J Supercomput (2017) 73:4407–4427
DOI 10.1007/s11227-017-2025-7

A game theoretic-based distributed detection method for VM-to-hypervisor attacks in cloud environment

Amin Nezarat[1] · **Yaser Shams**[2]

Published online: 30 March 2017

Abstract Cloud computing is a pool of scalable virtual resources serving a large number of users who pay fees depending on the extent of utilized service. From payment perspective, cloud is like electricity and water as people who use more of this shared pool should pay larger fees. Cloud computing involves a diverse set of technologies including networking, virtualization and transaction scheduling. Thus, it is vulnerable to a wide range of security threats. Some of the most important security issues threatening the cloud computing systems originate from virtualization technology, as it constitutes the main body and basis of these systems. The most important virtualization-based security threats include VM side channel, VM escape and rootkit attacks. The previous works on the subject of virtualization security rely on hardware approaches such as the use of firewalls, which are expensive, the use of schedulers to control the side channels along with noise injection, which impose high overhead, or the use of agents to collect information and send them back to a central intrusion detection system, which itself can become the target of attacker. In the method presented in this paper, a group of mobile agents act as the sensors of invalid actions in the cloud environment. They start a noncooperative game with the suspected attacker and then calculate the Nash equilibrium value and utility so as to differentiate an attack from legitimate requests and determine the severity of attack and its point of origin. The simulation results show that this method can detect the attacks with 86% accuracy. The use of mobile agents and their trainability feature has led to reduced system overhead and accelerated detection process.

✉ Amin Nezarat
aminnezarat@pnu.ac.ir

[1] Department of Computer Science and Engineering, Payame Noor University, Yazd, Islamic Republic of Iran

[2] Department of Computer Science and Engineering, Yazd Branch, Azad-e-Eslami University, Yazd, Islamic Republic of Iran

Keywords Cloud computing · Intrusion detection systems · Virtualization · Game theory · Nash equilibrium

1 Introduction

Cloud computing is a model for provision of easy on-demand network access to a set of variable and configurable computing resources such as networks, servers, storage spaces, applications and services that can be rapidly provisioned and released with minimal resource management or direct involvement of service provider [1]. With the advancement of information technology, computing tasks have found their way into almost all aspects of modern life. Meanwhile, ever rising competition and efficiency requirements have pushed organizations, enterprises and individuals to seek new cost-effective hardware and software solutions for their computing tasks. Cloud computing is the latest response of technology to these needs and requirements. This technology enables fast and convenient web-based access to computing resources and therefore leads to not only lower costs but also to reduced concerns regarding issues such as scalability, resource provisioning and flexibility. In the cloud, details are hidden from the user and therefore the users do not need any expertise regarding the cloud infrastructure and technology they use [2].

With the gradual rise of cloud-based computing services, there seems to be a growing concern about the adequacy of security provided for these services. Infrastructure of the cloud services largely depends on the virtualization technology. Virtualization and multi-tenancy allow multiple entities to work simultaneously on the same physical machine. Virtualization alters the relationship between the operating system and the hardware and adds another layer called "hypervisor," which must be properly configured, managed and secured. The connection of multiple servers to a single host and the lack of a physical partition between servers impose significant threats to the security of cloud architectures [3].

Attacks on virtual machines (VMs) can be divided into four categories: VM-to-VM attacks, VM-to-hypervisor attacks, hypervisor-to-VM attacks, and hypervisor-to-hypervisor attacks. A malicious VM can use hypervisor rather than physical network to create new communication channels; thus, the sole use of traditional network security measures such as firewalls, intrusion prevention systems (IPS) and intrusion detection systems (IDS) cannot contain the new threats. This highlights the importance of new structure to protect network security without network dependency. Currently, complete isolation and separation of two VMs using a VM monitor is practically impossible, and VM monitors have their own security shortcomings [4] such as Venom security hole discovered in 2015 in Xen and KVM. Various guest kernel monitoring software programs such as Secvisor, Lares, Ganglia, KVM-L4, whose aim is to monitor the behavior of virtual machine and kernel codes, lack any protection measure against security attacks [5]. The alternative measure is the use of intrusion prevention systems developed to protect network resources against attacks. However, the sheer breadth of internet-based attacks and the change in their form from centralized to distributed from have pushed the architecture of these systems toward distributed designs. There appears to be a day-by-day increase in the complexity of attacks and their technology,

and as a result, the computer networks are becoming more vulnerable. For example, attacks have now achieved such levels of sophistication that attacker can easily shut-down a small e-commerce company. Intrusion detection systems should be able to provide a decent protection against such attacks and mitigate the vulnerability [6].

The recent intrusion detection systems (IDS) can be divided into two groups:

- Reactive (concerned with signature detection or permission for login/logout)
- Proactive (secure overlay network, proxy)

The architecture of secure overlay network allows it to be used for provision of pre-emptive readiness against distributed attacks, but the method this architecture uses to communicate with nodes is based on constant contact. This constant contact imposes a large overhead on the network and acts as a barrier ahead of more widespread use of this architecture. The proactive approach could be much more efficient than the reactive method, but its effective implementation still faces many problems and needs further work [7].

Centralized intrusion detection systems are highly susceptible to single-point failures or holes, which can be discovered and exploited by attackers. To resolve this issue, a large number of intrusion detection systems can be used to reduce the number of undetected attacks, an approach which of course imposes further costs. The power of an IDS is directly related to its ability to create a balance between the number of defenders and the number of misdetection or detection failures. Therefore, it seems that the use of a distributed intrusion detection scheme can be a valid approach to handle the attacks and measure the degree of imposed security risk (the threat of an attack) and facilitate the decision-making process necessary for provision of adequate countermeasures.

2 Theoretical principles and concepts

2.1 Definition of cloud computing

Cloud computing is a new model based on the use of computer networks such as internet for provision, use and delivery of computing services (including infrastructure, software, platform, and other computing resources). The word “Cloud” in cloud computing is a metaphor for a network or a vast network of networks (e.g., the internet), wherein a normal user has no idea about what happens in the background. The internet is likened to a cloud because it hides its technical details from the user by putting an abstract layer between them. A cloud computing service or software provider could, for example, provide a set of online business applications working through a web browser or other software; in this type of service, applications and data are stored on the server and become available on demand, details are hidden from the user, and users do not need any expertise regarding the cloud infrastructure and technology that they allow to receive the service [8].

