J Inf Process Syst, Vol.14, No.1, pp.239~251, February 2018
https://doi.org/10.3745/JIPS.03.0089

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Study of Danger-Theory-Based Intrusion Detection Technology in Virtual Machines of Cloud Computing Environment

Ruirui Zhang* and Xin Xiao**

Abstract

In existing cloud services, information security and privacy concerns have been worried, and have become one of the major factors that hinder the popularization and promotion of cloud computing. As the cloud computing infrastructure, the security of virtual machine systems is very important. This paper presents an immune-inspired intrusion detection model in virtual machines of cloud computing environment, denoted I-VMIDS, to ensure the safety of user-level applications in client virtual machines. The model extracts system call sequences of programs, abstracts them into antigens, fuses environmental information of client virtual machines into danger signals, and implements intrusion detection by immune mechanisms. The model is capable of detecting attacks on processes which are statically tampered, and is able to detect attacks on processes which are dynamically running. Therefore, the model supports high real time. During the detection process, the model introduces information monitoring mechanism to supervise intrusion detection program, which ensures the authenticity of the test data. Experimental results show that the model does not bring much spending to the virtual machine system, and achieves good detection performance. It is feasible to apply I-VMIDS to the cloud computing platform.

Keywords

Artificial Immune, Cloud Computing, Danger Theory, Intrusion Detection, Virtual Machine

1. Introduction

Cloud computing is a new computing model, and it distributes computing tasks into the resource pool which is composed of a large number of computers. Then, many applications can get needed computing power, storage space and a variety of business services. In these cloud services which are already implemented, issues of information security and privacy protection have been concerned, and have become one of the major factors that hinder the popularization and promotion of cloud computing. At present, cloud computing is a hot studying topic, and research on the technologies of cloud computing security focuses on the following aspects: data security and privacy protection, safety of the virtualized computing environment, dynamic service authorization, access control and content auditing etc.

Manuscript received March 20, 2017; first revision June 9, 2017; accepted June 23, 2017.
Corresponding Author: Ruirui Zhang (zhangruiruisw@gmail.com)
* School of Business, Sichuan Agricultural University, Chengdu, China (zhangruiruisw@gmail.com)
** School of Computer Science, Southwest Minzu University, Chengdu, China (xiaoxin618@gmail.com)

As the infrastructure of cloud computing, the security of virtual machine (VM) systems is very important [1,2]. In the virtual machine system, VMM is between the upper VM and the lower hardware, and plays a very important role. In addition, there is a VM with a relatively high level of authority, which is called privileged VM, and it can manage and control other guest VMs to a certain extent. In the cloud computing platform, client VM provides services for users, and privileged VM and VMM are transparent for users and are managed by the cloud service provider. In this work, we use para-virtualized Xen system [3,4] as a prototype system, and use Linux as the operating system which is running in the client VM. In Xen, VMM is called hypervisor, and VM is called domain. The first domain which starts together with the hypervisor is called dom0, and other domains are called domU (non-privileged VM), as shown in Fig. 1. Due to the high privilege levels and relatively streamlined structure of dom0 and the hypervisor, it is assumed that these two are safe. The main intention of this model is to ensure the safety of user-level applications of domU.

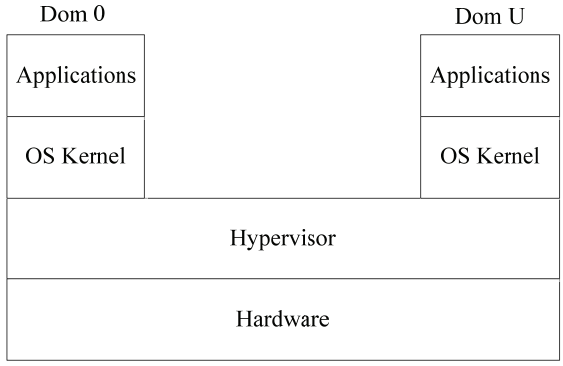

Fig. 1. Xen virtual machine system.

The remainder of this paper is organized as follows. The related work is described in Section 2. The theories of the model including description of the architecture, expressions of antigens and antibodies, implementation mechanism of danger signals, and implementation mechanism of information monitoring are described in Section 3. The effectiveness of I-VMIDS is verified using classic SPLASH-2 program group and common attacks in Section 4. Finally, the conclusion is given in the last section.

2. Related Work

In this section, two artificial immune theories being adopted in this paper are introduced, and they are the immune response mechanism and the danger theory. Then related virtual machine security technologies are briefly introduced.

