NAIST-IS-DD1261205

Doctoral Dissertation

Security Quantification and Risk-Adaptive Authorization Mechanism in Cloud Computing

Doudou Fall

September 16, 2015

Department of Information Science
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Doudou Fall

Thesis Committee:

Professor Suguru Yamaguchi	(Supervisor)
Professor Kazutoshi Fujikawa	(Co-supervisor)
Associate Professor Atsuo Inomata	(Co-supervisor)
Associate Professor Yuichi Kaji	(Co-supervisor)
Associate Professor Youki Kadobayashi	(Co-supervisor)

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 environment. 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 (CVSS), 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. While being infamous for its numerous security issues, cloud computing is also infamous for being highly dynamic. The dynamicity of cloud computing implies that dynamic security mechanisms are being employed to enforce its security, especially in regards to access decisions. However, this is surprisingly not the case. Static traditional authorization mechanisms are being used in cloud environments, leading to legitimate doubts on their ability to fulfill the security needs of the cloud. We propose a Risk-Adaptive Authorization Mechanism (RAdAM) for a simple cloud deployment, collaboration in cloud computing and federation in cloud computing. We use a fuzzy inference system to demonstrate the practicability of RAdAM. We complement RAdAM with a Vulnerability Based Authorization Mechanism (VBAM) which is a real-time authorization model based on the average vulnerability scores of the objects present in the cloud. We demonstrated the usefulness of VBAM in a use case featuring OpenStack.

Keywords: Cloud Computing, risk, security quantification, vulnerability, access control, fuzzy inference*

*Doctoral Dissertation, Department of Computer Science, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1261205, September 16, 2015.

Acknowledgments

First and foremost I thank Allah for giving me the light and for enabling me to complete this task.

I owe special thanks to my supervisor Professor Suguru Yamaguchi. He did for me what the best supervisors always find a way to do: put me in a position to be successful. I thank him for accepting me in his laboratory and for igniting and sustaining my interest in research.

I am grateful to Professor Kazutoshi Fujikawa, Associate Professor Yuichi Kaji and Associate Professor Atsuo Inomata for accepting positions as members of my thesis committee.

I want to extend particular thanks to Associate Professor Youki Kadobayashi, who always encouraged me to be more creative in research, providing wisdom, a sharp critical eye, and many more insights I fully took advantages of in my thesis. I wish to thank him as well for making use of his avuncular qualities to cheer me up whenever he sensed an air of demotivation.

My thanks go to Assistant Professor Takeshi Okuda, who urged me toward this thesis topic and patiently worked with me until it became clear to both of us that I am impossible to work with. He has been more patient with me than, I once thought, I would ever be towards myself or any other human being. Throughout, he has been generous with his advice, humility and enviable erudition.

I wish to thank Marius Liviu Georgescu and Ady Wahyudi Paundu for their valuable assistance, and weekly encouragements. I am grateful to the many provocative and serendipitous questions and responses I received from them, many of which opened doors to new horizons that I have greatly enjoyed exploring. I wish to reiterate my thanks to Marius Georgescu for the many hangout/eating plans he engineered and always welcomed me to join. I wish him and his wife the best of luck in the future.

I wish to thank Christopher Michael Yap who, since he joined the lab, has had a major impact in my English writing styles. Henceforth, every single page of my manuscripts has been improved by his 'yellow felt pen'. I wish him the best of luck for his Ph.D. and his likely illustrious future career.

I wish to thank Dr. Sirikarn Pukkawanna and soon-to-be Dr. Jane Louie Fresco Zamora for their endless support and encouragements. I deeply appreciated the feminine touch they brought to the laboratory and greatly enjoyed every single bit of the crazy conversations we were having. I wish them the best of luck in their professional careers.

I would like to bow down in thanks and prose to the Internet Engineering Laboratory current and past administrative staff members — Natsue Tanida, Yukika Nishitouge, Naoko Omori, Ami Ugo, and Emiko Okamoto — for their indefatigable aid.

I wish to thank all the remaining members of the Internet Engineering Laboratory for making the laboratory such a lovely place to not only conduct, but also to enjoy research.

I owe tremendous debt of gratitude to all the friends I have made in Japan. I apologize for not listing you all. Every single one of you has been, and I hope will

continue to be, a deep source of inspiration. I would especially like to thank Gustavo Garcia and Nelson Kibinge for supporting and tolerating the character I am since 2010 and for being present whenever I needed them. I hope, wish, and pray that our friendship will last forever. I also would like to extend my thanks to Salik M.D. Parwez for being such a good friend in a very short period of time. The best is the worst thing I could wish for him.

I owe a tremendous debt of gratitude to my friends in Senegal. You have been with me since the beginning, and your support is greatly appreciated.

For my parents, brothers, and sisters, I am running out of praise. There are no words that can testify my affection for you.

I would like to bow down in thanks and prose to NAIST International Student Affair Section staff members for their indefatigable aid.

Finally, I wish to thank the Japanese Ministry of Education, Culture, Sports, Science and Technology for their wonderful scholarship which I am a recipient of.

To my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Genesis of cloud computing

Cloud computing has revolutionized the information technology (IT) world. It promotes cheap computing resources delivered through the Internet as a utility like water, electricity, gas and telephone. It is beneficial for both users, who cut cost on infrastructures, and cloud service providers, who make profits by maximizing the resources in their datacenters. Cloud computing represents the dream of the pioneers of computer science who have always wanted computing to be served as a utility. Indeed, in 1966, Douglas Parkhill published a book titled *The Challenge of the computer utility* [3], where he expressed his will to see the process happening. That early desire of the pioneers sparked the creation of mainframe time-sharing technology that is considered as the ancestor of cloud computing as users were accessing to the main computer via dumb terminals. However, as mainframes were very big thus not very practical, people quickly realized that the technology will not be used as a utility. To aggravate the situation, the development of personal computers (PC) seemed to definitely ring the end of the idea of computing as a utility. But the development of the Internet brought back the idea as Leonard Kleinrock, one of the chief scientist of the Advanced Research Projects Agency Network (ARPANET) project, predicted it in 1969 by saying: “As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of ’computer utilities’ which, like electric and telephone utilities, will service individual homes and offices across the country.” In fact, the rapid development of the Internet allows companies to propose products that could be accessed via this very large network of computers. The history of cloud computing is intertwined with the history of the Internet. Salesforce was created in 1999 and was followed by Amazon Elastic Compute Cloud (EC2), which is the first to deliver infrastructure as a service. In 2006, Eric Schmidt, the CEO of Google at that time, first coined the term cloud computing. Afterwards, a flurry of definitions was proposed but one that has proven to be generic enough to stand the test of time was proposed by the National Institute of Science and Technology (NIST) [4] of the United States of America (USA): “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.” It has become clich to list the benefits of

cloud computing, as they are diverse and well known. It is equally clich to finger-point the security issues of cloud computing that represent the major obstacles to its wide adoption by many companies, specially the financial firms who prefer to implement their own cloud thus not fully profiting of the benefits of cloud computing.

1.2 Motivation and Problem Statement

In order to reap the stated benefits of cloud computing, it is necessary to investigate the disadvantages of cloud. Indeed, despite all the benefits, users are still reluctant to adopt it and the main cause of that reluctance is the security issues of cloud computing. Indeed cloud computing has many security issues identified in numerous survey papers [5, 6, 7, 8, 9, 10]. Those security issues range from the security of the multi-tenant aspect of Cloud Computing [11] to loss of control[12]. The current cloud computing model is based on an outsourcing strategy. Essentially, customers are generally reluctant to entrust the management of their data to a third party. Clearly the problem 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. We noticed that security issues come under many guises both technical and non-technical. There are existing efforts that look to provide a taxonomy over the aforementioned issues. The Cloud Security Alliance (CSA) [13] is a non-profit organization that thrive to promote the best practises for providing security assurance within the cloud computing milieu. In their report titled "The notorious nine: cloud computing top threats in 2013" [14], they identified nine threats to cloud computing:

- **Data breaches:** cloud computing introduces significant new avenues of attack related to data breaches.
- **Data loss:** the data stored in cloud can be lost due to many reasons. The cloud service provider (CSP) can be victim of a denial of service (DoS) attack, electrical outage can happen, the cloud administrator can inadvertently delete some users' data.
- **Account hijacking:** this type of attack is very prevalent in cloud systems where most users use a web client to access their data. Methods such as phising can be used to achieve the forfeiture. Attackers can steal the users credentials to commit malicious acts in the name of the user without his knowledge.
- **Insecure API:** the usage of application programming interfaces (APIs) in the cloud and many other modern IT systems is de rigueur. The cloud users use the APIs to interconnect with the cloud services. Most of the operations between the CSP and the users are done through the APIs. Therefore, there is prescient need to make them more secure.
- **Denial of Service:** attackers can cause a slowdown in a given system and thus prevent many cloud customers to access their data in the most efficient way possible.
- **Malicious insiders:** cloud administrators can be malicious. They can steal the users data

for monetary purposes or for spying reasons.

- **Abuse of cloud service:** we know that cloud computing give customers the impression that they have unlimited computing power. Enterprises take profit of it as well as attackers. Indeed, an attacker can use the computing power of a cloud system to decipher an encryption key in a record time.
- **Insufficient due diligence:** many organizations request the services of a cloud service provider without knowing the full functioning of the cloud. An organization should be responsible enough to gauge about incident response mechanisms, encryption, security monitoring and others prerequisites before hosting their data into a certain CSP.
- **Shared technology issues:** some of the aforementioned issues are caused by this unique characteristic of cloud computing. A number of research papers have been already published demonstrating how an attacker can leverage the shared environment provided by cloud computing to achieve a malicious act.

What we are exploring here is the fact that any kind of attack that can happen in the cloud will be the result of an exploitation of vulnerabilities related to the aforementioned security issues. In practice, many vulnerabilities may still remain in a cloud environment after they are discovered, 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 by leveraging the vulnerabilities seems to be the perfect solution.

Elsewhere, the many security issues of cloud computing make people overlook another important feature that can be as damaging: its dynamism. In fact cloud computing, like most modern computer systems, is highly dynamic. That dynamicity is illustrated by the elasticity of cloud computing that allows the user to scale in and scale out in an easy manner. A user can add computing power to its VM and reduce it at will or add more storage capacity and reduce and this have to be real time because the user cannot wait to get approval from the cloud administrator. The latter also cannot deal with those types of requests coming from thousand of users. Surprisingly, traditional static access control models like role-based access control, mandatory access control or discretionary access control are used to enforce the security in cloud computing. It is not to undermine the effectiveness of the aforementioned models; after all, through time, they have demonstrated their value in early computer systems. But the cloud brings a new environment, meaning that the traditional access control have to be moved, adapted or become obsolete. The last option is what currently prevails in cloud computing thus a new type of access models, as preconized by [15], is needed.

1.3 Contributions

Our contribution is twofold.
First, in order to prevent the issues mentioned above, we contend that the best solution is to quantify security risks in cloud computing. Our first 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: $R = L \times I$. 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.

Second, to solve the issue of the non-existence of an authorization mechanism that fits the dynamism of cloud computing, we propose a risk adaptive authorization mechanism (RAdAM) which is a dynamic risk-aware authorization mechanism for cloud computing. We ascertained that RAdAM follows the principles of a risk aware access control as defined in [16]. We proposed authorization mechanisms for the most prominent cases that exist in cloud computing. We consider the situation where the user is a simple cloud customer with basic services. Afterwards, we slightly complicated the atmosphere by proposing an authorization mechanism for collaboration in cloud computing. Finally, we designed an authorization mechanism for the most complicated case in cloud computing: cloud federation. We employed a fuzzy inference system (FIS) [17] to demonstrate how RAdAM functions. The main contribution of RAdAM is not in the risk determination in itself but in the way the authorization algorithms work and the novel parameters we introduce in order to allow the authorization to be as flexible as possible. To complement RAdAM, we also proposed a vulnerability-based authorization (VBAM) mechanism which is a variation of the Bell-Lapadula multi-level security (MLS) [18]. The most important point of VBAM is the usage of the vulnerability score of the objects to enforce a decision. We contend that modern authorization mechanisms should not only be dynamic, but also should be tailored to the vulnerabilities of the objects that are being accessed.

1.4 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2 : We briefly introduce cloud computing in this chapter and explain some fundamental notions like multi-tenancy, the economic aspect of cloud computing, the enabling technologies. We also performed a vulnerability analysis of OpenStack in order to confirm the security issues listed within this chaper as well as showcasing new security issues. The

purpose of this chapter is to provide the reader enough background information to understand the research landscape we are working on.

Chapter 3 : This chapter contains the details of our security quantification methodology.

Chapter 4 : Our second main contribution is described in detail in this chapter.

Chapter 5 : The future work is detailed in this chapter. Besides extensions of the work we accomplished in this dissertation, we propose different distinct projects that can be good avenues towards the securisation of cloud computing in particular, and computer systems in general.

Chapter 6 : This chapter concludes the thesis. We did a recapitulation of the work we have done by giving brief summaries of the different chapters.

Chapter 2

In-depth Understanding of Cloud Computing

In this Chapter, we will explain the background information required for understanding this thesis. The explanation of cloud computing given in Chapter 1 will be extended with futher technical details and clarifications. Multi-tenancy, the biggest aspect of controversy in cloud computing, will be explained from a theoretical/technical perspective including the different degrees of multi-tenancy that are possible.

2.1 Cloud Computing

Cloud computing is considered as the "fifth utility" after water, gas, electricity, and telephone [19]. It employs the same functioning mode where users pay their services as they go. In cloud, users have a panel of choices going to fully featured applications like Customer Relationship Management (CRM), to environments for software development and deployment, plus computing resources like storage, network bandwidth, and processing.

2.1.1 The many definitions of cloud computing

As stated in the introduction, the main idea behind cloud computing is not new. The pioneers of computer science have been advertising it since the 1960s. John McCarthy gave an early definition of the cloud as:"if computers of the kind I have advocated become computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry."

The term cloud computing was first coined in 2006 by the then Chief Executive Officer (CEO) of Google Erik Schmidt, and afterwards, as a domino effect, it started to gain massive popularity. The standard definition of cloud computing revolves around the one given by NIST [4], but there still exist other definition that can rival the NIST definition of cloud computing. Buyya et al. [19] provide the following definition: "A cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned

and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers."

Armbrust et al. [20] gave a more or less superficial definition of cloud computing as:

- The illusion of infinite computing resources available on demand
- The elimination of an up-front commitment by cloud users
- The ability to pay for use of computing resources on a short term basis as needed.

Despite the seemingly lack of consensus on the definition of cloud computing, all the researchers seem to agree on two characteristics: on-demand self service and rapid elasticity.

2.1.2 Characteristics

On-demand self-service: the cloud system is automated such that a consumer can conveniently make use of as many computer resources as he/she wants without any human interaction with the cloud service provider.

Broadband network access: standard network mechanisms are used to make the computing resources available online and accessible via different devices.

Resource pooling: multi-tenancy is the key technology to the economical benefits of cloud. The resources of the consumers are aggregated in the same physical infrastructure and separated logically. The resources vary from storage, processing, memory, to network bandwidth.

Rapid elasticity: this concerns the capabilities of a cloud service provider to quickly scale-in/out any computing resources.

Measured service: a metering capability is linked to each type of service provided by a CSP. For the needs of transparency, the resource usage can be monitored, controlled and reported.

2.1.3 Service Models

Cloud computing has several types of services offerings categorized as either Software, Platform, and Infrastructure as a Service. We briefly explain these three models as follows:

Software as a Service (SaaS): in this model, a consumer uses web-based applications running on a cloud infrastructure such as webmail or customer relationship management programs. Except for some specific application settings under the responsibility of the consumer, the management of the underlying infrastructure is completely dependent on the cloud service provider (CSP).

Platform as a Service (PaaS): in this model, a CSP offers a platform to deploy and develop applications using specific programming languages. The CSP is responsible for the under-

lying cloud infrastructure, while the consumer manages his/her applications.

Infrastructure as a Service (IaaS): in this model, a CSP provides necessary computing resources, generally in form of virtual machines, to the consumers who can deploy a system as they according to their needs and run their applications on it. The CSP has control over the underlying cloud infrastructure. Whereas, the consumer is responsible for the management of the operating systems, storage, deployed applications, and some network components.

2.1.4 Deployment Models

Along with the different service models present in Cloud Computing, there also are different deployment models. In this subsection, we will discuss the deployment models (*Private*, *Community*, *Public*, and *Hybrid* clouds) proposed by NIST [4].

Private Cloud: this type of cloud is generally deployed within a single organization with (or without) multiple cloud users. The organization, a CSP, or a combination of both can conduct the management of this type of cloud. The resources might be located inside the organization's facilities or off-premises.

Community Cloud: this deployment model is for a specific community of consumers that have shared concerns. The management can be done by one or more organizations in the community, entrusted to a third party, or a combination of the two. Similar to private cloud, the resources may be on-premises or off-premises.

Public Cloud: is the most popular deployment model. There are no restrictions on who can use the services as it is open to everybody and exists off-premises (i.e. on the premises of the cloud provider). The ownership can be governmental, academic, commercial, or a combination of these three entities.

Hybrid Cloud: this deployment model represents a mixture of two or more distinct cloud infrastructures.

2.2 Virtualization

Contrary to what many people may think, virtualization is not a new technology. It has been around since the 1960s and has become extremely evolved. However, despite being considered as an old technology, virtualization or more specifically server virtualization has been playing an important role in cloud computing, and more precisely, infrastructure cloud. In server virtualization, a virtualization software is used to create and manage multiple Virtual Machines (VMs) on a single server. The VMs function like normal physical machines. The virtualization software is called hypervisor, or virtual machine monitor (VMM) and is responsible of the abstraction of the hardware to the individual virtual machines.

There exist two types of hypervisor (Figure 2.1):

- Type 1 hypervisor: runs directly over the hardware. It is also known as bare metal or native.
- Type 2 hypervisor: runs over an operating system. It is also known as hosted hypervisor.

The hypervisor manages the VMs by providing security isolation as well as access to the shared resources in the physical server (CPU, memory, I/O). In the quasi-general case, a VM runs an operating system and applications which are not aware of the type of environment they are running in (virtual).

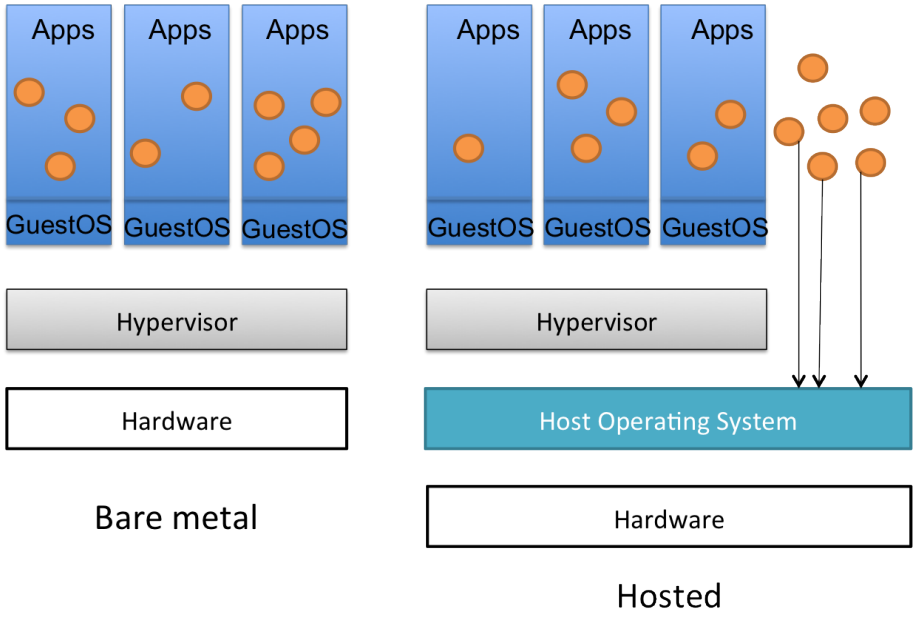

Figure 2.1: Different types of hypervisors in virtualization

2.3 Shared Resources in the Cloud: Multi-tenancy

At the beginning of cloud computing, multi-tenancy was considered as a feature unique to it. However, that was not the case, multi-tenancy has existed since the mainframe era where people were using terminals to gain access to the mainframe. With the rise of personal computer, it fell into obscurity. However with cloud computing, it is experiencing a rebirth.

Multi-tenancy is implemented in the cloud to reduce the cost of delivering the same application to many different sets of users. It is considered as the most direct path to cost minimization/cost maximization from a cloud application. A multi-tenant application is able to fulfill the needs of several tenants by using the hardware resources necessary to manage a single software instance. Tenants in a multitenant platform are separated logically and each of them can use and modify an application related to his individual needs.