The most important advantages of cloud computing include: hardware-independence, rapid software upgrades, global access to applications and data, unlimited storage capacity, lower cost, simplicity of group interactions, greater data reliability, dynamicity and agility, and measurability.

2.2 Cloud computing architecture

Three major cloud computing models are: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Each of these models has different effects on software security. These three models are briefly introduced in the following (see Fig. 1) [9,10]:

2.2.1 Software as a service (SaaS)

This architecture provides on-demand service in form of a single process of an application in a cloud environment simultaneously servicing multiple end-users.

2.2.2 Platform as a service (PaaS)

This service allows the client to put purchased or custom-built software on the cloud platform, and control, test or change it.

2.2.3 Infrastructure as a service (SaaS)

This service provides the processing power, network, storage as well as basic computing resources and therefore eliminates the clients need to purchase hardware and network equipment.

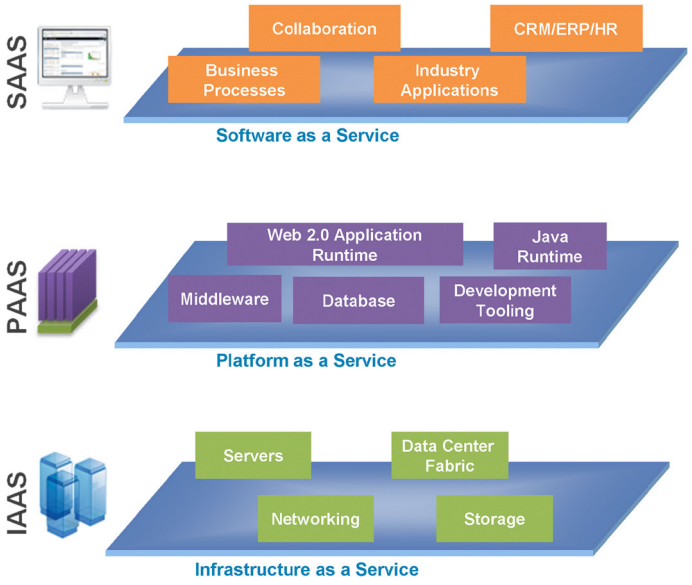

Fig. 1 Cloud computing architecture [10]

2.3 Cloud computing security

Cloud computing security is a subset of computer security, network security and in a broader sense the information security; it refers to a group of policies, technologies and controls that are used to protect data, applications and infrastructure related to cloud computing. The greatest concern of cloud computing security is the infrastructure, and the key element of cloud infrastructure is the virtualization technology. Virtualization alters the relationship between the hardware and the operating system and adds other elements that need to be properly configured, managed and secured.

Cloud computing security concerns can be divided into two categories: security issues on the part of cloud computing service provider (any organizations offering cloud-based infrastructure, platform, and software) and security issues on the part of clients. Cloud computing service provider should make sure that its infrastructure is secure and clients' data and programs are protected [11,12].

In VM attacks, attacker takes control of the virtual machine and keeps it running to monitor CPU cache or memory and examine the victim's behavior. This attack requires the victim's and attacker's virtual machines to be on the same hardware. The success of this procedure results in leakage of sensitive information about the client and the cloud provider. Clients are often not permitted to check the side channels, and although cloud providers can fully investigate the time and type of attacks, they may be reluctant to report them as it may hurt their brand and credibility [12].

2.4 Game theory

This study employs concepts such as game theory, noncooperative game and Nash equilibrium; thus, the basic principles of these concepts need to be explained. Game theory is a branch of applied mathematics with wide ranging applications in the social sciences, and especially in economics, biology, engineering, political science, international relations, computer science, marketing and philosophy. Game theory is an attempt to use mathematics to estimate the behavior in strategic situations (called the game), in which the success of an individual depends on choices made by the others. A game consists of a set of players, a set of moves or strategies and the known results of each combination of strategies. The prospect of winning is subject not only to chance but also to rules and principles of the game, and in the course of the game, each player tries to use these rules to increase his chance of winning [13].

2.5 Nash equilibrium in noncooperative games

Nash equilibrium is a concept in the game theory that has found many applications in the economy. Strategies adopted by players of a game result in a Nash equilibrium if no player has any incentive to change his strategy as long as other players hold on to their own strategies. In other words, in Nash equilibrium, strategy of each player is the best response to the strategy adopted by rivals; therefore, a player unilaterally deviating from his strategy will definitely fail to achieve a better outcome [14].

Of course, a Nash equilibrium does not necessarily mean that players are happy with the strategy of their rivals, but rather that their strategy is the best response they can come up with to counter their rivals, and this stability is what makes the Nash equilibrium more interesting.

To describe the mathematical definition of Nash equilibrium, assume the set (S, F) as a game with n players, where S_i is the set of strategies for player i, $S = S_1 * S_2 \ldots * S_n$ is the set of its strategy profiles, and $F(x)$ is its utility function. In addition, assume X_{-i} as strategy space of all players except player i. When a player selects the strategy X_i $(i \in N)$, the result is the strategy profile $x = x_1, \ldots, x_n$ and the utility function (FX_i). It should be noted that the utility function depends on the selected strategy profile. The strategy profile $X^* \in S$ is Nash equilibrium if no unilateral deviation in strategy by any single player can bring him more profit [22].

$$\forall_i, X_i \in S_i, X_i \neq X_i * : f_i\,(x_i *, x_{-i} *) > f_i\,(x_i, x_{-i} *) \tag{1}$$

3 Review of literature

Security improvement in the context of cloud computing and virtual machines has been the subject of numerous studies, of which most have used a number of similar methods to address the issue. Examples are as follows:

Liu et al. [15] showed that the VM isolation can be disrupted by side channel attacks. Their research suggests that an attacker can use memory bus manipulation and two malicious software in different virtual machines (but the same physical host) to exploit memory access latencies as a secret channel and extract sensitive security information, such as user passwords or credit card numbers by bypassing the access control policies (see Fig. 2). They proposed memory channel scheduling and addition of periodic noise as measures to counter this type of attack.

