" stands for high level integrity label, "L_{i_L}" stands for low level integrity label.)

Thus, all the situations in Table 6 are brought into the security features of BLP and Biba models for verification, it is confirmed that the flow of tenant information can meet the confidentiality and integrity requirements simultaneously.

Next, the system noninterference is analyzed. For this purpose, the elements involved in the system are defined in a mathematical form:

Let system m be a finite automaton, which mainly consists of the following components:

(a) $M(D, \mapsto)$: cloud tenant sensitive information flow security control system: D represents the security domain in the system, $D = \{\mu, \nu, \ldots\}$, where μ and ν represent the tenant security domain identified by the tenant ID, and "$\mapsto$" is the dynamic protection policy; (b) S: system state set: S $=\{S_0, \ldots, S_n\}$, where S_0 represents the initial state of the system; (c) A: Tenant action set A = {r, w, rw}; d. $obs_\mu(s_i) : s_i \times \mu \rightarrow O : obs_\mu(s_i)$: indicates the output observed by domain μ in state s_i, and O represents the output set; (e) $step(S, A) : S \times A \rightarrow S$: state transition function, $s \cdot \alpha$ represents the state of state s after dynamic sequence α, meanwhile, $s \times \varepsilon = s$, where ε is an empty sequence, $s \cdot \alpha a = step(s \cdot \alpha, a)$; (f) $dom(a)$: virtual domain corresponding to action a; (g) $val_\mu(s, n) : s \times n \rightarrow V$: in virtual domain μ, the value of resource named n in state s.

Next, the definition of TA function is given [44]: for system $M(D, \mapsto)$, $\nu \in D$, function $ta : A* \mapsto T(\{\varepsilon, A)$, the specific definition is described as equation 18:

$$ta_\upsilon(\alpha \cdot a) = \begin{cases} \varepsilon, & \alpha \cdot a \textit{ is the empty sequence}; \\ (ta_\upsilon(\alpha), ta_{dom(a)}(\alpha), a), & dom(a) \mapsto \upsilon; \\ ta_\upsilon(\alpha), & others. \end{cases} \tag{18}$$

According to the above definition, the judgment theorem of TA-safety is described as follows:

If the system $M(D, \mapsto)$ satisfies the weak unwinding Theorem about strategy "$\mapsto$", then $M(D, \mapsto)$ is TA safe about strategy "$\mapsto$". The weak unwinding needs to satisfy the following three conditions:

(1) Output consistency (OC): if $s \sim_\upsilon t$, then $obs\upsilon(s) = obs\upsilon(t)$; ($s \sim_\upsilon t$: the equivalent states s and t in the representation domain υ)

(2) Weak single-step consistency (WSC): if $(s \sim_\upsilon t) \cap (s \sim_{dom(a)} t)$, then $s \cdot a \sim_\upsilon t \cdot a$;

(3) Local compliance (LR): if $dom(a) \nrightarrow \nu$, then $s \sim_\upsilon s \cdot a$.

Theorem 1: Tenant sensitive data dynamic protection system $M(D, \mapsto)$ is TA-safe about policy "$\mapsto$".

For the proof of theorem 1, it is necessary to prove that the weak unwinding of system $M(D, \mapsto)$ about policy "$\mapsto$" satisfies OC, WSC, and LR according to the judgment theorem of TA-safety. The proof is as follows:

Suppose that the security labels in domain μ and domain ν are L_{c_u}, L_{i_u} and L_{c_v}, L_{i_v}, respectively.

a: OUTPUT CONSISTENCY (OC)

① When $(L_{c_\mu} \preccurlyeq L_{c_v}) \wedge (L_{i_v} \preccurlyeq L_{i_\mu})$ is satisfied, it can be seen from rule 2 that information I can flow from μ to ν. By observing the tenant information flow control matrix M, the following three situations exist: (a) When action a = r, the information value before and after the read-only

operation does not change, that is, $val_v(s, I) = val_v(t, I)$. Therefore, in the equivalent state (s $\sim_v$ t), there must be $val_v(s) = val_v(t)$; (b) When a = w, according to the integrity and confidentiality proof of tenant information flow, the confidentiality and integrity security of the original data and the new information before and after the information is written are guaranteed. According to rules 3 and 1, the security label will become stricter after writing and will not pose a security threat to other data. Therefore, in the equivalent state, $val_v(s) = val_v(t)$ holds the same; (c). When a = rw, the confidentiality and integrity of both sides of information flow are the same, which will not change the access relationship of the main body to the data in the tenant. Therefore, under $s \sim_v t$, $val_v(s) = val_v(t)$ must hold.