2.1 Immune Response Mechanism

Fig. 2 simulates the immune response mechanism of the biological immune system, including antibodies' own evolution process and intrusion detection process of antigens. In immune-based intrusion detection systems, we define an antigen as a network request, self-antigens as normal network

requests, and non-self-antigens as abnormal network activities (network attacks). Selves and non-selves constitute the collection of the entire antigen system. Antibodies are divided into immature ones, mature ones and memory ones. Antigens are tested by mature antibodies and memory antibodies. Immature antibodies are randomly generated from the gene pool, then go through the negative selection process, which means passing self-tolerance and evolving into mature antibodies. Mature antibodies go through the clonal selection process in their life cycle, match a certain number of antigens, then will be activated to evolve into memory antibodies. If they are not activated, they will die. Memory antibodies have unlimited life cycle, and have priority to match antigens, which is similar to the secondary response.

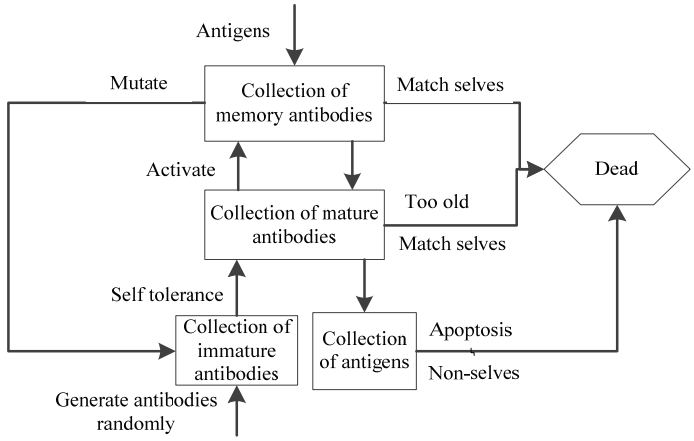

Fig. 2. Processes of antibody evolving and antigen testing.

2.2 Danger Theory

The danger theory proposed by biological immunologist Matzinger [5] believes that there are two death manners of cells in biological immune system, and they are apoptosis and necrosis. Apoptosis is a natural process, and is the result of environmental regulation in the body. Necrosis is irregular death, and is associated with stress cells or other means. This approach of death will lead to specific biochemical reactions of the body, is different from natural rules, and will produce distinct degrees of danger signals which form the basis of the immune response. Thus, the biological system produces danger signals, then conducts the immune response according to changes in the environment. Danger signals build a danger zone around them, where immune cells will be activated to take part in immune responses. Compared with the traditional CLONALG theory, danger theory introduces environmental factors of the body, describes some important characteristics of the biological immune system, and explains some immune phenomena which the traditional theory cannot explain, such as autoimmune diseases.

2.3 Related Virtual Machine Security Technologies

Currently, study on the security of virtual machine systems in cloud computing environment is not much, and existing research is briefly introduced in the following.

Haeberlen et al. [6] put forward the concept of accountable virtual machines (AVMs), where

programs were executed and related information was recorded to determine whether programs were normal. Payne et al. [7] presented the Lares system, and the system inserted a hook function in the client virtual machine which proactively monitored events of client virtual machine. Sharif et al. [8] put forward a common in-VM monitoring framework, which monitored and judged processes in untrusted guest VM. Wang et al. [9] put forward a lightweight system named HookSafe based on VM monitor, which was mainly used to monitor rootkit attacks on kernel spaces. The work in [10] and [11] are also used to detect kernel rootkits. The work in [10] monitors invariants in controlled flow transferring and constant relationships in data of uncontrolled flow. The work in [11] adopts the Daikon tool to deduce invariants from data structures which are extracted from memory pages, and monitors these invariants to determine the state of kernels. Bharadwaja et al. [12] analyzed the security issues which were raised by hypercalls in virtualized environments, and proposed a Xen-based distributed intrusion detection system. Srivastava et al. [13] studied the use of rootkit to fuzz system calls for virtual machine monitor attacks, and proposed a Xen-based monitoring system named Sherlock. Szefer et al. [14] proposed the NoHype system which runs VM directly on the underlying hardware, and maintains multiple virtual machines. Benzina and Goubault-Larrecq [15] proposed a role-based access control model for the virtual machine. Wang et al. [16] proposed a detection method for hidden processes which is based on VMM. Fang et al. [17] proposed a virtual machine based software protection method for defending against semantic attacks, named DAS-VMP. Liang et al. [18] studied the cache side channel attack technology about virtual machines in the cloud environment, and proposed a common model for driving these attacks. Zhu et al. [19] did the security research on virtualization software stack, and proposed an enhanced security solution for the stack in the aspects of software and hardware.

The above literatures studied user procedure security of VM and vulnerabilities of VMM, and proposed corresponding defense methods. However, through careful analysis, current methods cannot accurately determine the real-time status of client VM applications or the security vulnerabilities of VMM. Most of proposed methods are for particular attacks and vulnerabilities, and cannot effectively deal with threats of other attacks.