This scheduler is able to control time-based interferences in different virtual machines and also injects a periodic noise to reduce potential threat from side channels

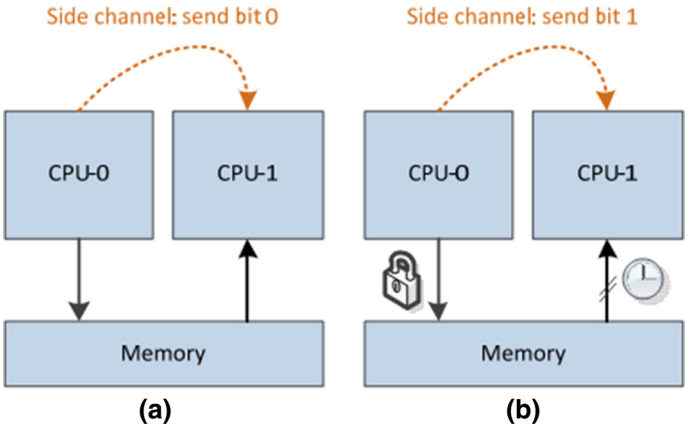

Fig. 2 Intrusion through side channel [24]

by increasing the error rate of colluding attackers. The drawback of this approach is that it only reduces the vulnerability to this attack and does not eliminate it and that the widespread use of side channel in the clouds makes the application of such scheduling scheme much more difficult. Another drawback of this method is that it is not intelligent, in the sense that there is no control on the behavior of VMs for attack detection, and one must only rely on the scheduler and the noise for protection. Another problem of this method is the challenge of finding the appropriate threshold for channel scheduling in order to establish a balance between security and performance. In case of selecting a short cycle period for scheduler, the number of switches will be high and this will slow down the system, and in case of selecting a long period system security will be compromised.

Kong et al. [16] and Aciicmez et al. [17] provided some hardware solutions for detection of side channel attack. However, hardware solutions are very expensive and cannot be used in existing systems.

Lombardi and Di Pietro [18] studied the virtualization in cloud computing services and a number of related threats and attacks and then proposed a new architecture and a technical solution for the studied system. Their proposal is a cloud system with incorporated firewalls, intrusion detection and prevention measures. In addition to academic research into this subject, industry-based researches have also contributed to addressing the discussed threats. Amazon Elastic Compute Cloud (EC2) has provided a special service allowing tenants to use specially assigned instances to ensure that their VM is running without other VMs working on the same host. This solution can effectively eliminate some of the side channels that use the shared hardware to transmit messages (including side channel cache and side channel memory). In this method, however, the hardware source cannot be fully exploited and this forces the user to pay higher prices for using this service. Our scheduler can provide the same level of isolation with a high hardware utilization ratio for cloud platform to ensure the cost-effectiveness of service.

Eid [19] proposed a mobile agent-based intrusion detection system that can detect intrusions from both inside and outside the network. In this method, sniffing is done by means of mobile agents which are tasked with collecting data and sending them back to a core system for analysis. The proposed model consists of three parts: intrusion detection processor, mobile agent and sniffers (see Fig. 3).

- Intrusion detection processor: this component is the cornerstone of framework and is responsible for network monitoring. This unit is placed on a strategic node and provides the following services: network traffic monitoring, integration of data sent to mobile agents as well as implementation of multi-point detection especially for distributed attacks from within the network, low level monitoring of connections within the network through packet scanning, and monitoring the network vulnerabilities by checking the local intrusion signatures.
- Mobile Agent Platform: this unit can create, interpret, run, transfer and terminate agents.
- Distributed sensors: these sniffers allow software or hardware to monitor network traffic.

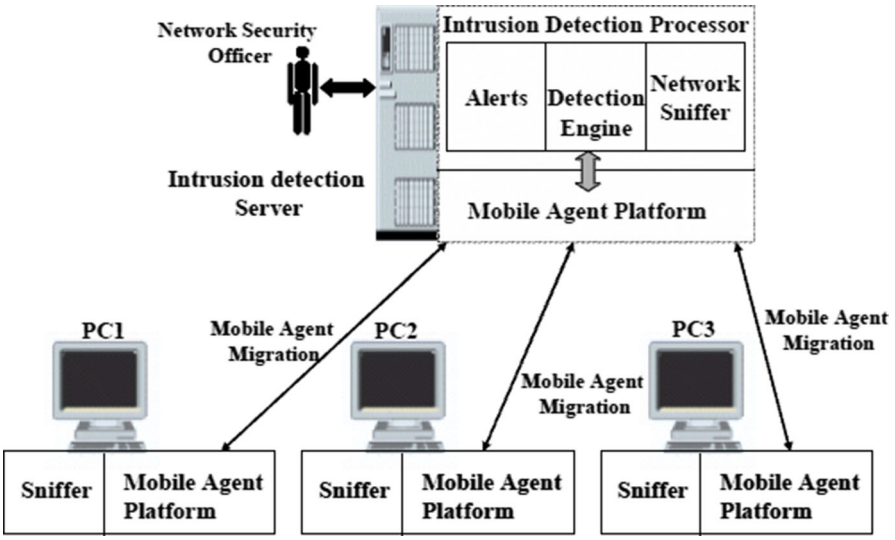

Fig. 3 Architecture of mobile agent-based intrusion detection system [20]

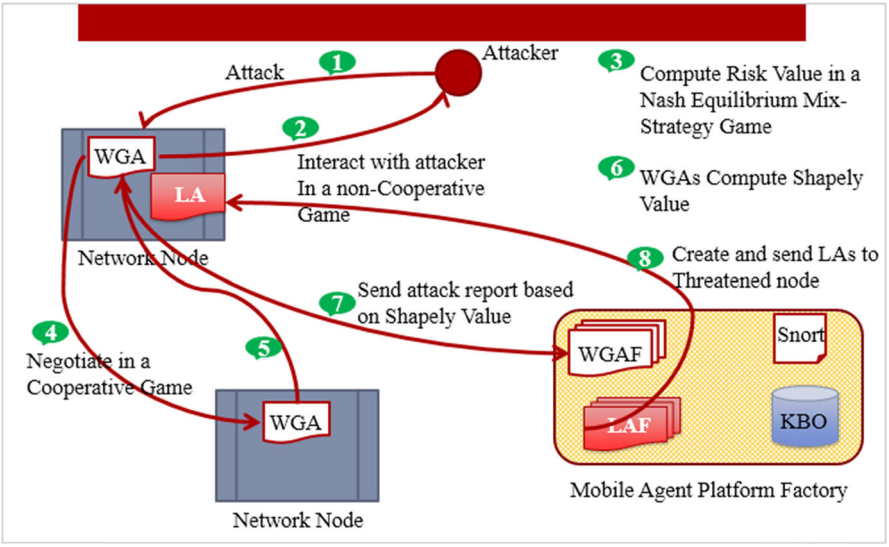

Fig. 4 Distributed intrusion detection system proposed in [20]

The advantage of this method is in the use of mobile agents to reduce network overhead and detect distributed attacks; however, its drawback is that its intrusion detection processor is concentrated on a single node, which can turn it into a new target for attackers.