② When $(L_{c_\mu} \preccurlyeq L_{c_v}) \wedge (L_{i_u} \preccurlyeq L_{i_v})$ or $(L_{c_v} \preccurlyeq L_{c_\mu}) \wedge (L_{i_v} \preccurlyeq L_{i_\mu})$ is satisfied, it can be seen from rule 2 that information is not allowed to flow, because it will destroy the integrity and confidentiality of information in the domain and avoid the occurrence of interference behavior. For the inflow and outflow operation without information in the system, $obs_v(s) = obs_v(t)$ must hold.

It can be seen that output consistency is satisfied

b: WEAK SINGLE-STEP CONSISTENCY (WSC)

According to the definition of WSC, the proof of WSC can be transformed into the proof that the value of the same operation $a \in A$ performed on the data in state $s \sim_\mu t$ is consistent; that is, $val_\mu(step(s, a)) = val_\mu(step(t, a))$, and the analysis is based on whether the operation domain of action a interferes with domain μ.

When $dom(a) \nrightarrow \mu$, $dom(a)$ has no effect on domain μ, because it will not change the value of data in domain μ, nor the access relationship to data, so $val_\mu(s, Data) = val_\mu(t, Data) \Longrightarrow val_\mu(step(s, a), Data) = val_\mu(step(t, a), Data)$;

When $dom(a) \mapsto \mu$, in state s, operation a is performed on the data, and the data value changes. According to the information flow control matrix M, it can be seen that the executed domain $dom(a)$ performs w or rw operation on domain μ.

a. When $(L_{c_{dom(a)}} \preccurlyeq L_{c_\mu}) \wedge (L_{i_\mu} \preccurlyeq L_{i_{dom(a)}})$ and action a = w, according to rule 2, the execution of action a is completed under the condition of ensuring the confidentiality and integrity of domain μ. According to rules 3 and 1, after the data is written to domain μ, $L_{c_{Data}} = L_{c_{Data}} \cup L_{c_\mu}$ and $L_{i_{Data}} = L_{i_{Data}} \cup L_{i_\mu}$ are obtained. It can be seen that the security label policy of data becomes stricter, and the writing of data will not change the access relationship of other data in domain μ, that is, it will not change the value of original data in domain μ.

Therefore, $val(s, Data) = val(t, Data) \Rightarrow val(step(s, a), Data) = val(step(t, a), Data)$ can be obtained from assumption $(s \sim_\mu t) \cap (s \sim_{dom(a)} t)$.

b. When $(L_{c_{dom(a)}} = L_{c_\mu}) \wedge (L_{i_\mu} = L_{i_{dom(a)}})$, according to rules 2 and 3, the integrity and confidentiality of both sides of data flow remain unchanged before and after data flow; that is, the integrity and confidentiality of domain μ will not change and will not affect the value of the original data in domain μ. Therefore, in the equivalent states s and t, after the completion of action, a, $val(step(s, a), Data) = val(step(t, a), Data)$.

It can be seen that WSC is satisfied.

c: LOCAL COMPLIANCE (LR)

According to the definition of LR, $dom(a) \nrightarrow u \Rightarrow obs_u(s) = obs_u(step(s, a))$. According to the tenant information flow control matrix, action a is divided into three situations, which are analyzed as follows:

When a = r, because the read-only operation is carried out under the constraint of rule 2, the data of domain u will not be disclosed, and the data value and label policy in domain u have not changed before and after the execution of a, that is, $dom(a) \nrightarrow \mu$, $obs_u(s) = obs_u(step(s, a))$.

When a = w$\vee$ rw, from the converse negative proposition, it is proved whether $obsu(s) \neq obs_u(step(s, a)) \Rightarrow dom(a) \longmapsto \mu$ is true or not. After action a is executed, the data is written to domain u, and the amount of data and the security label set of the domain u will change, that is, $val_\mu(s, Data) \neq val_\mu(step(s, a), Data) \Rightarrow obs_u(s) \neq obs_u(step(s, a))$.Obviously $dom(a) \longmapsto u$ is established, so $dom(a) \nrightarrow u \Rightarrow obs_u(s) = obs_u(step(s, a))$ is established.

