CLOSURE: A cloud scientific workflow scheduling algorithm based on attack-defense game model

Yawen Wang[1], Yunfei Guo[1], Zehua Guo[2*], Thar Baker[3], Wenyan Liu[1]

[1]National Digital Switching System Engineering Technology Research Center, Zhengzhou 450002, China

[2]Beijing Institute of Technology, Beijing 100081, China

[3]Liverpool John Moores University, Liverpool L3 3AF, U.K.

[*]Corresponding author: Zehua Guo (guolizihao@hotmail.com)

Abstract: The multi-tenant coexistence service mode makes the cloud-based scientific workflow encounter the risks of being intruded. For this problem, we propose a CLoud scientific wOrkflow SchedUling algoRithm based on attack-defensE game model (CLOSURE). In the algorithm, attacks based on different operating system vulnerabilities are regarded as different "*attack*" strategies; and different operating system distributions in a virtual machine cluster executing the workflows are regarded as different "*defense*" strategies. The information of the attacker and defender is not balanced. In other words, the defender cannot obtain the information about the attacker's strategies, while the attacker can acquire information about the defender's strategies through a network scan. Therefore, we propose to dynamically switch the defense strategies during the workflow execution, which can weaken the network scan effects and transform the workflow security problem into an attack-defense game problem. Then, the probability distribution of the optimal mixed defense strategies can be achieved by calculating the Nash Equilibrium in the attack-defense game model. Based on this probability, diverse VMs are provisioned for workflow execution. Furthermore, a task-VM mapping algorithm based on dynamic Heterogeneous Earliest Finish Time (HEFT) is presented to accelerate the defense strategy switching and improve workflow efficiency. The experiments are conducted on both simulation and actual environment, experimental results demonstrate that compared with other algorithms, the proposed algorithm can reduce the attacker's benefits by around 15.23%, and decrease the time costs of the algorithm by around 7.86%.

Key words: scientific workflow; workflow scheduling; attack-defense game; diverse operating systems; moving target defense

1 Introduction

Cloud computing uses virtualization technology to provide services. With cloud computing technologies, users can obtain the required computing and storage resources through the network. Due to the low costs and simple operation of cloud services, many scientific computing tasks have been moved to cloud platforms [1]-[4].

Scientific computing is a complex process which is composed of many sub-tasks connected by certain dependencies [5]. For example, Swinburne Astrophysics group has used the observation data from Parkes Radio Telescope [6] to conduct a pulsar searching survey [7]. Pulsar searching is a typical scientific computing task, which contains complex and time-consuming sub-tasks such as recording raw data, extracting beam, compressing beam, searching pulsar candidates, making a decision, and so on. These large-scale scientific problems are often modeled as scientific workflows, which can be distributed to multiple computing resources for a faster, and more effectual execution

[8].

The cloud-based scientific workflows face serious security threats. First, multi-tenant service mode makes many malicious tenants exist in clouds, therefore, the clouds are threatened by many kinds of attacks, such as co-residency attacks [9], side-channel attacks [10][11], virtual machine (VM) escape attacks [12]. Second, some security risks come from the characteristics of scientific workflows. Many scientific workflows are computation-intensive [13], which requires a lot of VMs. But, in clouds, VMs are easy to become attack targets [14]. Also, the scientific workflow execution often takes a long time [7], which provides sufficient preparation and intrusion time for attackers. Normally, the scientific workflow is described by a Directed Acyclic Graph (DAG), which is very sensitive to attacks, since any intermediate error will be inherited into the final result [15]. Furthermore, intermediate data of scientific workflows often contains the core secrets in some scientific fields. Once these data are stolen, it will cause huge losses to users [16]. Therefore, enhancing the security of cloud-based scientific workflows is challenging and meaningful work.

In multi-tenant clouds, many attacks are launched based on the operating system (OS) vulnerabilities. Attacks based on different OS vulnerabilities can be regarded as different attack strategies [17]. The scientific workflow execution requires many VMs, which constitute the VM cluster. To avoid the error propagation in the homogeneous computing environment [18], that is, one attack can compromise multiple VMs, it is necessary to build a VM cluster with different OSs [19]. In [17], different OS distribution in a VM cluster is considered as different defense strategies. However, in the attack and defense scenarios for cloud-based scientific workflows, the information of the attacker and defender is not symmetrical. The defender cannot obtain information about the attacker's strategies, while the attacker can acquire information about the defender's strategies through a network scan. Moving Target Defense (MTD) is a novel way to reverse this asymmetric situation between attacks and defenses. It keeps moving the attack surface of the protected system through dynamic shifting, which can disturb the attackers' reconnaissance [20].

Strategy formulation is an important technique in MTD [28]. In the network attack and defense scenario, the features of opposition, dependency and noncooperative are compatible with the game theory, thus making the strategy formulation based on game theory becoming the mainstream method in MTD [28]. In addition, how to switch the defense strategies is also a challenging job. In the cloud-based scientific workflows, the defense strategies refer to the OS distribution of VMs executing the scientific workflow sub-tasks, so switching the defense strategies is the process of VM recycling, which will influence the efficiency of the scientific workflows.

For these problems, we propose CLoud scientific wOrkflow SchedUling algoRithm based on attack-defensE game model (CLOSURE), which uses workflow scheduling as a defense method to improve the security of cloud-based scientific workflows. The scheduling is the core content of the scientific workflow [5][8][29]-[31], which requires not only meeting the users' needs but also improving the efficiency of the whole system [32]. However, there are few researches considering using the workflow scheduling as a defense method. The workflow scheduling contains two important parts: resource provisioning and task—VM mapping [8]. The job of resource provisioning is choosing the optimal configuration for the VM cluster executing the scientific workflow sub-tasks. Many researchers study flexible resource provisioning strategies that can dynamically scale up or down the size of the VM cluster according to the workflow execution requirements [33]-[35], which can increase workflow efficiency and reduce the costs. From the perspective of the security, we present the resource provisioning strategy based on attack-defense game (RPADG), which

generates a probability distribution of a set of VM cluster OS configurations, on the basis of this probability distribution, the clouds will continuously change the OS configuration of the VM cluster. The job of the task—VM mapping is placing the workflow sub-tasks into VMs according to some scheduling objectives. Based on traditional Heterogeneous Earliest Finish Time (HEFT) algorithm [36], we present dynamic HEFT task—VM mapping algorithm (DHEFT) to create opportunities for defense strategy switching, which can speed up the switching of defense strategies.

The main contributions are summarized below:

(1) Workflow scheduling is used for optimizing the performances of scientific workflows; however, inspired by MTD, we propose using workflow scheduling as a dynamic defense method to secure the cloud-based scientific workflows.

(2) We propose RPADG and get the optimal mixed defense strategy by calculating Nash Equilibrium of the model. Furthermore, we propose to dynamically switch the defense strategies by recycling and re-deploying VMs, which can weaken the reconnaissance effects of attackers.

(3) To speed-up the switching of defense strategies and adapt to the VM changes, based on traditional HEFT algorithm, we present DHEFT to map workflow sub-tasks to VMs, which can minimize the negative effects of switching defense strategies to the workflow efficiency.

(4) We conduct the experiments on both simulation environment-based on WorkflowSim [37] and actual environment-based OpenStack [38]. A famous security scanner named Nmap [39] is used for evaluating CLOSURE.

The rest of this paper is structured as follows: Section 2 introduces the related work. Section 3 states the security problem for scientific workflows in clouds. Section 4 presents the CLOSURE and Section 5 presents the experiments, followed by Section 6 that concludes this work.

2 Related work

The researches on workflow scheduling can be categorized into seven classes according to the scheduling objectives [8]: (1) Minimizing the monetary costs of executing workflows in public clouds [30]. These scheduling algorithms consider the costs of renting VM, transferring the intermediate data and storing the intermediate data and try to minimize it. (2) Minimizing workflow makespan [31]. (3) Maximizing the number of workflows executed with the given money or the specified deadline [40]. (4) Improving VM utilization during workflow execution [41]. The idle time slots in the leased VMs are deemed as a waste of money since they are paid for but not used, therefore, these scheduling algorithms try to avoid this problem. (5) Reducing the energy consumed by workflow execution [42]-[43]. Individuals and organizations worldwide have developed an increased concern to protect the environment by reducing carbon footprints [8], which makes some researchers consider the energy consumption when designing their scheduling algorithms. (6) Enhancing the reliability of scientific workflows [44]-[46]. The purpose of these scheduling algorithms is to ensure successful workflow execution even if resource or task failure occurs. (7) enhancing the security of scientific workflows [47]-[49]. Scientific workflows often involve secret data and sensitive computation, so some algorithms are presented to execute scientific workflows in a secure manner.

Clouds use virtualization technologies to maximize the utilization of the computing resources, e.g. many VMs run on a shared physical infrastructure, which causes high security risks. First, the virtualized environment introduces its own set of risks and vulnerabilities. The key module of

virtualization is a virtual machine monitor (VMM), which is responsible for VM management and isolation. There are many reported vulnerabilities that can be used for attacking VMM [50]. Once an attacker compromises the VMM, all the VMs managed by this VMM will be under the control of the attacker [51], and the VM metadata kept by this VMM will also be exposed to this attacker [52]. VM escape is a situation in which a malicious user or VM escapes from the control of VMM [53], which can provide the attacker access to other VMs or bring the VMM down [54]. Second, co-residency with other VMs cause high-security risks, such as side-channel attacks [55]. Cache side-channel attacks can infer secret information of a victim by measuring its cache usage patterns [56]. The researchers in [57][58] have shown that the last-level cache (LLC) attacks are powerful enough to extract fine-grained secret information with high resolution and low noise. Wang et al. [55] present a novel side-channel based on shared physical memory, which exploits the vulnerabilities of the balloon driver. Balloon driver is a very popular mechanism used by current VMMs to balance physical memory among several VMs.

