Journal of Information Processing Vol.23 No.4 465–475 (July 2015)

[DOI: 10.2197/ipsjjip.23.465]

Regular Paper

Security Risk Quantification Mechanism for Infrastructure as a Service Cloud Computing Platforms

Doudou Fall[1,a)] Takeshi Okuda[1,b)] Youki Kadobayashi[1,c)] Suguru Yamaguchi[1,d)]

Received: September 16, 2014, Accepted: March 4, 2015

Abstract: Cloud computing has revolutionized information technology, in that It allows enterprises and users to lower computing expenses by outsourcing their needs to a cloud service provider. However, despite all the benefits it brings, cloud computing raises several security concerns that have not yet been fully addressed to a satisfactory note. Indeed, by outsourcing its operations, a client surrenders control to the service provider and needs assurance that data is dealt with in an appropriate manner. Furthermore, the most inherent security issue of cloud computing is multi-tenancy. Cloud computing is a shared platform where users' data are hosted in the same physical infrastructure. A malicious user can exploit this fact to steal the data of the users whom he or she is sharing the platform with. To address the aforementioned security issues, we propose a security risk quantification method that will allow users and cloud computing administrators to measure the security level of a given cloud ecosystem. Our risk quantification method is an adaptation of the fault tree analysis, which is a modeling tool that has proven to be highly effective in mission-critical systems. We replaced the faults by the probable vulnerabilities in a cloud system, and with the help of the common vulnerability scoring system, we were able to generate the risk formula. In addition to addressing the previously mentioned issues, we were also able to quantify the security risks of a popular cloud management stack, and propose an architecture where users can evaluate and rank different cloud service providers.

Keywords: cloud computing, security, risk, quantification

1. Introduction

Cloud computing is revolutionizing the way society uses computing resources. It promises a new economic model, which is profitable for both cloud computing service providers and customers. Although many industry and academic definitions exist for cloud computing, the most widely accepted definition originates from the National Institute of Standards and Technology [1] (NIST) that defines cloud computing as "a new model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." The cloud computing business model permits non-IT-related companies to focus more on their specialties rather than their IT infrastructures. However, while cloud computing has numerous benefits, especially in regards to infrastructure costs, it also has some disadvantages that must be addressed.

In order to reap the stated benefits of cloud computing, it is necessary to investigate its disadvantages. Indeed, despite all the benefits, users are still reluctant to adopt cloud computing because of its security issues identified in numerous survey papers [2], [3], [4], [5], [6], [7]. These security issues range from the security of the multi-tenant aspect of cloud computing [8] to the loss of control [9].

The current cloud computing model is based on an outsourcing strategy. Essentially, customers are generally hesitant to entrust the management of their data to a third party. The issue of trust is magnified in this case and Cloud Service Providers (CSPs) must provide the necessary guarantees to convince the customers that their data will be safe and secure. Furthermore, the fact that the cloud environment is a shared platform (i.e., all the users' data are pooled up and stored in the same physical infrastructure) makes it a nightmare for CSPs in their endeavor to attract potential customers. It is this inherent multi-tenancy aspect of cloud computing that also dissuades customers from utilizing the cloud.

In this paper, we are exploring the fact that any kind of attack that can happen in the cloud is the result of an exploitation of unpatched vulnerabilities. In practice, many vulnerabilities remain in a cloud environment after they are discovered. This issue is due to environmental factors (latency in releasing vulnerability patches), cost factors (such as money and administrative efforts required for deploying patches), or mission factors (organizational preference for availability and usability over security). Therefore, addressing the problem of security in cloud computing is a huge challenge.

Cloud computing is widely accepted as having three preeminent service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In summary, the IaaS service model provides the virtual infrastructure (networks, bandwidth, virtual machines, volumes, etc.). The PaaS service model contains the middleware services. The SaaS service model provides software features that are used by end-

[1] Nara Institute of Science and Technology, Ikoma, Nara 630–0192, Japan
[a)] doudou-f@is.naist.jp
[b)] okuda@is.naist.jp
[c)] youki-k@is.naist.jp
[d)] suguru@is.naist.jp

users via a thin-client. Each of these service models has its inherent security issues but, in this paper, we have decided to validate this research over the IaaS service model. IaaS is the foundation of any cloud infrastructure and presents the highest level of multitenancy with the different tenants sharing storage, CPU, network bandwidth, and memory.

On the other hand, cloud computing has burgeoned to become the dominant paradigm in information technology (IT). The rapid development of cloud computing has permitted the emergence of other important paradigms like cloud management stacks. A cloud management stack is a set of components that work together to facilitate the management of a cloud infrastructure system. A cloud management stack is, at least, composed of the following components: an external application programming interface (API) that assures the communication between the cloud services and external users; a compute service which makes charge the management of the virtual machines (VMs) on the host machines in terms of features like creation, deletion or suspension of VMs; an image service for managing the deployment or registration of VM images; a volume service that maps persistent storage used by the VMs; and a network service that helps with the management of the networks used by the VMs. In addition to the intrinsic aforementioned components, cloud management stacks rely on some external services that are critical for its functioning. Among those external services, the hypervisor is regarded as the most important. Popular cloud computing management stacks include OpenStack [10], OpenNebula [11], CloudStack [12], or Eucalyptus [13]. In this paper, our focus is on OpenStack because it is the most deployed cloud management software, plus it has a vibrant community that provides all the necessary documentation. Since its inception, OpenStack has had numerous releases; this research considers the HAVANA edition. People in academia and industry often use OpenStack to deploy their private clouds. However, with its rapid adoption, OpenStack is also rapidly beginning to garner attention in the National Vulnerability Database (NVD) [14]. Indeed, OpenStack has a total of 101 vulnerabilities (as of September 2014) that have scores ranging from 9.0 to 1.9 in the Common Vulnerability Scoring System (CVSS) [15]. **Figure 1** gives a visual description of the former statement. Furthermore, the logical architecture of OpenStack [16] reveals a deep level of interconnectedness between its different components (services) and subcomponents. We contend that these two situations, mixed together, could jeopardize the security of the cloud systems of the different adopters of OpenStack. Due to this level of interconnectedness, a successful attack in one component can turn out to be a successful attack on the entire architecture.

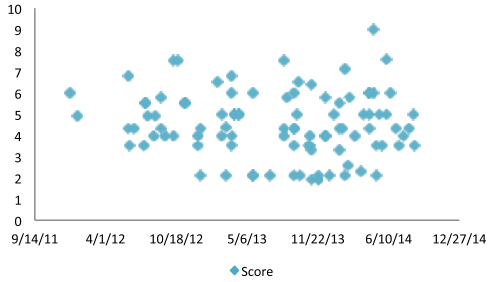

Fig. 1 OpenStack presence in the NVD.

In order to prevent the issues mentioned above, we contend that the best solution is to quantify security risks in cloud computing. Our main contribution is a novel approach for quantifying security in cloud computing, inspired by Fault Tree Analysis (FTA), which is commonly used in Probabilistic Risk Analysis (PRA) (which in turn is used to quantify the risk of failure in highly mission-critical systems like those of a nuclear power plant). In FTA, a fault tree is built out of the probable faults that could occur in the system. A further analysis of the fault tree concludes in a quantified result that could help decision makers to plan their future strategy. Our security quantification method consists of the vulnerabilities of an IaaS that we use to build a Boolean vulnerability tree while conforming to FTA rules. The analysis of the vulnerability tree leads to the extraction of the quantification formula. Risk (R) is generally defined as the likelihood (L) of an unwanted event to occur multiplied by the impact (I) that particular event would cause: $R = LxI$. Using the Common Vulnerability Scoring System (CVSS), we were able to generate an impact sub-formula and a likelihood sub-formula for any single vulnerability. We then use the vulnerability tree to generate the Impact and Likelihood formulas. The final risk value is calculated by using the traditional risk definition. We were able to prove how our solution works in a multi-tenant platform, and also presented a unique architecture for ranking cloud service providers. We finally used the security risk to perform an analysis of OpenStack logical architecture. We were able to generate the different security vulnerability trees of the components that compose the architecture, and we were also able to make some recommendations on a better nomenclature of OpenStack vulnerabilities.