Therefore, LR is satisfied.

In conclusion, the proof of Theorem 1 shows that the system $M\ (D, \mapsto)$ is TA safe to the strategy, and the system can ensure the security of tenant information flow without interference.

2) SECURITY VERIFICATION

Experiments were conducted for the security verification of tenant information flow control in DSCLoud, that is, for testing tenant process level communication and network level information flow control. The verification test consists of three parts: ① the tests of label propagation and file operation's information flow control in the virtual machine, ② the test of communication through the shared physical memory between processes in the same virtual machine, ③ the test of process communication between virtual machines, and ④ the test of system performance delay after adding security measures.

Experiment environment: the virtual operating system was Ubuntu 12.04, Linux kernel version 3.0.1, and the test virtual machine was ''Xen''. The test was implemented on the existing LSM framework in Linux. The security control of tenant information flow was programmed as a security module, which can be loaded and executed dynamically in a Linux kernel through the ''mod_reg_security()'' function [5], [45].

The related modules are realized as follows:

(1) Access to information

The virtual machine introspection (VMI) [46] is used to obtain the information of the internal process, module, memory, and network interface of the virtual machine, and the label information of the above objects is saved by creating

the corresponding data structure. For example, to obtain the current process and module information, the following process classes are defined in the VMI module:

```
class process {
public:
    uint32_t cr3; // Indicates the memory location of the process
    uint32_t pid; // Indicates the process identification number
    char name[VMI_MAX_MODULE]; // Modules loaded by the process
    uint32_t level; // The security level of the process
    unordered_map < uint32_t, module * > module_list; // The mapping table of the base address to the module pointer
};
```

(2) Label marking method

According to the control rules of information flow in the system, label marking can be realized by adding security labels with the security level on the processes and files.

In the experiment, 0×0, 0×1, 0×2, and 0×3 were used to represent the security levels of security labels in increasing order.

(3) Dynamic tracking of labels

To realize fine-grained label dynamic tracking at the instruction level, tiny code generation (TCG) technology is used to create a labeled global variable with the same size for each global variable (general register and label register) and the corresponding label dynamic tracking code is inserted in the TCG code generation phase. For example, the label of the global variable "eax" is "label_eax". If the data of the level "0×1" is stored in "eax", "label_eax" saves the value of the corresponding security level of "eax". If a certain type of instruction (such as a move instruction, an arithmetic instruction, or a logical operation instruction) generates information flow propagation, a corresponding label propagation instruction is inserted to track the label.

The experimental results are as follows:

① Test of label propagation and file operation's information flow control in the virtual machine:

The test files were named "public", "sensitive", and "secret", and the security levels of the security labels of the files were set to 0×0, 0×1, and 0×2, respectively. "gedit" and "office" were used as the test programs, and the security levels of the security labels were set to 0×1 and 0×3, respectively. "gedit" and "office" were used to try to read these three files. The experimental results are shown in Figure 16.

Figure 16 indicates that "gedit" could read the files "public" and "sensitive", but could not open the file "secret", and "office" could read the file "secret". Thus, the experimental expectation is met.

Propagation of security label: first, "office" read the file "secret" and then the file "public". Then, the file "public" was tried to open with "gedit"; this reading result is shown in Figure 17.