Nezarat [20,28,29] has provided a mobile agent-based model, wherein agents act as the sensors of invalid actions (see Fig. 4).

Table 1 Overview of literature

Author(s)	Goal	Cost	Overhead	Speed
Liu et al.	Preventing side channel attacks on memory by injecting noise	Low	High	Low
Kong et al.	Hardware method to prevent side channel attacks on memory	High	High	Low
Aciicmez et al.	Hardware method to prevent side channel attacks on storage devices	High	High	Low
Lombardi et al.	Preventing distributed attacks with firewalls and intrusion detection system	Low	Medium	Medium
Eid	Preventing distributed attacks with mobile agents	Low	Low	Medium
Nezarat et al.	Preventing distributed attacks with mobile agents and game theory	Low	Medium	High

Table 2 Probability and cost of intrusion detection by sensor

		Cost
Intrusion sensors		
P_d	Probability of intrusion detection	$-C_1$
$1 - P_d$	Probability of detection failure	C_2
P_f	Probability of misdetection	C_3

Mobile sensors named “White Globules Agents (WGA)” move sporadically from one node of the network to another, providing a secure overlay network by using a type of noncooperative/cooperative game and inter-agent connections which, after arriving at the Shapley value allow them to detect and report the extent and origin of the attack. Nezarat’s study proposes a method where WGA plays a noncooperative game with the attacker and tries to calculate the Nash value and achieve maximum utility in order not only to separate attacks from actual requests but also determine the extent and intensity of the attack with the help of other WGAs. The advantage of this method is the use of noncooperative and cooperative games for the detection of distributed attacks with high detection accuracy and low overhead. The disadvantage is this approach is that agents playing the cooperative game need to form a coalition and when there are high numbers of agents in a coalition, the process of calculating all inter-agent states to calculate the Shapley value becomes very time-consuming. Considering that the structure used in this paper has a distributed architecture, we decided to use its structure as the basis of our work to propose a localized model for cloud environment (Tables 1, 2).

4 The proposed model

The proposed model employs a game wherein players are attackers and intrusion sensors. The attackers are virtual machines leased to users, and the task of detecting the distributed attacks is carried out by intrusion sensors operating along with the hypervisor. The first step is to arrange a noncooperative game, where players take a number of steps representing their behavior on the cloud environment. The detector monitors the behavior of virtual machines and the game continues with creating utility matrix and finding the Nash equilibrium. The location at which Nash point is created is used as a measure to determine whether VM is an attacker or a regular user. To reduce the rate of false alarms, the proposed model is equipped with a threshold which prevents VMs with mostly normal behavior from being blocked after just a few abnormal steps. The components of architecture considered in this paper are described in the following subsections.

4.1 Sensor generation unit

Security systems should be able to carry out their duties with minimal network disruption as well as minimal overhead. Intrusion sensors are autonomous and proactive software units can move across the network without constant involvement of the main server and carry out their tasks such as data collection and initiation of countermeasures and then report the results without needing to backtrack to the server and occupy the band [21]. Sensor generation unit is tasked with creating an ID number, assigning and removing the sensors. This unit also holds all information regarding currently running sensors and their locations. At the onset of the process, this unit must determine the location of the sensors to be deployed in the cloud infrastructure, a task which can also be carried out manually by the system administrator. Sensors have a strong mobility, which means that they carry all previous states and results regarding their relationships with the attacker VMs. Intrusion sensors also carry a summary of network attack scenarios and can identify suspicious behavior and then connect to the suspicious VM to approve or correct their suspicion.

In the proposed method, the number of intrusion sensors is flexible, meaning that when under attack, the system deploys a larger number of sensors to provide higher level of security, and once critical situation is dealt with the number of sensors returns to normal.

4.2 Noncooperative game

Game theory has a wide range of practical applications, one of which is to determine how decision makers act in a competitive environment where results of decisions of each stakeholder affect the results obtained by the others. In most analyses, the main structure of game theory is represented by a multidimensional matrix. Each game consists of three types of basic elements:

- Players: players represent the decision makers. In the proposed method, players include attacker VMs and intrusion sensors.
- Actions: actions or strategies are defined as the set of decisions that each player can make. The action set of attacker VM includes pure attack, distributed attack and no attack, and the action set of sensors includes detection and no detection.
- Payoff function: payoff function of players' actions are denoted by $S_{\text{attack}} = S_1, S_2, S_3$ for attackers and by $d_{\text{Agent}} = d_1, d_2$ for sensors.

4.3 Problem formulation

In the game, every action has a positive or negative cost. For agents operating as defenders, we assume P_{d} as the probability of intrusion detection, (1-P_{d}) as the probability of detection failure, and P_{f} as the probability of misdetection. Also, $-C_1$ is the cost of detection of an attack by an agent, C_2 is the cost of detection failure, and C_3 is the cost of misdetection. We also assume m_i as the mobility rate of agent moving from one VM to another. Mobility rates is displayed by m_i is a concept in the model that shows the extent of displacement of an intrusion detection agent between virtual machines. This extent or amount is initially determined by the system administrator. The amount must be selected in such a way that, on the one hand, it is not too much to increase the amount of displacement of an agent since too much displacement of an agent may increases the cost and overhead of the system, and on the other hand, it should not be too less in such a way that virtual machines enjoy no intrusion detection agent during long intervals. Given the number of attacks, two important factors, namely the number of intrusion detection agents and their displacement, are subject to change during the lifetime of the system. In the model simulation, intrusion detection agents were selected to be one-fourth of the total number of virtual machines and mobility rate used was 0.4. This amount (i.e., 0.4) represents the movement of an intrusion detection agent among four virtual machines.