The remainder of the paper is structured as follows. Section 2 is about the related work. In Section 3, we detail our proposal. Section 4 contains a proof-of-concept of how our method works in a multi-tenant system. Section 5 is entirely dedicated to the ranking platform. In Section 6, we showcase the security analysis of the logical architecture of OpenStack. In Section 7, we propose a discussion of our findings and give a hint of our future work. Section 8 concludes the paper.

2. Related Work

Most of the work related to this study comes from reliability system analysis. Indeed, we make use of the fault tree to perform our security analysis, which is similar to applying fault tree in a highly critical system like nuclear power plant systems. To the best of our knowledge, besides Fall et al. [17], we are not aware of a similar work that has been conducted in cloud computing, particularly in OpenStack architecture. Nevertheless, there is some work related to the security of OpenStack that we can cite as reference.

Zhai el al. [18] produced the closest work to this research. They proposed a structural reliability auditing (SRA) technique that permits the quantification of the vulnerabilities of interdependent infrastructures in a cloud platform. The operating of the system is divided in three steps: infrastructure dependency data collection,

construction of a fault tree based on the gathered data, and analysis of the fault tree to estimate the probability of failure of the top event. They demonstrated the practicality of their system by implementing it, which also showed the lack of privacy measures for the data that is being used. Xiao et al. [19] fixed the issue by adding a privacy aspect to the SRA system. They re-engineered the 3-step process of Ref. [18]'s work by adding privacy to each step and using Secure Multi-Party Computation (SMPC). They were able to evaluate an implemented version of their proposal on the Sharemind SecretC platform.

Khan et al. [20] proposed an OpenId authentication mechanism for OpenStack. In their system, OpenId provides authentication for the user solely while the cloud provider manages the access control policies. Sasko et al. [21] performed an OpenStack security assessment. They set up a system running OpenStack with virtual machines that separately have Ubuntu, CentOS, Fedora and Windows operating systems. After running a vulnerability scan, they concluded that the latter operating system was more subject to vulnerabilities than the others.

In the same spirit, Donevski et al. [22] proposed a security assessment for virtual machines in open source clouds. They also used OpenStack and performed their assessment in two different network situations for the virtual machines: same IP addresses for floating and fixed IPs, and two different IPs for both. They were able to label out different test cases that gave different results that they classified qualitatively and quantitatively by using the CVSS. Aryan et al. [23] evaluated the degree of compromise of a cloud environment knowing that, at least one of the components is compromised.

A body of work that is similar to our initiative in the cloud level exists in the enterprise network domain and is called Attack graph [24]. An attack graph elucidates how multiple vulnerabilities may be combined to launch an attack in an enterprise network. Usually, an attack graph makes use of vulnerability on a host and evaluates the possibility of propagation towards different hosts. Different tools have been developed to automatically generate attack graphs. MulVAL [25] (Multihost, multistage Vulnerability Analysis) is a scalable attack graph generator that models the interaction of the constituents of an enterprise network (software bugs, system and network configuration). MULVAL uses Datalogs as modeling language contrarily to most of the other tools in this domain; they use model checking. NETSPA [26] (A Network Security Planning Architecture) is an attack graph tool that has a primarily preventive role. From the network topology of an enterprise, NETSPA generates an attack graph of all the possible paths an attacker can use, which permits network administrators to take necessary measures for plausible attacks. Topological Vulnerability Analysis (TVA) [27] is a tool that combines simulated attacker exploits on a network to discover attack paths that are used to assess the overall vulnerability of the network. There also exist commercial tools for vulnerability analysis and attack graph generation. As instances, Nessus [28], Retina [29] and Tripwire IP360 [30] are well-known network vulnerability scanners that provide vulnerability management, vulnerability analysis and network protection. Skybox security [31] and Red Seal Systems [32] propose attack graph generators where risk is calculated in the traditional way: likelihood multiplied by the impact.

On the other hand, despite the fact that it is out of the scope of this research, we wanted to mention that Failure Mode and Effects Analysis (FMEA) [33] and Root Cause Analysis (RCA) [34] compete with Fault Tree Analysis (FTA) [35] on modeling failures in a given system. We prefer FTA because we contend that it is more suitable for our research.

3. Proposal: Cloud Computing Security Risk Quantification

The concept of risk is very complex and the available definitions are sources of ambiguity. Indeed, risk is considered as "the probability that a particular adverse event occurs during a stated period of time, or results from a particular challenge" [36]. Another definition of risk considers it as "a combination of the probability or frequency of occurrence of a defined hazard and the magnitude of the consequences of the occurrence" [37]. These definitions seem different yet similar as they convey the idea of an undesired event linked with its consequences. In Information Technology (IT), risk is defined as "the net negative impact of the exercise of a vulnerability, considering both the probability and the impact of the occurrence" [38].

In this research, the concept of risk revolves around the concept of security vulnerability. A vulnerability is a security weakness that could be exploited to cause loss of or harm to the assets of a system. In line with the IT definition of risk, we consider risk as being the likelihood of a vulnerability being exploited times the impact of the previous case happening as highlighted in Eq. (1).

$$Risk = Likelihood \times Impact. \tag{1}$$

In the remainder of this section, we shed the light into our proposal. First we enumerate the security issues we want to tackle. Afterwards we explain in detail the methodology and the different tools that helped us to reach our goal.

3.1 Security Risks in Cloud Computing

It is a truism to single out the security issues of cloud computing as the main obstacles to its adoption, but that does not make it any less true. Indeed, since the inception of cloud computing as we know it in the modern era, until now, where it has matured, its security issues are still alarming. The most obvious security issue is the loss of control. The main feature of cloud computing is its capability to host users' computing resources. The users perceive that feature as a risk to their data. While it is really attractive, users are not yet ready to delegate the control of their data to the cloud administrators who can be malicious. The users' data can be leaked inadvertently or consciously to their competitors. In this research, we do not offer a direct solution for this security issue. However, we believe that the security issues that we are directly tackling can lead to a loss of control issue.

The other security point of contention is multi-tenancy. Cloud computing promotes cheap services for the users. But that implies some other concessions from the cloud service provider (CSP). In fact, to be able to host the users' data at the cheapest price possible, CSPs host their data on the same platform. In the early days

of cloud computing, Ristenpart et al. [8] demonstrated that attackers can take advantage of that shared platform to steal the data of their neighboring tenants. This nagging security issue of cloud computing is still unresolved. A tenant is quasi-totally oblivious of the maliciousness (respectively, honesty) of their other fellow tenants. They do not want to give to each other the benefit of the doubt and request the CSP to meticulously isolate their interests from other tenants' interests.

Another motivating issue is cloud computing ranking systems based on security. Indeed, as seen in numerous surveys, the main concern for users is security. Yet researchers continue to focus on proposing cloud computing ranking systems that are mainly based on the performances of the cloud service providers. We are not trying to undermine that valuable research by any means as they are important to a certain extent, but what really matters for the adopters of cloud computing is security. Thus, a security ranking system that allows users to securely evaluate a CSP before requesting its services is primordial nowadays. We look forward to propose a viable solution for this matter.