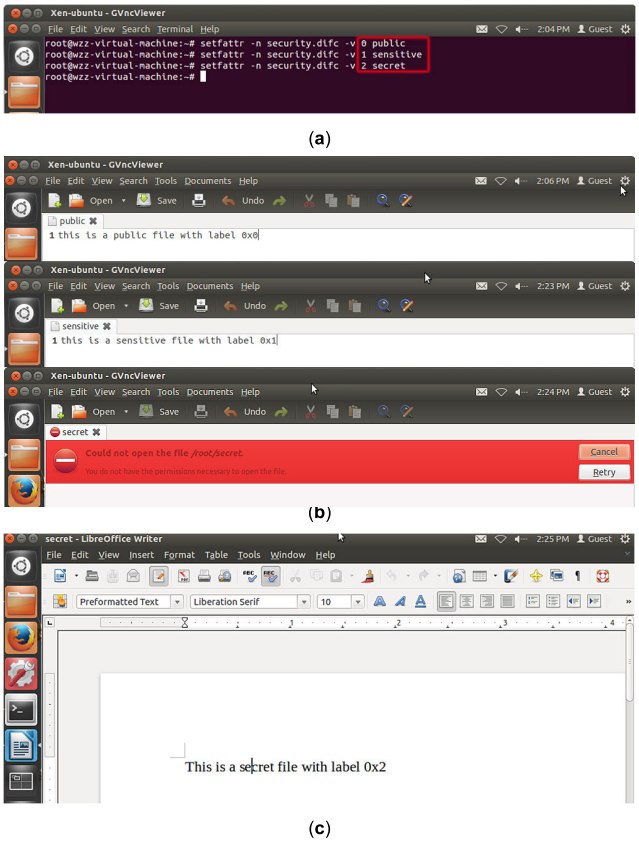

FIGURE 16. Information flow control of file operation by a process. (a) Set security level for the security labels of files; (b) Reading results of three files by "gedit"; (c) Reading results of the file "secret" by "office".

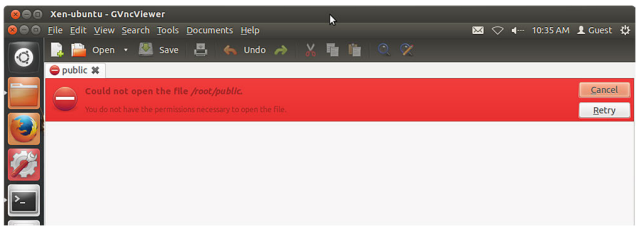

FIGURE 17. Reading result of the file "public" by "gedit".

TABLE 7. Security level settings of the security labels of the programs.

serverh	serverl	Clienth	clientl
0x3	0x1	0x3	0x0

Figure 17 indicates that "gedit" could not open the file "public", and this is because when "office" opened the file "secret" and then the file "public", the security level of the file "public" was updated to 0×2. The result meets the requirements of propagation rules of labels.

② Test of communication through the shared physical memory between processes in the same virtual machine:

The test programs used were "productor", "customerl", and "customerh", and the security levels of the security labels were set to 0×1, 0×0, and 0×1, respectively.

TABLE 8. System performance test results.

Operation	Original Linux system		Present system		Delay percentage (%)
	Mean value (ms)	Square value	Mean value (ms)	Square value	
Startup time	61984	5323	63401	4791	2.3
Program startup time	862	46	898	84	4.2
File opening	623	34	648	145	4.1
Keyboard input response	104	12	106	23	2.1
Network data transmission	843	56	863	64	2.4

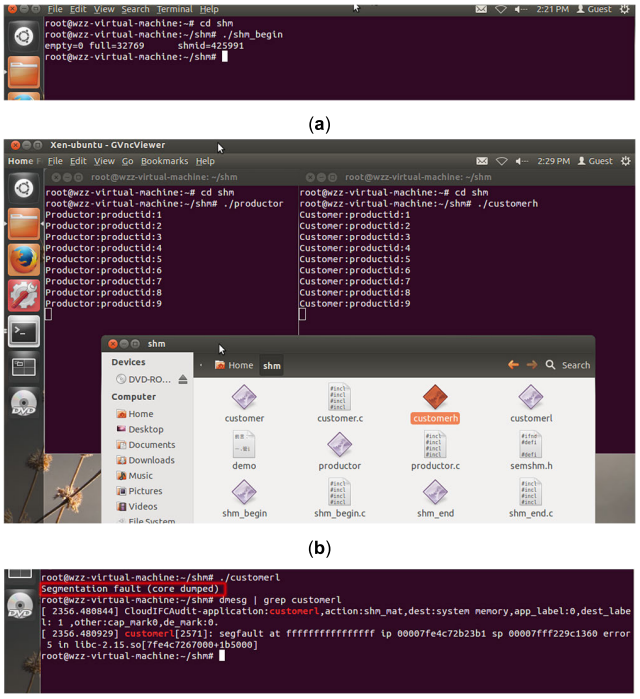

FIGURE 18. Test results of communication between processes in the same virtual machine. (a) Create shared information; (b) Normal communication between programs "productor" and "customerh" through shared memory; (c) Communication failure between programs "productor" and "customerl".