Cloud computing is commonly comprised of three services models: SaaS, PaaS, and IaaS. Therefore, we can naturally deduce three level of multi-tenancy: application level, middleware level, and hypervisor level. In our research, we mainly focus on the hypervisor level multi-tenancy. Multi-tenancy is defined as a continuum between isolation and sharing and can be divided in three approaches as depicted in Figure 2.2 .

Separate database: In this approach, each tenant has his/her own database where his/her data is stored, but he/she shares computation resources and application code with other tenants. The key benefits of this approach are that it is easy to meet tenants' individual needs and to restore tenants' data from backup in case of failure. Conversely, the drawback of this approach is that it is costly in terms of hardware and in terms of data backup.

Shared database, separate schema: several tenants are sharing the same database but each of them has his/her own schema, which is a set of tables. This approach is easy to implement and has the same benefits as the previous approach. But the disadvantage is that tenants' data cannot be recovered in the event of a failure because the backup is made at the database level and recovering from the backup implies completely overwriting the data of all the tenants, as opposed to just the affected tenants.

Shared database, shared schema: in this approach, the tenants are using the same database and the same set of tables. A table can contain records from multiple tenants identified by the tenants' ID. Among the three approaches, this one is the most cost-effective, but it is also the least secure. Futhermore it experiences the same problem of data backup similar to the second approach.

Each of these approaches appeals to some aspects of economic benefit and security, but each of these approaches is still based on a shared environment with different degrees of multi-tenancy.

2.4 Economics of cloud computing

Elasticity: like cloud computing, there is not a consensus on the definition of elasticity. In physics it is defined as the property of a material to be able to return to its initial state after a deformation. Herbs et al. [21] define elasticity in cloud computing as: "It is the degree of flexibility to which a system is able to adapt to workload changes by provisioning resources in a automatic manner, such that at each point in time the available resources match the current demand as closely as possible." Elasticity comes with some prerequisites that lie between the boundaries of the scalability of a cloud system including hardware, virtualization, and software layers. The following points factor in when evaluating elasticity in cloud computing: automatic scaling, elasticity dimensions, resource scaling units, and scalability bounds. Thanks to elasticity, the economic appeal of cloud computing is labeled as "converting capital expenses to operating expenses" (CapEx to OpEx). Elasticity in cloud computing allows customers to add or remove resources at fine grain and with

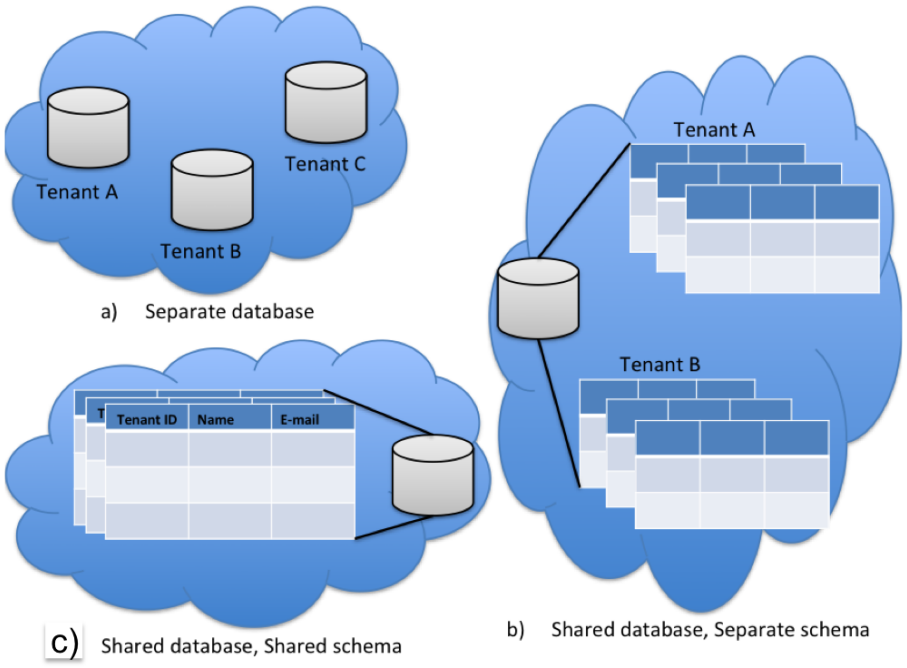

Figure 2.2: Different degrees of multi-tenancy

efficiency timing. It allows customers to match resources to workload more closely, hence mitigating the risk of overprovisioning or underprovisioning. The speed and precision on which the resources are scaled up/down are core aspects of elasticity.

Cost of the cloud: One of the main promise of cloud computing is the maximization of both the cloud service provider and the consumers. Figure 2.3 shows that that cloud computing can effectively help consumers save money. Additionally, there is a price war between the main cloud service providers Amazon, Google, Microsoft, which means the prices will continue to profit the consumers. In the cloud service provider side, whether or not they are making money depends on the Total Cost of Ownership (TCO). Indeed, many parameters factor in when it comes to evaluating the profits of a cloud provider. One of the main issue is the price of electricity. Data centers consume as much electricity as a developed country like France. An increase of the price of electricity can have a negative impact in a cloud service providers profit. But as they keep lowering the prices for their services, we can only assume that they are making profits.

Furthermore, cloud computing created new jobs in the market as shown in Figure 2.4 [22].

2.5 Cloud adoption obstacles

Availability of service: Cloud computing services are mainly delivered through the Internet. Prospective customers worry about whether the cloud service provider will have adequate

	Internal IT	Managed Services	The Cloud
Capital Investment	$40000	$0	$0
Setup Costs	$10000	$5000	$1000
Monthly Services	$0	$4000	$2400
Monthly Labor	$3200	$0	$1000
Cost over Three Years	$149000	$129000	$106000
Savings gained	0%	13%	29%

Figure 2.3: Cost comparison between internal IT, managed services and the cloud

availability. Cloud service providers and Internet service provider must collaborate in order to make sure there will no interruption of the connection. This is an important issue especially in regards to countries where Internet does not have a wide spread deployment. Cloud computing adoption will be even slower in those countries to date. Another lingering issue in the availability of services is power outage. In recent years, we have witnesses a dozen of power outages that hit different cloud service providers. Hence to ensure service availability, there should be a good management of the power grid.

Data lock-in: Customers can find themselves 'imprisoned' into a cloud service provider. It might difficult to extract the customers data from one cloud to another. A poignant example is users of the iPhone. It is impossible to transfer your personal data from an iPhone to an android device. As cloud is maturing, the experts are finding solution through standardization and policy management. Plus the development of different cloud deployment modes like cloud federation allows people to move their data freely between different cloud service providers.

Data confidentiality and auditability: This is the most obvious obstacle and lies directly into the motivation to pursue this thesis. Once the mechanism of the functioning of cloud computing is understood, customers immediately complain about the fact that their data will be managed by a third party. They might know the cloud service provider corporate name but they have to trust them and believe that they will not tamper with their data. Most companies with sensitive data adamantly refuse to use the services of a public cloud service provider because of this fact. Instead, they prefer building their own private cloud within their organizations boundaries. With the advent of cloud computing, the three principles of security: confidentiality, integrity, and availability; have stated the process of obsolescence. To add fuel to the fire, sometimes, users might not know the location of their data.

	Manufacturing		Trade		Services, hotels..		Transport&Com.		Finance&other		Total	
	Slow	**Fast**	**Slow**	**Fast**	**Slow**	**Fast**	**Slow**	**Fast**	**Slow**	**Fast**	**Slow**	**Fast**
Belgium	1002	5046	1004	5056	273	1376	387	1950	885	4457	***3589***	***18073***
Czech Rep.	4181	21055	2144	10796	489	2464	1042	5245	1546	7783	***9578***	***48228***
Denmark	854	4303	941	4736	215	1084	399	2010	800	4027	***3244***	***16336***
Germany	14634	73686	9848	49588	2708	13637	4048	20381	9188	46263	***40994***	***206418***
Ireland	544	2738	785	3955	367	1848	227	1144	550	2768	***2495***	***12565***
Greece	906	4562	2188	11017	688	3463	533	2686	761	3834	***5132***	***25839***
Spain	4265	21474	5531	27849	2074	10443	1735	8736	4514	22731	***18233***	***91810***
France	7529	37912	6833	34408	1884	9488	3187	16048	6882	34651	***26717***	***134530***
Italy	6595	33208	4962	24984	1607	8093	1783	8977	4037	20330	***19150***	***96425***
Hungary	1121	5646	841	4234	183	921	382	1923	696	3504	***3301***	***16620***
Netherlands	1603	8071	2847	14337	710	3575	978	4924	3292	16575	***9479***	***47731***
Austria	1028	5178	1017	5123	399	2012	398	2005	687	3461	***3582***	***18039***
Poland	8000	40283	6916	34822	715	3598	2351	11837	2992	15067	***21604***	***108784***
Portugal	1623	8174	1688	8500	540	2718	379	1907	1186	5973	***5463***	***27507***
Romania	3195	16089	1990	10019	250	1259	782	3937	880	4432	***7367***	***37097***
Slovakia	1358	6836	631	3176	71	360	341	1718	371	1867	***2899***	***14600***
Finland	1088	5479	711	3581	147	742	431	2168	600	3020	***3020***	***15208***
Sweden	1808	9104	1361	6851	282	1418	697	3508	1309	6592	***5525***	***27818***
UK	12933	65119	19577	98574	7932	39939	6425	32350	19591	98648	***67020***	***337464***
EU-27	**78592**	**395734**	**74373**	**374490**	**22305**	**112311**	**28077**	**141375**	**62572**	**315069**	***269765***	***1358350***

Creation of new jobs after a year from the beginning of adoption

Legenda: Slow stands for slow adoption of the new technology, while Fast stands for fast adoption

Figure 2.4: Net job creation of cloud computing

To further make more benefits out of the cloud, some cloud services providers build data centers in place where the overall cost is cheaper but where governments can, at any time, cease the customers data.

Performance unpredictability: The business model of cloud computing lies on multi-tenancy or the fact that users data are stored in the same physical platform hence they use the same computing resources. The cloud service provider must ensure that their hardware features: CPU, memory, I/O; are powerful enough to be able to be shared by multiple customers.

Bugs in large distributed systems: cloud computing systems are very large systems, hence prone to bugs and other vulnerabilities that can remain in the system undetected or even when they are detected, the cloud administrators do not patch them due to either situational factor or cost factors.

Reputation fate sharing: cloud computing is inherently multi-tenant. Customers are unaware of each other, thus a legitimate user may have their data hosted in the same physical infrastructure as a notable hacker. Ristenpart et al. [11] already demonstrated that it was possible to steal data from neighboring tenants in the cloud. Also, the bad behavior of a user in general can affect the reputation of the cloud service provider; others might perceive it as the cloud service provider is not doing a job to preserve its image. The reputation of a cloud provider can falter when they are found guilty of collaborating with the government by regularly handing them users data.

Software licensing: Formerly, the software licenses were commonly restricted to the computers on which the software is running. Users purchase the software and additionally pay a maintenance fee. Cloud computing changes this business model drastically and as it becoming more popular, and due to the fact most cloud providers were using open source software, the commercial are obliged to move, adapt or disappear. They have to restructure their licensing deals if they want to survive in this era of cloud computing.

2.6 Cloud Management Frameworks

Cloud management frameworks are groups of software that permit to ease the management of a cloud infrastructure. Figure **??** contains the taxonomy of a cloud management stack. Dukaric et al. [1] provided a taxonomy of cloud management frameworks by defining six quintessential layers that they should feature:

Resource abstraction layer: it comprises the compute, storage, volume and networking component. The *compute* component is in charge of the provisioning and management of virtual machines and physical hosts. The *storage* component provides a scalable and redundant object store. The *volume* component eases the management of persistent, block-level storage volumes, as the virtual machine instances do not provide persistent storage. The *network* component is responsible of managing networking tasks and performing system commands to manipulate the network.

Core service layer: it is made of five components, i.e. identity service, scheduling, image repository, charging and billing, and logging. The *identity service* component provides authen-

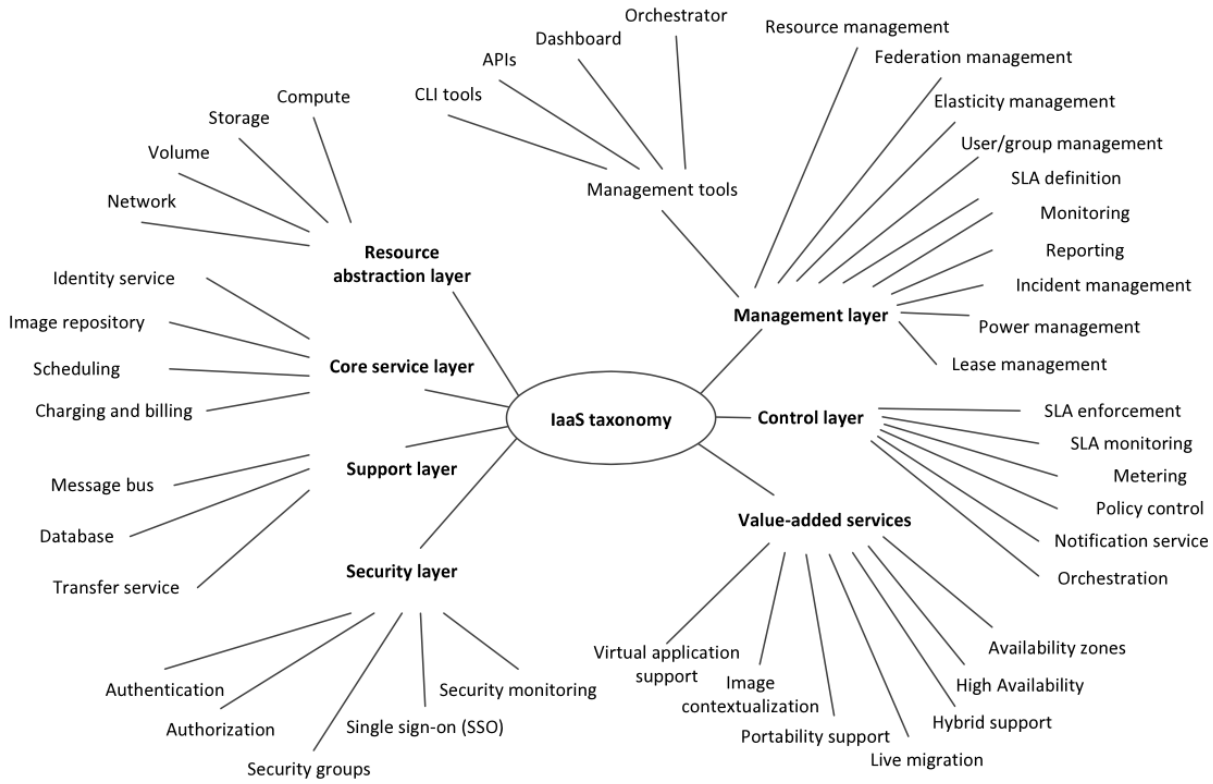

Figure 2.5: Unified taxonomy of cloud management stack architectures [1]

tication and authorization services for different cloud components; it also helps with the management of the users. The *scheduling* component is responsible fitting the virtual machine instances to their respective servers/hosts, which consequently lead to an efficient usage of the physical resources. The *image repository* provides a catalog service for storing and querying virtual disk images. The *charging and billing* component keeps track of the user activity and creates the bills. The *logging* component is mainly for auditing. It should make it impossible for any user to use the cloud without leaving any traces.

Support layer: it is the middleware layer. It is responsible for the communication between the other layers. It is comprised of the message bus, database and transfer services. The *message* bus acts as the main conveyor of messages between the cloud services. The *database* component stores the configuration and runtime state of cloud infrastructures like virtual machine instances, networks available, security users, etc. *Transfer services* is responsible of the transfer of the images of the virtual machines to the cluster nodes to allow the execution of the virtual machines without a degradation of performance.

Security layer: it is made of five components, i.e. authentication, authorization, security groups component, single sign-on and security monitoring. The *authentication* component mainly feature token-based access and uses well-known mechanism such as LDAP, SSH keypairs, Kerberos or X509 certificates. It is used for the authentication of the cloud users and

also for the provisioning of virtual machines. The *authorization* component is used to determine access decisions of the users when they attempt to perform specific tasks. The *security group* component contains sets of firewall rules that are applied to virtual machine instances linked with a particular group. The *single sign-on* component pairs with a technology accompanied with a set of standards named federated identity, as the cloud is multi-resources, identities should be shared securely across all the resources. The *security monitoring* provides the traditional network monitoring set of tools to the cloud.

Management layer: it is composed of eleven components, i.e. management tools, federation management, elasticity management, resource management, user and group management, SLA definition, reporting, monitoring, incident management, power management and lease management. *Management tools* is in charge of specifying application programming interfaces (APIs), tools, graphical user interfaces and a cloud orchestrator for interacting with underlying services and resources. The *federation management* component is responsible of putting together several cloud services with an SSO mechanism. The *elasticity management* component takes in charge the automation and dynamic provisioning of resources; all according to user-defined policies. *SLA definition* is primordial as a contract between the a consumer and a provider. A service level agreement (SLA) is a document that describes the agreed services. The *reporting* component generates cost, usage, and comparison reports for different users. *Power management* gives the cloud administrators the control over power management by providing features related to that service. The *resource management* incorporates virtual machine management (creation, suspension, and termination) and host management. The *user and group management* component features the addition and removal of users and groups, and the addition of users to a particular group. *Monitoring* should be able to vover wide cloud ecosystems. *Incident management* ensures that the cloud ecosystem can be resumed in an efficient timing in case of a disruption of a service. Finally, the *lease management* component is responsible of granting leases on resources to users.

Control layer: this layer is comprised of the components SLA enforcement, SLA monitoring, metering, policy control, notification service and orchestration.*SLA monitoring* specifies the prerequisites by which the service parameters can be monitored and the monitoring format. The *SLA enforcement* component ensures that the agreed quality of service (QoS) parameters are respected to best. The *metering* component provides metering capabilities to measure, analyze and report on resources used in the cloud environment. *Policy control* represents a paraphernalia of policies that cover wide ranges from quota to security and privacy control. The *notification* component allows the sending of notifications to the different stakeholders of the cloud. *Orchestration* provides a catalog of workflows that permit automated, configurable processes to manage the cloud and third-party technologies.

2.7 Security Issues in Cloud Computing

We have already listed the main security issues that hinder the development of cloud computing. We reiterate that the CSA published a work titled “the notorious 9” where they classify the se-

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
2011	1							1							
2012	23	3	1				1	1		5	1				
2013	42	11		5			1			7	9	2			
2014	55	13	2				7			6	14	6			
2015	12	4					1	1		1	2				
Total	133	31	3	5			10	3		19	26	8			
% Of All		23.3	2.3	3.8	0.0	0.0	7.5	2.3	0.0	14.3	19.5	6.0	0.0	0.0	

Figure 2.6: OpenStack vulnerabilities: number, types [2]

curity issues in the cloud as: data breaches, data loss, account hijacking, insecure API, denial of service, malicious insider, abuse of cloud service, insufficient due diligence, and shared technology issues. In this section, we want to demonstrate that, except *insufficient due diligence*, all the other security issues are represented in form of vulnerabilities available in the diverse vulnerability databases. Our modus operandi is to consider of the most famous cloud framework stack, and then conduct a vulnerability analysis. We choose to work with OpenStack, which, by far, has the most significant number of vulnerabilities among the cloud framework stacks. A simple query of the OpenStack vulnerabilities on the website www.cvedetails.com, returned the table shown in Figure 2.6. At first glance, we notice that the number of vulnerabilities augment every year, thus we can conclude that, in general, the number of vulnerabilities in cloud computing keeps increasing. Most importantly, we can say that all the security issues listed in the "notorious 9" are represented in this vulnerability dataset. We notice that DoS and information gain are very prevalent in the cloud. By analyzing the dataset, we have also discovered some security issues that were not addressed before, to the best of our knowledge.

Remote access: we are surprised to see that most (over 90%) of the vulnerabilities can be exploited by a remote attacker. The common knowledge in cloud computing is that most of the attacks are caused by legitimate consumers who take advantage of the shared platform to attack the other tenants.