Recently, we are assisting the rapid development of new technologies in cloud computing called cloud management stacks. These technologies facilitate the management of cloud infrastructure platforms by providing different layers of software solutions that permit a cloud administrator to easily handle the different required tasks in their datacenter. Organizations that are looking forward to building private clouds are euphorically adopting these software products without evaluating their security. This reminds us of the beginning of the Internet where researchers developed a flurry of applications without any care about their security. The result is that we are continuously trying to fix those early mistakes. We want this paper to serve as a wake-up call for all the aficionados of cloud management stacks to be more security oriented.

3.2 Fault Tree Analysis

A fault tree [35], [39] is a basic tool used as part of a quantitative analysis of a system. It gives rise to a pictorial representation of an undesirable event in a system in Boolean logic. The analysis of the fault tree is the process of developing a deterministic description of the occurrence of an undesirable event, the top event, in terms of the occurrence or non-occurrence of other events called intermediate events. Furthermore, the intermediate events are deeply explored until the basic events, which represent the lowest events of the tree, are reached. Each node in a fault tree represents either an event or a logic gate. The logic gates determine the logical relationship among the events. The events can be fundamentally different but should belong to the same family, i.e., when the top event is a successful attack on an infrastructure, the basics events are successful attacks on some of the components that constitute the infrastructure. Additionally, since fault trees are expressions in Boolean logic, their usage implies that the events are binary, that is, true or false. Fault tree construction requires different symbols and notations, some of them are illustrated in **Fig. 2**. Practically, the use of various gates can be helpful to construct a well-detailed fault tree but, in principle, it is possible to construct any fault tree from the combination of

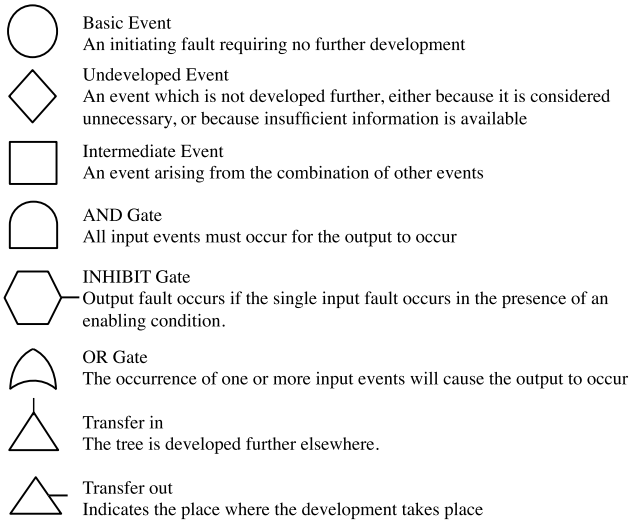

Fig. 2 List of standard fault tree symbols.

AND and *OR* gates. Hereafter, we provide some definitions that are necessary for a better comprehension of fault tree analysis.

Definition 1: A cut set is a collection of basic events such that if these events occur together then the top event will certainly occur.

Definition 2: A minimal cut set is a collection of basic events forming a cut set such that if any of the basic events are removed, then the remaining set is no longer a cut set.

Definition 3: A path set is a collection of basic events such that if none of these events occur then the top event will certainly not occur.

Definition 4: A minimal path set is a path set such that if any of the events are removed then the remaining set will no longer be a path set.

The minimal cut and path sets are primordial for quantifying the probability of the top event. Suppose that we have a fault tree representation with a top event T and several cut sets $C_1, \ldots, C_n$. From the aforementioned definitions, we know that T is a set composed of all the possible cut sets of the fault tree:

$$T = C_1 \cup C_2 \cup \ldots \cup C_n. \tag{2}$$

By applying the inclusion-exclusion law of probability [39] to Eq. (2), we obtain Eq. (4):

$$P[T] = P(C_1 \cup C_2 \cup \ldots \cup C_n). \tag{3}$$

$$P[T] = \sum_{i=1}^{n} P[C_i] - \sum_{i<j<k} P[C_i \cap C_j \cap C_k] - \ldots + (-1)^{n+1} P[C_1 \cap C_2 \cap \ldots \cap C_n]. \tag{4}$$

In the subsequent security analysis, we will use this formulation to perform the security quantification. The *Impact* and *Likelihood* equations are defined as follows:

$$I[T] = \sum_{i=1}^{n} I[C_i] - \sum_{i<j<k} I[C_i \cap C_j \cap C_k] - \ldots + (-1)^{n+1} I[C_1 \cap C_2 \cap \ldots \cap C_n]. \tag{5}$$

$$L[T] = \sum_{i=1}^{n} L[l_i] - \sum_{i<j<k} L[l_i \cap l_j \cap l_k] - \ldots + (-1)^{n+1} L[l_1 \cap l_2 \cap \ldots \cap l_n]. \tag{6}$$

We will also write vulnerability instead of a fault tree in order to be more in line with security, but the intrinsic concepts of fault tree remain intact. In the following section, we explain how we use the vulnerabilities in the National Vulnerability Database (NVD) and the Common Vulnerability Scoring System (CVSS) to achieve our proposal.

3.3 National Vulnerability Database and Common Vulnerability Scoring System

The National Vulnerability Database is a publicly available database for computer-related vulnerabilities. It is a property of the United States (US) government, which manages it throughout the computer security division of the U.S. National institute of Science and Technology (NIST). The NVD is also used by the U.S. government as a content repository for the Security Content Automation Protocol (SCAP). The primary sources of the NVD are as follows: Vulnerability Search Engine (Common Vulnerability Exposure (CVE) and CCE misconfigurations), National Checklist Program (automatable security configuration guidance in XCCDF and OVAL), SCAP and SCAP compatible tools, Product dictionary (CPE), Common vulnerability Scoring System for impact metrics, and Common Weakness Enumeration (CWE).

The Common Vulnerability Scoring System (CVSS) [15] is a vendor-neutral open source vulnerability scoring system. It was established to help organizations to efficiently plan their responses regarding security vulnerabilities. The CVSS is comprised of three metric groups classified as base, temporal, and environmental. The base metric group contains the quintessential characteristics of a vulnerability. The temporal metric group is used for non-constant characteristics of a vulnerability, and the environmental metric group defines the characteristics of a vulnerability that are tightly related to the user's environment. We want our proposal to be sufficiently generic so that it can be utilized at any time by any organization which expressly desires to adopt a secure cloud computing system. For that reason, we opted to make exclusive use of a base metric group which provides the constant characteristics of a vulnerability. In doing so, the vulnerabilities will not change in relation to either time or organization. Consequently, the temporal and environmental metric groups do not feature prominently in our research. The base metric group regroups essential metrics that are used to compute the score of a vulnerability: Access Vector (AV) is the metric reflecting how the vulnerability is exploited; Access Complexity (AC) is the metric that defines how difficult it is to exploit a vulnerability once an attacker has gained access to the target system; Authentication (Au) is the metric that reflects the number of times an attacker must authenticate to a target in order to exploit a vulnerability; Confidentiality Impact (C) is the metric that measures the impact on confidentiality of a successfully exploited vulnerability; Integrity Impact (I) is the metric that measures the impact to integrity of a successfully exploited vulnerability; and Availability Impact (A) is the metric that measures the impact to availability of a successfully exploited vulnerability. The base equation is the foundation of CVSS scoring; it contains two sub-equations that are of particular interest in our proposal:

```
AccessVector     = case AccessVector of
                       requires local access: 0.395
                       adjacent network accessible: 0.646
                       network accessible: 1.0

AccessComplexity = case AccessComplexity of
                       high: 0.35
                       medium: 0.61
                       low: 0.71

Authentication   = case Authentication of
                       requires multiple instances of authentication: 0.45
                       requires single instance of authentication: 0.56
                       requires no authentication: 0.704

ConfImpact       = case ConfidentialityImpact of
                       none:             0.0
                       partial:          0.275
                       complete:         0.660

IntegImpact      = case IntegrityImpact of
                       none:             0.0
                       partial:          0.275
                       complete:         0.660

AvailImpact      = case AvailabilityImpact of
                       none:             0.0
                       partial:          0.275
                       complete:         0.660
```