Inspired by the immune response mechanism and the danger theory of the biological immune system, this paper presents a danger-theory-based intrusion detection model in virtual machines of the cloud computing environment, named I-VMIDS. The main contributions of this model are as follows: (1) the model introduces the danger theory into VM intrusion detection, and defines the implementation of danger signals; (2) the model is capable of detecting attacks on processes which are statically tampered, and is able to detect attacks on processes which are dynamically running. Therefore, the model supports high real time; (3) the model monitors the intrusion detection process with less cost to ensure the authenticity of test data, so that the model has higher security. Experimental results show that the model does not bring much spending to the virtual machine system, and achieves good detection performance. It is feasible to apply I-VMIDS to the cloud computing platform.

3. Theories of the Model

3.1 Description of the Architecture

The architecture of I-VMIDS is shown in Fig. 3. This architecture is divided into four levels: the underlying hardware layer, the VMM layer, the VM kernel space layer, and the VM user space layer.

Modules of the model are distributed into these four levels. In order to reduce context switching between dom0 and domU and to do fine-grained monitoring, antigen presenting module and signal acquisition module are deployed in every guest VM. Immune response module and signal measurement module are deployed in the privileged VM. These two modules do not require communicating with domU, and are deployed separately in dom0, which can reduce the performance cost and improve the security of dom0. Information monitoring module is deployed in VMM. Because the guest VM is not credible, the model introduces the information monitoring module to supervise the running of antigen presenting module and signal acquisition module, in order to ensure the safety of the detection process.

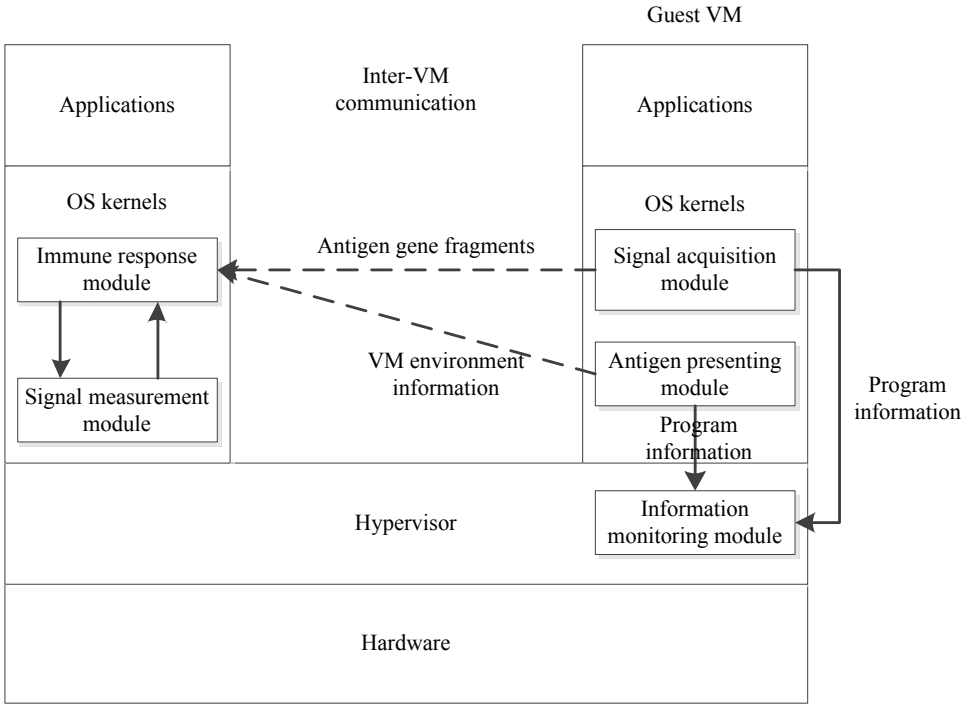

Fig. 3. Structure of the intrusion detection model.

The detection process is as follows. First, the antigen presenting module monitors executions of user-level applications in client VMs, extracts critical data as antigens, and delivers them to the immune response module in privileged VM through inter-VM communication mechanism. Meanwhile, the signal acquisition module collects environmental information when the program executes, and passes along to the signal measurement module in privileged VM. Then, the immune response module assesses whether to trigger secondary response or not based on the collection of memory antibodies. If the secondary response isn't triggered, the signal measurement module will evaluate the current environment's risk rating, produce danger signals of different degrees, and determine whether the invasion happens. If it happens, the model will start a further initial response to eliminate alien antigens. Information monitoring module periodically runs after the system starts, in order to ensure that antigen presentation module and signal acquisition module are not attacked.

3.2 Expressions of Antigens and Antibodies

Forrest et al. [20] found out that the execution of critical programs can be described by the sequence of system calls, which is also called the execution trace. The situation of system calls can reflect behavioral characteristics of the program to some extent, and the execution trace has local stability when the program is running [20, 21].