Vulnerabilities related to other vendors: this is also an issue that researchers have not thought about. By building extensions for some cloud software, the vendors introduce new vulnerabilities. The following vulnerability descriptions showcase the situation. CVE-20148333: *The VMware driver in OpenStack Compute (Nova) before 2014.1.4 allows remote authenticated users to cause a denial of service (disk consumption) by deleting an instance in the resize state*. CVE-2014-3632: *The default configuration in a sudoers file in the Red Hat openstack-neutron package before 2014.1.2-4, as used in Red Hat Enterprise Linux Open Stack Platform 5.0 for Red Hat Enterprise Linux 6, allows remote attackers to gain privileges via a crafted configuration file. NOTE: this vulnerability exists because of a CVE-2013-6433 regression.*

Inter-component vulnerabilities: We also found that there were vulnerabilities that were shared by many components. This can jeopardize an entire cloud environment as an attacker can

exploit the same vulnerability in different components. CVE-2014-3473: *Cross-site scripting (XSS) vulnerability in the Orchestration/Stack section in the Horizon Orchestration dashboard in OpenStack Dashboard (Horizon) before 2013.2.4, 2014.1 before 2014.1.2, and Juno before Juno-2, when used with Heat, allows remote Orchestration template owners or catalogs to inject arbitrary web script or HTML via a crafted template*. CVE-2014-7231: *The strutils.mask_password function in the OpenStack Oslo utility library, Cinder, Nova, and Trove before 2013.2.4 and 2014.1 before 2014.1.3 does not properly mask passwords when logging commands, which allows local users to obtain passwords by reading the log*.

Chapter 3

Security Risk Quantification in Cloud Computing

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.

3.1 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 [4] (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 [5, 6, 7, 8, 9, 10]. These security issues range from the security of the multi-tenant aspect of cloud computing [11] to the loss of control [12].

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 work, 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-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 multi-tenancy 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 [23], OpenNebula [24], CloudStack [25], or Eucalyptus [26]. 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) [27]. 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) [28]. Figure 3.1 gives a visual description of the former statement. Furthermore, the logical architecture of OpenStack [29] 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.

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: $R = L \times I$. 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 this Chapter is structured as follows. Section 3.2 is about the related work. In Section 3.3, we detail our proposal. Section 3.4 contains a proof-of-concept of how our method works in a multi-tenant system. Section 3.5 is entirely dedicated to the ranking platform. In section 3.6, we showcase the security analysis of the logical architecture of OpenStack. In Section 3.7, we propose a discussion of our findings and give a hint of our future work. Section 3.8 concludes the Chapter.

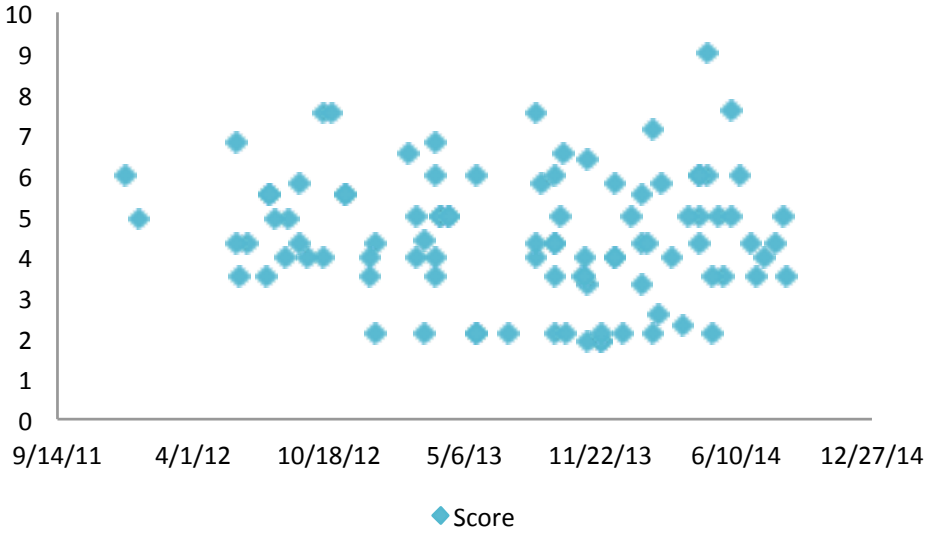

Figure 3.1: OpenStack presence in the NVD

3.2 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. [30], 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. [31] 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. [32] fixed the issue by adding a privacy aspect to the SRA system. They re-engineered the 3-step process of [31]'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. [33] 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. [34] 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 (Windows) was more subject to vulnerabilities than the others. In the same spirit, Donevski et al. [35] 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. [36] 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 [37]. 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 [38] (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 [39] (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) [40] 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 [41], Retina [42] and Tripwire IP360 [43] are well-known network vulnerability scanners that provide vulnerability management, vulnerability analysis and network protection. Skybox security [44] and Red Seal Systems [45] 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) [46] and Root Cause Analysis (RCA) [47] compete with Fault Tree Analysis (FTA) [48] on modeling failures in a given system. We prefer FTA because we contend that it is more suitable for our research.

3.3 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 [49]. 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 [50]. 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" [51].

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 Equation 3.1.

$$Risk = Likelihood \times Impact. \tag{3.1}$$

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.

3.3.1 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. [11] 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

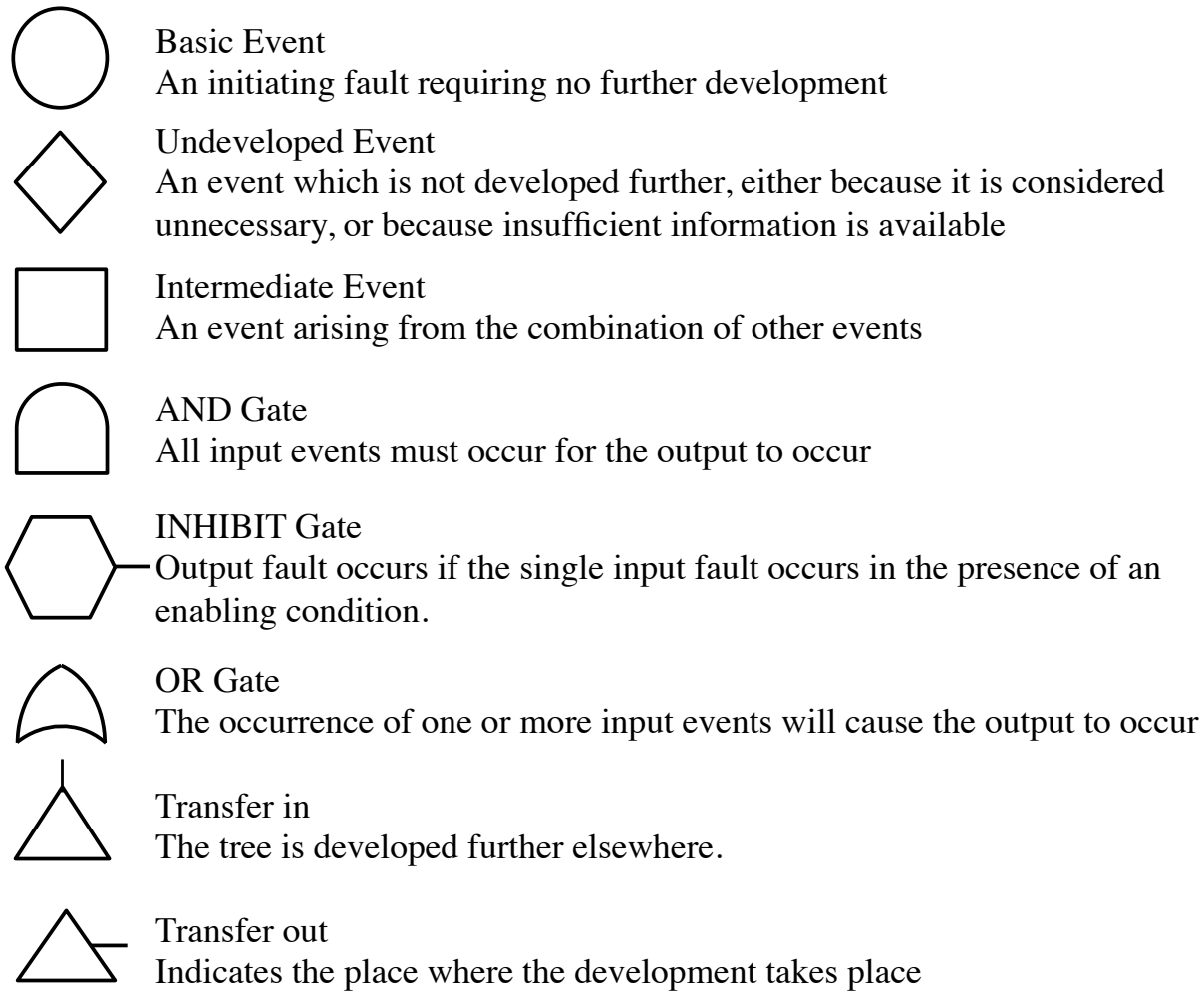

Figure 3.2: List of Standard Fault Tree Symbols

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.

3.3.2 Fault Tree Analysis

A fault tree [48, 52] 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 Figure 3.2. 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 *AND* and *OR* gates. Hereafter, we provide some definitions that are necessary for a better comprehension of fault tree analysis.

Definition 3.1 *: A cut set is a collection of basic events such that if these events occur together then the top event will certainly occur.*

Definition 3.2 *: 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 3.3 *: 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 3.4 *: 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, ..., C_n. From the aforementioned definitions, we know that T is a set composed of all the possible cut sets of the fault tree:

$$T = C_1 \cup C_2 \cup ... \cup C_n. \tag{3.2}$$

By applying the inclusion-exclusion law of probability [52] to Equation 3.2, we obtain Equation 3.4:

$$P[T] = P(C_1 \cup C_2 \cup ... \cup C_n). \tag{3.3}$$

$$P[T] = \sum_{i=1}^{n} P[C_i] - \sum_{i<j} P[C_i \cap C_j] + ... - (-1)^{n+1} P[C_1 \cap C_2 \cap ... \cap C_n]. \tag{3.4}$$

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

$$I[T] = \sum_{i=1}^{n} I[C_i] - \sum_{i<j} I[C_i \cap C_j] + ... - (-1)^{n+1} I[C_1 \cap C_2 \cap ... \cap C_n]. \tag{3.5}$$

$$L[T] \quad = \quad \sum_{i=1}^{n} L[l_i] \; - \; \sum_{i<j} L[l_i \; \cap \; l_j] \; + \; ... \; - \; (-1)^{n+1} L[l_1 \; \cap \; l_2 \; \cap \; ... \; \cap \; l_n]. \quad (3.6)$$

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.

The process of building a vulnerability tree is similar to the process of constructing a fault tree. It begins with a top event which is the attack of the cloud environment. We develop the tree by finding necessary and sufficient vulnerabilities of which their exploitation leads to the top event. In a simple case, all the vulnerabilities are somewhat independent and we use to Boolean logic OR to interpret their relationship. The situation can be more complicated with some dependencies in the tree as we show in Section 3.4.

3.3.3 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) [28] 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:

$$Impact = 10.41 \times (1 - (1 - ConfImpact) \times (1 - IntegImpact) \times (1 - AvailImpact)) \quad (3.7)$$

$$Exploitability = 20 \times AccessVector \times AccessComplexity \times Authentication \quad (3.8)$$

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 Equation 3.9 and Equation 3.10.

$$Impact = 1.041 \times (1 - (1 - ConfImpact) \times (1 - IntegImpact) \times (1 - AvailImpact)) \quad (3.9)$$

$$Likelihood = 2 \times AccessVector \times AccessComplexity \times Authentication \quad (3.10)$$

In this research, we consider an impact tree and a Likelihood tree. The analysis of the aforementioned trees lead to the Equations 3.5 and 3.6. Precious information about the parameters used in the CVSS equations are displayed in Figure 3.3. 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.

```
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
```


Figure 3.3: CVSS Parameter Details

Table 3.1: 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

3.4 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 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 3.1 *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 3.1 with their impact and likelihood values derived from Equation 3.9 and Equation 3.10 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 Figure 3.4. In the following, we expose the different vulnerabilities associated with the cloud system.

– xen-4.1: **CVE-2011-1898**
"Xen 4.1 before 4.1.1 and 4.0 before 4.0.2, when using PCI passthrough on Intel VT-d chipsets that do not have interrupt remapping, allows guest OS users to gain host OS

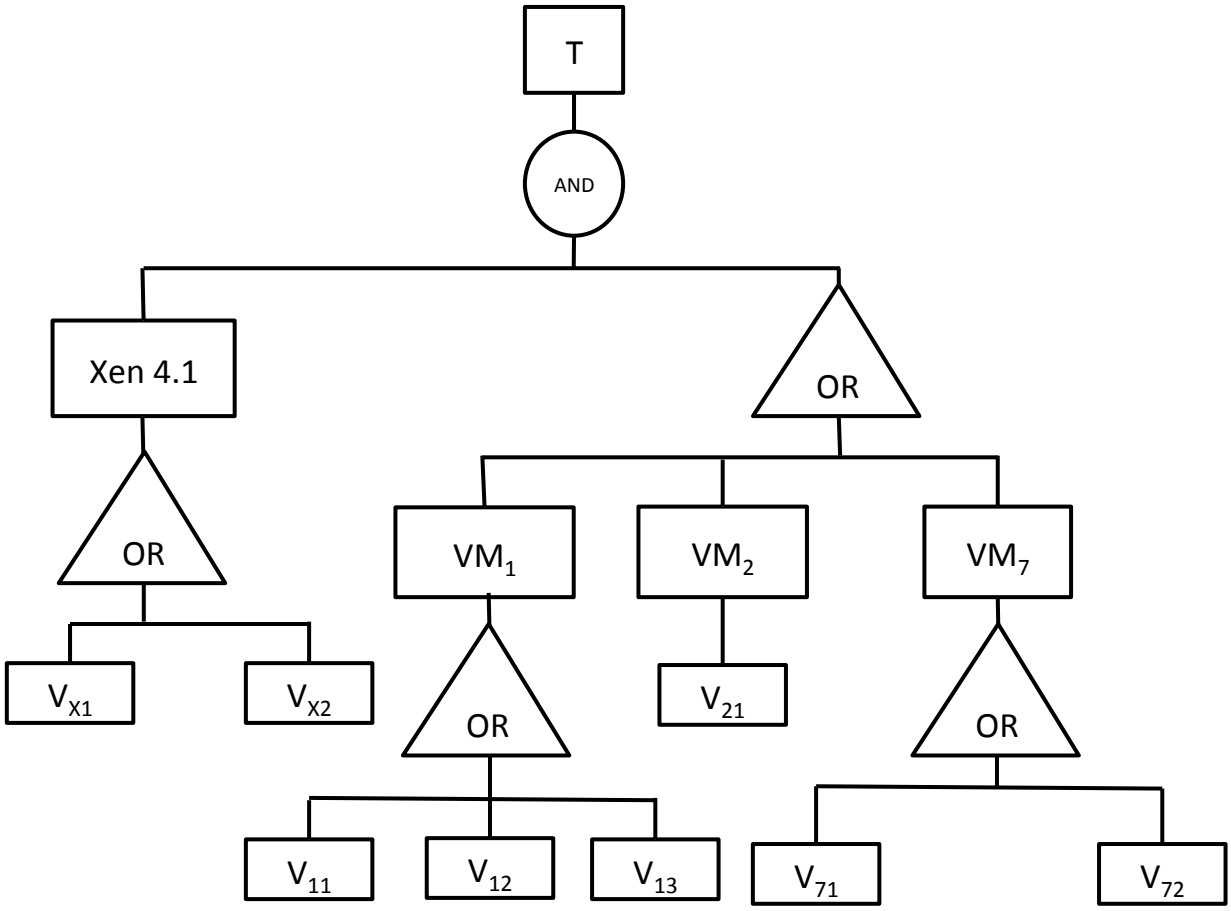

Figure 3.4: Vulnerability tree of the use case

privileges by using DMA to generate MSI interrupts by writing to the interrupt injection registers."

– xen-4.1: **CVE-2011-1583**
"Multiple integer overflows in tools/libxc/xc_dom_bzimageloader.c in Xen 3.2, 3.3, 4.0, and 4.1 allow local users to cause a denial of service and possibly execute arbitrary code via a crafted paravirtualised guest kernel image that triggers (1) a buffer overflow during a decompression loop or (2) an out-of-bounds read in the loader involving unspecified length fields."

– VM_1 (Apache 2.0): **CVE-2011-3192**
"The byterange filter in the Apache HTTP Server 1.3.x, 2.0.x through 2.0.64, and 2.2.x through 2.2.19 allows remote attackers to cause a denial of service (memory and CPU consumption) via a Range header that expresses multiple overlapping ranges, as exploited in the wild in August 2011, a different vulnerability than CVE-2007-0086."

– VM_1 (Apache 2.0)**CVE-2011-4317**
"The mod_proxy module in the Apache HTTP Server 1.3.x through 1.3.42, 2.0.x through

2.0.64, and 2.2.x through 2.2.21, when the Revision 1179239 patch is in place, does not properly interact with use of (1) RewriteRule and (2) ProxyPassMatch pattern matches for configuration of a reverse proxy, which allows remote attackers to send requests to intranet servers via a malformed URI containing an @ (at sign) character and a : (colon) character in invalid positions. NOTE: this vulnerability exists because of an incomplete fix for CVE-2011-3368."

- VM_1 (Apache 2.0)**CVE-2011-4415**
  "The ap_pregsub function in server/util.c in the Apache HTTP Server 2.0.x through 2.0.64 and 2.2.x through 2.2.21, when the mod_setenvif module is enabled, does not restrict the size of values of environment variables, which allows local users to cause a denial of service (memory consumption or NULL pointer dereference) via a .htaccess file with a crafted SetEnvIf directive, in conjunction with a crafted HTTP request header, related to (1) the "len +=" statement and (2) the apr_pcalloc function call, a different vulnerability than CVE-2011-3607."
- VM_2 (MySQL)**CVE-2010-1626**
  "MySQL before 5.1.46 allows local users to delete the data and index files of another user's MyISAM table via a symlink attack in conjunction with the DROP TABLE command, a different vulnerability than CVE-2008-4098 and CVE-2008-7247."
- VM_7 (Bind 9.8.0)**CVE-2011-2465**
  "Unspecified vulnerability in ISC BIND 9 9.8.0, 9.8.0-P1, 9.8.0-P2, and 9.8.1b1, when recursion is enabled and the Response Policy Zone (RPZ) contains DNAME or certain CNAME records, allows remote attackers to cause a denial of service (named daemon crash) via an unspecified query."
- VM_7 (Bind 9.8.0)**CVE-2011-2464**
  "Unspecified vulnerability in ISC BIND 9 9.6.x before 9.6-ESV-R4-P3, 9.7.x before 9.7.3-P3, and 9.8.x before 9.8.0-P4 allows remote attackers to cause a denial of service (named daemon crash) via a crafted UPDATE request."

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 3.2. We consider the vulnerabilities to be independent but not mutually exclusive.