Fig. 3 CVSS parameter details.

$$Impact = 10.41 \times (1-(1-ConfImpact) \times (1-IntegImpact) \times (1-AvailImpact)) \quad (7)$$

$$Exploitability = 20 \times AccessVector \times AccessComplexity \times Authentication \quad (8)$$

We are constrained to slightly amend the previous sub-equations because their default values range between 0 to 10 whereas we are looking for probabilities like values. The amendment results in Eq. (9) and Eq. (10).

$$Impact = 1.041 \times (1-(1-ConfImpact) \times (1-IntegImpact) \times (1-AvailImpact)) \quad (9)$$

$$Likelihood = 2 \times AccessVector \times AccessComplexity \times Authentication \quad (10)$$

In this research, we consider an impact tree and a Likelihood tree. The analysis of the aforementioned trees lead to the Eqs. (5) and (6). Precious information about the parameters used in the CVSS equations are displayed in **Fig. 3**. In the subsequent sections, we will explore three distinct cases that we deem as popular IaaS cloud deployment. The first case will feature the classic virtualization architecture where we have a server, a virtual machine monitor and virtual machines. We use that case to epitomize how to perform a security analysis of a multi-tenant cloud platform. Our second use case represents a modern architecture that allows users to compare the security of different cloud service providers (CSPs). The result of the ranking will give the user a good insight on which CSP they should use. The final use case focuses on the security quantification of OpenStack.

4. Multi-tenancy Security Quantification

Computing resources in Infrastructure as a Service (IaaS) cloud computing are typically consumed using virtual machines. By the magic of virtualization, a physical machine is represented by several virtual machines running their own operating system. The

Table 1 Vulnerabilities of this example and their impact and likelihood values.

Components	Vulnerabilities	Renaming	Impact	Likelihood
Xen 4.1	CVE-2011-1898	V_{X1}	1	0.44
	CVE-2011-1583	V_{X2}	1	0.34
VM_1 (Apache 2.0)	CVE-2011-3192	V_{11}	0.69	1
	CVE-2011-4317	V_{12}	0.29	0.86
	CVE-2011-4415	V_{13}	0.29	0.19
VM_2 (MySQL)	CVE-2010-1626	V_{21}	0.49	0.39
VM_7 (Bind 9.8.0)	CVE-2011-2465	V_{71}	0.29	0.49
	CVE-2011-2464	V_{72}	0.29	1

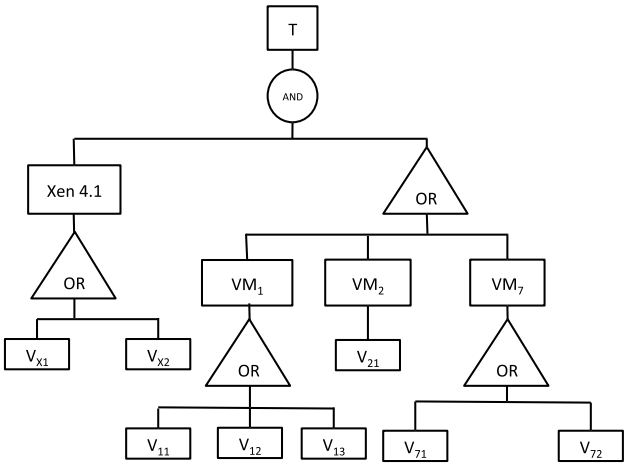

Fig. 4 Vulnerability tree of the use case.

virtual machines are isolated from each other in a multi-tenant environment. The system providing the abstraction of the hardware and managing the virtual machines is called the Hypervisor or Virtual Machine Monitor (VMM). Thus, the hypervisor plays a pivotal role in IaaS (machine virtualization) as it represents the most important link of the entire infrastructure chain system. The level of security of the shared resources significantly depends on the corresponding strength or weakness of the hypervisor. These factors culminate into the following hypothesis:

Hypothesis 1 In a multi-tenant IaaS cloud, unauthorized access occurs if and only if the attacker succeeds in exploiting a vulnerability on the virtual machine monitor.

This hypothesis is a pretext that we use to define complex attacks, which are any kind of attacks that involve multiple vulnerabilities (at least two). This hypothesis constitutes the cornerstone of this section and is only limited to this section.

Case study

In this case study, we use a cloud infrastructure that is running XEN-4.1 as hypervisor and has multiple virtual machines. After a security vulnerability scanner, the administrator discovers the vulnerabilities exposed in **Table 1** with their impact and likelihood values derived from Eq. (9) and Eq. (10) respectively.

The results of the scanner revealed that there are vulnerabilities in the hypervisor, VM_1, VM_2, and VM_7. The vulnerability tree of this scenario is shown in **Fig. 4**.

The significance of this scenario is the presence of multiple vulnerabilities in some components of the infrastructure: hypervisor (two), VM1 (three), and VM7 (two). Before applying our global formula to the entire system, we do the partial quantifications for those specific components. For each of the aforementioned components, we use the equations that we developed earlier to perform the partial Impact and Likelihood quantifications. Afterwards, we proceed to generate the actual risk quantification. The results are summarized in **Table 2**. We consider the vulnerabilities to be independent but not mutually exclusive.

Table 2 Use case: results summary.

Components	impact	likelihood
Xen 4.1	1	0.6304
VM_1	0.8437	1
VM_2 (MySQL)	0.49	0.39
VM_7	0.4959	1
I[T]	0.9598	0
L[T]	0	0.6304
Risk	0.605	

The NVD provides a vulnerability severity ratings which is as follows:

- "Low" for CVSS base scores that range from 0.0 to 3.9
- "Medium" if the vulnerabilities have base scores of 4.0–6.9
- "High" if the CVSS base scores range from 7.0 to 10

As we opted to work with probability-like values, we reduced the vulnerability severity rating of the NVD to the following:

- "Low" if the risk values range between 0.0–0.39
- "Medium" if the risk values range between 0.4–0.69
- "High" if the risk values range between 0.7–1

Therefore, the risk of this particular use case can be considered to be *MEDIUM*. This means that the administrator of the system has to take rapid actions to patch the vulnerabilities, especially if the reasons for not updating the system are mission or cost factors.

5. IaaSecEval: Cloud Security Evaluation Architecture

In this section, we propose a unique mechanism called IaaSecEval, that can allow a user to evaluate the security of a cloud service provider before using its services.

As stated in the Introduction of this paper, by a large margin, the first hurdle to the adoption of cloud computing is security. However, to the best of our knowledge most of the work that has been done in ranking cloud providers focuses more prominently on pricing, performance, sustainability, reliability and neglects security [40], [41], [42], [43]. We felt that this is a big issue in cloud computing that needed to be addressed.

Our ultimate motivation is to guide the customer to make a choice of provider solely based on security. The reality is that the customer should not worry about the prices of cloud services because there is a war of cloud prices between the providers [44] i.e., cloud services are getting cheaper. What it comes down to is performance versus security. In this work we have opted to furnish security evaluation to the user so that he can make a wiser choice of CSP.

The CSPs can also use our proposal to evaluate the security of their platform and make decisions on, for instance, which hypervisor they should deploy as primary hypervisors and which ones they should deploy as secondary hypervisors and so on.