Based on biological immune principles, the virtual machine platform is defined as a biological system, client virtual machines are defined as immune tissues, user programs in VM are defined as antigens, and the process ID and the short sequence of system calls are defined as gene fragments of antigens.

Therefore, an antigen is defined as a triple $ag = <gid,pid,<sid_1,sid_2,\ldots, sid_k>>$, $sid_i \in [0,100]$, $i=1,2,\ldots,k$. Antigens represent feature vectors in the solution space of the problem domain. *gid* is the unique ID which identifies the client VM. *pid* is the process ID. *sid* is the system call ID, and its value is normalized between 0 and 100. k is the length of the short system call sequence, that is to say, the code length of immune cells, which reflects order relationships of system calls during the execution process. From the information theory, $k=6$. The entire antigens in the space are expressed as $AG = \{ag_i \mid i=1,2,\ldots,n\}$.

Antibodies can recognize antigens and trigger specific immune responses. Antibodies have the same structure as antigens, and are used for detecting and matching antigens. They are expressed as $ab = <gid,pid,<sid_1,sid_2,\ldots, sid_k>>$. The set of antibodies is expressed as $AB = \{ab_i \mid i=1,2,\ldots,n\}$.

It is assumed that normal short sequences which can be recognized by the model constitute self set S, all the unknown short sequences are defined as N, abnormal short sequences which produce danger signals are defined as D, and short sequences which are judged as invasions are defined as I.

Then, $S \cap N=\emptyset$, $S \cup N=AG$. Danger theory does not distinguish between self and non-self, only recognizes intrusion set $I=D \cap N$ which triggers immune responses, and does not respond to harmless set $D \cap S$.

3.3 Implementation Mechanism of Danger Signals

Danger theory emphasizes that danger signals which are generated from environmental changes result in various degrees of immune responses, and the area around signals is called danger zone. The most important issue of introducing danger theory into intrusion detection systems is the definition of danger signals, which means how to determine the danger. In the virtual machine environment, we select three environmental values as assessments of danger signals, and normalize them to real-valued interval [0, 100]. The three values are the number of regular files in the system N_{reg}, the memory ratio used by a process *Rss*, and the number of files reported by lsof command N_{files}.

For antigen ag_i, we define the function of danger signal $DS(ag_i)$ as below. This function takes the three environmental values N_{reg}, *Rss* and N_{files} as inputs, then generates signal values.

$$DS(ag_i)=(k_1N_{reg}+k_2Rss-k_3N_{files})/(k_1+k_2+k_3) \quad (1)$$

As can be seen, N_{reg} and *Rss* will have a negative influence on the environment. The increases of N_{reg} and *Rss* show that the environment is damaged or the probability of being damaged grows. N_{files} will have a positive influence on the environment, and the increase of N_{files} gives the environment more chances to be normal.

The size of the danger zone limits the scope of the immune response, and immune cells in the region will be activated to participate in the immune response. For antigen ag_i, we define the function of danger zone $D(ag_i)$ as below. This function returns a collection of antibodies whose distance to ag_i is less than *r_danger*.

$$D(ag_i)=\{ab_j \mid dis(ag_i,ab_j)<r\_danger\} \quad (2)$$

where, *r_danger* is the radius of the danger zone. $dis(ag_i,ab_j)$ is the distance between antigen ag_i and antibody ab_j, and reflects the combined strength between antigen and antibody.

How to determine whether the environment is damaged according to danger signals? We took advantage of the cloud model. The cloud model [22] is a probabilistic reasoning tool, and is a mathematical transformation model between the qualitative concept expressed by language values and quantitative data. The cloud model has three numerical characteristics which are expectation *Ex*, entropy *En*, and hyper entropy *He*. Based on the danger signal modeling, we use cloud rule generator and reverse cloud generator to carry out qualitative analysis of environments in guest virtual machines. Rule generator can be divided into front cloud and rear cloud. IF part is the condition of the rule, which is achieved by the front cloud, while THEN part is the result of the rule, which is implemented by the rear cloud. The inputs of front cloud are values to be tested, and the output is the membership of a certain rule which is activated by samples. The output of front cloud is also input of rear cloud, and the output of rear cloud is the conclusion of the rule.

First, danger signals $DS(ag_i)$ were sampled *m* times in a safe state and an attacked state. Then, based on obtained cloud droplets we got numerical characteristics of front cloud $\{Ex_{si},En_{si},He_{si}\}$ and $\{Ex_{di},En_{di},He_{di}\}$ through reverse cloud generator which is expressed as Table 1.