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

Table 3.2: 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	

- "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. We show the details of the calculations of the *likelihood* and *impact*. Likelihood of the hypervisor Xen 4.1:

$$L[Xen4.1] = L[V_{X1}] + L[V_{X2}] - L[V_{X1}] * L[V_{X2}] \tag{3.11}$$

$$L[Xen4.1] = 0.44 + 0.34 - 0.44 * 0.34 = 0.6304 \tag{3.12}$$

Likelihood of VM_1:

$$\begin{aligned} L[VM_1] = {} & L[V_{11}] + L[V_{12}] + L[V_{13}] - L[V_{11}] * L[V_{12}] - L[V_{11}] * L[V_{13}] \\ & -L[V_{12}] * L[V_{13}] + L[V_{11}] * L[V_{12}] * L[V_{13}] \end{aligned} \tag{3.13}$$

$$\begin{aligned} L[VM_1] &= 1 + 0.86 + 0.19 - 1 * 0.86 - 1 * 0.19 - 0.19 * 0.86 + 1 * 0.86 * 19 \\ &= 1 \end{aligned} \tag{3.14}$$

Likelihood of VM_7:

$$L[VM_7] = L[V_{71}] + L[V_{72}] - L[V_{71}] * L[V_{72}] \tag{3.15}$$

$$L[VM_7] = 0.49 + 1 - 0.49 = 1 \tag{3.16}$$

Likelihood of the top event T:

$$\begin{aligned} L[T] &= L[Xen4.1] * L[VM_1] + L[Xen4.1] * L[VM_2] + L[Xen4.1] * L[VM_7] \\ &\quad -L[Xen4.1] * L[VM_1] * L[VM_2] - L[Xen4.1] * L[VM_1] * L[VM_7] \\ &\quad -L[Xen4.1] * L[VM_2] * L[VM_7] \\ &\quad +L[Xen4.1] * L[VM_1] * L[VM_2] * L[VM_7] \end{aligned} \tag{3.17}$$

$$\begin{aligned} L[T] &= 0.6304 * 1 + 0.6304 * 0.39 + 0.6304 * 1 - 0.6304 * 1 * 0.39 - 0.6304 * 1 * 1 \\ &\quad -0.6304 * 0.39 * 1 + 0.6304 * 1 * 0.39 * 1 \end{aligned} \tag{3.18}$$

$$L[T] = 0.6304 \tag{3.19}$$

Impact of the hypervisor Xen 4.1:

$$I[Xen4.1] = I[V_{X1}] + I[V_{X2}] - I[V_{X1}] * I[V_{X2}] \tag{3.20}$$

$$I[Xen4.1] = 1 + 1 - 1 = 1 \tag{3.21}$$

Impact of VM_1:

$$\begin{aligned} I[VM_1] &= I[V_{11}] + I[V_{12}] + I[V_{13}] - I[V_{11}] * I[V_{12}] - I[V_{11}] * I[V_{13}] \\ &\quad -I[V_{12}] * I[V_{13}] + I[V_{11}] * I[V_{12}] * I[V_{13}] \end{aligned} \tag{3.22}$$

$$\begin{aligned} I[VM_1] &= 0.69 + 0.29 + 0.29 - 0.69 * 0.29 - 0.69 * 0.29 - 0.29 * 0.29 + 0.69 * 0.29 * 0.29 \\ &= 0.8437 \end{aligned} \tag{3.23}$$

Impact of VM_7:

$$I[VM_7] = I[V_{71}] + I[V_{72}] - I[V_{71}] * I[V_{72}] \tag{3.24}$$

$$I[VM_7] = 0.29 + 0.29 - 0.29 * 0.29 = 0.4959 \tag{3.25}$$

Impact of the top event T:

$$
\begin{aligned}
I[T] &= I[Xen4.1] * I[VM_1] + I[Xen4.1] * I[VM_2] + I[Xen4.1] * I[VM_7] \\
&\quad -I[Xen4.1] * I[VM_1] * I[VM_2] - I[Xen4.1] * I[VM_1] * I[VM_7] \\
&\quad -I[Xen4.1] * I[VM_2] * I[VM_7] \\
&\quad +I[Xen4.1] * I[VM_1] * I[VM_2] * I[VM_7] \qquad (3.26)
\end{aligned}
$$

$$
\begin{aligned}
I[T] &= 0.8437 + 0.49 + 0.4959 - 0.8437 * 0.49 - 0.8437 * 0.4959 - 0.49 * 0.4959 \\
&\quad +0.8437 * 0.49 * 0.4959 \qquad (3.27)
\end{aligned}
$$

$$
I[T] = 0.9598 \qquad (3.28)
$$

Risk calculation:

$$
\begin{aligned}
Risk[T] &= L[T] * I[T] \\
&= 0.6304 * 0.9598 \\
&= 0.605 \qquad (3.29)
\end{aligned}
$$

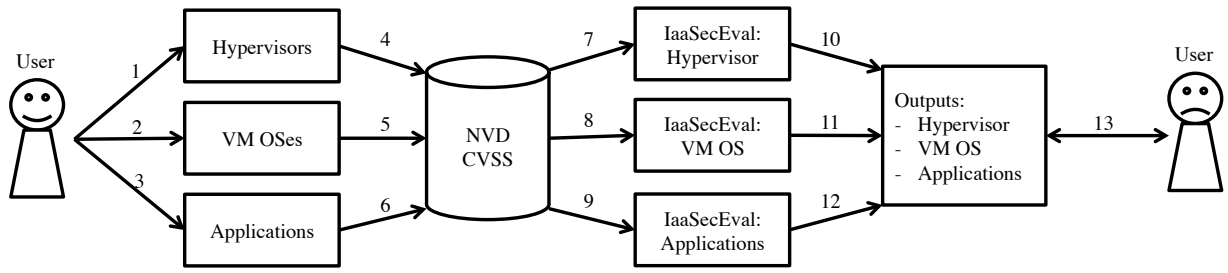

Figure 3.5: IaaSecEval architecture

3.5 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 Chapter, 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 [53, 54, 55, 56]. 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 [57] 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 Figure 3.5. The architecture is dispatched into 13 steps, which we explain hereafter: 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 3.6 contains the details of the use case and the final results. The descriptions of the vulnerabilities we considered in this section are as follows:

CVE-2014-1893: Multiple integer overflows in the (1) FLASK_GETBOOL and (2)

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

Figure 3.6: Use case Data and Results

FLASK_SETBOOL suboperations in the flask hypercall in Xen 4.1.x, 3.3.x, 3.2.x, and earlier, when XSM is enabled, allow local users to cause a denial of service (processor fault) via unspecified vectors, a different vulnerability than CVE-2014-1891, CVE-2014-1892, and CVE-2014-1894.

CVE-2014-1892: Xen 3.3 through 4.1, when XSM is enabled, allows local users to cause a denial of service via vectors related to a "large memory allocation," a different vulnerability than CVE-2014-1891, CVE-2014-1893, and CVE-2014-1894.

CVE-2011-1166: Xen, possibly before 4.0.2, allows local 64-bit PV guests to cause a denial of service (host crash) by specifying user mode execution without user-mode pagetables.

CVE-2014-3730: The django.util.http.is_safe_url function in Django 1.4 before 1.4.13, 1.5 before 1.5.8, 1.6 before 1.6.5, and 1.7 before 1.7b4 does not properly validate URLs, which allows remote attackers to conduct open redirect attacks via a malformed URL, as demonstrated by "http://djangoproject.com."

CVE-2014-3204: Unity before 7.2.1, as used in Ubuntu 14.04, does not properly handle keyboard shortcuts, which allows physically proximate attackers to bypass the lock screen and execute arbitrary commands, as demonstrated by right-clicking on the indicator bar and then pressing the ALT and F2 keys.

CVE-2014-2894: Off-by-one error in the cmd_smart function in the smart self test in hw/ide/core.c in QEMU before 2.0 allows local users to have unspecified impact via a SMART EXECUTE OFFLINE command that triggers a buffer underflow and memory corruption.

CVE-2014-0150: Integer overflow in the virtio_net_handle_mac function in hw/net/virtio-net.c in QEMU 2.0 and earlier allows local guest users to execute arbitrary code via a MAC

addresses table update request, which triggers a heap-based buffer overflow.

CVE-2013-4544: hw/net/vmxnet3.c in QEMU 2.0.0-rc0, 1.7.1, and earlier allows local guest users to cause a denial of service or possibly execute arbitrary code via vectors related to (1) RX or (2) TX queue numbers or (3) interrupt indices. NOTE: some of these details are obtained from third party information.

CVE-2014-1812: The Group Policy implementation in Microsoft Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows 8.1, and Windows Server 2012 Gold and R2 does not properly handle distribution of passwords, which allows remote authenticated users to obtain sensitive credential information and consequently gain privileges by leveraging access to the SYSVOL share, as exploited in the wild in May 2014, aka "Group Policy Preferences Password Elevation of Privilege Vulnerability."

CVE-2014-1807: The ShellExecute API in Windows Shell in Microsoft Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows 8.1, Windows Server 2012 Gold and R2, and Windows RT Gold and 8.1 does not properly implement file associations, which allows local users to gain privileges via a crafted application, as exploited in the wild in May 2014, aka "Windows Shell File Association Vulnerability."

3.6 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.

3.6.1 Overview of OpenStack Security Analysis

OpenStack logical architecture, Figure 3.7 [29], 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

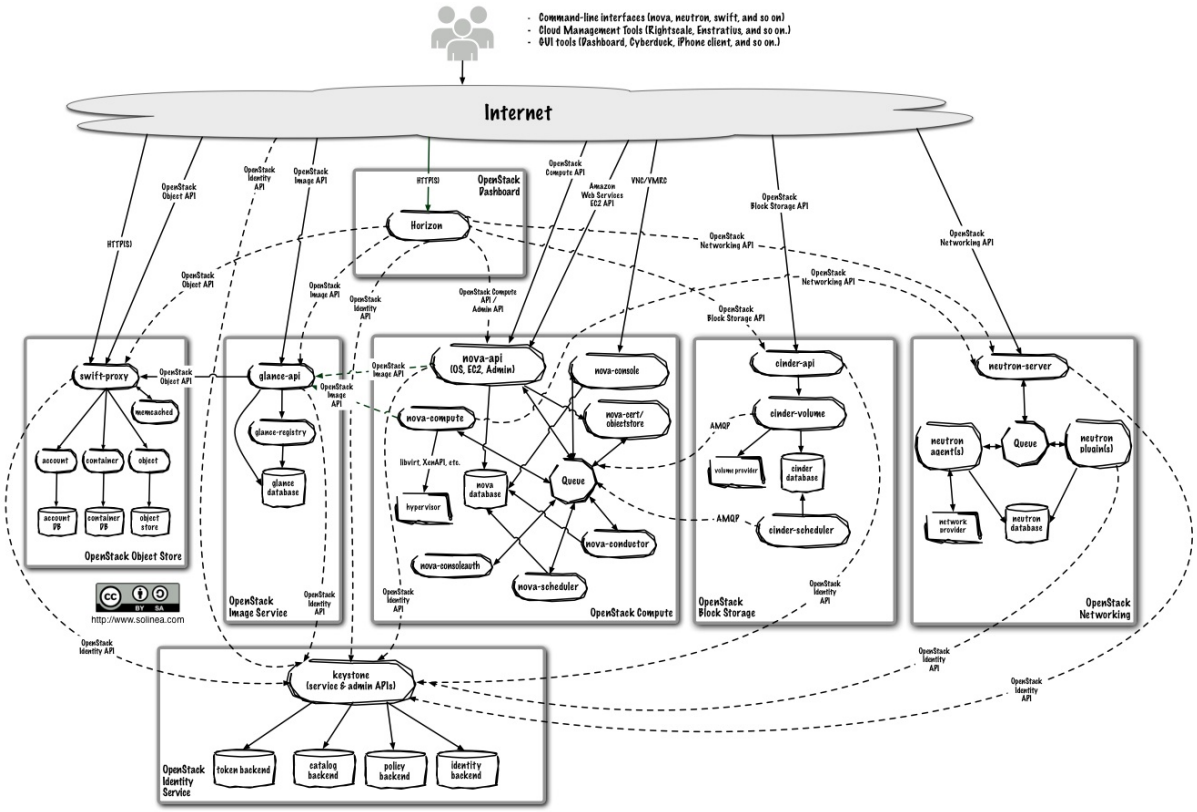

Figure 3.7: OpenStack logical architecture

to the construction of 7 vulnerability trees that we examine hereafter. To avoid redundancy, we only provide details of the top events for one case. A clear comprehension of Section 3.3 allows a better understanding of this section. Figure 3.2 shows a non-exhaustive list of standard fault tree symbols that we use to construct our vulnerability tree.

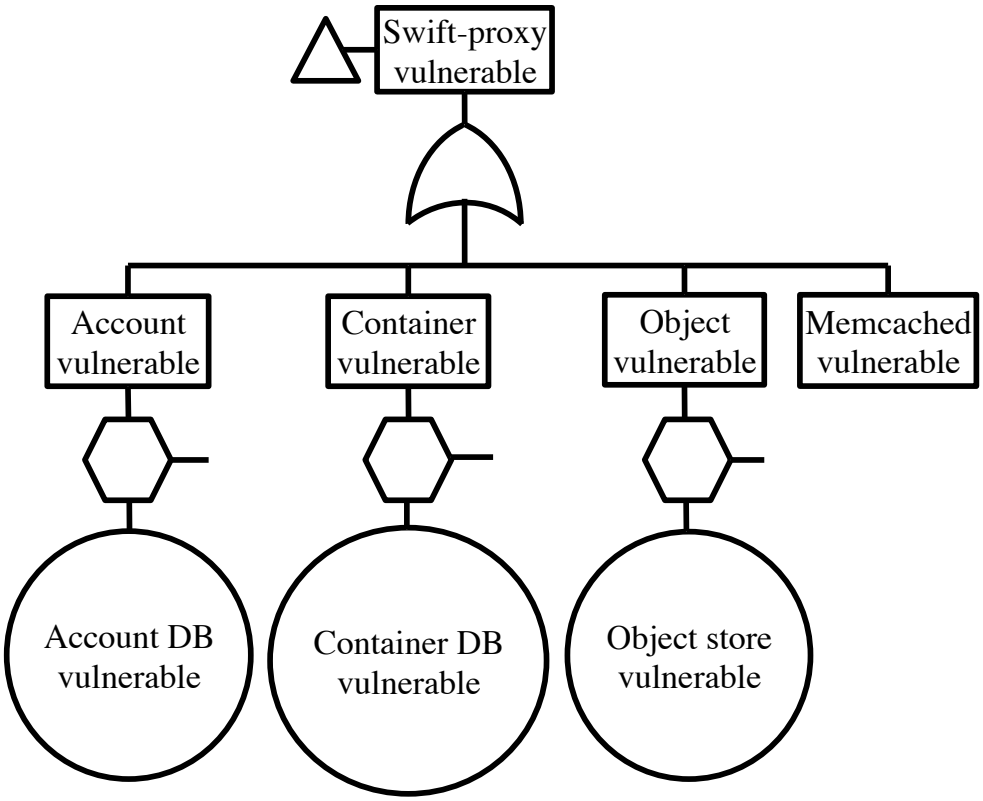

Figure 3.8: Swift Vulnerability Tree

3.6.2 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 Figure 3.8. As Swift is attached to Keystone, the vulnerability tree can be developed further in respect to that attachment.

3.6.3 Security Evaluation of Glance

Glance is very simple in its composition, consequently the vulnerability tree, which is schematized in Figure 3.9, 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.

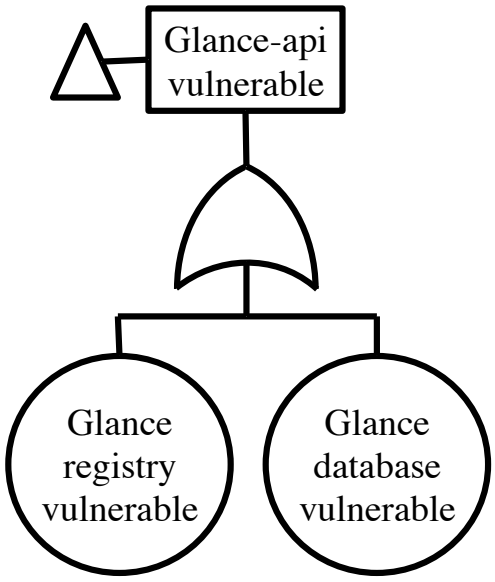

Figure 3.9: Glance Vulnerability Tree

3.6.4 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 Figure 3.10. We indicate that the tree can be extrapolated due the connections that Nova has with other components.

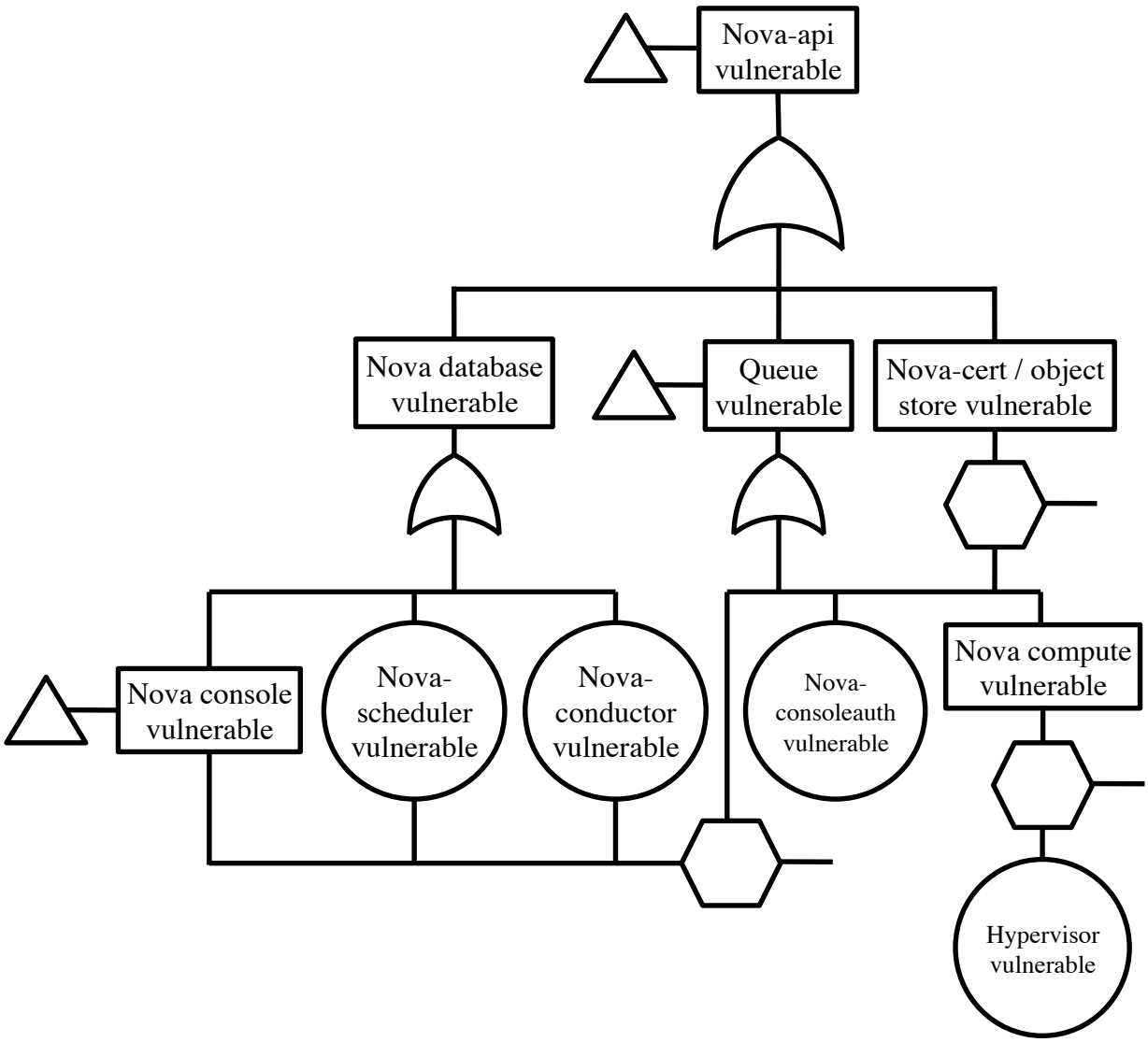

Figure 3.10: Nova Vulnerability Tree

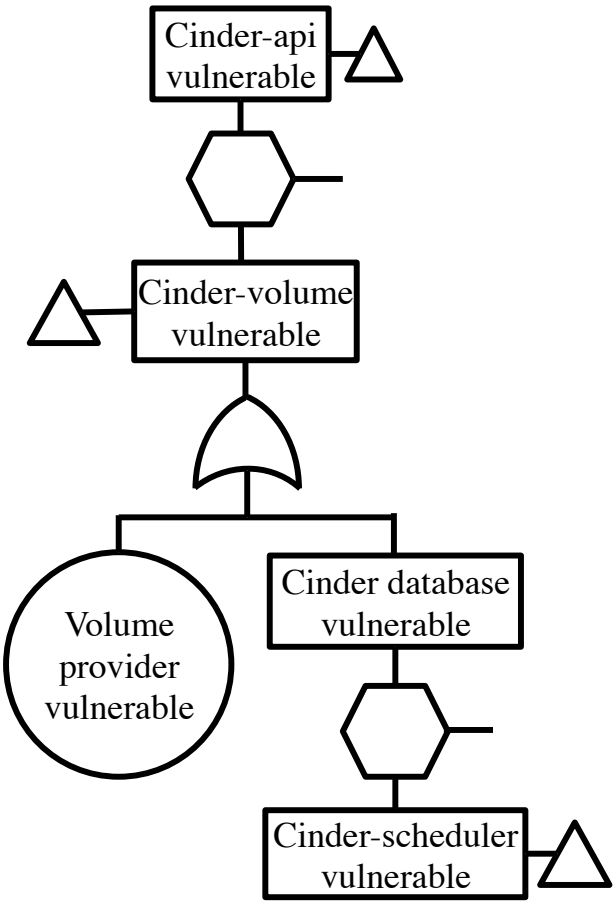

Figure 3.11: Cinder Vulnerability Tree

3.6.5 Security Evaluation of Cinder

The vulnerability tree of OpenStack Block Storage, also known as Cinder, is simple to construct and is represented in Figure 3.11. 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.

3.6.6 Security Evaluation of Neutron

OpenStack Network Service, codenamed Neutron, also has a simple composition that facilitates the construction of the vulnerability tree showed in Figure 3.12. Neutron has connections with Horizon, Nova, and Keystone. Consequently, the tree can be extended further towards those components.