The architecture of our framework is depicted in **Fig. 5**. The architecture is dispatched into 13 steps, which we explain here-

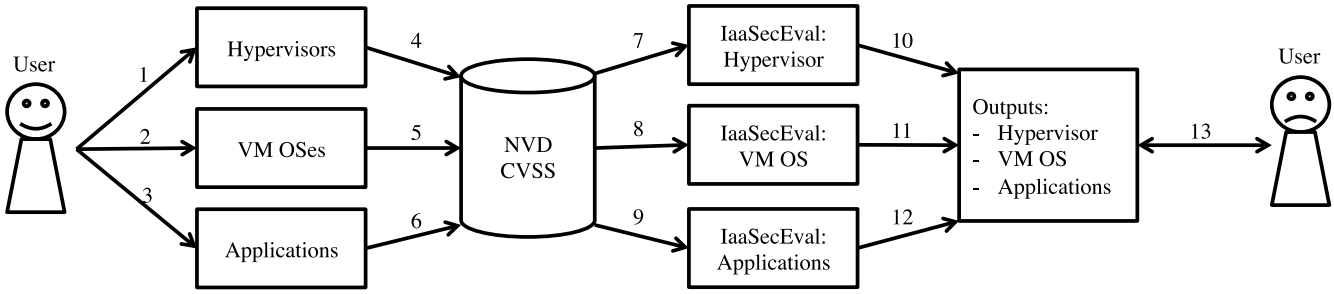

Fig. 5 IaaSecEval architecture.

CLOUD A				CLOUD B			
	CVE-ID	L	I		CVE-ID	L	I
Xen4.0.1	CVE-2014-1893	0.44	0.69	Qemu	CVE-2014-2894	0.39	1
	CVE-2014-1892	0.44	0.69	2.0.0	CVE-2014-0150	0.44	0.64
	CVE-2011-1166	0.51	0.69		CVE-2013-4544	0.44	0.64
		0.85	0.97			0.81	1
	Risk	0.8245			Risk	0.81	
Ubuntu 14.04	CVE-2014-3730	0.86	0.29	Windows 8	CVE-2014-1812	0.8	0.69
	CVE-2014-3204	0.34	0.64		CVE-2014-1807	0.39	1
		0.91	0.74			0.88	1
	Risk	0.6734			Risk	0.88	
Apache				Apache			

Fig. 6 Use case data and results.

after: Steps 1, 2, and 3 represent the users inputs. Actually, they can be done in no particular order. Once the user submits his requests, the system transfers the queries to the database (Steps 4, 5, and 6). In the database, the system executes the query to retrieve all the vulnerabilities associated with the hypervisor, the VM OS, and the Applications that the user inputted but also performs queries regarding the vulnerabilities of all the hypervisor (respectively VM OS and Applications) that are in the same family than the one that the user specified. Afterwards, the results of the queries are forwarded to the modules that are responsible of the partial evaluation (steps 7, 8, and 9). Finally, the modules in charge of the computation do their job, and forward their result to the interface of the user (steps 10, 11, 12, and 13).

We propose a use case where a user wants to use cloud services for his data. They have a choice between CLOUD A and CLOUD B and they use our architecture to compare the two cloud service providers. **Figure 6** contains the details of the use case and the final results.

6. Security Risk Quantification of OpenStack

In this section, we shed the light on the security levels of the most popular cloud management stack currently available in the market. We will elucidate the security interconnections that exist in OpenStack logical architecture.

6.1 Overview of OpenStack Security Analysis

OpenStack logical architecture [16] is made of seven main components (note that we use component instead of service to fit more into the spirit of vulnerability tree analysis). The components make use of their application programming interfaces (APIs) to communicate with each other. We use that architecture to run our security evaluation mechanism, which consists of using vulnerability trees into the different components of OpenStack. The architecture helps us understand the degree of interconnectedness that exists between the different components. The interconnectedness can compromise the security of the entire architecture as it favorites vulnerability propagation. The background of the main components has already been clarified in the previous section. We mostly used the Boolean operator OR to construct our vulnerability trees. That choice is made to give us more flexibility. The use of other Boolean operators like AND, for instance, would suggest a very strong dependency between the subcomponents, which implies that the failure of the entire component happens if and only if all the subcomponents are vulnerable. Nevertheless, we consider the Boolean operator used in our analysis to be inclusive. We made the assumption that all the components (respectively subcomponents) that have a direct connection to the Internet (the end users) are susceptible to being attacked. That assumption led to the construction of 7 vulnerability trees that we examine hereafter. Due to space limitations and the fact that the process of evaluation is similar, we only provide details of the top events for one case. A clear comprehension of Section 3 allows a better understanding of this section. Figure 2 shows a non-exhaustive list of standard fault tree symbols that we use to construct our vulnerability tree.

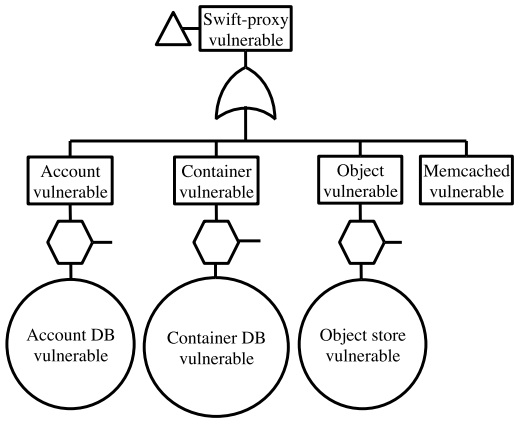

Fig. 7 Swift vulnerability tree.

6.2 Security Evaluation of Swift

Swift or OpenStack Object Store, is intrinsically composed of seven subcomponents that are named: *memcached*, *account*, *container*, *object*, *account DB*, *container DB*, and *Object DB*. The three last mentioned subcomponents are respectively bound to the three other subcomponents that precede them. The resulting vulnerability tree is described in **Fig. 7**. As Swift is attached to Keystone, the vulnerability tree can be developed further in respect to that attachment.

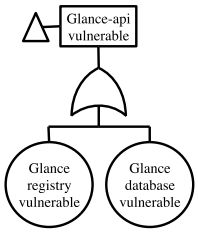

Fig. 8 Glance vulnerability tree.

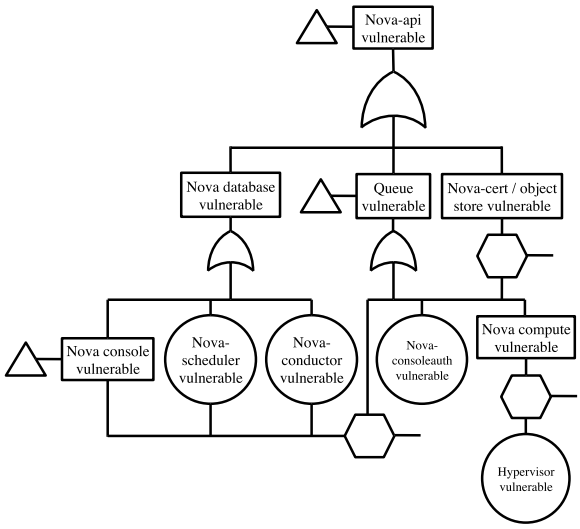

Fig. 9 Nova vulnerability tree.

6.3 Security Evaluation of Glance

Glance is very simple in its composition, consequently the vulnerability tree, which is schematized in **Fig. 8**, is easy to generate. Glance has connections with Swift, Horizon, Nova, and Keystone. As a result, the tree can be further expanded in any of those directions.

6.4 Security Evaluation of Nova

OpenStack Compute or Nova turns out to be the most complicated component of OpenStack in terms of the high level of interconnection between its contents plus the fact that it can be accessed from the Internet in two ways. We have constructed one vulnerability tree that describes the former situation. The subcomponent nova-api, which we consider as the main subcomponent, is linked to the subcomponents *nova-database*, *Queue*, and *nova-cert/objectstore*. Queue, in its turn, is linked to the subcomponents *nova-consoleauth*, *nova-scheduler*, *nova-conductor*, *nova-compute*, *nova-console*. The vulnerability tree that resumes this narrative is depicted in **Fig. 9**. We indicate that the tree can be extrapolated due the connections that Nova has with other components.