Table 1. The algorithm of reverse cloud generator

Obtain cloud's digital characteristics according to droplets. (Take concentration of consanguinity class family A_1 as an example)
Input: sample points A_{11}, …, A_{1k}.
Output: (Ex_1, En_1, He_1)
Step1. Calculate sample mean $\overline{A_1} = (1/k)\sum_{i=1}^{k} A_{1i}$, sample variance $S^2 = 1/(k-1)\sum_{i=1}^{k}(A_{1i}-\overline{A_1})^2$.
Step2. $Ex_1 = \overline{A_1}$.
Step3. $En_1 = \sqrt{\pi/2} \times (1/k)\sum_{i=1}^{k}\lvert A_{1i} - Ex\rvert$
Step4. $He_1 = \sqrt{S^2 - En^2}$.

If the secure state cloud and dangerous state cloud cover the entire state space, we can use these two clouds to determine the status of the system. This is an ideal situation. If these two clouds cannot cover the whole state space, we need to divide the empty part, and it can be divided into weak secure state cloud and weak dangerous state cloud. In general, the closer to the center of discourse domain the cloud is, the smaller entropy and hyper entropy of the cloud are; the farther from the center the cloud is, the greater entropy and hyper entropy are. Entropies and hyper entropies of clouds are next to each other, and the smaller is 0.618 times of the greater, which is from experiences. So, we can get En_{lsi}, En_{ldi}, He_{lsi}, and He_{ldi}. According to the "3En rules" of the cloud model, we can estimate expectations of weak secure state cloud and weak dangerous state cloud. Formulas are as follows.

$$Ex_{lsi}=Ex_{si}+3En_{lsi}=Ex_{si}+3*0.618En_{si} \quad (3)$$

$$Ex_{ldi}=Ex_{di}-3En_{ldi}=Ex_{di}-3*0.618En_{di} \quad (4)$$

We design rules listing in the following to build the rule generator. Then we can get the environment status and the level of membership according to actual values of danger signals.

Rule 1: IF danger signal indicator is low, THEN the system is safe. The system will not elicit the immune response, and the corresponding antibody can be deleted;

Rule 2: IF danger signal indicator is comparatively low, THEN the system is relatively safe. The system will not elicit the immune response;

Rule 3: IF danger signal indicator is comparatively high, THEN the system is relatively in danger. The system will elicit the immune response;

Rule 4: IF danger signal indicator is high, THEN the system is in danger. The system will elicit the immune response, and add corresponding mature antibody into the memory antibody collection.

When the system triggers the secondary response or danger signals trigger the initial response, antibodies will mutate based on the immune response mechanism to generate new antibodies which have higher affinity with original antigens in order to identify danger more quickly, and generate new antibodies which have lower affinity and are added into immature antibody collection in order to ensure the diversity of the immune system.

3.4 Implementation Mechanism of Information Monitoring

Antigen presenting module and signal acquisition module are deployed in domU. Because Linux is an open-source operating system, we can add these two modules into domU's kernel. Information monitoring module is deployed in VMM. To ensure antigen presentation module and signal acquisition module's safety, the model accesses memory spaces where they belong to, and performs hash computing of the memory data. The implementation mechanism needs to solve two important issues. The first one is how to find the memory space where the two modules belong to, and the second is how to use hashing to ensure that the two modules are not attacked.

VMM is responsible for managing and distributing various hardware resources, and provides virtual hardware resources for the upper operating system kernel. domU accesses the physical memory through VMM. In Linux system, system.map file is a specific kernel symbol table, and lists all the kernel symbolic names and their corresponding virtual addresses. A kernel symbol may be a variable name or a function name. Since antigen presenting module and signal acquisition module are in domU's kernel space, all the variables and functions which they contain can be found in system.map, that is to say, we can find virtual memory addresses of these variables and functions in domU. In Xen system, there are three memory structures which are virtual memory, pseudo-physical memory and machine memory. Virtual memory means that each process has a separate virtual memory address space. Pseudo-physical memory locates between virtual memory and machine memory, and each operating system of domUs believes that pseudo-physical memory is "physical memory". In fact, machine memory is real physical memory. VMM maintains a M2P (Machine to Physical) global conversion table, and each domU maintains a P2M (Physical to Machine) partial conversion table. As can be seen, we can find the pseudo-physical address which is corresponding to virtual memory address through domU's page table, and find

machine address which is corresponding to pseudo-physical address through domU's P2M table.