The agent can make two types of errors: The first is to approve an attacker as a normal user and the second is to mistake a normal user for an attacker; and here, we aim to use the game theory to minimize these errors.

On the other hand, an attacker VM successfully infiltrating the system gains the cost $-b_1$, and in case of an unsuccessful attack, the attacker should pay cost b_2. In the case that an attacker VM attacks sensor i located on one of the nodes, we assume i_1 to be the extent of attack and i_2 to be the bandwidth that the attacker occupies to attack agent i. It can be concluded that a propagation attack occupies λf_i of bandwidth.

The minus in the formula reflects the success of the agent or the attacker. Applying minus in the formula indicates that a player that succeeded should pay less. It is the same as $-b_1$ that indicates the success of the attacker's intrusion. This success is calculated in Table 3 in such a way that the attacker must pay a lower cost and that means success; if the attacker cannot intrude or the intrusion is detected by the agent, the cost that must be paid by the attacker increases and this indicates the failure of the attacker.

Table 3 Cost of attack by VM attacker

Cost	Outcome
Attacker VM	
$-b_1$	Successful infiltration
b_2	Unsuccessful infiltration

Finally, the following utility function is defined for sensors and attacker. Utility function of attacker VM is also shown below. Here, r_1 is a factor representing the maliciousness of attacker behavior.

$$\text{payoff}_{\text{att}} = r_1[p_d b_2 - b_1 (1 - p_d)] \tag{2}$$

Meanwhile, utility function of intrusion sensor is as follows:

$$\text{payoff}_{\text{WGA}} = r_1 c_3 + p_d c_2 - r_1 p_d (c_1 + c_2 + c_3) \tag{3}$$

Utility function represents the formula for the generated status in attacker and agent. According to the assumed strategies, the utility function was extended to obtain the data presented in Table 4. The reason for the extension of utility function is the kind of the selected game type of attacker and agent of intrusion detector, each of which having their own strategies. After obtaining the utility function of intrusion sensors and attacker VM, the next step is to determine the actions or strategies of each player. In the game theory, strategy is defined as "optimal use of skills in the game"; in other words, strategy is the skill of playing well or the aptitude of the player to optimally apply his skills. In the following, we introduce the strategies or actions of attackers and agents. The action set of attacker VM is $S_{\text{attacker}} = \{u_1, u_2, u_3\}$ where:

- u_1 represents a pure attack with the probability of r_1;
- u_2 represents a propagation attack with the probability of r_2;
- u_3 represents the absence of any action with the probability of $1 - r_1 - r_2$.

The action set of intrusion sensor is $S_{\text{agent}}=\{d_1, d_2\}$ where:

- d_1 represents detection of an intrusion with the probability of q;
- d_2 represents the absence of any action in response to attack with the probability of $1 - q$.

The next step is to create the strategic form of the game. In games with limited number of players and strategies, the selected strategies can be represented by a matrix. This matrix is created through the following procedure: Rows of the matrix represent the strategies or actions of the first player, and columns of the matrix represent the actions of the second player. Each element of the matrix is composed of two numbers: The first number (from the left) is the outcome of this strategy state for the first player and the second one is the outcome for the second player. With the above description, the attacker–agent matrix is as shown in Table 4.

In the above table:

Table 4 Outcome matrix for the game played by intrusion sensor and attacker's VM

Intrusion sensor i		
Attacker VM	d_1	d_2
u_1	$b_1 f_i, -c_1 (1 + p_d + m_i)$	$-b_2 f_i, c_3 (1 + (1 - p_d))$
u_2	$b_1 (1 + \lambda f_i), -c_1 (1 + p_d + m_i)$	$b_2 (1 + \lambda f_i), -c_3 (1 + (1 - p_d))$
u_3	$0, c_2 (1 + p_f + m_i)$	0, 0

- m_i is the mobility of intrusion sensor among virtual machines;
- fi is the bandwidth occupied for the attack;
- λf_i is the propagation of attack.

The above matrix shows the utility value of the two players in different strategy scenarios. It is assumed that both players have access to this matrix and can determine the state of other player. In other words, the defender has full knowledge about its own actions and the response of the attacker. At this stage, each player must select an action based on which there would be no incentive to change the play and the other player must also arrive at the same point (Nash equilibrium), denoted by $(r*, q*)$. By definition, a set of strategies adopted by the players of a game constitute a Nash equilibrium if no player has any incentive to change his strategy as long as other players hold on to their own strategies. In other words, in Nash equilibrium, the strategy of each player is the best response to the strategy adopted by the other players; thus, a player unilaterally deviating from his strategy will stand to lose its current gains. Nash equilibrium of the above matrix can be found by the use of the following equations. The following pseudocode shows the location of the establishment of Nash equilibrium in the matrix. It is in such a way that the column and the row in which the Nash equilibrium is established are the basis of our decision on whether the virtual machine is an attacker or a normal virtual machine. For example, if the location of the establishment of Nash equilibrium is in column d1, which corresponds to the detection of an attack by intrusion detection sensor, and row u2, which corresponds to proliferative attack, it indicates that an attack has taken place.

$$\textbf{if} \quad b_1 f_i > b_1(1+\lambda f_i) \textbf{ and } \quad b_1 f_i > 0 \quad \textbf{then}$$
$$r_1{}^* = b_1 f_i$$
$$\textbf{else if} \quad b_1 f_i < b_1(1+\lambda f_i) \textbf{ and } \quad b_1(1+\lambda f_i) > 0 \quad \textbf{then}$$
$$r_1{}^* = b_1(1+\lambda f_i)$$
$$\textbf{else} \quad r_1{}^* = 0$$

$$\textbf{if} \quad -b_2 f_i > b_2(1+\lambda f_i) \quad \textbf{and} \quad b_1 f_i > 0 \quad \textbf{then}$$
$$r_2{}^* = -b_2 f_i$$
$$\textbf{else if} \quad -b_2 f_i < b_2(1+\lambda f_i) \quad \textbf{and} \quad b_2(1+\lambda f_i) > 0 \quad \textbf{then}$$
$$r_2{}^* = b_2(1+\lambda f_i)$$
$$\textbf{else} \quad r_2{}^* = 0$$

$$\textbf{if} \quad -c_1(1+p_d+m_i) > c_3(1+(1-p_d)) \quad \textbf{then}$$
$$q_1{}^* = -c_1(1+p_d+m_i)$$
$$\textbf{else} \quad q_1{}^* = c_3(1+(1-p_d))$$

$$\textbf{if} \quad -c_1(1+p_d+m_i) > -c_3(1+(1-p_d)) \quad \textbf{then}$$
$$q_2{}^* = -c_1(1+p_d+m_i)$$
$$\textbf{else} \quad q_2{}^* = -c_3(1+(1-p_d))$$

$$\textbf{if} \quad c_2(1+p_f+m_i) > 0 \quad \textbf{then}$$
$$q_3{}^* = c_2(1+p_f+m_i)$$
$$\textbf{else} \quad q_3{}^* = 0$$