6.5 Security Evaluation of Cinder

The vulnerability tree of OpenStack Block Storage, also known as Cinder, is simple to construct and is represented in **Fig. 10**. Cinder is composed of the subcomponents *cinder-api*, *cinder volume*, *volume provider*, *cinder database* and *cinder scheduler*. The tree can be developed further as Cinder has connections with Nova and Keystone.

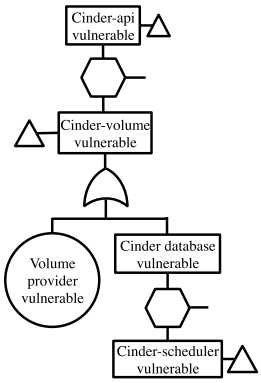

Fig. 10 Cinder vulnerability tree.

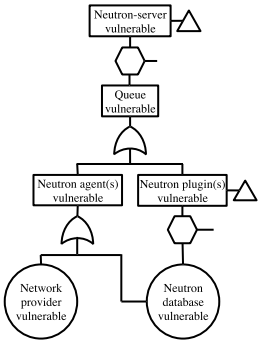

Fig. 11 Neutron vulnerability tree.

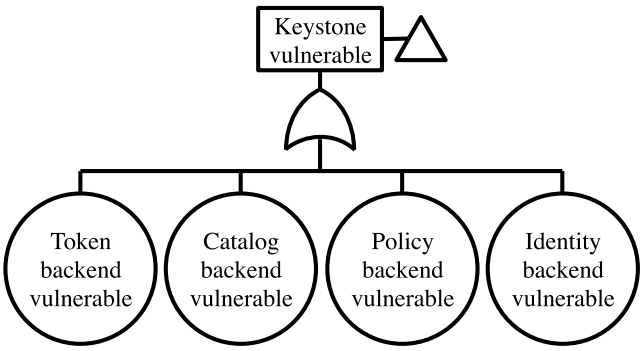

Fig. 12 Keystone vulnerability tree.

6.6 Security Evaluation of Neutron

OpenStack Network Service, codenamed Neutron, also has a simple composition that facilitates the construction of the vulnerability tree showed in **Fig. 11**. Neutron has connections with Horizon, Nova, and Keystone. Consequently, the tree can be extended further towards those components.

6.7 Security Evaluation of Keystone

Keystone, which is the security guard of OpenStack, is composed of the subcomponents *tocken backend*, *catalog backend*, *policy backend*, and *identity backend*. The vulnerability tree is described in **Fig. 12**. Keystone is connected to all the other components in that manner. The tree is subject to be developed further to accomplish a deeper analysis. We denote the top event (Keystone vulnerable) K, the basics events: Token backend vulnerable, Catalog backend vulnerable, Policy backend vulnerable, and Identity backend vulnerable, are respectively denoted K_1, K_2, K_3, and K_4. By following the details in Section 3, we are able to

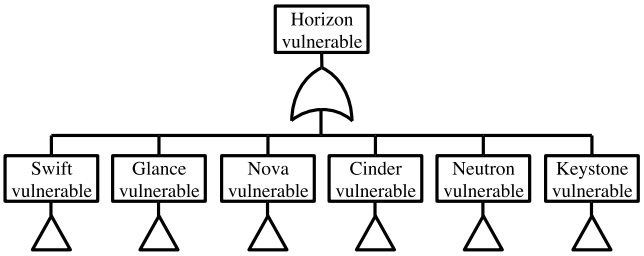

Fig. 13 Horizon vulnerability tree.

derive the security evaluation, which is given by Eq. (11).

$$\begin{aligned} P[K] = {} & P[K_1] + P[K_2] + P[K_3] + P[K_4] - P[K_1]P[K_2] \\ & - P[K_1]P[K_3] - P[K_1]P[K_4] - P[K_2]P[K_3] \\ & - P[K_2]P[K_4] - P[K_3]P[K_4] + P[K_1]P[K_2]P[K_3] \\ & + P[K_1]P[K_3]P[K_4] + P[K_1]P[K_2]P[K_4] \\ & + P[K_2]P[K_3]P[K_4] - P[K_1]P[K_2]P[K_3]P[K_4]. \quad (11) \end{aligned}$$

Let us reiterate that to compute the risk quantification for Keystone, we compute first the Likelihood and Impact by using the previous equation. With the values of the impact and the likelihood for each vulnerability derived as explained in Section 3. Unfortunately, we cannot have a use case because of the vulnerability issue (Section 7) we encountered in this study.

6.8 Security Evaluation of Horizon

Horizon or OpenStack dashboard is very intriguing because it does not have any particular subcomponent but is linked to all the other major components, which makes it one of the most critical components of the architecture. Its vulnerability tree is depicted in **Fig. 13**. All the events are deemed intermediate because they could be extended further.

6.9 Summary

Overall, the security analysis of the logical architecture of OpenStack is a daunting task. One must know the intricacies of each component and their different liaisons to effectively perform the analysis. We decided to use the Boolean operator OR to give ourselves more room to flexibly operate the security analysis but a deeper analysis of the architecture can yield a more on-the-point security analysis by using more precise Boolean operators. Some components contain subcomponents that have redundant connections with other subcomponents. That situation was hard to design in the vulnerability tree, and we forcibly have to ignore that redundancy while generating the likelihood and impact of the top event.

7. Discussion and Future Work

We have developed our proposal based on industry and consumer needs and evaluated its applicability with three different application scenarios described in Sections 4, 5, and 6. Currently, many administrators of cloud systems use the CVSS to evaluate potential reported vulnerabilities, with the resulting score helping to quantify the severity of the vulnerabilities and to prioritize their responses. They do not have a response in case of mixed, combined vulnerabilities. Our proposal is a response to these particular cases.

```
<cpe-lang:fact-ref name="cpe:/a:openstack:swift:1.4.6"/>
<cpe-lang:fact-ref name="cpe:/a:openstack:swift:1.4.7"/>
<cpe-lang:fact-ref name="cpe:/a:openstack:swift:1.4.8"/>
```

Fig. 14 Current vulnerability naming of OpenStack.

However, we do not argue that our proposal is the ultimate security solution that will solve all the security problems in IaaS cloud systems. The fact that we are only using the base metric group can be subject to discussion. In fact, in our proposal, we completely omitted the temporal and environmental metric groups. We believe that their presence in our proposal is a nonsense as we want to solely use the intrinsic score of the vulnerabilities as reported in the NVD, plus the fact that they are optional gives us a better leverage to ignore them. Nevertheless, if an organization wants to adopt our method, we recommend them to include the two omitted metrics as they can help to generate more accurate impact and likelihood values.

In Section 4, security quantification of multi-tenancy, we made a strong assumption, which is represented by our hypothesis. That assumption can be a polarizing notion. Some experts may not agree that the hypervisor plays such a pivotal role in an IaaS system. Researchers in academia and industry acknowledge that the hypervisor is responsible of the security of the virtual machines but they would not go as far as we did in our argumentation.

In Section 6, we deployed our security analysis mechanism and generated the different vulnerability trees that could allow someone to quantify the security of OpenStack depending on how many components they wish to use. One of the first issues we noticed is the complication of the interconnectedness of the components. Indeed, if they are taken individually, we can affirm that the vulnerability trees developed are sound. But when we take them collectively, we have some components that come back redundantly, hence compromising our vulnerability tree. The result of the security evaluation in this case will not be optimal because we do not really know how that redundancy is impacting the evaluation.