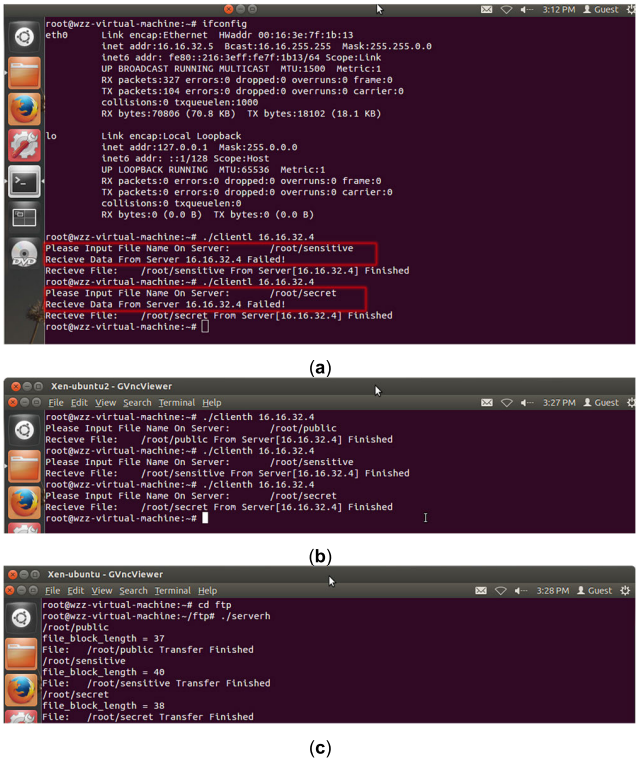

FIGURE 19. Test results of communication between virtual machines. (a) Result of communication between "clientl" and "serverl"; (b), (c) Result of communication between "clienth" and "serverh".

"productor" was used to send messages to "customerh" and "customerl". The test results are shown in Figure 18.

Figure 18 indicates that "productor" and "customerh" with the same security level could communicate normally, while "productor" and "customerl" with different security levels could not communicate; this meets the experiment expectation.

③ Test of process communication between virtual machines:

We configured two virtual machines "ubuntu" and "ubuntu2", with system IPs 16.16.32.4 and 16.16.32.5, respectively. The programs "serverh" and "serverl" ran in "ubuntu", while the programs "clienth" and "clientl" ran in "ubuntu2". The security level settings of the security label of the programs are shown in Table 7.

The settings of the transfer file were the same as those in experiment ①. In the experiment, we used "clientl" to connect with "server" to receive the files "sensitive" and "secret" from the ftp server and used "clienth" to connect with "serverh" to receive the files "public", "sensitive", and "secret" from the ftp server. The experimental results are shown in Figure 19.

Figure 19 indicates that "server" failed to transfer the files "sensitive" and "secret" to "clientl", whereas "clienth" and "serverh" successfully transferred the files "public", "sensitive", and "secret". This meets the experiment expectation. This result can be attributed to the fact that under the constraints of the information flow control rules, when two virtual machines communicate with each other, they cannot send the data of a high security level to the process of a low security level.

④ Test of system performance delay

To test the effect of adding the security measures on the actual performance of the system, this study tested the system startup time, program startup time, file opening time, keyboard input response time, and network data transmission

time between the original Linux system and the present system. The results are shown in Table 8.

The table 8 shows that the security measures delayed the operation performance of the system by an average of approximately 3.0%. The security measures showed the greatest effect on the performance of the startup program (from inputting ''gedit'' on the command line till the program starts and displays) and file opening (from entering ''gedit file path'' on the command line till the file is opened and displayed),whereas they had no effect on the operation of the system. In summary, the performance delay of the system after adding security measures can be ignored.