For the cloud-based scientific workflow security problem, Poola et al. [46] propose an adaptive, just-in-time workflow scheduling algorithm, which reduces costs and enhances fault-tolerant capabilities of scientific workflows by dynamically selecting spot and on-demand instances. Yao et al. [45] believe the rescheduling of scientific workflows is similar to the biological immune system, so an immune system inspired failure-aware rescheduling algorithm is suggested to improve the reliability of scientific workflows. Ding el al. [44] present a primary-backup workflow scheduling algorithm to achieve fault-tolerant elastic workflow scheduling. In [34], a novel fault-tolerant workflow scheduling algorithm is presented, which combines the resubmission and replication strategy together to play their respective advantages. To secure the data of scientific workflows, in [47], the workflow datasets are divided into moveable and immovable datasets based on the security requirements. Movable datasets have no security restrictions and can be moved between data centers. While immovable datasets are restricted to a single data center and cannot be moved. In [48], the authors build a security overhead model to reasonably measure the security overheads incurred by the sensitive data, and propose a data placement strategy to dynamically deploy the intermediate data for the scientific workflows. Chen et al. [49] develop a task-scheduling framework for security sensitive workflows, in which the intermediate data is encrypted by effectively exploiting tasks' laxity time. Li et al. [13] consider the heterogeneity of workflow sub-tasks, such as data-intensive sub-tasks, memory-intensive sub-tasks, and computation-intensive sub-tasks, and propose security and cost aware scheduling algorithm for heterogeneous sub-tasks of scientific workflow in clouds. Nepal et al. [16] design TruXy system, which provides trusted storage services for scientific workflows. Sujana et al. [59] present SPSO to find an optimized schedule that tradeoff between security and minimum makespan and cost for workflow execution in clouds. Wen et al. [60] propose a novel algorithm to deploy workflow applications on federated clouds, in which an extension of the Bell-LaPadula Multi-Level security model is applied to meet application security requirements.

Although the above works have considered the scientific workflow security, these works generally use traditional protection methods such as encryption or hash check to improve security. Inspired by MTD, we achieve a dynamic defense method by workflow scheduling to secure the scientific workflows in clouds.

Research on the MTD strategy formulation can be categorized into three classes: strategy formulation based on game theory [21]-[23], strategy formulation based on machine learning control theory [24][25] and strategy formulation based on evolution theory [26][27]. In the network attack

and defense scenario, the features of opposition, dependency and noncooperative are compatible with the game theory, thus making the strategy formulation based on game theory becoming the mainstream method [28]. Manadhata et al. [61] construct a two-player stochastic dynamic game model and analyze the impact of attack surface transformation on both offensive and defensive behaviors. Zhu et al. [62] establish a game-theoretic model to provide a formal analysis of security strategies and guide the design of MTD. Then they develop a feedback system framework for strategically shifting attack surface based on observation. Carter et al. [63] research dynamic platform defense by using a game-theoretical approach modeling an attacker who can monitor the defender's moves. Feng et al. [64] combine Stackelberg game model with Markov decision process and design an iterative algorithm to select optimal strategy under worst-case by abstracting strategy selection into the minimum-maximum problem. Ahmed et al. [65] formulate an APT defense game using evolutionary game theory to research the dynamic behavior of the APT attacker and defender with replicator dynamics.

3 Problem statement

3.1 Cloud-based scientific workflow system

A workflow is often modeled as a Directed Acyclic Graph (DAG), which can be represented as $G = (V, E)$, where V denotes the set of sub-tasks making up the workflow. $E = \{(a_i, a_k) | a_i, a_k \in V, i \neq k\}$ with no cycles $a_i \rightsquigarrow a_k \rightsquigarrow a_i$ is the set of arcs between sub-tasks in V. Each arch $(a_i, a_k) \in E$ represents the dependence constraint between sub-tasks a_i and a_k, such that a_i must be completed prior to the initiation of the execution of a_k [66][67].

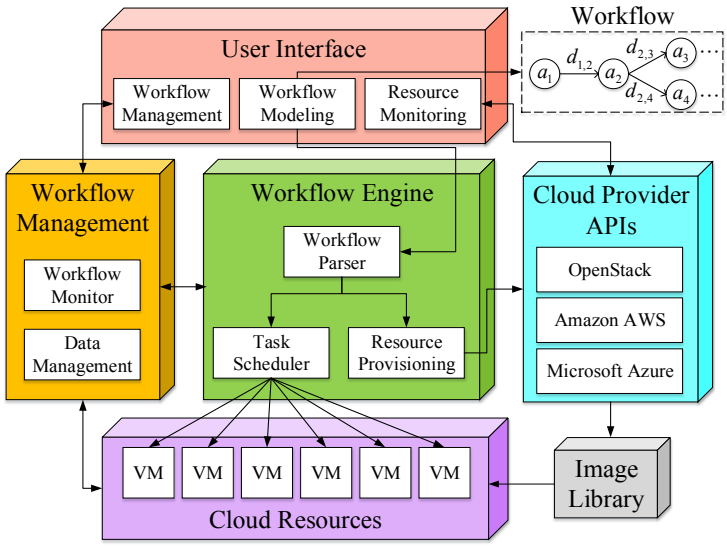


Figure 1 The structure of a typical cloud-based scientific workflow system [8]

A typical cloud-based scientific workflow system mainly comprises 5 modules: workflow engine, user interface, workflow management module, cloud provider APIs and cloud resources [8], as shown in Figure 1. The users model their workflows in the user interface and submit the defined workflow to the workflow engine. The workflow engine is the core of the cloud-based scientific workflow system, which contains three main functions, workflow parsing, workflow scheduling and resource provisioning. The workflow parser can analyze the required resources for the

submitted workflow; according to the resource requirement, resource provisioning module will use cloud provider APIs to produce VMs. Furthermore, the workflow parser also can translate each abstract workflow into an internal executable workflow representation [68]; then, the task scheduler will dispatch the workflow sub-tasks to VMs to execute based on loaded scheduling strategies. Different scheduling strategies can be selected based on different requirements of the users, such as minimizing the monetary costs of executing workflows, minimizing workflow makespan, and so on. During the workflow execution, the workflow management module will monitor the workflow execution status and manage the generated intermediate data; then, reflects this information to the user interface.

3.2 The threats of the scientific workflows in clouds

Many scientific workflows belong to important scientific computing tasks, such as high-energy physics, bioinformatics, atmospheric science, and so on [69]. Therefore, workflow intermediate data involves core secrets in some areas, it will cause huge losses if the data information is stolen [70]. However, due to the multi-tenant coexistence, data leakage is one of the serious threats in clouds [11].

For cloud-based scientific workflow systems, the workflow is executed by VMs. After completing each workflow sub-task, the intermediate data will be generated in the VMs, which is used for the execution of successor sub-tasks. Since these intermediate data are only temporarily stored in the VM, they are usually not encrypted. As long as the attacker enters these VMs, these intermediate data can be stolen easily.

In executing a scientific workflow, VMs need to transfer data frequently, so these VMs are usually placed in the same tenant network. Different tenant networks are separated by tunneling protocol, if the tunneling protocol fails, all tenants' traffic is visible [71]. Therefore, attackers can penetrate other tenants' networks through damaging the tunneling protocol.

If an attacker has succeeded in penetrating a tenant network in which many VMs are executing sub-tasks of scientific workflows, he can use a scanning tool to gain information about the OSs of these VMs. Then, the attacker can develop an attack strategy with the greatest attack benefits (penetrating as many VMs as possible). For example, the attacker uses Nmap to scan the entire tenant network and find 8 Windows VMs and 2 Ubuntu VMs. After that, the attacker can use the vulnerabilities of the Windows OS to occupy 8 VMs and steal the workflow data.

4 Cloud scientific workflow scheduling algorithm based on attack-defense game model

4.1 Overview of CLOSURE

A network attacker is an intelligent and rational individual, which pursues the maximization of attack benefits while considering the costs of attacks [72]. In the attack scenario for scientific workflows described in Section 3.2, to steal as much workflow intermediate data as possible, the attacker will try to use few attacks to invade as many VMs as possible (the number of attacks is regarded as the attack costs in this paper). To defend these intelligent and rational attackers, we propose CLOSURE, which contains three main contents: diverse VMs, resource provision algorithm based on attack-defense game model (RPADG) and task—VM mapping algorithm based

on DHEFT, the relationships of these three contents are shown in Figure 2, and the basic ideas are:

(1) Generating diverse VMs with different OS images to increase attack costs. If all VMs executing the scientific workflow are Ubuntu based, the attacker only needs one attack based on Ubuntu OS vulnerabilities to invade all VMs (it is assumed that the attacker can attack multiple VMs at a time). However, if the VMs use different OSs, it is difficult for the attacker to invade all VMs by only one attack. We introduce this content in Section 4.2.

(2) We regard the different OS distribution in the VM cluster as different defense strategies, and build the attack-defense game model for the scientific workflows. Then, we can get the probability distribution of the optimal mixed defense strategies by solving the model. Based on the mixed defense strategies, we propose RPADG to switch the defense strategy by recycling and re-deploying VMs, which can make it difficult for attackers to obtain defense information through the reconnaissance. Because the defense strategy obtained by the reconnaissance at the previous moment will change at the next moment. We introduce this content in Section 4.3.

(3) In RPADG, the defense strategy switching is achieved by recycling idle VMs. However, when the number of available VMs is small, the frequency of the idle state of VMs is very low, which will greatly reduce the switching period of the defense strategy. So, based on traditional HEFT (heterogeneous earliest finish time) algorithm, we propose DHEFT, which can corporate with the RPADG to speed up the switching period of defense strategies, and dynamically adjust the scheduling strategy according to the VM changes, reducing the adverse effect of VM changes on workflow efficiency.

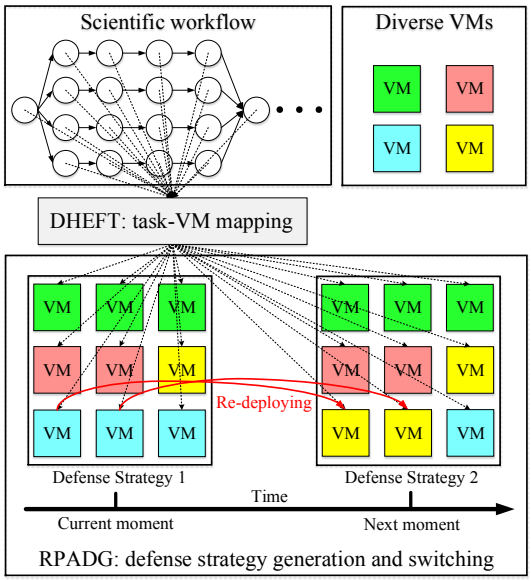


Figure 2 Overview of the defense scheme in CLOSURE

4.2 Diverse VMs

4.2.1 Workflow software deployment method

One of the difficulties in executing scientific workflows in diverse VMs is the workflow software deployment since users need to develop multiple versions of workflow software to be compatible with different OSs. However, the emerging container technology in recent years has solved this problem. Hence, we propose a workflow software deployment method based on

container technology, the principle is shown in Figure 3. There is a workflow consisting of 4 sub-tasks, a_1, a_2, a_3, a_4. Users, firstly, need to deploy the software corresponding to each workflow sub-task in the containers. Then, these containers will be placed in the diverse VMs. The mapping between the container port and the VM port should be established. During the workflow execution, the software will be activated by transmitting the data to the corresponding port. In Figure 3, if the data is transferred to port 1, the a_1 software will be activated to execute sub-task a_1. VM snapshot will be created and uploaded to the image library.