5 Architecture of the proposed model

One of the most important parts of this model is the intrusion detection sensors that are both flexible in numbers and capable of training. The number of sensors can be initially determined by the system administrator or by the model itself according to the number of virtual machines. In this simulation, the ratio of one-fourth was used for the number of agents against the number of virtual machines. In the next step, once SNORT detects an invalid action, intrusion sensor starts a noncooperative game with the attacker. Snort is a widely used IDS that can detect attacks by examining network packets and matching them with attack scenarios in its knowledge bank.

One of the key features of the agents is their trainability in the real environment. In this simulation, this ability was used to reduce the agents' reference to the central intrusion detection system. This was used in such a way that the agent reacted to the

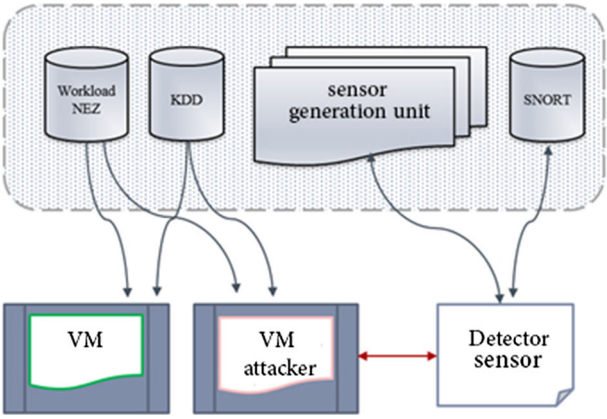

Fig. 5 Architecture of the proposed model

positive responses of intrusion detected by the intrusion detection system as an attack during the communication with the central intrusion detection system; if this attack pattern is repeated for several times, it is kept in the system's memory and in the next cases, the systems initially search in its memory and in case the answer is not found, the attack pattern will be sent to the central intrusion detection system. It is the same behavior that CPU does in fetching the information firstly through the cache and in the absence of that, through RAM. This feature accelerates the detection process and reduces the system overhead. Agents use the data obtained from interaction with the attacker to adopt a strategy in order to reach the point of Nash equilibrium, which will be assumed as the risk level of the attack. After detecting an attacker VM, the sensor sends the information to the sensor generation unit, allowing it to adjust the number of sensors depending on the attack level. There is also an alternative for suspending the attacker virtual machine following a successful detection. A system administrator specified threshold is used to increase the accuracy of intrusion detection system and reduce the rate of false alarms. This threshold is defined in form of the ratio of normal actions to aggressive actions and any VM crossing this threshold will be recognized as an attacker and will be treated accordingly. This parameter prevents a VM with mostly normal behavior from being recognized as an attacker after just a few abnormal behaviors. Figure 5 shows the architecture of the proposed model.

6 Evaluation of the proposed architecture

The proposed method was tested using CloudSim simulator, which was implemented on the Sim java module. CloudSim was created by a team under the administration of Dr. Buyya in the Cloudbus Laboratory at the University of Melbourne, Australia [21]. The experiment was conducted on hardware with the following configurations: 8-core CPU and 8 GB RAM. To create a simulation environment for the distributed attacks, 51,987 virtual machines were created in CloudSim, each of which could have normal

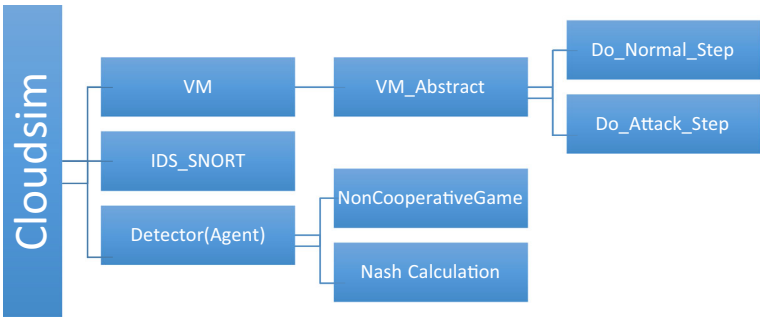

Fig. 6 Classes of attack and detection simulator in CloudSim

or abnormal steps. Attacker virtual machines can work together to take various attack steps separately and thus complete an attack.

By default, CloudSim lacks any plugins for attack and detection simulation; therefore, first a plugin in form of Fig. 6 was coded for CloudSim to simulate the normal and attack behaviors, attack detection, noncooperative game and Nash calculations.

- VM_Abstract: it is a class derived from basic VM class of CloudSim; it creates a virtual machine and adds features like normal and attack behaviors.
- IDS_SNORT: to start a noncooperative game, intrusion sensors should have a general knowledge about the behavior of virtual machine, and this task is carried out by IDS_SNORT class. However, as mentioned earlier, one of the features of the sensors is their trainability, which allows them to learn the most frequent attack patterns and to develop an ability to detect these patterns without the involvement of SNORT. This feature reduces the system overhead and the involvement of central intrusion detection system.
- Detector: this class represents the intrusion sensor and the process of starting a noncooperative game to reach a Nash equilibrium.
- Nash: it determines whether the matrix resulted by noncooperative game has reached a Nash equilibrium.

The virtual machine was simulated by the use of SWF workload. Next, virtual machines needed to exhibit normal or abnormal behaviors, and this task was carried out using KDD CUP 1999 data [23]. These data are provided by Lincoln Laboratory of MIT under the license of DARPA 1998 with the aim of facilitating research on attack detection. These data include one million records, of which 218,011 were used in our simulation.

The results of the simulation are presented and discussed below. The simulation of the performance of the proposed model required a numerical instance. Simulation was carried out by using four number groups shown in Table 5 for the main parameters b_1 and b_2.