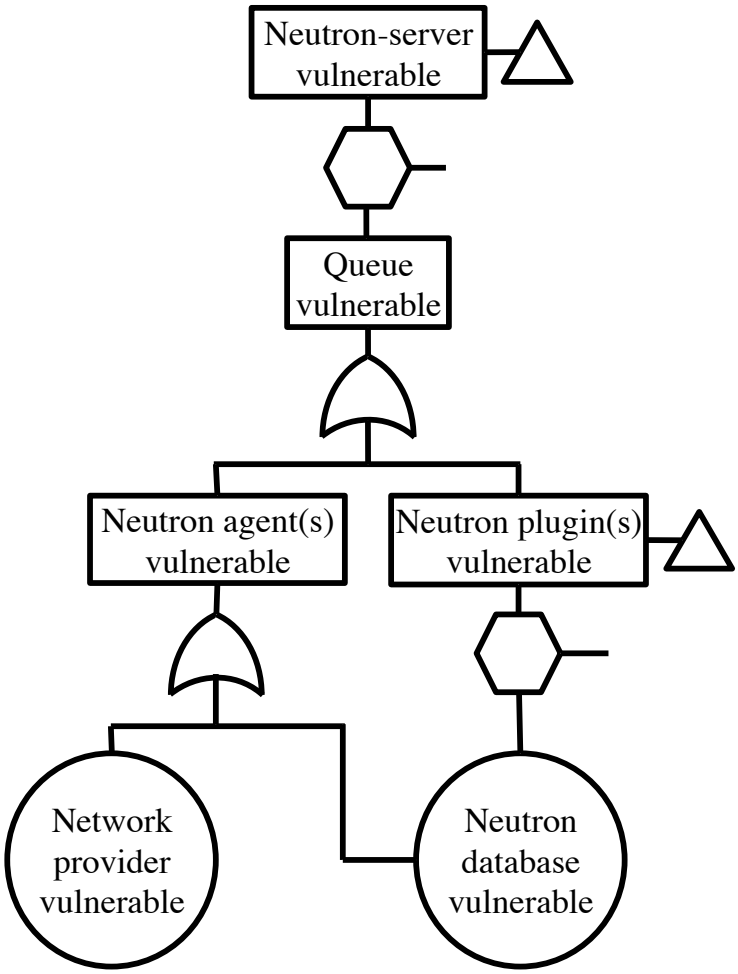

Figure 3.12: Neutron Vulnerability Tree

3.6.7 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 Figure 3.13. 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 3.3, we are able to derive the security evaluation, which is given by Equation 3.30.

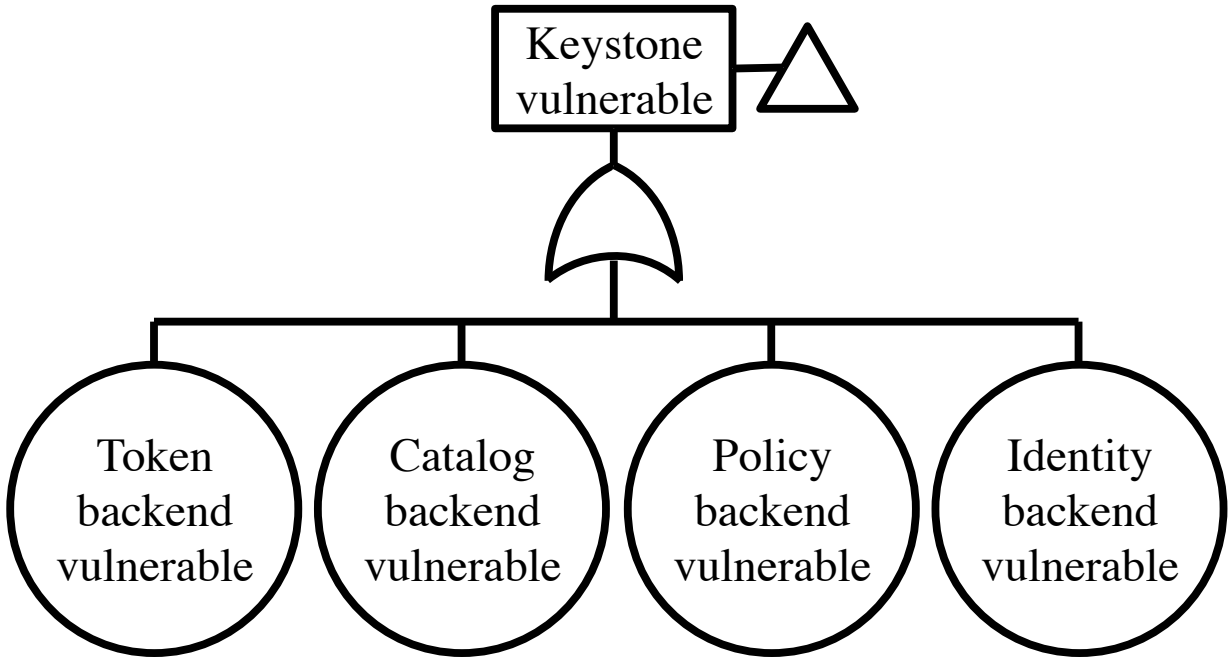

Figure 3.13: Keystone Vulnerability Tree

$$
\begin{aligned}
P[K] \quad = \quad & P[K_1] + P[K_2] + P[K_3] + P[K_4] - P[K_1]P[K_2] \\
& -P[K_1]P[K_3] - P[K_1]P[K_4] - P[K_2]P[K_3] \\
& -P[K_2]P[K_4] - P[K_3]P[K_4] + P[K_1]P[K_2]P[K_3] \\
& +P[K_1]P[K_3]P[K_4] + P[K_1]P[K_2]P[K_4] + \\
& P[K_2]P[K_3]P[K_4] - P[K_1]P[K_2]P[K_3]P[K_4]. \qquad (3.30)
\end{aligned}
$$

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 3.3. Unfortunately, we cannot have an ideal use case because of the vulnerability issue (Section 3.7) we encountered in this study. Nevertheless, we provide a particular case of security quantification of Keystone. We query the NVD to retrieve four vulnerabilities of Keystone and their respective impact and likelihood values as shown in Table 3.3.

Likelihood of Keystone L[K]:

Table 3.3: Keystone vulnerabilities likelihood & impact

Vulnerabilities	Impact	Likelihood
CVE-2014-0204 (K_1)	0.64	0.8
CVE-2014-3520 (K_2)	0.64	0.68
CVE-2014-7144 (K_3)	0.29	0.86
CVE-2014-3621 (K_4)	0.29	0.8

$$\begin{aligned} L[K] &= L[K_1] + L[K_2] + L[K_3] + L[K_4] - L[K_1]L[K_2] \\ &\quad -L[K_1]L[K_3] - L[K_1]L[K_4] - L[K_2]L[K_3] \\ &\quad -L[K_2]L[K_4] - L[K_3]L[K_4] + L[K_1]L[K_2]L[K_3] \\ &\quad +L[K_1]L[K_3]L[K_4] + L[K_1]L[K_2]L[K_4] + \\ &\quad L[K_2]L[K_3]L[K_4] - L[K_1]L[K_2]L[K_3]L[K_4]. \end{aligned} \tag{3.31}$$

$$\begin{aligned} L[K] &= 0.8 + 0.68 + 0.86 + 0.8 - 0.8 * 0.68 \\ &\quad -0.8 * 0.86 - 0.8 * 0.8 - 0.68 * 0.86 \\ &\quad -0.68 * 0.8 - 0.86 * 0.8 + 0.8 * 0.68 * 0.86 \\ &\quad +0.8 * 0.86 * 0.8 + 0.8 * 0.68 * 0.8 + \\ &\quad 0.68 * 0.86 * 0.8 - 0.8 * 0.68 * 0.86 * 0.8. \end{aligned} \tag{3.32}$$

$$L[K] = 0.9981. \tag{3.33}$$

Impact of Keystone I[K]:

$$\begin{aligned} I[K] &= I[K_1] + I[K_2] + I[K_3] + I[K_4] - I[K_1]I[K_2] \\ &\quad -I[K_1]I[K_3] - I[K_1]I[K_4] - I[K_2]I[K_3] \\ &\quad -I[K_2]I[K_4] - I[K_3]I[K_4] + I[K_1]I[K_2]I[K_3] \\ &\quad +I[K_1]I[K_3]I[K_4] + I[K_1]I[K_2]I[K_4] + \\ &\quad I[K_2]I[K_3]I[K_4] - I[K_1]I[K_2]I[K_3]I[K_4]. \end{aligned} \tag{3.34}$$

$$\begin{aligned} I[K] &= 0.64 + 0.64 + 0.29 + 0.29 - 0.64 * 0.64 \\ &\quad -0.64 * 0.29 - 0.64 * 0.29 - 0.64 * 0.29 \\ &\quad -0.64 * 0.29 - 0.29 * 0.29 + 0.64 * 0.64 * 0.29 \\ &\quad +0.64 * 0.29 * 0.29 + 0.64 * 0.64 * 0.29 + \\ &\quad 0.64 * 0.29 * 0.29 - 0.64 * 0.64 * 0.29 * 0.29. \end{aligned} \tag{3.35}$$

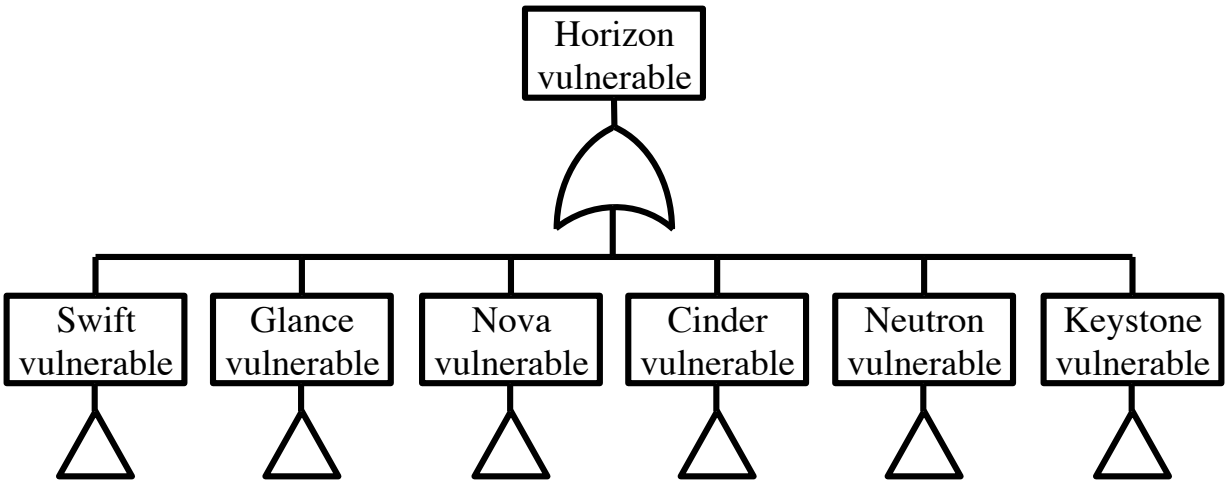

Figure 3.14: Horizon Vulnerability Tree

$$I[K] = 0.9347. \tag{3.36}$$

Risk of Keystone R[K]:

$$R[K] = L[K] * I[K] = 0.9981 * 0.9347 = 0.9329. \tag{3.37}$$

3.6.8 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 Figure 3.14. All the events are deemed intermediate because they could be extended further.

Let's denote the top event (Horizon vulnerable) by H. The intermediate events Swift vulnerable, Glance vulnerable, Nova vulnerable, Cinder vulnerable, Neutron vulnerable, and Keystone vulnerable are respectively denoted S, G, No, C, Ne, and K. The ensuing security evaluation is given in Equation 3.38. As in the previous subsection, Equation 3.38 can not be used with the current naming of OpenStack's vulnerabilities.

Let us remind that P[S], P[G], P[No], P[C], P[Ne], and P[K] respectively represents the security evaluation of Swift, Glance, Nova, Cinder, Neutron, and Keystone. The security evaluation of Horizon, P[H], can be seen as the security evaluation of the entire OpenStack logical architecture.

$$
\begin{aligned}
P[H] \;=\;& P[S] + P[G] + P[No] + P[C] + P[Ne] + P[K] - P[S]P[G] - \\
& P[S]P[No] - P[S]P[C] - P[S]P[Ne] - P[S]P[K] - P[G]P[No] - \\
& P[G]P[C] - P[G]P[Ne] - P[G]P[K] - P[No]P[C] - P[No]P[Ne] - \\
& P[No]P[K] - P[C]P[Ne] - P[C]P[K] - P[Ne]P[K] + P[S]P[G]P[No] + \\
& P[S]P[G]P[C] + P[S]P[G]P[Ne] + P[S]P[G]P[K] + P[S]P[No]P[C] + \\
& P[S]P[No]P[Ne] + P[S]P[No]P[K] + P[S]P[C]P[Ne] + \\
& P[S]P[C]P[K] + P[S]P[Ne]P[K] + P[G]P[No]P[C] + P[G]P[No]P[Ne] + \\
& P[G]P[No]P[K] + P[G]P[C]P[Ne] + P[G]P[C]P[K] + P[G]P[Ne]P[K] + \\
& P[No]P[C]P[Ne] + P[No]P[C]P[K] + P[No]P[Ne]P[K] + P[C]P[Ne]P[K] - \\
& P[S]P[G]P[No]P[C] - P[S]P[G]P[No]P[Ne] - P[S]P[G]P[No]P[K] - \\
& P[S]P[No]P[C]P[Ne] - P[S]P[No]P[C]P[K] - P[S]P[C]P[Ne]P[K] - \\
& P[G]P[No]P[C]P[Ne] - P[G]P[No]P[C]P[K] - \\
& P[No]P[C]P[Ne]P[K] + P[S]P[G]P[No]P[C]P[Ne] + \\
& P[S]P[G]P[No]P[C]P[K] + P[G]P[No]P[C]P[Ne]P[K] - \\
& P[S]P[G]P[No]P[C]P[Ne]P[K]. \qquad (3.38)
\end{aligned}
$$

3.6.9 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.

3.7 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 3.4, 3.5, and 3.6. 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.

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

```
<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"/>
```

Figure 3.15: Current Vulnerability Naming of OpenStack

```
<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:*:*:*:*:*"/>
```

Figure 3.16: Proposed Nomenclature for OpenStack Vulnerabilities

and environmental metric groups. We believe that their presence in our proposal is a non-sense 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 3.4, 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 3.6, 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 (Figure 3.15). The naming of the vulnerabilities is effectuated by using the Naming specification of the Common Platform Enumeration (CPE) [58]. 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 [59]; that is why it has a monopolistic 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 3.16 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 3.6, 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.

3.8 Summary

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.

Chapter 4

Risk Adaptive Authorization Mechanism for Cloud Computing

Cloud computing provides many advantages for both the cloud service provider and the clients. It is also infamous for being highly dynamic and for having numerous security issues. The dynamicity of cloud computing implies that dynamic security mechanisms are being employed to enforce its security, especially in regards to access decisions. However, this is surprisingly not the case. Static traditional authorization mechanisms are being used in cloud environments, leading to legitimate doubts on their ability to fulfill the security needs of the cloud. We propose a risk adaptive authorization mechanism (RAdAM) for a simple cloud deployment, collaboration in cloud computing and federation in cloud computing. We use a fuzzy inference system to demonstrate the practicability of RAdAM. We complement RAdAM with a Vulnerability Based Authorization Mechanism (VBAM) which is a real-time authorization model based on the average vulnerability scores of the objects present in the cloud. We demonstrated the usefulness of VBAM in a use case featuring OpenStack.

4.1 Introduction

Modern computer systems are highly dynamic. This fact is epitomized by cloud computing, which is a novel paradigm in Information Technology (IT) that features data externalization and low cost of computing infrastructures. The National Institute of Standards and Technology (NIST) [4] defines cloud computing as "a model for enabling ubiquitous on demand network access to a shared pool of configurable resources (e.g., networks, servers, storage, applications, and servers) that can be rapidly provisioned and released with minimal management effort or service provider interaction." Despite the fact that cloud computing is profitable for both adopters and providers, its security issues remain a concern; particularly, the fact that traditional static authorization mechanisms are employed to enforce access decisions. The current situation reflects an unsuitability between cloud computing and the authorization mechanisms to date. Novel authorization mechanisms are needed. In the early 2000s, the JASON report [15] pioneered the usage of risk in modern authorization mechanisms in order to keep with the dynamicity of modern IT platforms. Their idea supposes that every access request is associated with a risk decision

that will be challenged by a threshold to allow (resp. deny) the access request. That marks the genesis of a number of work around risk aware access controls. Among those researches, a seminal paper [16] defined the principles of a risk aware access control. In that regard, we propose a risk adaptive authorization mechanism (RAdAM) which is a dynamic real-time risk-aware authorization mechanism for cloud computing. We ascertained that RAdAM follows the principles of a risk aware access control as defined in [16]. We proposed authorization mechanisms for the most prominent cases that exist in cloud computing. We consider the situation where the user is a simple cloud customer with basic services. Afterwards, we slightly complicated the atmosphere by proposing an authorization mechanism for collaboration in cloud computing. Finally, we designed an authorization mechanism for the most complicated case in cloud computing: cloud federation. We employed a fuzzy inference system (FIS) [17] to demonstrate how RAdAM functions. The main contribution of RAdAM is not in the risk determination in itself but in the way the authorization algorithms work and the novel parameters we introduce in order to allow the authorization to be as flexible as possible. To complement RAdAM, we also proposed a vulnerability-based authorization (VBAM) mechanism which is a variation of the Bell-Lapadula multi-level security (MLS) [18]. The most important point of VBAM is the usage of the vulnerability score of the objects to enforce a decision. We contend that modern authorization mechanisms should not only be dynamic, but also should be tailored to the vulnerabilities of the objects that are being accessed. We demonstrated the usefulness of VBAM by explaining how it could be used in an OpenStack environment. The remaining of this Chapter is structured as follows: Section 4.2 contains the motivation and the background explanation of a risk aware access control. The related work is exposed in Section 4.3. We explain the details of RAdAM in Section 4.4. We expand upon the usage of RAdAM in a fuzzy inference system in Section 4.5. Section 4.6 contains the explanation of VBAM and a use case of a possible usage. In Section 4.7, we discuss the shortcomings of our proposals and future work that lies ahead. Section 4.8 concludes the paper.

4.2 Background: Risk Aware Access Control

Information sharing has always been a subject of controversy. That is why it drew the attention of the security experts in the early stages of computing. Many security mechanisms have been developed for the purpose of information sharing; starting from the basic Access Control Lists (ACLs) to Mandatory Access Control (MAC) [60], Discretionary Access Control (DAC) [61], and Role-Based Access Control (RBAC) [62]. Currently, we are facing new challenges as the shared infrastructure became dynamic and those aforementioned ACs are not flexible enough to meet the new challenges. It became an urgent matter to engineer access controls that can cope with the new challenges brought by the modern computer systems. The JASON report [15] pioneered the idea of a modern access control by defining three fundamental principles that should be included in their design:

- Measure risk: if you can measure it, you can manage it. In other words, knowing the risk associated with an event permits a better handling.
- Establish an acceptable risk level

- Ensure that the information is tailored to the level of the acceptable risk.

An authorization mechanism that is engineered by mirroring these guidelines can deal with the reality of today's information sharing platforms. A seminal paper about an access control that follows theses guideline was proposed by NIST and titled Risk-Adaptable Access Control (RAdAC) [16]. RAdAC is a real-time, adaptable risk-aware access control built by a combination of Attribute-Based Access Control [63], Policy-Based Access Control, Machine Learning, and heuristics. Six factors are indispensable to make RAdAC decisions:

- **Operational need**: this factor is one of the factors that RAdAC borrowed from traditional access controls. It involves the notion of *need to know*, which means that a requester should have a relation with the object he is requesting.
- **Security risk**: is the cornerstone of RAdAC. This factor requires the use of machine learning for a real time probabilistic determination of risk associated to a request.
- **Situational factors**: sometimes the access decisions are made depending on the actual situations; it is possible that the *Operational need* outweighs the *Security risk*.
- **Access Control Policy**: as any normal AC, policies need to be enforced. The policies should respect the mechanisms of RAdAC by, among other things, setting the acceptable risk level and defining the conditions on which the *Operational Need* can outweigh the *Security Risk*.
- **Heuristics**: the goal is to use past access control decisions to make present and future decisions. The utilization of past decisions will help to better determine the *Security Risk* and *Operational Need* and will increase the positive number of access control decisions.