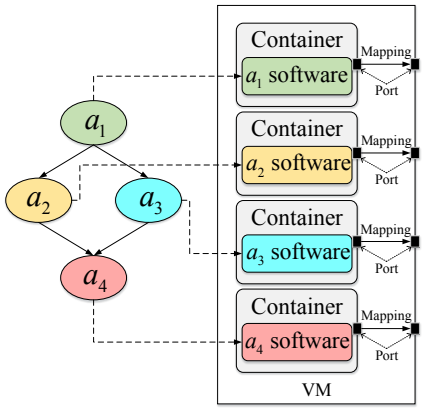


Figure 3 Workflow software deployment method based on container technology

4.2.2 The vulnerability information about diverse OSs

In [19], the authors summarize the vulnerability information about 11 OSs (OpenBSD, NetBSD, FreeBSD, Windows Server 2000, Windows Server 2003, Windows Server 2008, Ubuntu, Debian, Red Hat, Solaris and OpenSolaris) and the common vulnerability information between them, which are shown in Table 1 and Table 2, respectively. These vulnerabilities can be divided into four categories: the vulnerabilities of kernel, system software, driver and application, as shown in Figure 4. Driver vulnerabilities are distributed in the network cards, audio cards, and play devices. These vulnerabilities do not affect the execution of scientific workflows, so, we define the threat level of this kind of vulnerabilities as Level 1. Application vulnerabilities are distributed in databases, text editors, FTP clients and servers, and so on. We define the threat level of application vulnerabilities as Level 2. Kernel vulnerabilities are distributed in TCP/IP stack, file system, task management and core libraries, which can affect the workflow sub-task execution, intermediate data transmission, so we define the threat level of kernel vulnerabilities as Level 3. System software vulnerabilities are distributed in software providing common OS functionalities such as login, shells and basic daemons. In this paper, the main threat we face is the attackers' illegal login to the VM, so we define the threat level of vulnerabilities as Level 4.

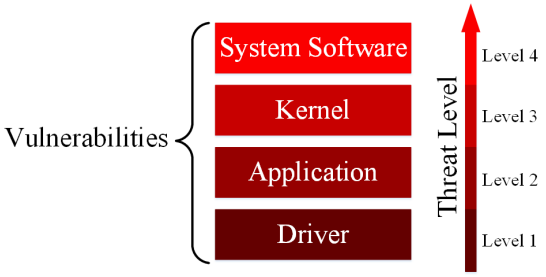


Figure 4 Vulnerability classification and threat level

Table 1 Vulnerabilities per OS from 1994 to 2011 [19]

OS	Sys. Soft.	Kernel	Application	Driver
OpenBSD	37	76	38	2
NetBSD	39	64	31	9
FreeBSD	61	153	61	4
OpenSolaris	9	15	7	0
Solaris	120	155	149	2
Debian	39	25	148	1
Ubuntu	8	22	58	2
Red Hat	108	94	237	5
Win2000	135	146	211	3
Win2003	96	171	291	2
Win2008	36	123	175	0

Table 2 Vulnerabilities between OSs from 1994 to 2011 [19]

A pair of OSs	Sys. Soft.	Kernel	Application	Driver
OpenBSD-NetBSD	8	8	11	1
OpenBSD-FreeBSD	18	14	8	1
OpenBSD-OpenSolaris	0	0	0	0
OpenBSD-Solaris	1	5	3	0
OpenBSD-Debian	0	0	0	0
OpenBSD-Ubuntu	0	0	2	0
OpenBSD-Red Hat	3	1	5	0
OpenBSD-Win2000	0	3	0	0
OpenBSD-Win2003	0	2	0	0
OpenBSD-Win2008	0	1	0	0
NetBSD-FreeBSD	10	13	13	2
NetBSD-OpenSolaris	0	0	0	0
NetBSD-Solaris	6	4	4	0
NetBSD-Debian	4	0	1	0
NetBSD-Ubuntu	0	0	0	0
NetBSD-Red Hat	6	0	3	0
NetBSD-Win2000	0	2	0	1
NetBSD-Win2003	0	1	0	0
NetBSD-Win2008	0	1	0	0
FreeBSD-OpenSolaris	0	0	0	0
FreeBSD-Solaris	3	5	8	0
FreeBSD-Debian	1	0	3	0
FreeBSD-Ubuntu	0	0	0	0
FreeBSD-Red Hat	4	1	7	0
FreeBSD-Win2000	0	3	0	1
FreeBSD-Win2003	0	2	0	0
FreeBSD-Win2008	0	1	0	0
OpenSolaris-Solaris	3	3	5	0
OpenSolaris-Debian	0	0	0	0
OpenSolaris-Ubuntu	0	0	0	0
OpenSolaris-Red Hat	0	0	0	0
OpenSolaris-Win2000	0	0	0	0
OpenSolaris-Win2003	0	0	0	0
OpenSolaris-Win2008	0	0	0	0
Solaris-Debian	1	1	0	0
Solaris-Ubuntu	0	0	0	0
Solaris-Red Hat	3	3	7	0
Solaris-Win2000	0	3	7	0
Solaris-Win2003	0	1	7	0
Solaris-Win2008	0	0	1	0
Debian-Ubuntu	2	0	9	0
Debian-Red Hat	8	5	38	0
Debian-Win2000	1	0	0	0
Debian-Win2003	1	0	0	0
Debian-Win2008	0	0	0	0
Ubuntu-Red Hat	1	0	20	0
Ubuntu-Win2000	1	0	0	0
Ubuntu-Win2003	1	0	0	0

Ubuntu-Win2008	0	0	0	0
Red Hat-Win2000	1	0	1	0
Red Hat-Win2003	1	0	1	0
Red Hat-Win2008	0	0	0	0
Win2000-Win2003	43	42	145	0
Win2000-Win2008	8	8	51	0
Win2003-Win2008	22	18	157	0

4.3 Resource provisioning based on attack-defense game model

In the attack-defense environment for cloud-based scientific workflows, the relationship between the attacker and defender is non-cooperative. In the process of attack and defense, the attacker and defender will not inform the opponent of the decision information in advance. The purpose of the attackers is trying to intrude as many VMs as possible to get the most benefits, while defenders try to minimize the affected VMs, which can be modeled as the attack-defense game model for scientific workflows (ADGSW).

4.3.1 Related definition of ADGSW

The definition of ADGSW: In this paper, we only consider that there is only one attacker and one defender in the game. Therefore, ADGSW can be represented as $G = \big((N_a, N_d), (S_a, S_d), (U_a, U_d)\big)$, where N_a represents the attacker and N_d represents the defender (scientific workflow users). $S_a = (s_1^a, s_2^a, \cdots, s_m^a)$ denotes the attack strategy set of the attacker and $S_d = \big(s_1^d, s_2^d, \cdots, s_n^d\big)$ denotes the defense strategy set of the defender. U_a and U_d are utility functions of the attacker and defender, $U_a\big(s_i^a, s_j^d\big)$ and $U_d\big(s_i^a, s_j^d\big)$ respectively represent the utility of the attacker and defender when the attacker adopts the attack strategy s_i^a and the defender adopts the defend strategy s_j^d. In ADGSW, the defender's loss is the attacker's gain, so $U_a\big(s_i^a, s_j^d\big) + U_d\big(s_i^a, s_j^d\big) = 0$, that is, ADGSW is a zero-sum game. The most commonly used solution concept in game theory is that of Nash Equilibrium.

The definition of pure strategy Nash Equilibrium: Nash Equilibrium captures a steady state of the play of a strategic game in which each player holds the correct expectation about the other players' behavior and acts rationally [73]. In ADGSW, the pair of attack-defense strategy s_*^a and s_*^d is a Nash Equilibrium if it meets the requirement that for $\forall s_{any}^a \in S_a$, $U_a(s_*^a, s_*^d) \geq U_a\big(s_{any}^a, s_*^d\big)$, and for $\forall s_{any}^d \in S_d$, $U_d(s_*^a, s_*^d) \geq U_d\big(s_*^a, s_{any}^d\big)$, which means that s_*^a, s_*^d is the optimal strategy that can maximize the utility of both the attacker and defender. But there may not be a pure strategy Nash Equilibrium in ADGSW [73]. At this time, both the attacker and defender must consider the mixed attack-defense strategies, since every finite strategic game has a mixed strategy Nash Equilibrium [73].

The definition of mixed attack-defense strategies: In ADGSW, the probability distribution of $S_a = (s_1^a, s_2^a, \cdots, s_m^a)$ and $S_d = \big(s_1^d, s_2^d, \cdots, s_n^d\big)$ are respectively defined as $P_a = (p_1^a, p_2^a, \cdots, p_m^a)$ and $P_d = \big(p_1^d, p_2^d, \cdots, p_n^d\big)$, $0 \leq p_i^a \leq 1$, $0 \leq p_i^d \leq 1$, $\sum_{i=1}^m p_i^a = 1$, $\sum_{i=1}^n p_i^d = 1$.

The definition of mixed strategy Nash Equilibrium: In ADGSW, the probability distribution of the attacker and defender's mixed strategies is $P_a = (p_1^a, p_2^a, \cdots, p_m^a)$ and $P_d = \big(p_1^d, p_2^d, \cdots, p_n^d\big)$, respectively, so, the utility expectation of the attacker and defender can be calculated by (1) and (2),

$$E_a(P_a, P_d) = \sum_{i=1}^m p_i^a \left[\sum_{j=1}^n p_j^d U_a\big(s_i^a, s_j^d\big)\right] = \sum_{i=1}^m \sum_{j=1}^n p_i^a p_j^d U_a\big(s_i^a, s_j^d\big) \quad (1)$$

$$E_d(P_a, P_d) = \sum_{j=1}^{n} p_j^d \left[\sum_{i=1}^{m} p_i^a U_d(s_i^a, s_j^d)\right] = \sum_{j=1}^{n}\sum_{i=1}^{m} p_i^a p_j^d U_d(s_i^a, s_j^d) \quad (2)$$

The condition that the mixed strategies having the probability distribution (P_a^*, P_d^*) is Nash Equilibrium is that for $\forall P_a,\ E_a(P_a^*, P_d^*) \geq E_a(P_a, P_d^*)$, and for $\forall P_d,\ E_d(P_a^*, P_d^*) \geq E_d(P_a^*, P_d)$.