Through the above method, we can find the memory space where antigen presenting module and signal acquisition module belong to. Information monitoring module reads contents of all initialized data, read-only data and functions' memories in the order which is in accordance with the system.map file, and considers them as hash input. Hash computing can map an arbitrary-length binary value to a shorter fixed-length binary value, and two different inputs cannot be mapped to the same value. Therefore, we use hash computing to ensure the integrity of memory spaces of antigen presenting module and signal acquisition module. In Hypervisor, we define two variables hd_{ag} and hd_{sig}, to store cumulative hash values of antigen presenting module and signal acquisition module. They are calculated as follows.

$$hd_{ag}(i+1)=hash(hd_{ag}(i)\&r_{ag}(i+1)) \tag{5}$$

$$hd_{sig}(j+1)=hash(hd_{sig}(j)\&r_{sig}(j+1)) \tag{6}$$

In Eq. (5), $hash(x)$ is the hash function, & is a binary string concatenation operator, $r_{ag}(i)$ is the content of the i^{th} memory segment of antigen presenting module, and $hd_{ag}(i)$ is the accumulative value after i times hash computing for antigen presenting module. Meaning of Eq. (6) is by analogy. We mark the final cumulative hash values of antigen presenting module and signal acquisition module stored by hypervisor in a safe state as standard values hd_{ag}' and hd_{sig}'. Information monitoring module periodically executes. Through comparing hash values hd_{ag} and hd_{sig} which are obtained when the program is running with standard values, we can determine the security status of antigen presenting module and signal acquisition module.

4. Experimental Results and Analysis

In this section, we verified the validity of I-VMIDS through experiments, including effects on the performance of programs after joining I-VMIDS into the Xen virtual machine system and intrusion detection efficiencies of I-VMIDS. Experimental environment is as follows. All tests were performed on the ThinkPad T540p notebook. The hardware configuration is: an Intel Core i5-4300M 2.60 GHz quad-core CPU, and 8G physical memory. Xen version number is 4.4.1. There are two domains, privileged VM dom0 and guest VM dom1. These two virtual machines run Ubuntu system with the version 14.04, and the kernel version of Linux is 3.13.0.19. dom0 is allocated four VCPU and 4G physical memory, and CPU scheduling weight is set to 256, while dom1 is allocated four VCPU and 1G physical memory, and CPU scheduling weight is set to 256.

In I-VMIDS, parameters are set as follows. Danger signal parameters k_1=1, k_2=0.5, k_3=-1.5, and the radius of danger zone r_danger=5.

Experiments run 10 times and averaged results were acquired.

4.1 Performance Evaluations of the Model

The introduction of I-VMIDS to a virtual machine system will obviously bring some performance cost. In cloud computing, many applications are executed concurrently. Therefore, this section uses the

appropriate performance test to assess the impact of I-VMIDS on parallel programs. In our tests, we used the classic SPLASH-2 program group [23,24]. The programs are written in C, are composed of 12 benchmarks, and use PThread parallel mode. We randomly select five procedures for testing and Table 2 gives a brief introduction.

Table 2. Illustrations of tested parallel programs

Program name	Meaning	Parameter setting
FFT	Computing a fast Fourier transform	m = 22, p = 2, n = 65536, l = 4
LU	Splitting a sparse matrix into a product of a lower triangular matrix and an upper triangular matrix	p = 2, n = 2048, b = 16
Ocean	Simulating movements of an entire ocean through the edge of the ocean currents (non-contiguous block allocation method)	p = 4, n = 258, t = 380, e = 1e – 09
Raytrace	Path simulation of lights	p = 4, envfile = ball4
Barnes	Simulating a three-dimensional multi-body system (for instance galaxies)	p = 2, fleaves = 2

Fig. 4 shows contrasts of the five benchmarks between loading I-VMIDS and unloading I-VMIDS. As can be seen from Fig. 4, the calculation time of dom1 is longer than the original system, and the average increased time is 7.33%, up to 10.86% on LU program. This indicates that the additional cost of virtual machine system with integrated I-VMIDS is very small, and in the acceptable range. Applying I-VMIDS to cloud computing platforms will not have significant impact on parallel applications.

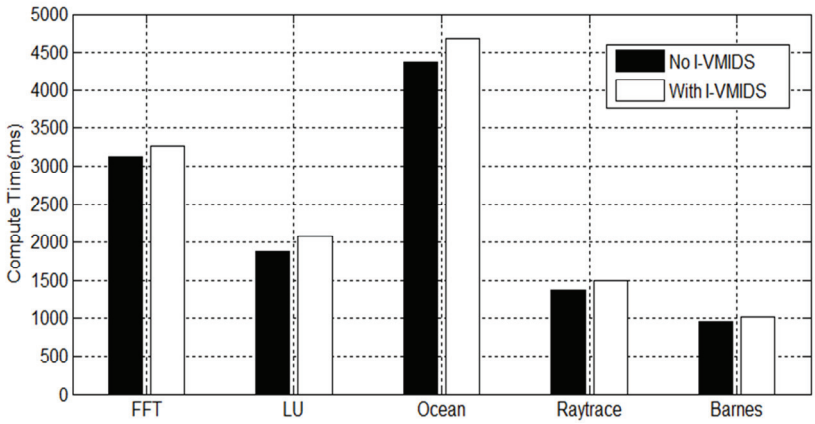

Fig. 4. Testing of parallel programs.