4.3 Related Work

To the best of our knowledge, this technique is novel as it has not been employed by other researchers. Nevertheless, a quick perusal over the literature of risk based access control revealed interesting facts.

Despite the fact that cloud computing is the archetypical dynamic system, there is not a tremendous amount of research related to risk aware access control in cloud. In fact, Fall et al. [64] and Dos Santos et al. [65, 66] are the only researchers who are evangelizing the usage of this method in cloud. Fall et al. [64] proposed a semantic for adapting RAdAC [16] in cloud computing. Unfortunately they did not propose any implementation of their idea. Santos et al. [65] proposed a risk-based access control architecture for a highly scalable cloud federation. They demonstrated the usefulness of their proposal through a prototype by using a combination of tools including XACML [67]. In their other paper [66], they extended [65] by adding three modules to the XACML 3.0 architecture. A risk policy module that explains to the cloud service provider how the access control must be handled. The risk engine module handles the processing of risk policies. The risk quantification and web services module quantifies the risk for every access request.

Cheng et al. [68] extended the Bell-Lapadula multi-level security (MLS) access control model with the concept of risk and named their proposal FuzzyMLS. In FuzzyMLS, risk is the value of utility loss, multiplied by its leakage probability. The leakage probability is jointly

dependent on the temptation and inadvertent leakage probabilities. The temptation is calculated by the security level of the subject and object, and the inadvertent leakage by a fuzzy approach. They use different risk thresholds to make access decisions. Ni et al. [69] proposed a risk based access control built on fuzzy inference. They claim that the inflexibility of traditional access control is a major inhibitor for information sharing. They first proposed fuzzy BLP and compared it to FuzzyMLS [68]. Moreover, they argue that despite fuzzy inference being an excellent solution for modern access control, it raises new issues that have to be addressed. Particularly, the fact that, generally, risk aware access controls are time-consuming thus malicious users can take advantage of the time window that exists between an access request and the risk mitigation. Further, they proposed some algebraic adjustment to solve the issues of conjunction and disjunction in fuzzy inference.

Burnett et al. [70] proposed TRAAC a trust and risk aware access control that provides a policy coverage, dynamic access control decisions, appropriate denial of risk and rights delegation. They claim that their system fits the healthcare domain perfectly though it can be extended to other domains. First, they defined a zone policy model where the data owner has total control of the privileges on his data which he can share with other users, hence including them in the domain. The trust is defined in terms of sharing trust and obligation trust that permits the verification of whether the requester respected the obligations that are assigned to him or not. They used a probabilistic computational trust model called *Subjective Logic* to formulate their trust assessment. The risk is computed in the classical manner of expected loss in terms of unwanted disclosure. Khambhammettu et al. [71] designed a risk aware framework for decision making where risk is evaluated as the product of threats and impact scores. The authorization is made by weighing the security risk against the operational-need and security factors. The framework embraces four approaches based on the sensitivity level of the object, the trustworthiness of the subject, and the remaining are made of the combination of the two aforementioned approaches. Throughout their paper, they proposed successful instances of their proposal. Shaikh et al. [72] proposed two methods of risk aware access control based on trust and risk. The formulation of trust and risk depends on rewards and penalties which are updated either positively (rewards) or negatively (penalties) after an access request decision is made (allow or deny, respectively). An increase of the trust causes a decrease of the risk and vice versa. The method is based on Multi-Level Security (MLS) systems [18]. Due to the slow responses in access rights, they proposed another risk aware access control based on Exponentially Weighted Moving Average (EWMA) [73]. This method has similar functionalities as the first method.

4.4 Risk-Adaptive Authorization Mechanism: RAdAM

In this section, we will shed some light on our proposal. We propose an authorization mechanism for three cloud deployment methods: user-mode, collaboration mode and federation mode.

Our authorization mechanism respects the aforementioned principles of RAdAC. We reiterate that, in this model, every access request is evaluated by a risk that will be challenged against a threshold. We introduce a number of parameters that are handy while designing the authorizations. If the access request is allowed (resp. denied), the parameters are 'positively' (resp. negatively) updated in a way that risk will be better (resp. worse) for the next access request. We

assume the existence of a set of objects O and a set of subjects S. An object represents a piece of data which is controlled by the system.

4.4.1 RAdAM: Cloud user

We are aware of the fact that in basic cloud environments, the prominent actors are the users and the cloud administrator. We suppose that the cloud administrator is honest so our authorization mechanism focuses on the user. We interchangeably say user or subject. We consider the data of the user as objects. Thus in our authorization mechanism, we have the tuple <Subject, Object>. As we strive to build a real time risk adaptive authorization mechanism, we introduce two tokens that will be used to compute the risk associated with a request:

- α: represents the dynamic token of the user that is linked with the object the subject is trying to access.
- β: stands for the token of the object that is being accessed.

In our system, every single access is associated with a risk calculation. The risk is calculated in function of the token of the user α and the token of the object β. Risk is represented by $Risk(\alpha, \beta)$. To be in line with the principles of a risk adaptive authorization mechanism, we introduce a threshold $T(\alpha, \beta)$. The threshold is also inherently attached to the subject and that particular object. For *situational factor* purposes, we also introduce two other parameters:

- The average risk, which is the sum of the risk accumulated by the subject divided by the number of objects:
  $Risk(\alpha, \beta)_A = \sum Risk(\alpha, \beta)$ / n.
- The average risk threshold, which is the sum of the thresholds to all subject/object risk attempts divided by the number of objects:
  $T(\alpha, \beta)_A = \sum T(\alpha, \beta)$ / n.

The average risk gives us the general behavior of the user. In case the $Risk(\alpha, \beta)$ is equal to the threshold $T(\alpha, \beta)$, the system verifies whether the average risk is greater than or equal to the average risk threshold to subsequently allow the access request or deny it otherwise.

We want to clarify how the system works in detail as the parameters defined above are for generic purposes. We consider that we have a set of subjects S. A subject $s_i \in S$ has a set of objects $O_i = \{o_{i1}, \ldots, o_{in}\}$. So for an access request to object o_{ik}, the risk is defined by $Risk(\alpha_{ik}, \beta_{ik})$. Similarly the threshold is $T(\alpha_{ik}, \beta_{ik})$. The average risk would be given by:
$Risk(\alpha_i, \beta_i)_A = \sum_{j=1}^{n} Risk(\alpha_{ij}, \beta_{ij})$ / n.
The average risk threshold is: $T(\alpha_i, \beta_i)_A = \sum_{j=1}^{n} T(\alpha_{ij}, \beta_{ij})$ / n. The authorization algorithm is highlighted in Equation 4.1.

$$Au(s_i, o_{ik}) = \begin{cases} \text{if } Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) & \text{allow} \\ \text{if } Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ then, if} & \\ Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A & \text{allow} \\ otherwise, \text{ deny} & \end{cases} \tag{4.1}$$

4.4.2 RAdAM: Cloud Collaboration

Data owner

In our understanding, collaboration in cloud computing reflects the idea of a subject expressing a desire to share their data with other subjects. While collaboration creates added values, resource misuse and irresponsibility inflict costs and damages on affected resources. Permission misuse and legitimate user attacks are among the most serious threats which users in cloud computing collaboration face today. The situation is made even worse due to the dynamic aspect of cloud computing. The authorization mechanism that we display in this section solely concerns the data owner. We discuss the authorization mechanism for the other subjects in the appropriate section. The parameters we defined in the previous section are carried over in this section. We also introduce new parameters that are inherent to collaboration in cloud computing:

- γ: represents the dynamic collaboration token of the user that is attached to the object which the user is trying to access.
- δ: represents the token of the object in collaboration mode.
- $Risk(\gamma, \delta)$: is the risk associated with a collaboration access request.
- $T(\gamma, \delta)$: the risk threshold to which the risk will be evaluated against.
- $Risk(\gamma, \delta)_A$: the average collaboration risk is the sum of the accumulated collaboration risk divide by their number (m): $Risk(\gamma, \delta)_A = \sum Risk(\gamma, \delta)$ / n.
- $T(\gamma, \delta)_A$: The average collaboration risk threshold is the sum of the risk collaboration thresholds divided by their number: $T(\gamma, \delta)_A = \sum T(\gamma, \delta)$ / n.

The above-introduced parameters are solely used in the case of collaboration. Similar to the previous section, $Risk(\gamma, \delta)_A$ and $T(\gamma, \delta)_A$ are only used for situational factors.

For an instance of the authorization mechanism in this collaboration case, we still consider the user s_i with his set of object O_i=$\{o_{i1},\ldots,o_{in}\}$. If the user wants to access the object o_{ik}, the collaboration risk is given by: $Risk(\gamma_{ik}, \delta_{ik})$. The collaboration threshold is: $T(\gamma_{ik}, \delta_{ik})$. The average collabration risk is determined by: $Risk(\gamma_i, \delta_i)_A = \sum_{l=1}^{t} Risk(\gamma_{il}, \delta_{il})$ / t; t being the number of objects, of the user, that are in collaboration mode. The average collaboration risk threshold is: $T(\gamma_i, \delta_i)_A = \sum_{l=1}^{t} T(\gamma_{il}, \delta_{il})$ / t. The authorization mechanism is displayed in Equation 4.2.

Collaborative user

In this situation we assume that the data owner wants to collaborate with another user in the cloud and thus share their data with them. As there are different levels of permission (i.e, read, write or execute), we introduce risk bands that define the level of permission which the user belongs to.

When a user attempts an access request, the system checks first whether they has the collaboration parameters. If so, the system then proceeds and computes the risk based on the parameters. The result will determine to which risk band the user belongs to hence their level of permission over the data they is trying to access.

$$Au(s_i, o_{ik}, C) = \begin{cases} \text{if } \begin{cases} Risk(\alpha_{ik}, \beta_{ik}) > T(\alpha_{ik}, \beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik}, \delta_{ik}) > T(\gamma_{ik}, \delta_{ik}) \end{cases} & allow \\ \text{if } \begin{cases} Risk(\alpha_{ik}, \beta_{ik}) = T(\alpha_{ik}, \beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik}, \delta_{ik}) > T(\gamma_{ik}, \delta_{ik}) \text{ then,} \\ \text{if } Risk(\alpha_i, \beta_i)_A \geq T(\alpha_i, \beta_i)_A \end{cases} & allow \\ \text{if } \begin{cases} Risk(\alpha_{ik}, \beta_{ik}) = T(\alpha_{ik}, \beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik}, \delta_{ik}) = T(\gamma_{ik}, \delta_{ik}) \text{ then,} \\ \text{if } \begin{cases} Risk(\alpha_i, \beta_i)_A \geq T(\alpha_i, \beta_i)_A \text{ and,} \\ Risk(\gamma_i, \delta_i)_A \geq T(\gamma_i, \delta_i)_A \end{cases} \end{cases} & allow \\ \text{if } \begin{cases} Risk(\alpha_{ik}, \beta_{ik}) > T(\alpha_{ik}, \beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik}, \delta_{ik}) = T(\gamma_{ik}, \delta_{ik}) \text{ then,} \\ \text{if } Risk(\gamma_i, \delta_i)_A \geq T(\gamma_i, \delta_i)_A \end{cases} & allow \\ otherwise, deny \end{cases} \tag{4.2}$$

4.4.3 RAdAM: Cloud Federation

Data owner

Cloud federation is understood as indicating subjects from different cloud providers being able to access each other's data [74, 75, 76]. Cloud computing federation should cope with a heterogeneous environment and dynamic sets of users and access requests, and are under multiple administrative domains. There is a great deal of identity management that is involved in this paradigm and technologies like OpenID [77] appear to be quintessential in these settings. In our authorization mechanism, we ignore all the noise caused by the federation and only focus on the user who is sharing their data with another user from a different cloud. We believe that cloud federation incorporates both the simple cloud and the collaboration. Therefore, the parameters defined in the previous section are carried over to this section. To those parameters, we add the following:

- ϵ: represents the dynamic federation token for the user and the object he is trying to access.
- η: represents the token of the object that user is trying to access.
- $Risk(\epsilon, \eta)$: represents the risk of the federation request access.
- $T(\epsilon, \eta)$: represents the threshold of the federation risk access to which the federation risk will be compared to.
- $Risk(\epsilon, \eta)_A$: stands for the average federation risk and is calculated by dividing the sum of all the federation risks by the number of objects: $Risk(\epsilon, \eta)_A = \sum Risk(\epsilon, \eta)$ / n.
- $T(\epsilon, \eta)_A$: represents the average federation threshold risk and is computed by dividing the sum of all the federation risk thresholds by the number of objects: $T(\epsilon, \eta)_A = \sum T(\epsilon, \eta)$ / n.

The access request is allowed (respectively denied) if the risk for federation is strictly greater than (respectively, strictly lower than) the risk threshold for federation. The other parameters are used in the *situational factor*.

For an instance of the authorization mechanism in this federation case, we still consider the user s_i with his set of object O_i=$\{o_{i1},\ldots,o_{in}\}$. If the user wants to access the object o_{ik}, the risk federation is given by: $Risk(\epsilon_{ik}, \eta_{ik})$. The federation threshold is: $T(\epsilon_{ik}, \eta_{ik})$. The average federation risk is determined by: $Risk(\epsilon_i, \eta_i)_A = \sum_{m=1}^{z} Risk(\epsilon_{im}, \eta_{im})$ / z; z being the number of objects that are in federation mode. The average federation risk threshold is: $T(\epsilon_i, \eta_i)_A = \sum_{m=1}^{z} T(\epsilon_{im}, \eta_{im})$ / z. The authorization mechanism is represented in Equation 4.3.

Federative user

Cloud federation can be described as a cloud collaboration on a wider scale. In fact, Tang et al [78] designed an RBAC model for collaboration cloud services but exclusively used cloud federation, throughout their paper, to test their solution. In our system, a data owner in cloud federation has the right to allow a user from another cloud (federative user) to have access to their data. In this case, the data owner decides the risk level (user token and object token) of the federative user and can, at any moment, revoke their rights. The authorization mechanism for the federative user works as in Equation 4.2.

4.4.4 Outsourcing data without outsourcing control

It is a common knowledge that cloud computing prospective adopters are not comfortable knowing that the cloud service provider can access their data at anytime. This explains why many financial institutions prefer to build their own private cloud instead of using the public clouds. In this Section, we show that our model can be used to limit the power of the cloud administrator over the user's data. In a perfect world, the cloud admin should be limited to managerial rights over the user's data. We can use the risk bands as in collaboration to allow the user to have more power over his data that the cloud provider. But the concern that lingers is that the cloud admin administers the cloud without being able to have a minimal access to the users data.

$$
Au = \begin{cases}
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) > T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) > T(\epsilon_{ik},\eta_{ik}) \end{cases} \quad allow \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) > T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) > T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A \end{cases} \quad allow \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) = T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) > T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } Risk(\gamma_i,\delta_i)_A \geq T(\gamma_i,\delta_i)_A \end{cases} \quad allow \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) > T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) = T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } Risk(\epsilon_i,\eta_i)_A \geq T(\epsilon_i,\eta_i)_A \end{cases} \quad allow \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) = T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) > T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } \begin{cases} Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A \text{ and,} \\ Risk(\gamma_i,\delta_i)_A \geq T(\gamma_i,\delta_i)_A \end{cases} \end{cases} \quad allow \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) > T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) = T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } \begin{cases} Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A \text{ and,} \\ Risk(\epsilon_i,\eta_i)_A \geq T(\epsilon_i,\eta_i)_A \end{cases} \end{cases} \quad allow \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) = T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) = T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } \begin{cases} Risk(\gamma_i,\delta_i)_A \geq T(\gamma_i,\delta_i)_A \text{ and,} \\ Risk(\epsilon_i,\eta_i)_A \geq T(\epsilon_i,\eta_i)_A \end{cases} \end{cases} \quad allow \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) = T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) = T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } \begin{cases} Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A \text{ and,} \\ Risk(\gamma_i,\delta_i)_A \geq T(\gamma_i,\delta_i)_A \text{ and,} \\ Risk(\epsilon_i,\eta_i)_A \geq T(\epsilon_i,\eta_i)_A \end{cases} \end{cases} \quad allow \\
\text{deny otherwise}
\end{cases} \tag{4.3}
$$

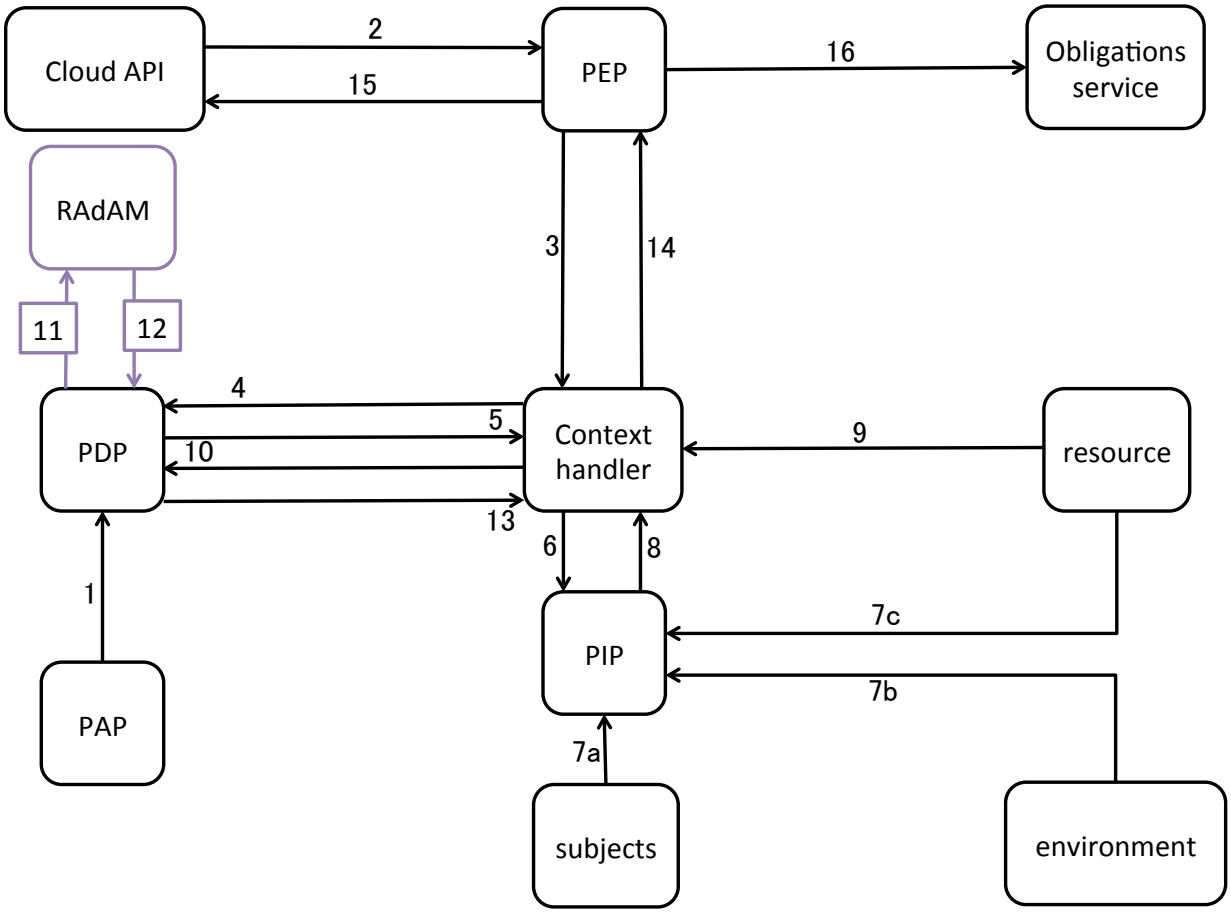

Figure 4.1: RAdAM/XACML architecture

4.4.5 Architecture of RAdAM

We use XACML 3.0 to leverage the architecture for our authorization mechanism. XACML is an extensible, XML-encoded language that provides format for authorization policies and access control decision request and responses. XACML 3.0 was approved as an OASIS standard on 22 January 2013 [67]. It includes a non-normative data flow model that describes the major components involved in processing access requests. Prior to any access request, the **policy administration point** (PAP) writes *policies* and *policy sets*, that represent the complete *policy* for a specified *target*, and transmits them to the **policy decision point** (PDP) (step 1). Via the cloud API, the user performs an access request (step 2). The **policy encryption point** (PEP) intercepts the request and forwards it to the **context handler** (step 3). The context handler converts the request into an XACML request contest and sends it to the PDP (step 4). The PDP requests to the context handler any supplemental attributes it might deem necessary to correctly evaluate the XACML request (step 5). The context handler request the attributes from a **policy information point** (PIP) (step 6). The PIP retrieves the requested attributes which include, among other things, all the parameters that we introduced in our authorization mechanism and then trans-

mits them to the context handler (steps 7-8). The context handler forwards the attributes to the PDP, and can optionally include the **resource** (steps 9-10). The PDP sends the attributes to the RAdAM module for evaluation (step 11). After evaluation as per of our authorization mechanism, the RAdAM module returns the decision to the PDP, which forwards it to the context handler (steps 12-13). The context handler translates the response to the native response format of the PEP and then sends it to the latter (step 14). The PEP transmits the response to the cloud API and fulfills the obligations, which include updating the tokens of the users (steps 15-16).