4.3.2 The attack and defense strategies

Firstly, we use $o_i, i = 1,2,\cdots,11$ to respectively represent the OS of OpenBSD, NetBSD, FreeBSD, Windows Server 2000, Windows Server 2003, Windows Server 2008, Ubuntu, Debian, Red Hat, Solaris and OpenSolaris.

In ADGSW, we define a defense strategy as $s_k^d = (nv_1, nv_2, \cdots, nv_{11})$, nv_i denotes the number of VMs with the OS of o_i. For example, $s_k^d = (10,0,0,0,0,0,0,0,10,0,0)$ represents the defense strategy having 10 OpenBSD VMs and 10 Red Hat VMs.

Then, we define an attack strategy as $s_k^a = o_j$, which represents to use a vulnerability from o_j as the attack strategy. The attack strategy set is defined as $S_a = (s_1^a, s_2^a, \cdots, s_{11}^a) = (o_1, o_2, \cdots, o_{11})$.

4.3.3 The quantification of utility functions

In section 4.3.1, we have discussed that ADGSW is a zero-sum game, so as long as the attacker's utility function U_a is known, the defender's utility function U_d can be obtained easily.

In this section, we propose to use the common vulnerabilities between OSs to quantify the utility functions. We use $vs_i, vk_i, va_i, vd_i, i = 1,2,\cdots,11$ to respectively represent the number of system software, kernel, application and driver vulnerabilities of o_i. Using $vs_{i,j}, vk_{i,j}, va_{i,j}, vd_{i,j}, i, j = 1,2,\cdots,11$ to represent the number of system software, kernel, application and driver common vulnerabilities between o_i and o_j, and $vs_{i,i} = vs_i, vk_{i,i} = vk_i, va_{i,i} = va_i, vd_{i,i} = vd_i, i = 1,2,\cdots,11$. Furthermore, we use ws, wk, wa, wd to respectively represent the weight of the system software, kernel, application and driver vulnerabilities, in this paper, $ws = 4, wk = 3, wa = 2, wd = 1$. If the attacker adopts the $o_k, k \in N, 1 \leq k \leq 11$ as the attack strategy, the U_a is defined as (3),

$$U_a = \sum_{i=1}^{11} \left[nv_i \cdot \left(ws \cdot vs_{i,k} + wk \cdot vk_{i,k} + wa \cdot va_{i,k} + wd \cdot vd_{i,k}\right)\right] \quad (3)$$

nv_i denotes the number of VMs with the OS of o_i. (3) denotes that the attack is more effective if the target has more vulnerabilities with high threat level. Because if a VM has many vulnerabilities with high threat level, the security of the VM is low, so the probability of successfully intruding the VM will be high. Furthermore, (3) also considers the common vulnerabilities between OSs. For example, if the attacker uses the vulnerability from o_1 to attack the VM whose OS is o_2, the success probability is related to the number and type of common vulnerabilities between o_1 and o_2.

4.3.4 The optimal mixed defense strategy

In ADGSW, there are many defense strategies, we cannot consider all the defense strategies. Therefore, a certain number of defense strategies should be randomly generated firstly. The number of generated defense strategies we set is 100.

Equations (1) and (2) are used for explaining the definition of the mixed strategy Nash Equilibrium. When actually solving the Nash Equilibrium, we do not use (1) and (2), but use *minimax* method [74]. In ADGSW, the pair of attack-defense strategy s_*^a and s_*^d is a mixed strategy Nash Equilibrium if and only if every player's every pure strategy supports that s_*^a and

s_*^d are the best strategies [73]. Therefore, the defender just needs to consider the situation where the attacker adopts pure strategies. In the special case of two-player zero-sum games, computing an optimal mixed strategy is equivalent to computing a *minimax* strategy, which minimizes the maximum expected utility that the opponent can obtain [74]. Therefore, the object of the defender is selecting the mixed attack strategy $P_d = (p_1^d, p_2^d, \cdots, p_{nd}^d)$ to minimize *od*,

$$od = \max_{1 \le i \le 11} \sum_{j=1}^{nd} p_j^d \cdot U_a(s_i^a, s_j^d) \tag{4}$$

The problem of selecting the mixed attack strategy can be transformed into linear programming problem:

$$\begin{gathered} \text{Minimize } e \\ \text{s.t. } e \ge \sum_{j=1}^{nd} p_j^d \cdot U_a(s_i^a, s_j^d), i = 1,2,\cdots,11 \\ \sum_{j=1}^{nd} p_j^d = 1 \\ 0 \le p_j^d \le 1, j = 1,2,\cdots,nd \end{gathered} \tag{5}$$

4.3.5 Resource provisioning strategy

Based on ADGSW, we can get the probability distribution of the optimal mixed defense strategies, then, RPADG uses the probability to select a series of defense strategies. To prevent the attacker from obtaining the information about defense strategies, RPADG switches the defense strategies by recycling and re-deploying a part of VMs. To preserve the data in the VMs which are going to be recycled, block storage service such as OpenStack Cinder is needed. The newly deployed VMs can get the data as long as the volume containing the data is mounted. But, the process of recycling VMs will cause a decrease in the number of available VMs in a short period, which can delay the workflow makespan, especially when multiple VMs are recycled at the same time. In [77], the authors conclude that deploying an Amazon EC2 instance requires around 50 seconds. For this problem, RPADG recycles only one VM at one time, the pseudo-code of RPADG is shown in Algorithm 1.

Algorithm 1 RPADG

```
1   sd[nd] = randomly generating nd defense strategies; //sd represents the defense strategies
2   pd[nd] = calculating the probability of mixed defense strategies by (5); //pd represents the
    probability of mixed defense strategies
3   s = selecting a defense strategy according to pd[nd]; //s denotes the selected defense
    strategy
4   d = deploying VMs according to the defense strategy s; //d denotes the actual distribution
    of VMs
5   while (the workflow does not end)
6       s = selecting a defense strategy according to pd[nd];
7       while (d does not match with s)
8           if (the VM needing to be recycled is idle && no VM is being recycled)
9               d = recycling the VM and re-deploying new VMs with specific OS;
10          end
11      end
12  end
```

First, RPADG need to randomly generate *nd* defense strategies (step 1), then solve the linear programming problem (5) to obtain the mixed defense strategies (step 2). After that, RPADG selects a defense strategy based on the probability distribution obtained by step 2 (step 3). Afterwards, RPADG deploys the VMs according to the selected defense strategy, and these VMs will execute the workflow. During the workflow execution, RPADG will select new defense strategies and transform the current VM distribution into the new defense strategy. The process of this transformation is realized by recycling the idle VMs and re-deploying new VMs with specific OS (step 9). For example, it is assumed that the new defense strategy is the distribution of 2 Ubuntu VMs and 2 FreeBSD VMs and the current VM distribution is 2 Ubuntu VMs, 1 Window Server 2008 VM and 1 FreeBSD VM, when the Windows Server 2008 VM is idle this VM will be recycled and the recycled resource will be used for deploying a new FreeBSD VM.

4.4 Task—VM mapping based on DHEFT

Besides the resource provisioning strategy, the task—VM mapping is also essential to the efficiency and security of the workflows. HEFT is a traditional task—VM mapping algorithm, which has been extensively used in scientific workflow scheduling in clouds [79]. However, traditional HEFT cannot be applied to CLOSURE, since traditional HEFT is a static scheduling algorithm, which calculates the scheduling strategy before workflow execution, the calculated scheduling strategy does not change during the workflow execution. RPADG will randomly recycle some VMs during the workflow execution, so the task—VM mapping algorithm is necessary to be self-adjusting in time. Therefore, we propose DHEFT, it is a dynamic HEFT algorithm, which will adjust the task scheduling strategy when resources changing. The principle of DHEFT is shown in Algorithm 2.

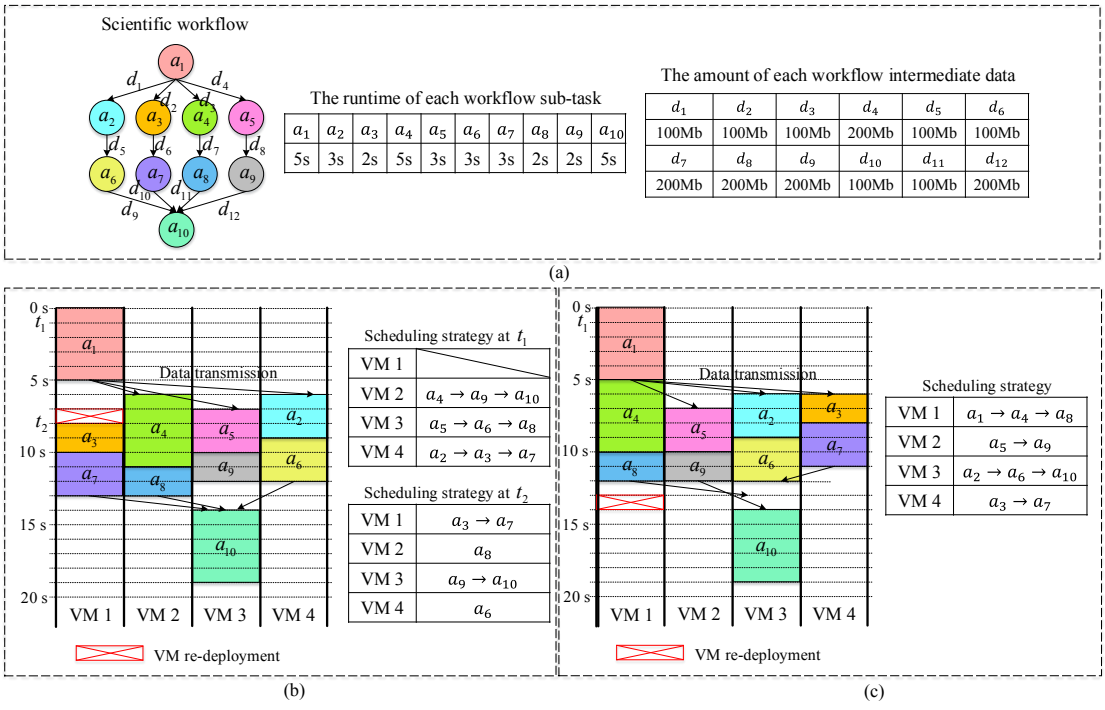


Figure 5 (a) A workflow used for illustrating DHEFT (b) The illustration of the task scheduling process of DHEFT (c) The illustration of the task scheduling process of traditional HEFT

We use a workflow that is shown in Figure 5 (a) to illustrate DHEFT. It is assumed that the bandwidth between VMs is 100 Mbps, the number of available VMs is 4, the time of re-deploying a VM is one second. According to HEFT, the workflow sub-task scheduling order is (6)

$$a_1 \rightarrow a_4 \rightarrow a_5 \rightarrow a_2 \rightarrow a_3 \rightarrow a_6 \rightarrow a_7 \rightarrow a_9 \rightarrow a_8 \rightarrow a_{10} \quad (6)$$

At t_1, the system receives the command of switching the defense strategy, and VM1 needs to be recycled. In order to make VM1 idle, DHEFT will calculate the scheduling strategy without considering VM1. The scheduling strategy calculated at t_1 is that a_4, a_9 and a_{10} are assigned to VM2, a_5, a_6 and a_8 are assigned to VM3, a_2, a_3 and a_7 are assigned to VM4. At 5 seconds, a_1 is finished and VM1 will transmit the intermediate data to other VMs to execute successor sub-tasks. After data transmission, VM1 is idle and will be recycled by RPADG. At t_2, the VM re-deployment is finished and DHEFT will re-calculate a new scheduling strategy. The process of task scheduling of DHEFT is shown in Figure 5 (b), and it takes 7 seconds from receiving the defense strategy switching command to competing the strategy switching. However, if HEFT is used for task scheduling, it will take 13 seconds to finish the strategy switching, as shown in Figure 5 (c). Therefore, DHEFT can speed up the switching period of defense strategies.