The other point of contention, which is also considered as a future work, is the nomenclature of the vulnerabilities. In our security evaluation, the equations depend heavily on the subcomponents; whilst the naming of the vulnerabilities in the NVD does not give any indication on which subcomponent was affected by the vulnerability. The descriptions of OpenStack vulnerabilities often only indicate the components that are vulnerable (**Fig. 14**). The naming of the vulnerabilities is effectuated by using the Naming specification of the Common Platform Enumeration (CPE) [45]. CPE is a standard that is used for the identification and the description of classes of applications, operating systems, and hardware devices. The latest version of CPE (CPE 2.3) uses the well-formed CPE name (WFN), which is an abstract logical construction, to represent the name of the classes of products. There are two methods for binding WFNs into machine-readable encodings: Uniform Resource Identifier (URI) binding and formatted string binding. URI binding is used for backward compatibility with CPE 2.2 [46]; that is why it has a monopo-

```
<cpe-lang:fact-ref name="cpe:2.3:a:openstack:swift:1.10.0:*:*:*:*:*:*:*"/>
<cpe-lang:fact-ref name="cpe:2.3:a:openstack:swift:1.10.0:memcached:*:*:*:*:*:*:*"/>
<cpe-lang:fact-ref name="cpe:2.3:a:openstack:swift:1.10.0:account:*:*:*:*:*:*:*"/>
<cpe-lang:fact-ref name="cpe:2.3:a:openstack:swift:1.10.0:account:accountDB:*:*:*:*:*:*:*"/>
<cpe-lang:fact-ref name="cpe:2.3:a:openstack:swift:1.10.0:container:*:*:*:*:*:*:*"/>
<cpe-lang:fact-ref name="cpe:2.3:a:openstack:swift:1.10.0:container:containerDB:*:*:*:*:*:*:*"/>
<cpe-lang:fact-ref name="cpe:2.3:a:openstack:swift:1.10.0:object:*:*:*:*:*:*:*"/>
<cpe-lang:fact-ref name="cpe:2.3:a:openstack:swift:1.10.0:object:objectDB:*:*:*:*:*:*:*"/>
```

Fig. 15 Proposed nomenclature for OpenStack vulnerabilities.

listic presence in the NVD. Based on these facts, the equations for each component would be 'mono-parametric'. Additionally, the equation of the entire architecture will be simpler yet hiding much information i.e., it will not be accurate. Therefore, a new way of naming OpenStack vulnerabilities is needed. The novel enumeration should take into account all the different subcomponents that compose OpenStack. As future work, we will propose the use of a nomenclature system that is adequate to our security evaluation. The formatted string binding appear to be a good choice. **Figure 15** gives a hint of what a better nomenclature for OpenStack's vulnerabilities should look like by using the formatted string binding. But, the intrinsic definition of the CPE forbids the usage of its binding methods to name a class of product in a very detailed way. That means that a new binding method is definitely needed for our proposal.

Our last discussion point revolves around the case of vulnerability masking. Indeed, one might argue that in case of a networked-system the vulnerabilities might not factor in i.e., the security evaluation is useless in that situation. That theory is true, and that is why in the introduction of Section 6, we made the assumption that only the Internet-facing components are considered in our security evaluation. In our research, an Internet-facing component is a component that has a direct connection to the Internet-there is no intermediary infrastructure like a firewall.

Finally, the architecture we considered in this work does not contain all the services of OpenStack. Indeed, Heat and Ceilometer are not part of the architecture. Consequently, we did not consider them in our security evaluation.

8. Conclusion

Cloud computing is a promising technology that needs to be protected in order to sustain or further accelerate its adoption. In that momentum, we proposed a unique mechanism to quantify security risks in cloud computing. Our solution consists of building vulnerability (impact and likelihood) trees of IaaS systems and perform an analysis that results in impact and likelihood formulas that lead to the quantified security risk. We built our vulnerability trees in regards to the rules and regulations of fault tree analysis. We showed how our method works in a multi-tenant cloud system. We made an assumption that hypothesizes that in a multi-tenant system, an attacker cannot exploit a vulnerability of a targeted virtual machine without bypassing the hypervisor. The use case that we provided represents an archetype of how to use our technique in that particular environment. Furthermore, we proposed a security ranking architecture that allows users to rank cloud service providers. The architecture allows the user to input the features of the different cloud provider they want to compare. Afterwards, the system automatically queries the vulnerabilities associated with those features and computes their risk values. The example we provided represents an epitome. We finished by applying the method on the logical architecture of OpenStack. We were able to generate security vulnerabilities for the different components of OpenStack but we were not able to properly quantify the security risk due to the current nomenclature of OpenStack vulnerabilities. We proposed the usage of a new nomenclature scheme to allow better security quantification of OpenStack and other similar software products.

Acknowledgments This work was supported by JSPS KAKENHI Grant Number 24700067.

References

[1] Mell, P. and Grance, T.: *The NIST definition of cloud computing*, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, United States Department of Commerce (2011).
[2] Takabi, H., Joshi, J.B. and Ahn, G.: Security and Privacy Challenges in Cloud Computing Environments, *IEEE Security & Privacy*, Vol.8, No.6, pp.24–31 (2010).
[3] Vaquero, L.M., Rodero-Merino, L. and Morán, D.: Locking the sky: A survey on IaaS cloud security, *Computing*, Vol.91, No.1, pp.93–118 (2011).
[4] Almorsy, M., Grundy, J. and Muller, I.: An analysis of the cloud computing security problem, *Proc. APSEC 2010 Cloud Workshop*, Sydney, Australia (Nov. 2010).
[5] Catteddu, D.: *Cloud Computing: Benefits, risks and recommendations for information security*, Springer (2010).
[6] Zhou, M., Zhang, R., Xie, W., Qian, W. and Zhou, A.: Security and privacy in cloud computing: A survey, *2010 6th International Conference on Semantics Knowledge and Grid* (*SKG*), pp.105–112, IEEE (2010).
[7] Pearson, S. and Benameur, A.: Privacy, security and trust issues arising from cloud computing, *2010 IEEE 2nd International Conference on Cloud Computing Technology and Science* (*CloudCom*), pp.693–702, IEEE (2010).
[8] Ristenpart, T., Tromer, E., Shacham, H. and Savage, S.: Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds, *Proc. 16th ACM Conference on Computer and Communications Security*, pp.199–212, ACM (2009).
[9] Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R. and Molina, J.: Controlling data in the cloud: Outsourcing computation without outsourcing control, *Proc. 2009 ACM Workshop on Cloud Computing Security*, pp.85–90, ACM (2009).
[10] OpenStack Community: OpenStack Cloud software, OpenStack Foundation (online), available from ⟨http://www.openstack.org⟩ (accessed 2014-09-10).
[11] OpenNebula Community: OpenNebula Cloud, OpenNebula (online), available from ⟨http://www.opennebula.org⟩ (accessed 2014-09-10).
[12] CloudStack Community: Apache CloudStack Software, Apache Foundation (online), available from ⟨http://www.cloudstack.apache.org⟩ (accessed 2014-09-10).
[13] Eucalyptus Community: Eucalyptus Cloud, Eucalyptus Systems (online), available from ⟨http://www.eucalyptus.com⟩ (accessed 2014-09-10).
[14] Computer Security Division: National Vulnerability Database, US National Institute of Standards and Technology (online), available from ⟨http://www.nvd.nist.gov⟩ (accessed 2014-09-10).
[15] Mell, P., Scarfone, K. and Romanosky, S.: A complete guide to the common vulnerability scoring system version 2.0, *Published by FIRST-Forum of Incident Response and Security Teams*, pp.1–23 (2007).
[16] OpenStack Community: OpenStack Logical Architecture, OpenStack Foundation (online), available from ⟨http://goo.gl/SemROL⟩ (accessed 2014-09-11).
[17] Fall, D., Chaisamran, N., Okuda, T., Kadobayashi, Y. and Yamaguchi, S.: Security Quantification of Complex Attacks In Infrastructure as a Service Cloud Computing, *Proc. 3rd International Conference on Cloud Computing and Services Science* (2013).
[18] Zhai, E., Wolinsky, D.I., Xiao, H., Liu, H., Su, X. and Ford, B.: Auditing the structural reliability of the clouds, Technical Report, Technical Report YALEU/DCS/TR-1479, Department of Computer Science, Yale University (2013), available from ⟨http://www.cs.yale.edu/homes/zhai-ennan/sra.pdf⟩.
[19] Xiao, H., Ford, B. and Feigenbaum, J.: Structural cloud audits that protect private information, *Proc. 2013 ACM Workshop on Cloud Computing Security Workshop*, pp.101–112, ACM (2013).
[20] Khan, R.H., Ylitalo, J. and Ahmed, A.S.: OpenID authentication as a