VI. CONCLUSION

By analyzing the characteristics of the cloud environment and the security requirements of sensitive information flow of cloud tenants, a dynamic control method for tenants' sensitive information flow based on virtual boundary recognition was proposed. First, an automatic recognition algorithm of tenant virtual boundaries based on the dynamic spiking neural network was designed. Based on the analysis of tenant behavior and operation log, the feature vector of tenant behavior was constructed. Through the learning and training of the neural network, the automatic recognition of tenants' virtual security boundary was realized, which provides the basis for the security control of tenants' sensitive information flow. By implementing a dynamic control method for sensitive information flow of cloud tenants, the security strategy of tenant information flow was formulated, and security labels were used to track and control the tenant information flow inside and outside the boundary in order to realize the tenant's independent control of the information flow within the boundary as well as the dynamic control and security sharing of information flow between tenants. Based on the identification n of tenants' virtual security boundaries and the security control of information flow, a dynamic security control application system for sensitive information flow was constructed. Finally, a cloud platform was built using OpenStack and sample data were collected for experiments. The accuracy of the recognition algorithm was verified, and the safety and effectiveness of the system were confirmed by the intransitive noninterference theory and the experiment of information flow control. In the future work, we hope to achieve higher accuracy and efficiency in boundary recognition. So, we will further optimize the ''D_SNNBAR'' algorithm and build a more complete dataset. In addition, we will complete the overall implementation of the ''DSCloud'' system to achieve better application.

REFERENCES

[1] A. Lele, ''Cloud computing,'' in *Disruptive Technologies for the Militaries and Security*, vol. 132. Singapore: Springer, 2019, pp. 167–185.

[2] A. Cook, M. Robinson, M. A. Ferrag, L. A. Maglaras, Y. He, K. Jones, and H. Janicke, ''Internet of cloud: Security and privacy issues,'' in *Cloud Computing for Optimization: Foundations, Applications, and Challenges*. Berlin, Germany: Springer, 2018, pp. 271–301.

[3] D. Puthal, B. P. S. Sahoo, S. Mishra, and S. Swain, ''Cloud computing features, issues, and challenges: A big picture,'' in *Proc. Int. Conf. Comput. Intell. Netw.*, Bhubaneshwar, India, Jan. 2015, pp. 116–123.

[4] M. K. Walia, M. N. Halgamuge, N. D. Hettikankanamage, and C. Bellamy, ''Cloud computing security issues of sensitive data,'' in *Handbook of Research on the IoT, Cloud Computing, and Wireless Network Optimization*. Hershey, PA, USA: IGI Global, 2019, pp. 60–84.

[5] C. Wright, C. Cowan, J. Morris, S. Smalley, and G. Kroah-Hartman, ''Linux security modules: General security support for the Linux kernel,'' in *Proc. 11th USENIX Secur. Symp.*, Los Alamitos, CA, USA, Aug. 2002, pp. 17–31, doi: 10.1109/FITS.2003.1264934.

[6] H. Zhou, H. Ba, Y. Wang, Z. Wang, J. Ma, Y. Li, and H. Qiao, ''Tenant-oriented monitoring for customized security services in the cloud,'' *Symmetry*, vol. 11, no. 2, p. 252, Feb. 2019.

[7] B. Medeiros, M. Simplicio, and E. R. Andrade, ''Designing and assessing multi-tenant isolation strategies for cloud networks,'' in *Proc. 22nd Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Paris, France, Feb. 2019, pp. 214–221.

[8] Y. Shi, Y. Guo, J. Q. Liu, Z. Han, W. Ma, and L. Chang, ''Trusted cloud tenant separation mechanism supporting transparency,'' (in Chinese), *J. Softw.*, vol. 27, no. 6, pp. 1538–1548, 2016.

[9] Y. Hu, ''Research on multi-tenant scalable network in cloud data center,'' (in Chinese), Nat. Univ. Defense Technol., Changsha, China, Tech. Rep., 2014.

[10] J. Kinoshita, K. Maeda, H. Yabusaki, K. Akune, and N. Komoda, ''Realization of VXLAN gateway-based data center network virtualization,'' in *Proc. 5th IIAI Int. Congr. Adv. Appl. Informat. (IIAI-AAI)*, Kumamoto, Japan, Jul. 2016, pp. 884–887, doi: 10.1109/IIAI-AAI.2016.121.

[11] K. Salah, J. M. A. Calero, S. Zeadally, S. Al-Mulla, and M. Alzaabi, ''Using cloud computing to implement a security overlay network,'' *IEEE Secur. Privacy*, vol. 11, no. 1, pp. 44–53, Jun. 2012, doi: 10.1109/MSP.2012.88.

[12] V. Patil, C. Patil, and R. N. Awale, ''Security challenges in software defined network and their solutions,'' in *Proc. 8th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Delhi, India, Dec. 2017, pp. 1–5, doi: 10.1109/ICCCNT.2017.8203978.