Algorithm 2 DHEFT

```
    Input: na (the number of available VMs), vm[na] (VM ID), nc (the number of VMs that
    need to be converted), cvm[nc] (ID of VMs that need to be converted)
1   while (the workflow does not end)
2       if (receiving the defense strategy switching command || a new VM has been deployed)
3           order[] = ordering the unscheduled sub-tasks by HEFT;
4           for i = 1:size(order[])
5               for j = 1:na
6                   if (nc >= 1)
7                       if (vm[j] == cvm[nc-1])
8                           continue; //Do not schedule sub-tasks on VMs that need to be
    converted
9                       end
10                  end
11                  if (order[i] has the earliest start time on vm[j])
12                      order[i] will be assigned to vm[j];
13                      nc = nc-1;
14                  end
15              end
16          end
17      end
18  end
```

5 Experiments

5.1 Experimental setup

The experiments are conducted based on simulation environment based on WorkflowSim and

actual environment based on OpenStack.

5.1.1 Simulation experiment environment

WorkflowSim is an open source software devised for workflow scheduling simulation in clouds. In WorkflowSim, the workflows are described by a XML file, which includes the information about the number of sub-tasks, the dependent relationship between sub-tasks, the number of intermediate data, the size of each intermediate data and each sub-task runtime.

The workflows used for simulation are CyberShake, Epigenomics, Montage, and Inspiral, which are published by Pegasus project [75]. The structures of these workflows are shown in Figure 6, and the parameters are shown in Table 3.

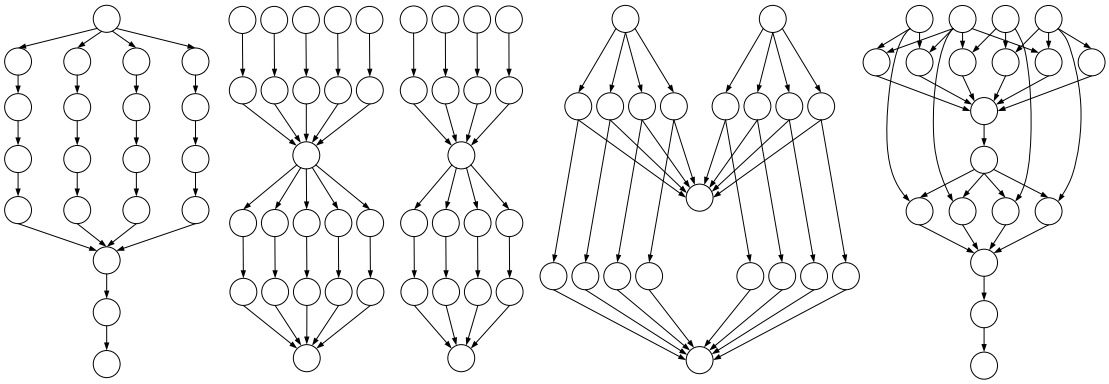


Figure 6 The structures of workflow Epigenomics, Inspiral, CyberShake and Montage

Table 3 The parameters of workflows used for tests

Workflow	Number of sub-tasks	Number of intermediate data	Average data size	Average sub-task runtime
Epigenomics (997)	997	1234	388.59 MB	3858.67 s
Inspiral (1000)	1000	1233	8.90 MB	227.25 s
CyberShake (1000)	1000	1988	102.29 MB	22.71 s
Montage (1000)	1000	2485	3.21 MB	11.36 s

5.1.2 Actual experiment environment

To evaluate CLOSURE in an actual environment, we use two physical servers (24 core processor, 32 G memory, 2 T storage space) to build a small scientific workflow system based on OpenStack, as shown in Figure 7. The system contains one control node and one computing node. In the control node, OpenStack Keystone (authentication module), OpenStack Glance (image module), OpenStack Neutron (network module), OpenStack Cinder (block storage module) and OpenStack Nova (computing module). Besides these OpenStack necessary components, we build workflow parser, workflow scheduler, and resource provisioning module to implement CLOSURE. An actual workflow, object detection in images, is used for evaluating CLOSURE, the structure of the workflow is illustrated in Figure 8. In the actual test, we describe the actual workflow by a XML file. The workflow parser reads the XML file and activates resource provisioning module to deploy VMs. Resource provisioning module executes RPADG through Nova API and image library, which can dynamically switch the defense strategies. We assume that there are 3 existing VMs to execute

the workflow, and we build a VM to simulate an attacker, which is not controlled by the resource provisioning module. The workflow scheduler uses DHEFT to dispatch the workflow sub-tasks to the VMs.

In Section 4.2.1, we discussed that the workflow software deployment requires the container technology, but some OSs listed in Table 1 do not support container technology, such as OpenSolaris, Windows Server 2000, 2003, 2008, and so on. Therefore, we construct the image library contains the VM images of Windows 10, Windows 7, Solaris 10, FreeBSD 9.0, Debian 9.5, Ubuntu 16.04 and CentOS 7.3, and the OSs of Windows 10, Windows 7, Solaris 10, FreeBSD 9.0, Debian 9.5, Ubuntu 16.04 and CentOS 7.3 respectively use the vulnerability data of Win2008, Win2003, Solaris, FreeBSD, Debian, Ubuntu and Red Hat in Table 1 and Table 2.

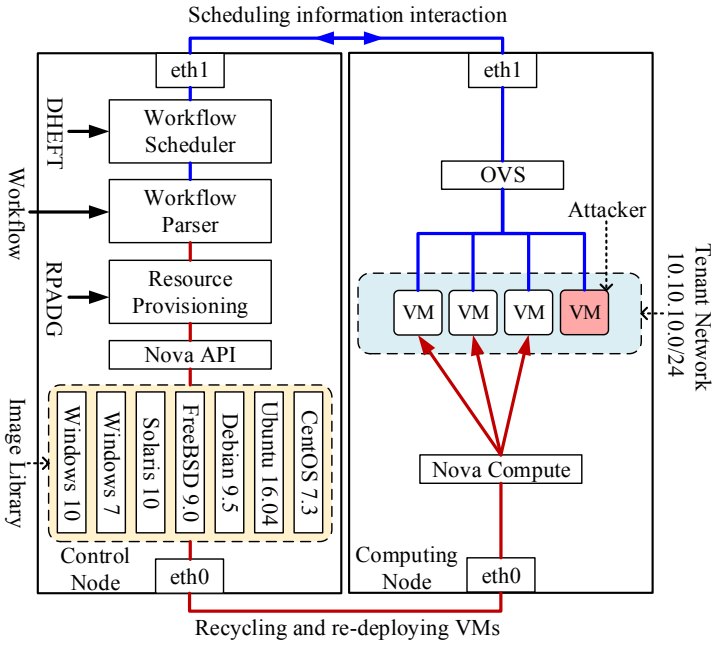


Figure 7 A small scientific workflow system based on OpenStack

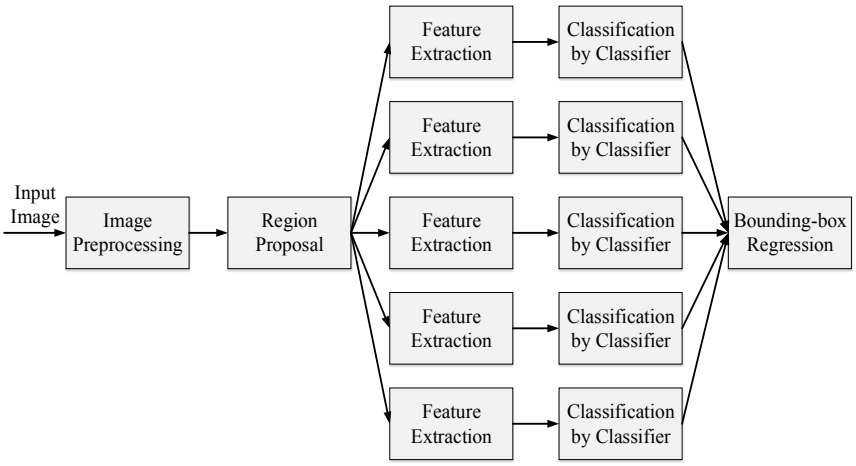


Figure 8 An actual workflow used for tests

5.2 The comparison algorithm

We compare CLOSURE with the following 2 algorithms:

WSH (workflow scheduling based on VM heterogeneity) [76]: In this algorithm, each VM finishing sub-task execution will be recycled, and a new VM will be re-deployed to execute the next sub-task. The OSs of the VMs before and after scheduling have maximum heterogeneity. The available OSs are listed in Table 1.

WSR (workflow scheduling based on random defense strategy): This algorithm is designed by us to compare the performance with CLOSURE. WSR is similar to CLOSURE, the only difference is that in CLOSURE, the probability distribution of the mixed defense strategies is obtained by solving the attack-defense game model, while in WSR, the probability distribution of the mixed defense strategies is uniform.

5.3 Experiment results

5.3.1 Switching period of defense strategies

The switching period of defense strategies is an important factor to measure the security gains brought by CLOSURE, since if the switching period of defense strategies is long, CLOSURE cannot effectively avoid the attacker's reconnaissance.