In I-VMIDS, the main performance overhead of domU is from antigen presenting module and signal acquisition module, and the operation of passing data to dom0 through inter-virtual machine communication mechanism. These acts are performed regularly, and the cost is limited. For example, antigen presenting module is a proactive monitoring program on system call sequence, and is not triggered by every system call. Signal acquisition module is the same. Through the event channel, domU puts antigens and environmental status into the ring buffer. Only if the ring buffer is empty, it will

notify dom0, which will cause a context switch between domU and dom0. If there is data in the ring buffer, dom0 will keep reading, and domU's notification is not required. So, the overhead of context switching is limited. In addition, implementations of immune response module, signal measurement module and information monitoring module will increase performance overhead of dom0, and the impact on domU can be ignored.

4.2 Comparisons of Detection Rates and False Alarm Rates

To test the capability of I-VMIDS to detect attacks, this section adopts wu-ftpd2.6.0 program and sendmail8.12.0 program in Linux which are widely deployed as anomaly detection applications. Attacks against wu-ftpd include the scripting attack of file name matching vulnerability, the attack of getting around access restrictions, the scripting attack of site exec vulnerability, and so on. Attacks against sendmail include the sccp attack, decode attack, remote buffer overflow attack, and so on. In the experiments, we use the detection rate *DR* and the false alarm rate *FAR* to measure the effectiveness of the model.

Experiments were divided into two phases, a training phase and a testing phase. In the training phase, the model gathers the system status and the corresponding data when the system is under normal circumstances and under attacks. The system constructs rule generators by modeling normal data of the cloud model, produces immune cell collections of various types after learning (including memory antibodies, mature antibodies, selves, etc.), and calculates standard hash values of antigen presenting module and signal acquisition module. The main job of the detection phase is to determine whether domU is under attack.

Table 3 lists comparisons of detection rates and false alarm rates of I-VMIDS, HookSafe [9] and Sherlock [13] when wu-ftpd or sendmail is under attack, and variances are in parentheses. As can be seen from Table 3, I-VMIDS has higher detection rates and lower false alarm rates under various attacks, and is feasible in practical applications.

Table 3. Comparisons of I-VMIDS, HookSafe, and Sherlock

		I-VMIDS		HookSafe		Sherlock	
		DR%	**FAR%**	**DR%**	**FAR%**	**DR%**	**FAR%**
wu-ftpd	File name matching vulnerability	96.55 (1.14)	7.22 (1.22)	75.14 (3.16)	14.28 (3.64)	84.58 (4.52)	9.50 (3.99)
	Site exec vulnerability	97.31 (1.23)	6.65 (2.01)	76.04 (4.53)	12.36 (3.04)	87.15 (3.78)	11.27 (3.50)
	Attack of getting around access restrictions	97.02 (1.08)	7.43 (1.67)	78.30 (3.01)	13.08 (3.55)	87.24 (4.71)	9.87 (3.89)
sendmail	sccp attack	98.11 (1.25)	5.15 (1.63)	80.09 (5.22)	11.46 (4.71)	90.14 (3.25)	8.77 (4.13)
	Decode attack	98.35 (1.01)	5.42 (1.69)	79.01 (6.02)	15.82 (5.27)	85.04 (4.74)	7.48 (3.20)
	Remote buffer overflow attack	98.78 (1.14)	5.80 (1.28)	83.32 (4.44)	8.87 (3.54)	91.15 (3.81)	7.09 (3.29)

Variances are in parentheses.

5. Conclusions

In existing cloud services, information security and privacy concerns have been worried, and have become one of the major factors that hinder the popularization and promotion of cloud computing. As the cloud computing infrastructure, the security of virtual machine systems is very important. This paper presents an immune-inspired intrusion detection model in virtual machines of cloud computing environment to ensure the safety of user-level applications in client virtual machines. The model extracts system call sequences of programs, abstracts them into antigens, fuses environmental information of client virtual machines into danger signals, and implements intrusion detection by immune mechanisms. During the detection process, the model introduces information monitoring mechanism to supervise intrusion detection program, which ensures the authenticity of the test data. So, the model has higher security. Experimental results show that the model does not bring much spending to the virtual machine system, and achieves good detection performance. It is feasible to apply I-VMIDS to the cloud computing platform.

Acknowledgement

The authors would like to thank Sichuan Agricultural University Double Support Project for providing financial aid.

References

[1] Z. Y. Qin, R. S. Shen, Q. F. Zhang, and Y. X. Di, "Survey on virtual machine system security," *Application Research of Computers*, vol. 29, no. 5, pp. 1618-1622, 2012.

[2] L. M. Cao and F. Y. Zhao, "Security detection of virtual machine process in private cloud platform," *Application Research of Computers*, vol. 30, no. 5, pp. 1495-1499, 2013.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, 2003, pp. 164-177.