[13] S. Papavassiliou, ''Software defined networking (SDN) and network function virtualization (NFV),'' *Future Internet*, vol. 12, no. 1, p. 7, Jan. 2020.

[14] K. Chandra, ''Software-as-a-service (SaaS),'' in *Encyclopedia of Database Systems*. Berlin, Germany: Springer, 2017, pp. 1–2.

[15] M. Li, ''Research and implementation of data isolation mode customization system for SaaS multi-tenants,'' Southwest Jiaotong Univ., Chengdu, China, Tech. Rep., 2018.

[16] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li, and J. Li, ''TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds,'' *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

[17] H. Yuan, J. Bi, M. Zhou, and K. Sedraoui, ''WARM: Workload-aware multi-application task scheduling for revenue maximization in SDN-based cloud data center,'' *IEEE Access*, vol. 6, pp. 645–657, 2017.

[18] L.-C. Jiao, S.-Y. Yang, F. Liu, S.-G. Wang, and Z. X. Feng, ''Seventy years beyond neural networks: Retrospect and prospect,'' (in Chinese), *Chin. J. Comput.*, vol. 39, no. 8, pp. 1697–1717, 2016.

[19] S. Chen, ''Research and application of dynamic and self-adaptive model based on spiking neural network,'' Univ. Electron. Sci. Technol. China, Chengdu, Chain, Tech. Rep., 2018.

[20] N. Kasabov, K. Dhoble, N. Nuntalid, and G. Indiveri, ''Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition,'' *Neural Netw.*, vol. 41, pp. 188–201, May 2013, doi: 10.1016/j.neunet.2012.11.014.

[21] S. Thorpe and J. Gautrais, ''Rank order coding,'' in *Proc. Conf. Comput. Neurosci., Trends Res.* New York, NY, USA: Plenum, 1998, pp. 113–118.

[22] J. Wang, A. Belatreche, L. P. Maguire, and T. M. McGinnity, ''SpikeTemp: An enhanced rank-order-based learning approach for spiking neural networks with adaptive structure,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 1, pp. 30–43, Jan. 2017, doi: 10.1109/TNNLS.2015.2501322.

[23] C.-S. Feng, Z. G. Qin, D. Yuan, and Y. Qing, ''Key techniques of access control for cloud computing,'' (in Chinese), *Acta Electron. Sinica*, vol. 43, no. 2, pp. 312–319, 2015.

[24] F. Cai, N. Zhu, J. He, P. Mu, W. Li, and Y. Yu, ''Survey of access control models and technologies for cloud computing,'' *Cluster Comput.*, vol. 22, no. S3, pp. 6111–6122, May 2019.

[25] D. E. Denning, ''A lattice model of secure information flow,'' *Commun. ACM*, vol. 19, no. 5, pp. 236–243, May 1976, doi: 10.1145/360051.360056.