4.4.6 Risk assessment in XACML

XACML employs three top-level policy elements: <Rule>, <Policy> and <PolicySet>. The <Rule> element contains a Boolean expression that is meant to be evaluated in isolation within the XACML PAP. It is the most elementary unit of a policy. A <Rule> comprises an optional <Target> and <Condition> elements and an Effect attribute. The <Condition> defines a Boolean expression that further restricts the applicability of the rule that determines the outcome: either Permit or Deny. A <Rule> may also include obligation expressions that refer to operations that must be performed by the PEP in addition to enforcing the PDP's decision. Note that these are systems obligations, not user obligations. More than one rule defined by a policy may be relevant to a request, and so the rule-combining algorithm (Deny-overrides, as instance), is used to combine outcomes of these rules into a single decision. The <Policy> element is the basic unit of policy used by the PDP. It is composed of a set of <Rule> elements and plays a pivotal role in an authorization decision, a <Target> and a rule-combining algorithm. The <Target> defines a set of conditions that the attribute values in an access request must meet for the policy to apply to the request. The <PolicySet> element is the standard apparatus for combining separate policies into a single policy. It is made of a set of <Policy> and/or other <PolicySet> elements with a sound process to make the combination works by using a policy combining algorithm that defines the manner of combining the results of evaluating the policies. An instance of RAdAM in XACML expression is shown in Figure 4.2.

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- RAdAM Risk Mititgation Policy for the singleton user -->

<Policy
    xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
    http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
    PolicyId="urn:oasis:names:tc:xacml:3.0:radam:useradminpolicy"
    Version="1.0"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable">

    <VariableDefinition VariableId="risk">
        <AttributeValue>risk</AttributeValue>
    </VariableDefinition>

    <VariableDefinition VariableId="riskthreshold">
        <AttributeValue>threshold</AttributeValue>
    </VariableDefinition>

    <VariableDefinition VariableId="generalriskthreshold">
        <AttributeValue>generalriskthreshold</AttributeValue>
    </VariableDefinition>

    <VariableDefinition VariableId="generalrisk">
        <AttributeValue>generalrisk</AttributeValue>
    </VariableDefinition>

    <Rule RuleId="radam" Effect="Permit">
        <Condition>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-greater-than">
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-one-and-only">
                    <VariableReference VariableId="useradminriskthreshold"/>
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">riskvalue</AttributeValue>
                </Apply>

                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-one-and-only">
                    <VariableReference VariableId="riskthreshold"/>
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">threshold</AttributeValue>
                </Apply>
            </Apply>

            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">

                    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-equal">
                        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-one-and-only">
                            <VariableReference VariableId="useradminriskthreshold"/>
                            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">riskvalue</AttributeValue>
                        </Apply>

                        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-one-and-only">
                            <VariableReference VariableId="riskthreshold"/>
                            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">threshold</AttributeValue>
                        </Apply>
                    </Apply>