[4] D. Chisnall, *The Definitive Guide to the Xen Hypervisor*. Upper Saddle River, NJ: Prentice-Hall, 2007.

[5] P. Matzinger, "The danger model: a renewed sense of self," *Science*, vol. 296, no. 5566, pp. 301-305, 2002.

[6] A. Haeberlen, P. Aditya, R. Rodrigues, and P. Druschel, "Accountable virtual machines," in *Proceedings of 9th USENIX Symposium on Operating Systems Design and Implementation*, Vancouver, Canada, 2010, pp. 119-134.

[7] B. D. Payne, M. Carbone, M. Sharif, and W. Lee, "Lares: an architecture for secure active monitoring using virtualization," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 2008, pp. 233-247.

[8] M. Sharif, W. Lee, W. Cui, and A. Lanzi, "Secure in-VM monitoring using hardware virtualization," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, Chicago, IL, 2009, pp. 477-487.

[9] Z. Wang, X. Jiang, W. Cui, and P. Ning, "Countering kernel rootkits with lightweight hook protection," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, Chicago, IL, 2009, pp. 545-554.

[10] O. S. Hofmann, A. M. Dunn, S. Kim, I. Roy, and E. Witchel, "Ensuring operating system kernel integrity with OSck," in *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems*, New Beach, CA, 2011, pp. 279-290.

[11] A. Baliga, V. Ganapathy, and L. Iftode, "Detecting kernel-level rootkits using data structure invariants," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 5, pp. 670-684, 2011.

[12] S. Bharadwaja, W. Q. Sun, M. Niamat, and F. Shen, "Collabra: a Xen hypervisor based collaborative intrusion detection system," in *Proceedings of the 8th International Conference on Information Technology: New Generations*, Toledo, OH, 2011, pp. 695-700.

[13] A. Srivastava, A. Lanzi, J. Giffin, and D. Balzarotti, "Operating system interface obfuscation and the revealing of hidden operations," in *Proceedings of the 8th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Amsterdam, the Netherlands, 2011, pp. 214-233.

[14] J. Szefer, E. Keller, R. B. Lee, and J. Rexford, "Eliminating the hypervisor attack surface for a more secure cloud," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, Chicago, IL, 2011, pp. 401-412.

[15] H. Benzina and J. Goubault-Larrecq, "Some ideas on virtualized system security, and monitors," in *Proceedings of the 5th International Workshop on Data Privacy Management*, Athens, Greece, 2010, pp. 244-258.

[16] L. Wang, H. Gao, W. Liu, and P. Yang, "Detecting and managing hidden process via hypervisor," *Journal of Computer Research and Development*, vol. 48, no. 8, pp. 1534-1541, 2011.

[17] D. Fang, H. Zhang, Z. Tang, and X. Chen, "DAS-VMP: a virtual machine-based software protection method for defending against semantic attacks," *Journal of Sichuan University (Engineering Science Edition)*, vol. 49, no. 1, pp. 159-168, 2017.

[18] X. Liang, X. L. Gui, H. J. Dai, and C. Zhang, "Cross-VM cache side channel attacks in cloud: a survey," *Chinese Journal of Computers*, vol. 40, no. 2, pp. 317-336, 2017.

[19] M. Zhu, B. B. Tu, and D. Meng, "The security research of virtualization software stack," *Chinese Journal of Computers*, vol. 40, no. 2, pp. 481-504, 2017.

[20] S. Forrest, A. S. Perelason, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 1994, pp. 202-212.

[21] X. Tian, L. Gao, C. Sun, and A. Zhang, "Anomaly Detection of Program Behaviors Based on System Calls and Homogeneous Markov Chain Models", *Journal of Computer Research & Development*, vol. 44, no. 9, 2007, pp. 1538-1544.

[22] D. Y. Li, C. Y. Liu, Y. Du, and X. Han, "Artificial intelligence with uncertainty," *Journal of Software*, vol. 15, no. 11, pp. 1583-1594, 2004.

[23] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," in *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, S. Margherita Ligure, Italy, 1995, pp. 24-36.

[24] J. P. Singh, W. D. Weber, and A. Gupta, "SPLASH: Stanford parallel applications for shared-memory," *ACM SIGARCH Computer Architecture News*, vol. 20, no. 1, pp. 5-44, 1992.

Ruirui Zhang https://orcid.org/0000-0003-1898-1487

She received B.S., M.S., and Ph.D. degrees in School of Computer Science from Sichuan University in 2004, 2007, and 2012, respectively. Her current research interests include network security, wireless sensor networks, intrusion detection and artificial immune systems.

Xin Xiao https://orcid.org/0000-0001-8703-4243

She received B.S., M.S., and Ph.D. degrees in School of Computer Science from Sichuan University in 2004, 2009, and 2015 respectively. Her current research interests include network security, intrusion detection and artificial immune systems.