[26] A. C. Myers and B. Liskov, "A decentralized model for information flow control," *ACM SIGOPS Oper. Syst. Rev.*, vol. 31, no. 5, pp. 129–142, Dec. 1997, doi: 10.1145/269005.266669.
[27] S. Smalley, C. Vance, and W. Salamon, "Implementing SELinux as a Linux security module," NAI Labs Rep. 1, Dec. 2001, pp. 139–146, no. 43.
[28] P. Efstathopoulos, M. Krohn, S. VanDeBogart, C. Frey, D. Ziegler, E. Kohler, D. Mazières, F. Kaashoek, and R. Morris, "Labels and event processes in the asbestos operating system," in *Proc. 20th ACM Symp. Oper. Syst. Princ. (SOSP)*, New York, NY, USA, 2005, pp. 17–30.
[29] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazières, "Making information flow explicit in HiStar," in *Proc. 7th Symp. Oper. Syst. Design Implement.* Berkeley, CA, USA: USENIX, 2006, pp. 263–278.
[30] M. Krohn and E. Tromer, "Noninterference for a practical DIFC-based operating system," in *Proc. 30th IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2009, pp. 61–76.
[31] N. Adwait, B. Andow, W. Enck, and S. Jha, "Practical DIFC enforcement on Android," in *Proc. Conf 25th USENIX Secur. Symp. (USENIX Secur.)*, 2016, pp. 1119–1136.
[32] Z. Wu, X. Chen, Z. Yang, and X. Du, "Reducing security risks of suspicious data and codes through a novel dynamic defense model," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 9, pp. 2427–2440, Sep. 2019.
[33] V. Pappas, V. P. Kemerlis, A. Zavou, M. Polychronakis, and A. D. Keromytis, "CloudFence: Data flow tracking as a cloud service," in *Research in Attacks, Intrusions, and Defenses*. Berlin, Germany: Springer, 2013, pp. 411–431.
[34] T. F. J. M. Pasquier, J. Bacon, and D. Eyers, "FlowK: Information flow control for the cloud," in *Proc. IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, Singapore, Dec. 2014, pp. 70–77.
[35] T. F. J. M. Pasquier, J. Bacon, and B. Shand, "FlowR: Aspect oriented programming for information flow control in ruby," in *Proc 13th Int. Conf. Modularity*, New York, NY, USA, 2014, pp. 37–48.
[36] C. Priebe, D. Muthukumaran, D. O'Keeffe, D. Eyers, B. Shand, R. Kapitza, and P. Pietzuch, "CloudSafetyNet: Detecting data leakage between cloud tenants," in *Proc 6th Ed. ACM Workshop Cloud Comput. Secur.*, New York, NY, USA: 2014, pp. 117–128.
[37] Z.-Z. Wu, X.-Y. Chen, and X.-H. Du, "Enhancing sensitive data security based-on double-layer information flow controlling in the cloud," *Acta Electron. Sinica*, vol. 46, no. 9, pp. 2245–2250, Sep. 2018.
[38] T. F. J.-M. Pasquier, J. Singh, and J. Bacon, "Clouds of things need information flow control with hardware roots of trust," in *Proc. IEEE 7th Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Vancouver, BC, Canada, Nov. 2015, pp. 467–470.
[39] C. Lü, G. Qian, and T. Chen, "A cloud computing security model based on noninterference," *Wuhan Univ. J. Natural Sci.*, vol. 24, no. 3, pp. 194–200, Jun. 2019.
[40] W. Ma, H. Li, and D. Witarsyah, "A cloud computing separation model based on information flow," *Open Phys.*, vol. 17, no. 1, pp. 128–134, Apr. 2019.
[41] A. Asuncion and D. J. Newman. (Mar. 20, 2018). *UCI Machine Learning Repository[OL]*. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[42] C.-D. Lv, "Research on information flow security of cloud computing based on noninterference models," (in Chinese), Beijing Jiaotong Univ., Beijing, Chain, Tech. Rep., 2016.
[43] J. A. Goguen and J. Meseguer, "Inference control and unwinding," in *Proc. Symp. Secur. Privacy*. Oakland, CA, USA: IEEE Computer Society, 1984, pp. 75–86.
[44] R. Van Der Meyden, "What, indeed, is intransitive noninterference?" in *Computer Security*. Berlin, Germany: Springer-Verlag, 2007, pp. 235–250.
[45] L.-Y. Tian, X. Rong, and T. T. Liu, "Design and implementation of Linux file mandatory access control," in *Proc. Int. Conf. Netw. Comput. Inf. Secur.* Berlin, Germany: Springer, 2012, pp. 15–22, doi: 10.1007/978-3-642-35211-9_3.
[46] H. W. Baek, A. Srivastava, and J. V. D. Merwe, "CloudVMI: Virtual machine introspection as a cloud service," in *Proc. IEEE Int. Conf. Cloud Eng. (ICE)*, Mar. 2014, pp. 153–158.

XIN LU received the B.S. degree from the Zhengzhou Science and Technology Institute, Zhengzhou, China, in 2017, where he is currently pursuing the M.S. degree. His research interests include information security and cloud computing security.

LIFENG CAO received the Ph.D. degree from the Zhengzhou Science and Technology Institute, Zhengzhou, China, in 2013. He is currently an Associate Professor with the Zhengzhou Science and Technology Institute. His research interests include network security and information security.

XUEHUI DU received the Ph.D. degree from the Zhengzhou Science and Technology Institute, Zhengzhou, China, in 2012. She is currently a Professor with the Zhengzhou Science and Technology Institute. Her research interests include cloud computing and big data security.