One of the most important criteria of intrusion detection system is the rate of successful attack detection. The results pertaining to this important parameter, which are obtained based on the values of Table 5, are shown in Fig. 7.

Table 5 Parameters of agent and attacker

Group	b_1	$-b_2$
1	10	−90
2	70	−40
3	100	−11
4	5	−10

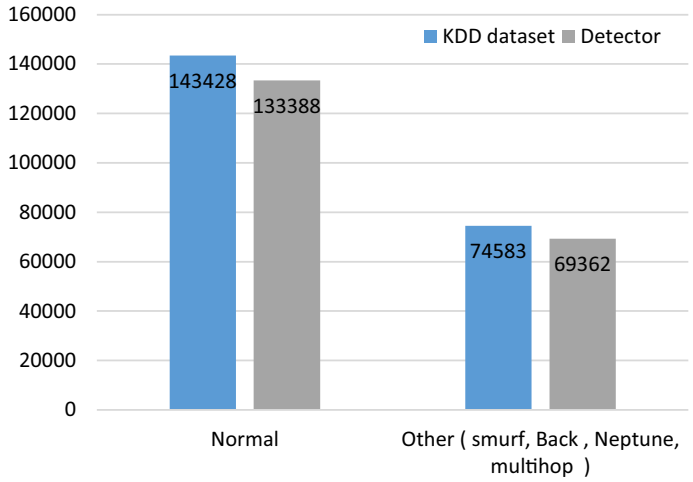

Fig. 7 The rate of successful detection of benchmark records

As can be seen, intrusion sensors have managed to successfully identify 133,388 out of 143,428 normal records and 69,362 out of 74,583 attack records of KDD file. Consequently, it can be said that game theory-based intrusion sensors have correctly identified about 86% of the behaviors.

Another important criterion of intrusion detection system is the rate of misdetection of the attack. The results obtained for this parameter based on values of Table 5 are shown in Fig. 8.

As Fig. 8 demonstrates, intrusion sensors failed to identify 5221 of 74,583 attack records and made a false alarm in 10,040 out of 143,428 normal records of KDD file.

In this paper, a threshold value is used to reduce the number of false alarms. As previously mentioned, this threshold prevents VMs with mostly normal behavior from being recognized as attackers for just a few abnormal behaviors.

Threshold value used in the study was based on trial-and-error method, and the simulator was obtained after performing it for several times. The threshold value was selected in the following way: If more than 20% of the behavior of a virtual machine was detected as an attack by intrusion detection sensor, that virtual machine was regarded as an attacker. Figure 9 shows that applying the threshold reduces the number of false alarms by about 43%.

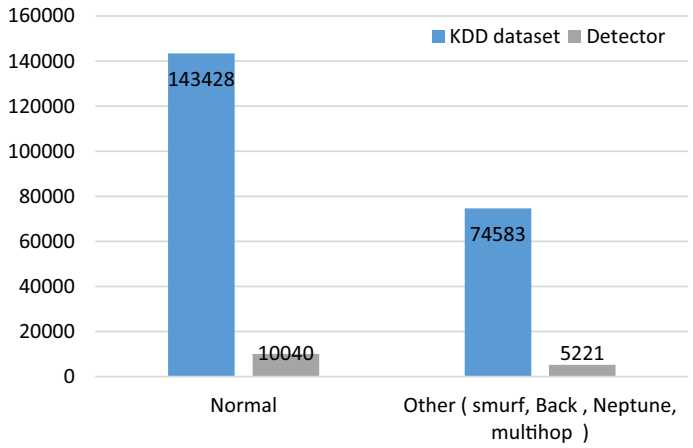

Fig. 8 The rate of misdetection of benchmark records

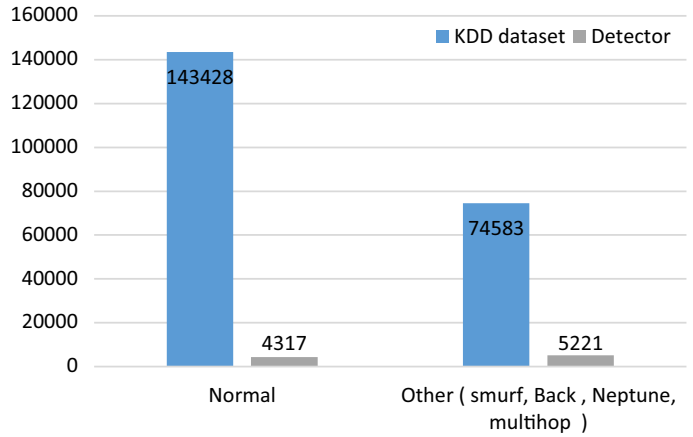

Fig. 9 The rate of misdetection of benchmark records after applying the threshold

7 Comparison and evaluation of the model

Regarding the number of attacks detected by the proposed model, a comparison was made between the obtained data in the study and the same input data of Menzel's [24], in which he defines a general structure for grouping and applying security restrictions in a distributed system. According to this model, certain restrictions are determined that accurately define the required information and security mechanisms based on the identification, authentication, confidentiality and integrity. The ontology performance offered by this model is using a set of rules derived from JAM algorithm, which is obtained by defining the desired model in a cloud environment to detect 78% of the attacks.

In a study by Burney [25], an intrusion detection system is presented using parallel neural networks. In this model, the amount of redundancy was first decreased by

Table 6 Comparing attack detection in intrusion detection models

Model	Attack detection (%)
Model for service-oriented architectures	78
Parallel neural networks	82.5
Genetic algorithms and fuzzy logic	75.18
Genetic algorithms and neural networks	78.6
The proposed model	86

reducing the investigations for the features and then a parallel multilayer neural network was implemented. One of the methods for neural network training is to compare an attack with a set of normal data. These comparisons are made in a parallel fashion in four stages. According to this model, the result of attack detection was 82.5%.

In a research conducted by Kandeeban [26], genetic algorithm and neural networks were used to obtain the quick and efficient model for intrusion detection system. In this model, eight features are used to detect the attacks. A set of rules are also used to detect the attacks. Genetic algorithm is used for training and classifying the rules. These rules which are shown as chromosomes can be combined with the other chromosomes based on the genetic algorithm during the system's lifetime and produce better rules. Using the same input data, this model can detect 78.6% of the attacks.