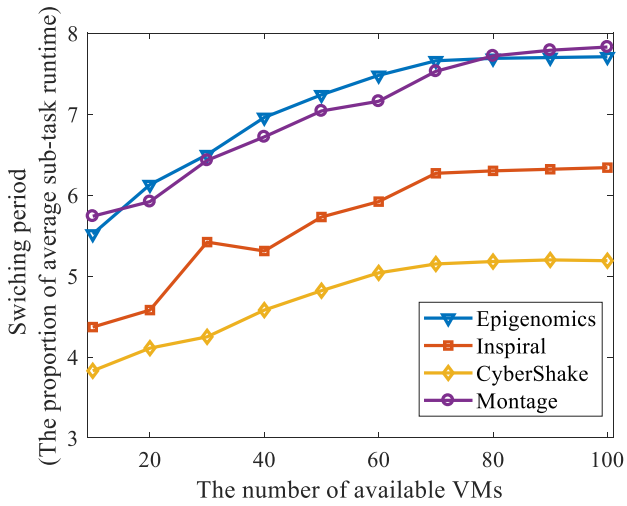


Figure 9 Defense strategy switching period with different number of available VMs

So, in this section, we evaluate the switching period of defense strategies, each experiment is repeated 100 times for different workflow instances, the average evaluation result is shown in Figure 9. In order to ignore the impact of the differences in workflow sub-task runtimes, the switching period is represented by the proportion of the average sub-task runtime. For example, if the switching period is 4, it means that the switching of defense strategies takes four times the average sub-task runtime. From the figure, we can find that with the increase of the number of available VMs, the switching period will show an upward trend first, then stabilize. Because the more VMs are available, the more VMs that need to be converted when switching defense strategies. But when the number of VMs is saturated, the redundant VMs are idle. These idle VMs can be converted without waiting for the sub-task completion, so these redundant VMs will not have effects on the

switching period of defense strategies.

Then, we record the times of switching defense strategies during the workflow execution, the results are shown in Figure 10. Before reaching the saturation point, the increase of the available VMs can accelerate the workflow completion, therefore the times of switching defense strategies shows a downward trend. But, considering the costs, users normally do not choose to rent too many VMs. If there are 20 available VMs, at least 10 times defense strategy switching will be launched, which is enough for avoiding the attacker's reconnaissance.

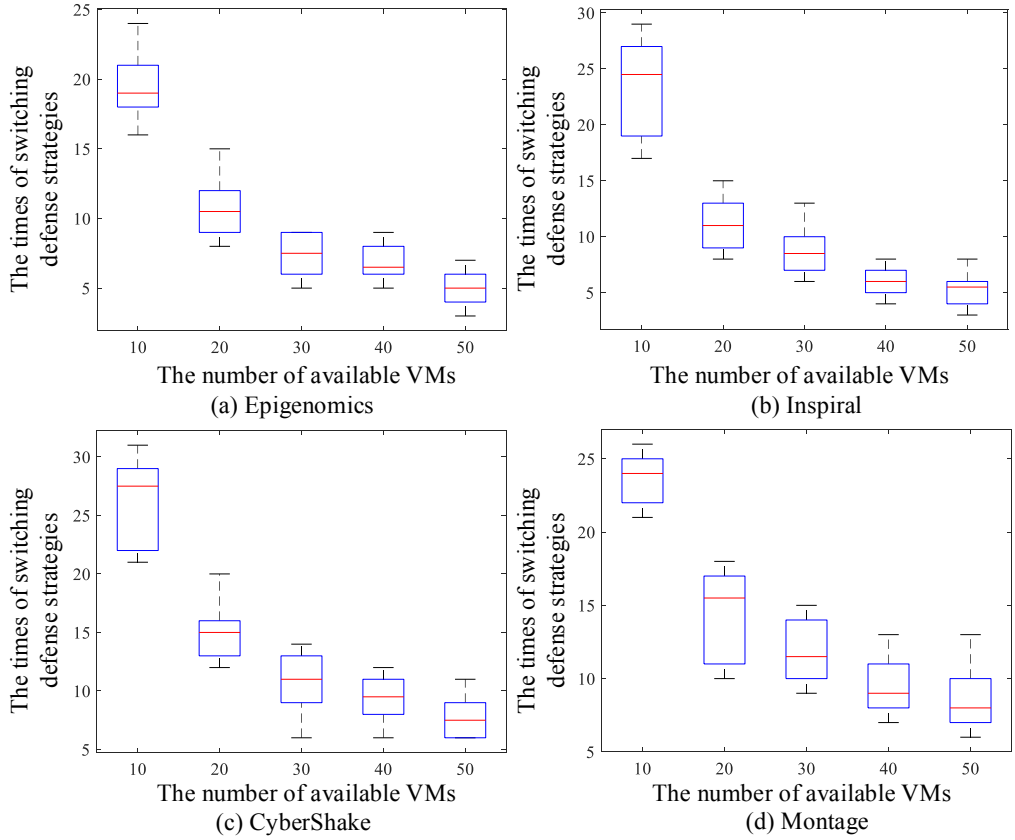


Figure 10 The times of switching defense strategies with different number of available VMs

5.3.2 Benefits of attackers

In this section, we use (3) to quantify the benefits of attackers. We define that the starting point of the attack period is that the attacker starts network reconnaissance and the end point is that the attacker successfully penetrates into the target VM. When one attack cycle ends, a new attack cycle will begin immediately. It is assumed that a scientific workflow makespan is T. If the attack period is $T/2$, the attacker can launch two complete attacks during the scientific workflow execution. So the longer the attack period, the less the number of attacks launched in a limited time. It is assumed that there are 20 available VMs to execute the workflow, and the attacker is a rational individual who uses the mixed attack strategy to attack the scientific workflows, the probability distribution of the mixed attack strategy can be obtained by solving (5). Workflows Epigenomics, Inspiral, CyberShake and Montage are used for this test, and the attack period is a variable which is from 100 to 500 seconds, each experiment is repeated 100 times for different workflow instances, the average results of the attacker's benefits are shown in Figure 11. The longer the attack period, the

less the number of attacks launched, and the fewer the benefits are. Compared with the WSR and WSH, CLOSURE can reduce the benefits of attackers more effectively. Because in this test, the attacker is a rational individual, he will consider the attack benefits and tend to launch high-benefit attacks. CLOSURE will make defense strategies according to this characteristic. According to the results in Figure 11, we use the results of CLOSURE to subtract the results of WSH and average them, the calculation result is 15.23%, so compared with WSH, CLOSURE can reduce the attacker's benefits by around 15.23%.

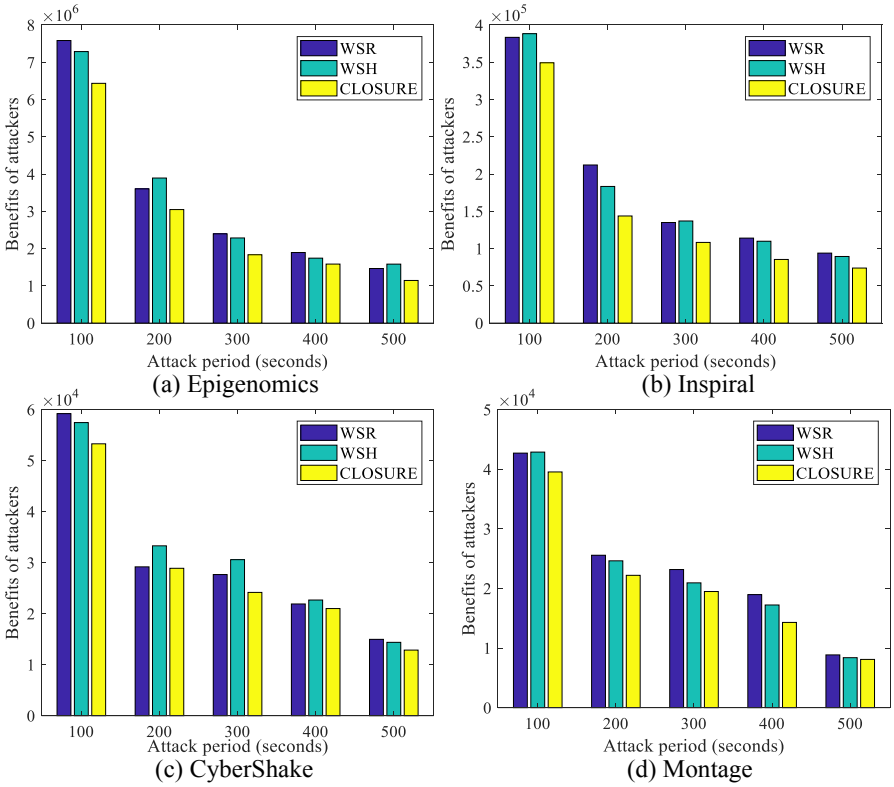


Figure 11 The comparison of the benefits of attackers in WSR, WSH and CLOSURE

5.3.3 Time costs of scheduling algorithm

In this section, we use workflow Epigenomics to compare the time costs of WSH, WSR, and CLOSURE and the experiment is repeated 100 times. It is assumed that re-deploying a VM requires 50 seconds. The timeframe is similar to the time required when deploying an Amazon EC2 instance which is around 50 seconds [77]. We use the ratio of the increased workflow makespan to represent the time costs, and record the time costs of WSH, WSR, and CLOSURE in Table 4. From the table, we can find that the difference of the time cost of WSR and CLOSURE is small because the time costs are mainly from the VM re-deployment during the defense strategy switching. The principles of switching the defense strategy of WSR and CLOSURE are exactly the same, so, the time costs of WSR and CLOSURE are similar. The time cost of WSH is higher than WSR and CLOSURE, since each VM completing the sub-task execution has to be re-deployed, which will generate a lot of time costs. We also can find that the time cost shows an upward trend with the increase of the number of available VMs. Since increasing the number of VMs can shorten the workflow makespan, the shorter the workflow makespan, the larger effects of VM deployment time to the entire workflow. Many scientific workflows take a long time for execution [7], which will make the time cost of

CLOSURE very small. According to the results in Table 4, we use the results of CLOSURE to subtract the results of WSH and average them, the calculation result is 7.86%, so compared with WSH, CLOSURE can reduce the time costs by around 7.86%.