service in OpenStack, *2011 7th International Conference on Information Assurance and Security* (*IAS*), pp.372–377, IEEE (2011).
[21] Ristov, S., Gusev, M. and Donevski, A.: OpenStack cloud security vulnerabilities from inside and outside, *CLOUD COMPUTING 2013, The 4th International Conference on Cloud Computing, GRIDs, and Virtualization*, pp.101–107 (2013).
[22] Donevski, A., Ristov, S. and Gusev, M.: Security assessment of virtual machines in open source clouds, *2013 36th International Convention on Information & Communication Technology Electronics & Microelectronics* (*MIPRO*), pp.1094–1099, IEEE (2013).
[23] TaheriMonfared, A. and Jaatun, M.G.: As Strong as the Weakest Link: Handling compromised compoenents in OpenStack, *Proc. 3rd IEEE International Conference on Cloud Computing Technology and Science* (*CloudCom*) (2011).
[24] Ou, X., Boyer, W.F. and McQueen, M.A.: A scalable approach to attack graph generation, *Proc. 13th ACM Conference on Computer and Communications Security*, pp.336–345, ACM (2006).
[25] Ou, X., Govindavajhala, S. and Appel, A.W.: MulVAL: A Logic-based Network Security Analyzer (2005).
[26] Artz, M.L.: Netspa: A network security planning architecture, PhD Thesis, Massachusetts Institute of Technology (2002).
[27] Jajodia, S., Noel, S. and O'Berry, B.: Topological analysis of network attack vulnerability, *Managing Cyber Threats*, Springer, pp.247–266 (2005).
[28] Tenable Network Security: NESSUS vulnerability scanner, Tenable network security (online), available from ⟨http://www.tenable.com/products/nessus⟩ (accessed 2015-01-01).
[29] BeyondTrust Corporation: Retina network security scanner, BeyondTrust (online), available from ⟨http://www.beyondtrust.com/Products/RetinaNetworkSecurityScanner/⟩ (accessed 2015-01-01).
[30] Tripwire Company: Tripwire IP360 Vulnerability & Risk management, Tripwire (online), available from ⟨http://www.tripwire.com/it-security-software/enterprise-vulnerability-management/tripwire-ip360/⟩ (accessed 2015-01-01).
[31] Skybox Corp: Risk analytics for cyber security, Skybox security (online), available from ⟨http://www.skyboxsecurity.com/⟩ (accessed 2015-01-01).
[32] ReadSeal Inc.: Read Seal systems, ReadSeal (online), available from ⟨https://www.readseal.co/⟩ (accessed 2015-01-01).
[33] Ben-Daya, M.: *Failure Mode and Effect Analysis*, Springer (2009).
[34] Rooney, J.J. and Heuvel, L.N.V.: Root cause analysis for beginners, *Quality progress*, Vol.37, No.7, pp.45–56 (2004).
[35] Vesely, W.E., Goldberg, F.F., Roberts, N.H. and Haasl, D.F.: Fault tree handbook, Technical report, DTIC Document (1981).
[36] Sandra, M.E. Wint: An Overview of Risk, RSA Risk Commission (online), available from ⟨http://www.thersa.org/__data/assets/pdf_file/0005/286790/Risk-Commission-An-Overview-of-Risk.pdf⟩ (accessed 2014-09-10).
[37] Ellis, L.: Reliability of systems, equipment and components - Guide to fault tree analysis, British Standard (online), available from ⟨http://www.sre.org/At_Large/News/61025%20FTA.pdf⟩ (accessed 2014-09-10).
[38] Stoneburner, G., Goguen, A. and Feringa, A.: Risk management guide for information technology systems, *Nist special publication*, Vol.800, No.30, pp.800–30 (2002).
[39] Bedford, T. and Cooke, R.: *Probabilistic risk analysis: Foundations and methods*, Cambridge University Press (2001).
[40] Li, A., Yang, X., Kandula, S. and Zhang, M.: CloudCmp: Comparing public cloud providers, *Proc. 10th ACM SIGCOMM Conference on Internet Measurement*, pp.1–14, ACM (2010).
[41] Garg, S.K., Versteeg, S. and Buyya, R.: Smicloud: A framework for comparing and ranking cloud services, *2011 4th IEEE International Conference on Utility and Cloud Computing* (*UCC*), pp.210–218, IEEE (2011).
[42] Qian, H., Zu, H., Cao, C. and Wang, Q.: CSS: Facilitate the cloud service selection in IaaS platforms, *2013 International Conference on Collaboration Technologies and Systems* (*CTS*), pp.347–354, IEEE (2013).
[43] Repschlaeger, J., Wind, S., Zarnekow, R. and Turowski, K.: A reference guide to Cloud Computing dimensions: Infrastructure as a service classification framework, *2012 45th Hawaii International Conference on System Science* (*HICSS*), pp.2178–2188, IEEE (2012).
[44] Ovide, S.: A Price War Erupts in Cloud Services, The Wall Street Journal (online), available from ⟨http://goo.gl/7g5fK3⟩ (accessed 2014-09-10).
[45] Cheikes, B.A., Waltermire, D. and Scarfone, K.: Common Platform Enumeration: Naming Specification Version 2.3, *NIST Interagency Report 7695, NIST-IR*, Vol.7695 (2011).
[46] Buttner, A. and Ziring, N.: Common Platform Enumeration (CPE) - Specification Version 2.2, The MITRE Corporation - National Security Agency (2009).

Doudou Fall received his B.Sc. degree in Physics, a M.E. degree in Data Transmission and Information Security from University Cheikh Anta Diop of Dakar, Senegal in 2007 and 2009, respectively, and M.E. degree in Information Science from Nara Institute of Science and Technology (NAIST), Japan in 2012. He is currently a Ph.D. student in the Graduate School of Information Science, NAIST. His research interests include cloud computing security, vulnerability and security risk analysis.

Takeshi Okuda received his M.E. and D.E. degrees in Information Science from Osaka University, Japan in 1998 and 2011 respectively. He is currently an Associate Professor in the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. His research interests include virtual machine, virtual network and their security. He is a member of IEEE.

Youki Kadobayashi received his Ph.D. degree in Computer Science from Osaka University, Japan, in 1997. He is currently an Associate Professor in the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. Since 2009, he has also been working as an Associate Rapporteur of ITUT Q.417 for cybersecurity standardization. His research interests include cybersecurity, web security, and distributed systems.

Suguru Yamaguchi received his M.E. and D.E. degrees in Computer Science from Osaka University, Japan, in 1988 and 1991, respectively. Since 2000, he has been a Professor in the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST). He has been working aggressively for making and running JPCERT/CC since 1996, and APCERT since 2002. From 2004 to 2010, he was appointed Advisor on Information Security to the Cabinet, government of Japan. His research interests include technologies for information sharing, multimedia communication over broadband channels, large-scale distributed computing systems including cloud computing technology, network security and network management for the Internet.