In Mostaque's study [27], the genetic algorithm is used to detect a variety of attacks. Genetic algorithm is also used in the phases of training and detecting intrusion. Fuzzy logic is used for classifying the rules into two classes of ordinary rules and unnatural rules. Using KDD 99 input data, this model is able to detect 75.18% of the attacks (Table 6).

8 Conclusion

With the development of cloud-based technologies, the attacks on virtual machines and hypervisors of cloud platforms have become a growing concern. The use of virtual machine technology allows several tenants to gain access to shared computing resources of cloud network. Although this resource pooling can significantly reduce the computing costs, it also poses several security vulnerabilities, as VM isolation can be disrupted through side channels. Furthermore, changing the attacks from centralized to distributed from has highlighted the importance of developing security measures to counter distributed attacks and detect abnormal behaviors. In the method discussed in this paper, a group of mobile agents act as the sensors of invalid actions and start a noncooperative game with the suspected attacker, and then seek to calculate the Nash value and maximum utility so as to differentiate an attack from legitimate requests and determine the severity of the attack and its point of origin. The simulation results showed that this method can achieve a good level of detection, and applying a threshold value can greatly reduce the number of false alarms. The trainability feature of sensors allows them to learn the frequent attack patterns and prevents the direct involvement

of SNORT in security management of all requests. This trainability feature accelerates the process of detection and reduces the system overhead.

References

1. Modi C, Patel D, Borisaniya B, Patel A, Rajarajan M (2013) A survey on security issues and solutions at different layers of cloud computing. J Supercomput 63(2):561–592
2. Gritzalis S, Liu L (2013) Requirements engineering for security, privacy and services in cloud environments. Requir Eng 18(4):297
3. Srinivasan MK, Sarukesi K, Rodrigues P, Manoj MS, Revathy P (2012) State-of-the-art cloud computing security taxonomies: a classification of security challenges in the present cloud computing environment. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics. ACM, pp 470–476
4. Zissis D, Lekkas D (2012) Addressing cloud computing security issues. Future Gener Comput Syst 28(3):583–592
5. Fatema K, Emeakaroha VC, Healy PD, Morrison JP, Lynn T (2014) A survey of cloud monitoring tools: taxonomy, capabilities and objectives. J Parallel Distrib Comput 74(10):2918–2933
6. Keromytis AD, Misra V, Rubenstein D (2004) SOS: an architecture for mitigating DDoS attacks. IEEE J Sel Areas Commun 22(1):176–188
7. Wa Wang Z, Lee RB (2008) A novel cache architecture with enhanced performance and security. In: 41st IEEE/ACM International Symposium on Microarchitecture, 2008. MICRO-41. IEEE, pp 83–93
8. Avram MG (2014) Advantages and challenges of adopting cloud computing from an enterprise perspective. Procedia Technol 12:529–534
9. Carroll M, Van Der Merwe A, Kotze P (2011, August) Secure cloud computing: benefits, risks and controls. In: Information Security South Africa (ISSA), 2011. IEEE, pp 1–9
10. Ertaul L, Singhal S, Saldamli G (2010) Security challenges in cloud computing. In: International Conference on Security and Management, Las Vegas, pp 36–42
11. Yang J, Chen Z (2010) Cloud computing research and security issues. In: 2010 International Conference on Computational Intelligence and Software Engineering (CiSE). IEEE, pp 1–3
12. Khalil IM, Khreishah A, Azeem M (2014) Cloud computing security: a survey. Computers 3(1):1–35
13. Gibbons R (1992) A primer in game theory. Harvester Wheatsheaf, Birmingham
14. Von Neumann J, Morgenstern O (2007) Theory of games and economic behavior. Princeton University Press, Princeton
15. Liu F, Ren L, Bai H (2014) Mitigating cross-vm side channel attack on multiple tenants cloud platform. J Comput 9(4):1005–1013
16. Kong J, Aciiçmez O, Seifert JP, Zhou H (2009) Hardware–software integrated approaches to defend against software cache-based side channel attacks. In: IEEE 15th International Symposium on High Performance Computer Architecture, 2009. HPCA 2009. IEEE, pp 393–404
17. Aciicmez O, Kong J, Seifert JP, Zhou H (2008) Deconstructing new cache designs for thwarting software cache-based side channel attacks. In: Proceedings of the 2nd ACM Workshop on Computer Security Architectures. ACM, pp 25–34
18. Lombardi F, Di Pietro R (2011) Secure virtualization for cloud computing. J Netw Comput Appl 34(4):1113–1122
19. Eid M (2004) A new mobile agent-based intrusion detection system using distributed sensors. In: Proceeding of FEASC, pp 114–125
20. Nezarat A (2013) A novel model for detecting intrusion with mobile agent and game theory. In: Fourth International Conference on Information and Communication Technology, Tehran, pp 120–134
21. Maskat K, Shukran MAM, Khairuddin MA, Isa MRM (2011) Mobile agents in intrusion detection system: review and analysis. Mod Appl Sci 5(6):218
22. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50
23. Chandolikar NS, Nandavadekar VD (2012) Selection of relevant feature for intrusion attack classification by analyzing KDD Cup 99. MIT Int J Comput Sci Inf Technol 2(2):85–90
24. Menzel M, Meinel C (2009) A security meta-model for service-oriented architectures. In: IEEE Conference on Services Computing, pp 1–9

25. Burney SMA, Khan MSA, Jilani TA (2010) Feature deduction and ensemble design of parallel neural networks for intrusion detection system. IJCSNS 10(10):259
26. Kandeeban SS, Rajesh RS (2010) Integrated intrusion detection system using soft computing. IJ Netw Secur 10(2):87–92
27. Hassan MMM (2013) Network intrusion detection system using genetic algorithm and fuzzy logic. Int J Innov Res Comput Commun Eng 1(7):1435–1445
28. Nezarat A, Dastghaibyfard G (2016) A game theoretical model for profit maximization resource allocation in cloud environment with budget and deadline constraints. J Supercomput 72:4737. doi:10.1007/s11227-016-1782-z
29. Nezarat A, Dastghaibyfard G (2016) A game theoretic method for resource allocation in scientific cloud. Int J Cloud Appl Comput (IJCAC) 6(1). doi:10.4018/IJCAC.2016010102