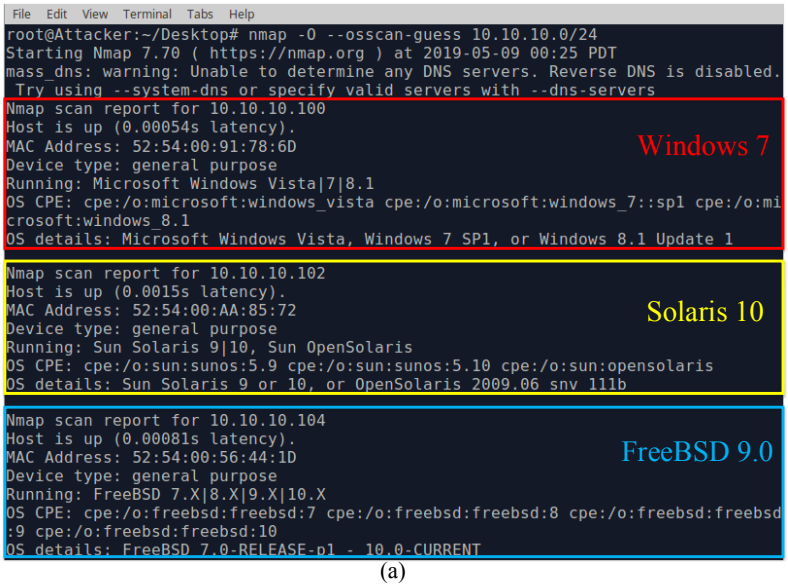
Table 4 The comparison of time costs of WSH, WSR and CLOSURE, time costs are represented by the ratio of the increased workflow makespan

The number of VMs	Workflow makespan	WSH	WSR	CLOSURE
10	426125 s	3.40%	0.92%	1.04%
20	262192 s	5.65%	1.42%	1.25%
30	194785 s	7.53%	1.39%	1.28%
40	165382 s	7.76%	1.57%	1.50%
50	139040 s	9.06%	1.65%	1.84%
60	127718 s	10.82%	1.63%	1.73%
70	113413 s	10.62%	1.47%	1.55%
80	132062 s	12.52%	1.59%	1.55%
90	109890 s	13.17%	1.81%	1.60%
100	107424 s	13.28%	2.04%	1.88%

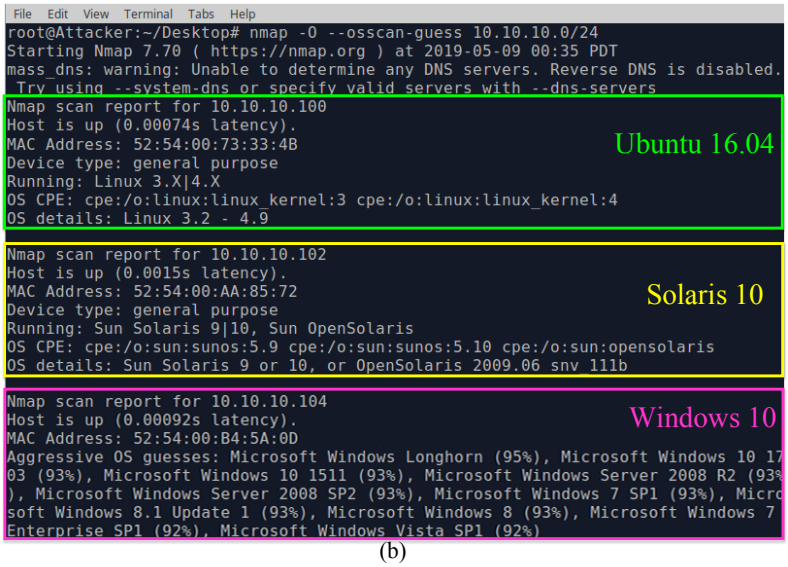
5.3.4 Nmap scan test

In this section, we build a small scientific workflow system as shown in Figure 7 and an actual workflow shown in Figure 8 is used for the test. Nmap is used to simulate the attacker's reconnaissance. Nmap is a free and open source software for network discovery and security auditing, which uses raw IP packets in novel way to determine what hosts are available on the network, what services those hosts are offering and what OSs they are running [39].

In this test, the first defense strategy generated by CLOSURE is one Windows 7 VM, one Solaris 10 VM and one FreeBSD 9.0 VM. Then the attacker used Nmap to scan the whole tenant network (10.10.10.0/24) at 2019-05-09 00:25 and found 3 VMs with the OSs of Windows 7, Solaris 10 and FreeBSD 9.0, as shown in Figure 12 (a). Then, the second defense strategy generated by CLOSURE is one Ubuntu 16.04 VM, one Solaris 10 VM and one Windows 10 VM. So, when the Windows 7 VM and FreeBSD 9.0 VM had finished the sub-task, it would be replaced by an Ubuntu 16.04 VM and a Windows 10 VM. So, 10 minutes later, the attacker used Nmap to scan the whole tenant network again and found that the OS distribution had been changed, as shown in Figure 12 (b). Therefore, CLOSURE can effectively avoid the attacker's reconnaissance, making it difficult for the attackers to obtain information about the defense strategies.



(a)



(b)

Figure 12 (a) Nmap scan test results at 2019-05-09 00:25 (b) Nmap scan test results at 2019-05-09 00:35

6 Conclusions and future work

In order to secure the scientific workflows in clouds, we propose CLOSURE to increase the difficulties for the attackers to infiltrate into VMs executing workflow sub-tasks. Most of the network attacks are launched based on OS vulnerabilities, we regard the attacks based on different OS vulnerabilities as different attack strategies. A homogeneous VM cluster environment can easily cause error propagation, one attack can compromise multiple VMs. So, diverse VMs are used for workflow execution and different OS distributions are regarded as different defense strategies. However, in the attack and defense scenarios for scientific workflows in clouds, the information of the attacker and defender is not balanced. The defender cannot obtain information about the attacker's strategies, while the attacker can acquire information about the defender's strategies through a network scan. For this problem, we propose to dynamically recycle and re-deploy VMs to switch the defense strategies during the workflow execution, which can weaken the attacker's

reconnaissance effects and transform the scientific workflow security problem into the attack-defense game problem. Then, the probability distribution of the optimal mixed defense strategy is acquired by calculating the Nash Equilibrium in the attack-defense game model. Furthermore, task scheduling algorithm based on dynamic HEFT is presented to accelerate the defense strategy switching and improve workflow efficiency. The experiments are conducted on both simulation and actual environment, experimental results demonstrate that compared with the other algorithm, CLOSURE can reduce the attacker's benefits by around 15.23% and the time costs of the algorithm by around 7.86%.

However, only one attacker is considered in CLOSURE. If there are multiple attackers, a multi-player game model needs to be established. We will attempt to solve this problem in the future.

7 Acknowledgements

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB0804004 and Grant 2018YFB1003700, in part by the Foundation for Innovative Research Groups of National Natural Science Foundation of China under Grant 61521003, in part by the Beijing Natural Science Foundation under Grant Z170003, and in part by the Beijing Institute of Technology Research Fund Program for Young Scholars.

Reference

[1] Y. Zhao, Y. Li, I. Raicu, et al. Migrating Scientific Workflow Management Systems from the Grid to the Cloud. Cloud Computing for Data-Intensive Applications. Springer, New York, NY, 2014: 231-256.

[2] J. J. Rehr, F. D. Vila, J.P. Gardner, et al. Scientific computing in the cloud. Computing in science & Engineering, 2010, 12(3): 34.

[3] A. Iosup, S. Ostermann, M. N. Yigitbasi, et al. Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Transactions on Parallel and Distributed systems, 2011, 22(6): 931-945.

[4] E. Deelman, K. Vahi, M. Rynge, et al. Pegasus in the cloud: Science automation through workflow technologies. IEEE Internet Computing, 2016, 20(1): 70-76.

[5] H. Chen, X. Zhu, D. Qiu, et al. Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds. IEEE Transactions on Parallel and distributed systems, 2017, 28(9): 2674-2688.

[6] http://astronomy.swin.edu.au/pulsar/

[7] D. Yuan, Y. Yang, X. Liu, et al. A data dependency based strategy for intermediate data storage in scientific cloud workflow systems. Concurrency and Computation: Practice and Experience, 2012, 24(9): 956-976.

[8] M. A. Rodriguez, R. Buyya. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. Concurrency and Computation: Practice and Experience, 2017, 29(8): 1-23.

[9] A. O. F. Atya, Z. Qian, S. V. Krishnamurthy, et al. Malicious co-residency on the cloud: Attacks and defense. IEEE INFOCOM 2017-IEEE Conference on Computer Communications. IEEE, 2017: 1-9.

[10] Y. Zhang, A. Juels, M. K. Reiter, et al. Cross-tenant side-channel attacks in PaaS clouds.

Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 990-1003.

[11] Z. Wang, J. Wu, Z. Guo, et al. Secure virtual network embedding to mitigate the risk of covert channel attacks. 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2016: 144-145.

[12] J. Wu, Z. Lei, S. Chen, et al. An access control model for preventing virtual machine escape attack. Future Internet, 2017, 9(2): 20.

[13] Z. Li, J. Ge, H. Yang, et al. A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. Future Generation Computer Systems, 2016, 65: 140-152.

[14] K. S. Narayana, S. K. Pasupuleti. Trusted Model for Virtual Machine Security in Cloud Computing. Progress in Computing, Analytics and Networking. Springer, Singapore, 2018: 655-665.

[15] Y. Wang, J. Wu, Y. Guo, et al. Scientific workflow execution system based on mimic defense in the cloud environment. Frontiers of Information Technology & Electronic Engineering, 2018, 19(12): 1522-1536.

[16] S. Nepal, R. O. Sinnott, C. Friedrich, et al. TruXy: Trusted storage cloud for scientific workflows. IEEE transactions on cloud computing, 2015, 5(3): 428-442.

[17] M. Guo, P. Bhattacharya. Diverse virtual replicas for improving intrusion tolerance in cloud. Proceedings of the 9th Annual Cyber and Information Security Research Conference. ACM, 2014: 41-44.

[18] J. Ai, H. Chen, Z. Guo, et al. Mitigating malicious packets attack via vulnerability-aware heterogeneous network devices assignment. Future Generation Computer Systems, https://doi.org/10.1016/j.future.2019.04.034, 2019.

[19] M. Garcia, A. Bessani, I. Gashi, et al. Analysis of operating system diversity for intrusion tolerance. Software: Practice and Experience, 2014, 44(6): 735-770.

[20] G. Cai, B. Wang, W. Hu, et al. Moving target defense: state of the art and characteristics. Frontiers of Information Technology & Electronic Engineering, 2016, 17(11): 1122-1153.

[21] X. Feng, Z. Zheng, P. Mohapatra, et al. A stackelberg game and markov modeling of moving target defense. International Conference on Decision and Game Theory for Security. Springer, Cham, 2017: 315-335.

[22] S. Sengupta, S. G. Vadlamudi, S. Kambhampati, et al. A game theoretic approach to strategy generation for moving target defense in web applications. Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. International Foundation for Autonomous Agents and Multiagent Systems, 2017: 178-186.

[23] X. Feng, Z. Zheng, D. Cansever, et al. A signaling game model for moving target defense. IEEE INFOCOM 2017-IEEE Conference on Computer Communications. IEEE, 2017: 1-9.

[24] J. Rowe, K. N. Levitt, T. Demir, et al. Artificial diversity as maneuvers in a control theoretic moving target defense. National Symposium on Moving Target Research. 2012.

[25] M. D. Adams, S. D. Hitefield, B. Hoy, et al. Application of cybernetics and control theory for a new paradigm in cybersecurity. arXiv preprint arXiv:1311.0257, 2013.

[26] M. Crouse, E. W. Fulp, D. Canas. Improving the diversity defense of genetic algorithm-based moving target approaches. Proceedings of the National Symposium on Moving Target Research. 2012.

[27] D. J. John, R. W. Smith, W. H. Turkett, et al. Evolutionary based moving target cyber defense. Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. ACM, 2014: 1261-1268.

[28] C. Lei, H. Q. Zhang, J. L. Tan, et al. Moving target defense techniques: A survey. Security and Communication Networks, 2018.

[29] F. Wu, Q. Wu, Y. Tan. Workflow scheduling in cloud: a survey. The Journal of Supercomputing, 2015, 71(9): 3373-3418.

[30] A. C. Zhou, B. He, C. Liu. Monetary cost optimizations for hosting workflow-as-a-service in IaaS clouds. IEEE Transactions on Cloud Computing, 2016, 4(1): 34-48.

[31] H. Jiang, M. Song. Dynamic scheduling of workflow for makespan and robustness improvement in the iaas cloud. IEICE TRANSACTIONS on Information and Systems, 2017, 100(4): 813-821.

[32] L. Zuo, L. Shu, S. Dong, et al. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing[J]. Ieee Access, 2015, 3: 2687-2699.

[33] M. A. Rodriguez, R. Buyya. A responsive knapsack-based algorithm for resource provisioning and scheduling of scientific workflows in clouds. 2015 44th International Conference on Parallel Processing. IEEE, 2015: 839-848.

[34] G. Yao, Y. Ding, K. Hao. Using imbalance characteristic for fault-tolerant workflow scheduling in cloud systems. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(12): 3671-3683.

[35] E. K. Byun, Y. S. Kee, J. S. Kim, et al. BTS: Resource capacity estimate for time-targeted science workflows. Journal of Parallel and Distributed Computing, 2011, 71(6): 848-862.

[36] H. Topcuoglu, S. Hariri, M. Y. Wu. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE Transactions on Parallel & Distributed Systems, 2002, 13(3):260-274.

[37] W. Chen, E. Deelman. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. 2012 IEEE 8th International Conference on E-Science. IEEE, 2012: 1-8.

[38] O. Sefraoui, M. Aissaoui, M. Eleuldj. OpenStack: toward an open-source solution for cloud computing. International Journal of Computer Applications, 2012, 55(3): 38-42.

[39] https://nmap.org/

[40] I. Pietri, M. Malawski, G. Juve, et al. Energy-constrained provisioning for scientific workflow ensembles. 2013 International Conference on Cloud and Green Computing. IEEE, 2013: 34-41.

[41] Y. C. Lee, H. Han, A. Y. Zomaya, et al. Resource-efficient workflow scheduling in clouds. Knowledge-Based Systems, 2015, 80: 153-162.

[42] X. Xu, W. Dou, X. Zhang, et al. EnReal: An energy-aware resource allocation method for scientific workflow executions in cloud environment. IEEE Transactions on Cloud Computing, 2016, 4(2): 166-179.

[43] Z. Guo, S. Hui, Y. Xu, et al. Dynamic flow scheduling for power efficient data center networks. IEEE/ACM International Symposium on Quality of Service, Beijing, China, 2016, pp. 1-10.

[44] Y. Ding, G. Yao, K. Hao. Fault-tolerant elastic scheduling algorithm for workflow in cloud systems. Information Sciences, 2017, 393: 47-65.

[45] G. Yao, Y. Ding, L. Ren, et al. An immune system-inspired rescheduling algorithm for

workflow in Cloud systems. Knowledge-Based Systems, 2016, 99: 39-50.
[46] D. Poola, K. Ramamohanarao, R. Buyya. Enhancing reliability of workflow execution using task replication and spot instances. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2016, 10(4): 30.
[47] L. Zeng, B. Veeravalli, X. Li. SABA: A security-aware and budget-aware workflow scheduling strategy in clouds. Journal of parallel and Distributed computing, 2015, 75: 141-151.
[48] W. Liu, S. Peng, W. Du, et al. Security-aware intermediate data placement strategy in scientific cloud workflows. Knowledge and information systems, 2014, 41(2): 423-447.
[49] H. Chen, X. Zhu, D. Qiu, et al. Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds. IEEE Transactions on Parallel and distributed systems, 2017, 28(9): 2674-2688.
[50] M. Ali, S. U. Khan, A. V. Vasilakos. Security in cloud computing: opportunities and challenges. Information sciences, 2015, 305: 357-383.
[51] J. Szefer, E. Keller, R. B. Lee, et al. Eliminating the hypervisor attack surface for a more secure cloud. Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011: 401-412.
[52] F. Zhang, H. Chen. Security-preserving live migration of virtual machines in the cloud. Journal of network and systems management, 2013, 21(4): 562-587.
[53] M. H. Song. Analysis of risks for virtualization technology. Applied Mechanics and Materials. Trans Tech Publications, 2014, 539: 374-377.
[54] J. Sen. Security and privacy issues in cloud computing. Cloud Technology: Concepts, Methodologies, Tools, and Applications. IGI Global, 2015: 1585-1630.
[55] Z. Wang, R. Yang, X. Fu, et al. A shared memory based cross-VM side channel attacks in IaaS cloud. 2016 IEEE conference on computer communications workshops (INFOCOM WKSHPS). IEEE, 2016: 181-186.
[56] F. Liu, Q. Ge, Y. Yarom, et al. Catalyst: Defeating last-level cache side channel attacks in cloud computing. 2016 IEEE international symposium on high performance computer architecture (HPCA). IEEE, 2016: 406-418.
[57] F. Liu, Y. Yarom, Q. Ge, et al. Last-level cache side-channel attacks are practical. IEEE Symposium on Security and Privacy. IEEE, 2015: 605-622.
[58] G. Irazoqui, T. Eisenbarth, B. Sunar. S $ A: A shared cache attack that works across cores and defies VM sandboxing--and its application to AES. IEEE Symposium on Security and Privacy. IEEE, 2015: 591-604.
[59] J. A. J. Sujana, T. Revathi, T. S. S. Priya, et al. Smart PSO-based secured scheduling approaches for scientific workflows in cloud computing. Soft Computing, 2019, 23(5): 1745-1765.
[60] Z. Wen, J. Cała, P. Watson, et al. Cost effective, reliable and secure workflow deployment over federated clouds. IEEE Transactions on Services Computing, 2016, 10(6): 929-941.
[61] P. K. Manadhata. Game theoretic approaches to attack surface shifting. Moving Target Defense II. Springer, New York, NY, 2013: 1-13.
[62] Q. Zhu, T. Basar. Game-theoretic approach to feedback-driven multi-stage moving target defense. International Conference on Decision and Game Theory for Security. Springer, Cham, 2013: 246-263.
[63] K. M. Carter, J. F. Riordan, H. Okhravi. A game theoretic approach to strategy determination for dynamic platform defenses. Proceedings of the First ACM Workshop on Moving Target

Defense. ACM, 2014: 21-30.
[64] X. Feng, Z. Zheng, P. Mohapatra, et al. A stackelberg game and markov modeling of moving target defense. International Conference on Decision and Game Theory for Security. Springer, Cham, 2017: 315-335.
[65] A. A. A. Abass, L. Xiao, N. B. Mandayam, et al. Evolutionary game theoretic analysis of advanced persistent threats against cloud storage. IEEE Access, 2017, 5: 8482-8491.
[66] J. Musial, M. Guzek, P. Bouvry, et al. A note on the complexity of scheduling of communication-aware directed acyclic graph. Bulletin of the Polish Academy of Sciences: Technical Sciences, 2018.
[67] D. Kliazovich, J. E. Pecero, A. Tchernykh, et al. CA-DAG: Modeling communication-aware applications for scheduling in cloud computing. Journal of Grid Computing, 2016, 14(1): 23-39.
[68] C. Lin, S. Lu, X. Fei, et al. Architecture for Scientific Workflow Management Systems and the VIEW SOA Solution, IEEE Transactions on Services Computing, 2009, 2(1): 79-92.
[69] S. Bharathi, A. Chervenak, E. Deelman, et al. Characterization of scientific workflows. In IEEE Workshop on Workflows in Support of Large-Scale Science, Austin, USA, 2008: 1-10.
[70] Y. Wang, Y. Guo, Z. Guo, et al. Securing the intermediate data of scientific workflows in clouds with ACISO. IEEE Access, 2019, 7(1): 126603-126617.
[71] V. Del Piccolo, A. Amamou, K. Haddadou, et al. A survey of network isolation solutions for multi-tenant data centers. IEEE Communications Surveys & Tutorials, 2016, 18(4): 2787-2821.
[72] M. Campanelli, R. Gennaro. Sequentially composable rational proofs. International Conference on Decision and Game Theory for Security. Springer, Cham, 2015: 270-288.
[73] M. J. Osborne, A. Rubinstein. A course in game theory. MIT press, 1994.
[74] V. Conitzer, T. Sandholm. Computing the optimal strategy to commit to. Proceedings of the 7th ACM conference on Electronic commerce. ACM, 2006: 82-90.
[75] E. Deelman, K. Vahi, G. Juve, et al. Pegasus, a workflow management system for science automation. Future Generation Computer Systems, 2015, 46: 17-35.
[76] Y. Wang, Y. Guo, W. Liu, et al. A Task Scheduling Method for Cloud Workflow Security. Journal of Computer Research & Development, 2018, 55(6): 66-75.
[77] S. Ostermann, A. Iosup, N. Yigitbasi, et al. A performance analysis of EC2 cloud computing services for scientific computing. International Conference on Cloud Computing. Springer, Berlin, Heidelberg, 2009: 115-131.
[78] R. Zhuang, S. A. DeLoach, X. Qu. Towards a theory of moving target defense. Proceedings of the First ACM Workshop on Moving Target Defense. ACM, 2014: 31-40.
[79] A. Verma, S. Kaushal. A hybrid multi-objective Particle Swarm Optimization for scientific workflow scheduling. Parallel Computing, 2017, 62:1-19.