                    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
                        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-greater-or-equal">
                            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-one-and-only">
                                <VariableReference VariableId="generalrisk">
                                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">generalrisk</AttributeValue>
                            </Apply>
                            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-one-and-only">
                                <VariableReference VariableId="generalriskthreshold">
                                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">generalriskthreshold</AttributeValue>
                            </Apply>
                        </Apply>
                </Apply>
            </Apply>
        </Condition>
        <ObligationExpressions>User token update</ObligationExpressions>
    </Rule>
```

Figure 4.2: RAdAM expression in XACML

4.5 RAdAM Based on Fuzzy Inference System

Fuzzy inference systems (FIS) derive from fuzzy logic. Profesor L. A. Zadeth of the University of California Berkley invented fuzzy logic in 1965 [17]. Fuzzy logic is a multivalued logic that permits intermediate values to be expressed in conventional evaluations like true/false, yes/no, high/low, etc. Fuzzy systems are an alternative to traditional notions of set membership and logic. The very basic notion of fuzzy systems is a fuzzy set. The concept of fuzzy set extends the concept of a classical crisp set. A classical crisp set is a collection of objects in a given range with a sharp boundary, which means that a member either belongs to that set or does not. The fuzzy set is fundamentally a more flexible set as it allows its members to have a smooth boundary i.e., a member can belong to a set to some partial degree. The fuzzy set convincingly provides interpretations that are similar to a human being thought processes. The implementation of fuzzy logic requires the three following steps:

- *Fuzzification*: the first step to apply a fuzzy inference system. It involves two processes: derive the membership functions for input and output variables and represent them with linguistic variables. This process is equivalent to converting or mapping classical set to fuzzy set to varying degrees. The membership function is a graphical representation of the magnitude of the participation of each input. It associates a weight with each of the inputs that are processed, defines functional overlap between inputs, and utltimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion. In practice, membership functions can have a multitude of different types, such as the triangular waveform, trapezoidal waveform, Gaussian waveform, bell-shaped waveform, sigmoidal waveform and S-curve waveform. The exact type depends on the actual applications. A triangular or trapezoidal waveform should be utilized for systems that necessitate significant dynamic variations in a short period of time. Systems that require a very high control accuracy are better suited with a Gaussian or S-curve waveform.
- *Fuzzy control rules*: The fuzzy classifiers represent one application of fuzzy theory. Expert knowledge is used and can be expressed in a very human-like way using linguistic variables, which are described by fuzzy sets. The expert knowledge can be formulated as rules:
  IF A is X **AND** B is Y **THEN** C is Z
  Linguistic rules describing the control system consist of two parts; an antecedent block (between the **IF** and **THEN**) and a consequent block (following **THEN**).
- *Defuzzification*: The two steps explained above constitute the fuzzy inference system. The fuzzy conclusion (output) is a linguistic variable and needs to be reverted back to the crisp value: the process is denominated defuzzification.

4.5.1 RAdAM risk estimation in FIS

In order to explain RAdAM risk estimation in FIS, it is helpful to consider the following example. Let's suppose that we have a tuple (subject, object) in a cloud environment. The objects are defined by their sensitivity: unclassified (U), public trust (PT), confidential (C), secret (S),

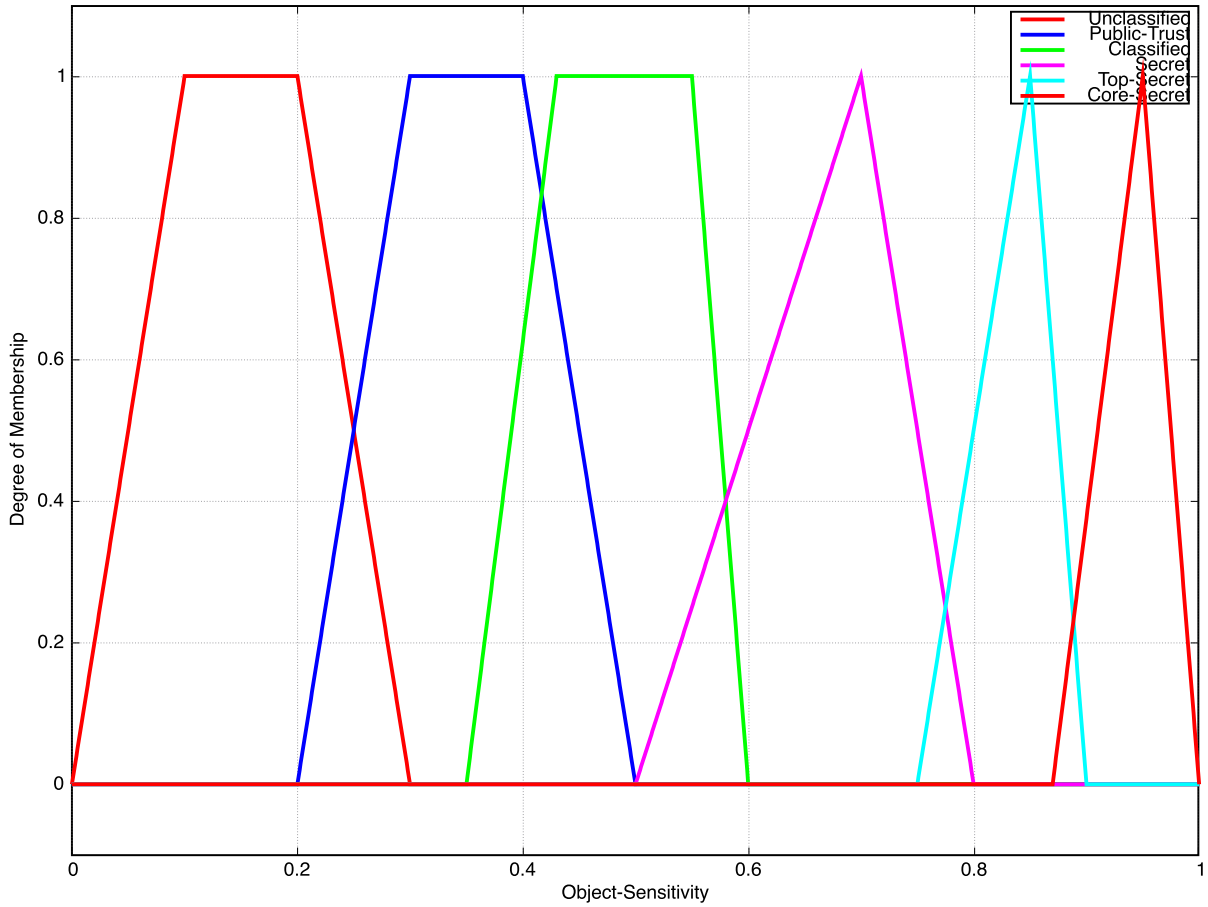

Figure 4.3: Membership functions of the object sensitivty

top secret (TS), and core secret (CS). Similarly, the different clearance levels for the subjects are as follows: unclassified (U), public trust (PT), confidential (C), secret (S), top secret (TS), and core secret (CS). As stated in previous sections, the risk is calculated in function of the subject clearance and object sensitivity and can be valued as negligible (N), low (L), medium (M), high (H), and very high (VH). The first step of the process of FIS consists of defining the input and output variables of our system. The inputs are the object sensitivity and the subject clearance and the output is the risk. We have decided to represent the clearance of the user in a trapezoidal membership function and the object sensitivity in a combination of trapezoidal and triangular membership functions. The risk is represented as a Gaussian membership function. The different membership functions are depicted in Figure 4.3, Figure 4.4, Figure 4.5. We subsequently establish the fuzzy classifier (if ... then), which is shown in Table 4.1. Different values of the risk are evaluated in Figure 4.6 in function of the subject clearance and the object sensitivity.

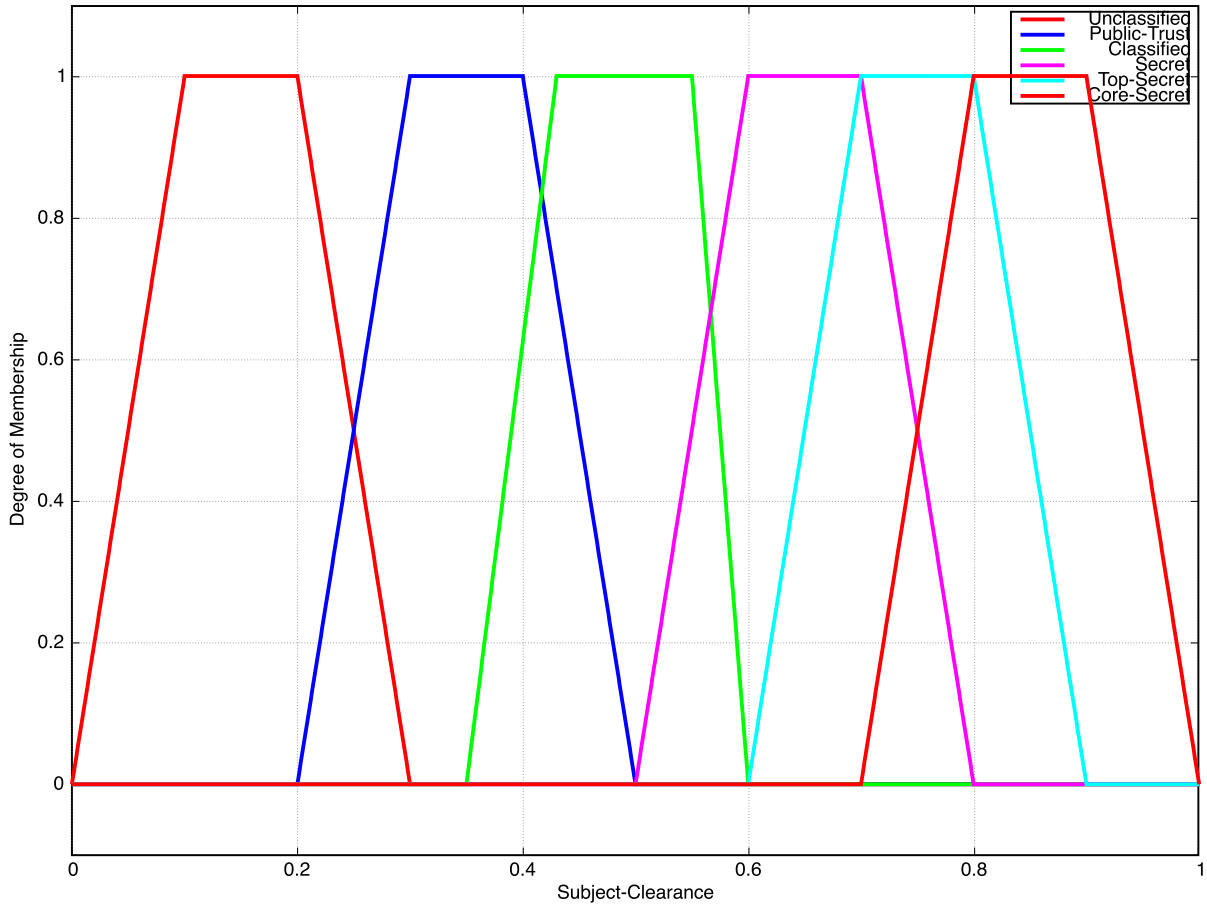

Figure 4.4: Membership functions of the subject clearance

4.5.2 RAdAM decision evaluation in FIS

The same process is conducted in order to evaluate the decisions of our authorization mechanism. It is important to note that the threshold is also a risk. In this fuzzy system, the risk and the threshold are the inputs and the decision (allow or deny) is the output. The output is considered a constant as represented in Figure 4.7. The schema of the decision results is depicted in Figure 4.8. We can clearly see the decisions depending of the values of the risk and the threshold.

4.6 Enforcement of RAdAM with Vulnerability Based Authorization (VBAM) Mechanism

Until now, researchers in the access control fields have not thought of using vulnerabilities score to determine access decisions. We propose such solution by introducing Vulnerability-Based Access Control. The modus operandi is to tailor an access to an object to the vulnerability level of both the requester and the object that is being requested. In traditional access control,

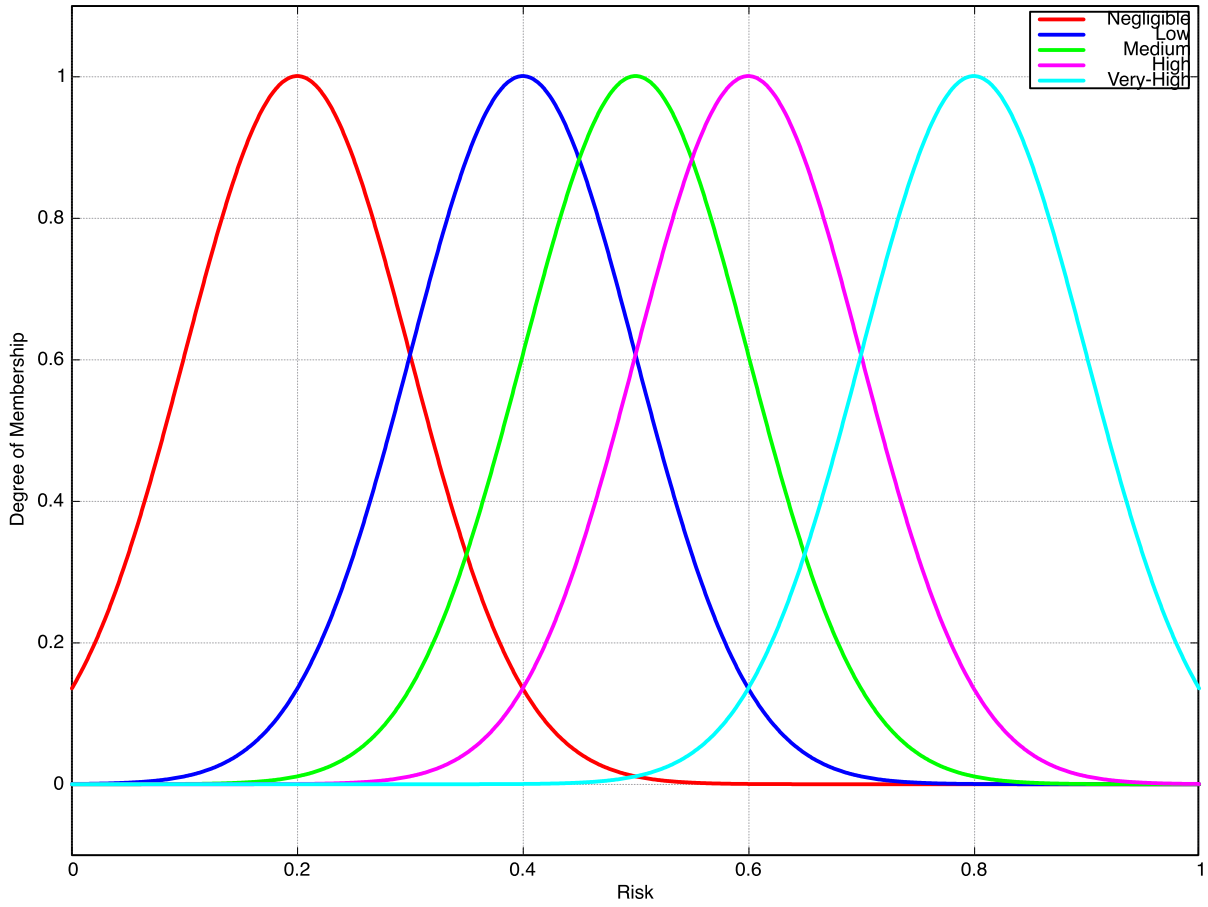

Figure 4.5: Membership functions of the risk level

users are granted access to the objects regarding the privileges (read, write, or execute). We contend that these types of access controls are obsolete in modern computer systems like cloud computing. For instance, let us say we have a user who has the privilege to read an object. In simplistic thinking we might think that the user has a low privilege over the object because he can only read it. But the reality is that, if the object has a vulnerability with a high impact, the user, if malicious, can exploit the vulnerability and damage the system. That is the reason why we are advocating the usage of a Vulnerability (Risk) Based Authorization Mechanism (VBAM) where the users access to an object will be decided by the security level of the user and the average vulnerability score of the object that is being accessed. In our model, there is need to be a standard vulnerability framework. The objects are classified according to their vulnerability score: risk level 1, risk level 2, . . . , risk level n. The users of the system are also classified according to the same multi-level security system. Whenever a user attempts an access, the system checks the security clearance of the user before computing the average vulnerability score of the object that is being accessed. If the user has a clearance level higher or equal to the clearance level of the object then the access is granted otherwise, it is denied. In this situation, if a

Table 4.1: List of the rules for RAdAM risk evaluation

S/O	U	PT	C	S	TS	CS
U	L	M	H	VH	VH	VH
PT	N	L	H	H	VH	VH
C	N	L	L	H	H	VH
S	N	L	L	L	H	VH
TS	N	N	N	M	L	H
CS	N	N	N	L	L	M

user can only access objects with a low risk level, and if they succeed to exploit the vulnerability, then the damage would not be significant.

4.6.1 Practical: VBAM with the Common Vulnerability Scoring System

The basic premise of the traditional MLS Bell Lapadula model is to determine if a subject is trustworthy enough and has the legitimate *need-to-know* to access an object. A subject is usually a person or an application running on behalf of a person. An object is usually a piece of information such as a file. Each subject or object is tagged with a *security label* which represents the sensitivity level. A subject's sensitivity level reflects the degree of trust placed on the subject. An object sensitivity level indicates how sensitive the object is or the magnitude of the damage incurred by an unauthorized disclosure of the object. A subject can *read* an object if their label dominates the object's label

4.6.2 NVD and CVSS

The National Vulnerability Database (NVD) [27] is a publicly available database for computer-related vulnerabilities. It is the 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) [28] 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.

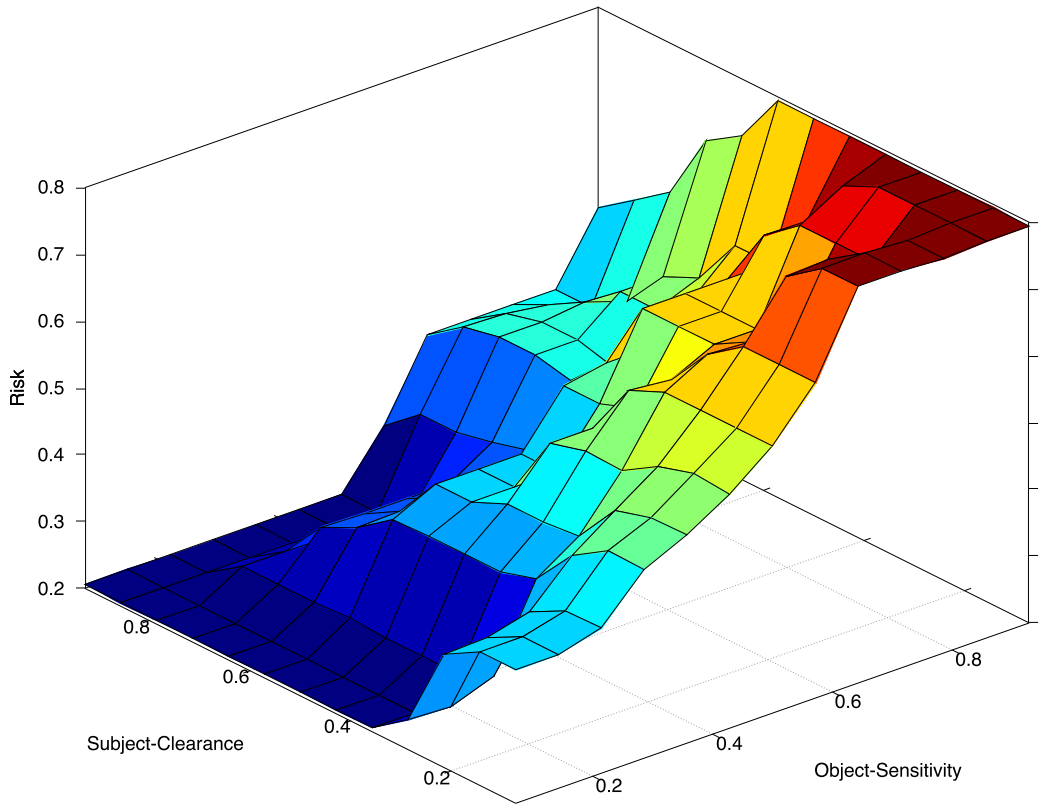

Figure 4.6: RAdAM risk evaluation

4.6.3 VBAM and CVSS

We aim to make use of the CVSS to demonstrate the suitability of our proposal. We suppose that an OpenStack environment was installed in an Internet Engineering laboratory; we have to use VBAM to determine which group of users (faculty, Ph.D. students, master students) can access to which OpenStack services in the cloud environment. OpenStack is the most popular open-source cloud management platform. In this study, we consider ten of its services, which are briefly described hereafter. Dashboard, also called *Horizon*, provides a web portal for the management of the underlying OpenStack services. Compute or *Nova* facilitates the management of OpenStack's instances. Networking, codenamed *Neutron*, this service, not only permits network connection between OpenStack's services, but also allows users to configure networks by putting an API into their disposition. Object Storage helps with the storage and retrieval of arbitrary unstructured data objects. It is also known as *Swift*. Block storage or *Cinder* provisions persistent block storage to running instances. The identity service is responsible of the identity management (authentication, authorization, endpoints) for the rest of OpenStack's services. This service is codenamed *Keystone*. The image service, codenamed *Glance*, takes charge of the stor-

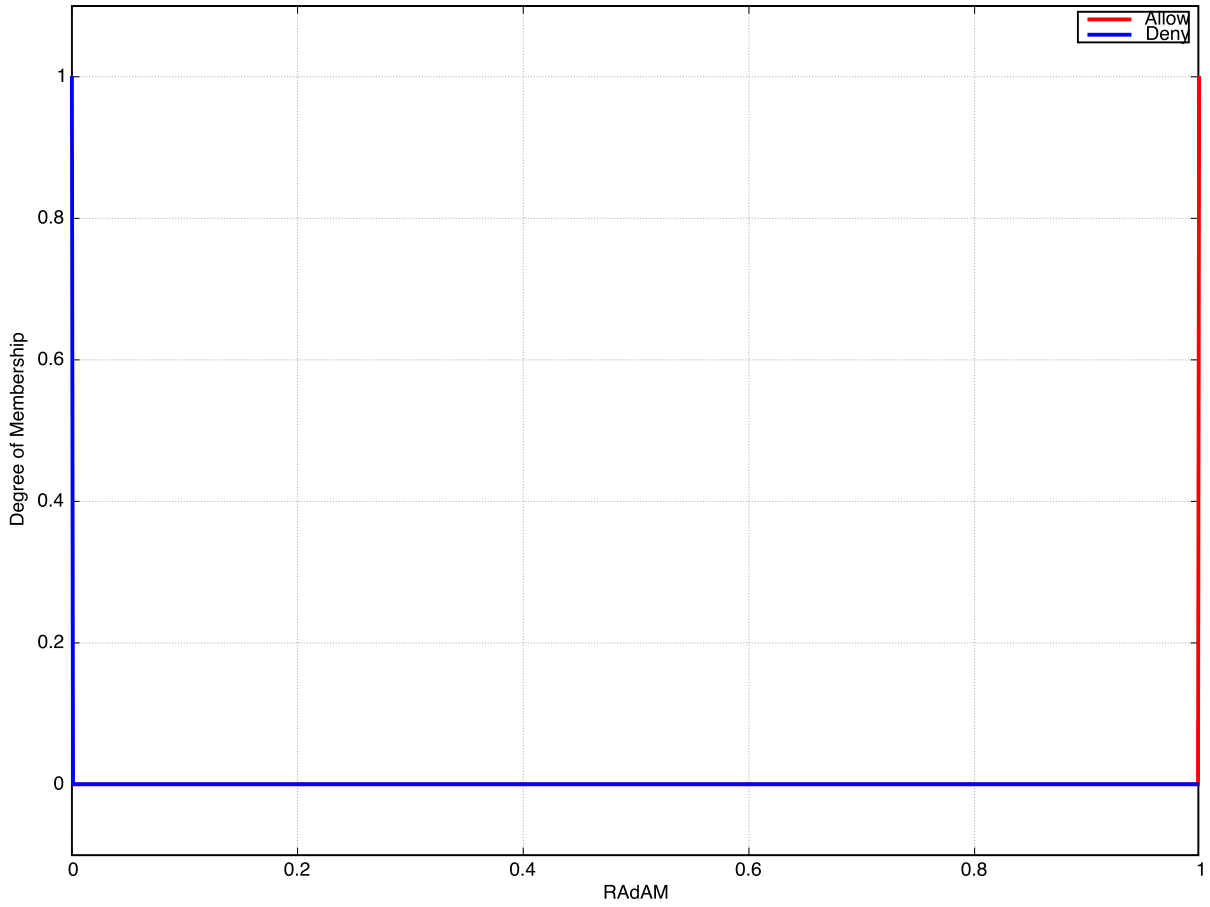

Figure 4.7: Membership functions of RAdAM decisions

age and the retrieval of virtual machine disk images. Telemetry, codenamed *Ceilometer*, helps monitoring and metering the business aspects of OpenStack like billing or benchmarking. Orchestration, or *Heat*, facilitates the orchestration of multiple composite cloud applications. The Database as a Service, named *Trove*, provides a scalable and reliable cloud database provisioning functionality.

We know that the CVSS provides vulnerability scores that range between 0 to 10 and following the NVD severity ranking we have: LOW (0 - 3.9), MEDIUM (4 - 6.9), HIGH (7 - 10). We apply the same classification to the users of the laboratory to get the following:

- Master students → LOW
- Ph.D. students → MEDIUM
- Faculty members → HIGH

This means that users with a security level of LOW can only access objects with a similar security level. Users with a security level of MEDIUM, can access objects with security levels of MEDIUM and LOW. Finally, users with a security level of HIGH, can access all the services. We conducted a static method by retrieving from the NVD all the vulnerabilities of the different

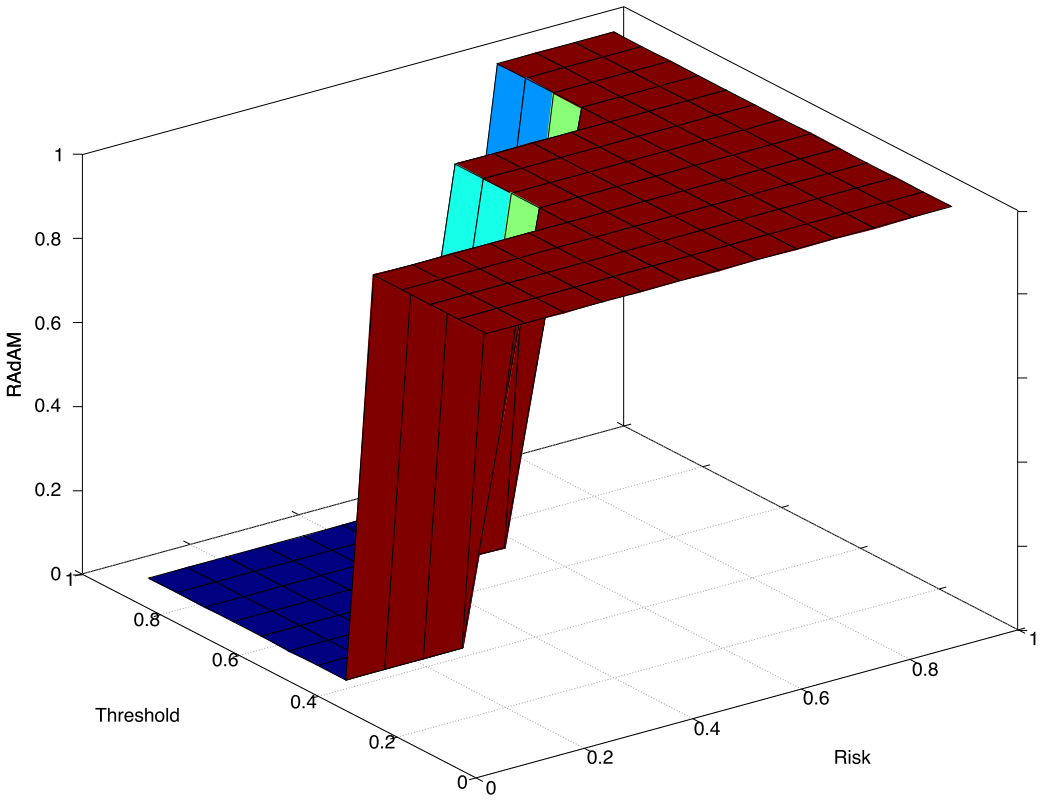

Figure 4.8: RAdAM decision evaluation

services of OpenStack that are in play in this paper, and their respective vulnerabilities. These details are recorded in Table 4.2. The last row of the table represents the average score of the vulnerabilities of the different services. Henceforth, a straightforward analysis of the table reveals that VBAM allows the faculty members to access all the services while the master students can only access *Cinder*, *Trove* and *Ceilometer*.

4.7 Discussion

In this section, we elaborate on different scenarios in which RAdAM could be used. Furthermore, we propose a solution to thwart some of the limitations of RAdAM.

We believe that the main weakness of RAdAM resides in the fact that we did not take into account many aspects of cloud federations, especially with regards to federated identity management. In the future, we will try to find how to integrate RAdAM in federation mode with federated identity management. The main issue with risk aware access control in general is the cost of computation. Our proposal does not solve the issue. The excitement about RAAC re-

Table 4.2: Vulnerabilities of the different services of OpenStack

	H	G	Ne	N	K	S	C	T	Ce	H
Average Score	4.3	4.4	5.1	4.3	4.8	5.3	3.4	2.1	3.5	4.1

volves around the fact that in each access request, the risk will be evaluated. This certainly helps in making better access decisions but it is costly in terms of resource consumption. Furthermore, as stated by Ni et al. [69], an attacker can launch attacks in the time window between the access request and risk mitigation. In the future, we will try to see how we can leverage the computing power offered by cloud computing to resolve this issue. We believe that if our proposal is implemented in a system with 'unlimited' computer power, as the cloud offer, the issue will become obsolete.

We also proposed VBAM, a vulnerability (or risk) based authorization mechanism that can cope with the dynamicity of modern computer systems like cloud computing. We contend that the future of access control revolves around tailoring the access decisions to the vulnerability score of the objects. We are aware that most attacks are the results of exploited vulnerabilities. Therefore, to limit the damages of a probable attack, we must adopt VBAM and other similar authorization mechanisms. One of the main issues of risk aware access control in general is computational cost: our proposal is not an exception to that rule. Indeed, the fact that the system is at work every time a user makes a request results in high resource usage. The second point of contention relates to how our method can be used in an environment. We contend that it can be used in a kind of 2-step access control. Where VBAM will be used as primary authorization mechanism and a second authorization mechanism will decide whether the user can read write or execute the objects they is allowed to access to by VBAM. One might argue that our proposal does not solve the problem of authorization mechanism because a user with a high security level can exploit a vulnerability that has an equally high damage factor. While this is true, we argue that, in our system, users with a high security level are trusted users and should be a limited number. Therefore detecting the culprit in any of these types of attacks is easy, as we already know the possible damages that each user can make. The same applies for the other security levels. As future work, we seek to implement our system in combination with the XACML framework in order to provide a more flexible authorization mechanism for the cloud. The security risk model we developed in Chapter 3 could be used to express risk in VBAM. The principle is simple and does not necessarily require building a vulnerability tree. The modus operandi is as follows: first we query all the vulnerabilities related to the object that is being accessed and then we apply the inclusion-exclusion law to determine the likelihood and the impact before doing the calculation for the risk. This operation can be tedious in case we have a big number of vulnerabilities.

4.8 Summary

We proposed a risk-aware authorization mechanism flexible enough to deal with the dynamicity of cloud computing. We proposed algorithms for simple user access, collaboration and federation modes. To complement RAdAM, we proposed a variation of MLS model where the risk is represented by the vulnerability score of the objects. We showed the applicability of our methods by using fuzzy inference systems for RAdAM and by proposing an OpenStack use case for VBAM. We contend that a combination of authorization mechanisms similar to the one we proposed in this paper represents the future of highly dynamic systems.

Chapter 5

Future Work

The work we have done in this research could be extended to produce further research; in Chapters 3 and 4 we have already highlighted some avenues for probable future work. Risk aware access controls (RAAC) constitute the present and future of access control mechanisms. Other techniques like machine-learning (ML) can be incorporated into the mix to make RAAC more efficient. We believe that we have realized a decent work in this thesis; we also believe that what we have is the tip (a big tip nevertheless) of the iceberg of what is to come in the future.

In Chapter 4, we have developed our approache based on industry and consumer needs and evaluated its applicability with the application scenarios. Our proposal is a solution to quantify sophisticated security attacks in IaaS as a service cloud computing. 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 vulnerability and to prioritize their response. The difference is they do it with single isolated vulnerabilities. They don't have a response in case of mixed, combined vulnerabilities, our proposal is a response in those particular cases. The first to benefit from this proposal is obviously an administrator of an IaaS cloud system. In practice, what often happens is vulnerabilities are discovered in systems but are not patched due to environmental, cost, or mission factors. Our proposal allows the administrator to prioritize the patch of the vulnerabilities. In the case of environmental factors, where there are no patches for the discovered vulnerabilities, the admin can use our model to evaluate the security of different scenarios where those vulnerabilities can be combined and used by an attacker. Depending on the results the admin can take simple to drastic decisions going from a real time monitoring of the concerned services to shutting them down until patches become available. In the case of cost factors, where the patches for the scanned vulnerabilities are monetized, the result from our quantification model can tell the admin if he/she can be frugal or not. In the case of mission factors, the administrator can use our model to convince the consumers of the necessity to stop some services momentarily to secure the system by patching the vulnerabilities.

A consumer of a cloud service provider can use our model to check the security level of the CSP. Indeed a consumer of an IaaS CSP has some information about the type of resources that are used. Knowing what kind of hypervisor the CSP uses and the different services he/she is running there, he/she can simulate some scenarios by using vulnerabilities from the NVD related to the services he/she subscribed for. For example if a user wants to subscribe to amazon web services (AWS), knowing that they are using a highly modified version of Xen version 3, he/she

can simulate some plausible attack scenarios and obtain their security level by using our method. Cloud computing is here to stay and will inevitably have more users. The development of security models where users will be able to check the security of a cloud provider before subscribing to its services will be indispensable. Those models will allow the users to classify, security-wise, the different cloud service providers available in the market. Our proposal is a good premise for this kind of model.

Henceforward, we try to explain some of the projects that we want to pursue in the short and long term.

5.1 Classification of the security bugs in the cloud

There is a prescient need to classify the security bugs in the cloud as the technology keep growing and, by ricochet, is generating other emergent paradigms. Gunawi et al. [79] presented a work that set the tone in the study of bugs in cloud computing. The title of their paper is somehow misleading, What Bugs Live in the Cloud, as they only cover bugs related to the bug data tools: Hadoop MapReduce, Hadoop File System (HDFS), HBase, Cassandra, ZooKeeper and Flume. Additionally, they focused on any types of bugs that exist within the tools cited above. We promote a wider coverage of the cloud software by including others like cloud management stacks and we also want to limit to study to security bugs solely. At the end of the study, we should be able to identify all the security bugs or vulnerabilities that exist in cloud computing and provide a guideline for best practices when it comes to developing cloud software. In the same occasion, we can curate security vulnerabilities databases by eliminating any redundant vulnerability. In fact, we have thousands of vulnerabilities that are references in vulnerabilities databases. There might be numerous vulnerabilities that are intrinsically similar but appear in different products. We believe that there should be a common semantic, for all the vulnerabilities, that will help developers, regardless of their programming languages, to provide more secure software.

5.2 Crankenstein: towards the most efficient cloud management stack

Cloud management frameworks are on the rise. Organizations are adopting the different software on the market in order to manage their private cloud or develop a business to provide cloud services to the users. Currently, most organizations are directed towards the open source cloud management frameworks like OpenStack, CloudStack, or Eucalyptus. We have already shown in Chapter 2 that these software are composed of different stacks or component. In this project, we want to select the best cloud frameworks and evaluate their different corresponding components. Dukaric et al [1] have already done an excellent job in establishing a taxonomy of cloud software and classifying the components of the most popular cloud software. The evaluation will be diverse: performance, scalability, security, etc. The goal is to assemble the best cloud management framework out of the best performing components: C-rankenstein. We are aware of the fact the

code of some of these software, CloudStack, is monolithic; this might implies an exclusion of any software that presents that specificity.

5.3 Elo rating method for ranking cloud service providers

The Elo rating is very popular for ranking entities (players, teams, etc.) in world-class competition like chess, football or tennis. It has a very good reputation and allows tournament organizers to efficiently pair the competitors so that they will not be having the best competitors competing against each other early in the tournament. We believe that interdisciplinary techniques can help solve issues. In this situation, we contend that the Elo rating method can be adopted and adapted in the cyber security field so that we will have a ranging based on different parameters like number of vulnerabilities, vulnerability scores, vulnerability density, etc. to the best of our knowledge, only Pieters et al. [80] attempted such an adaptation. We contend that such a ranking platform will help make the cloud in particular, any computer system more secure.

5.4 Game-theoretic approach: Stackelberg Security Games and Cloud Computing

In game theory, the work of Jon Nash is the most recognized and has been used in many different domains. However, recently, Stackelberg security games (so named due to their origins in the work of Heinrich von Stackelberg) are drawing a lot of attention when it comes to game-theoretic approaches to security. In the Stackelberg security game, the defender acts first by committing to a strategy afterwards, the attacker choses where and how to attack after observing the defenders choice. The leader-follower paradigm appears to fit many real world security situations. Tambe et al. [81] used Bayesian Stackelberg games to successfully deploy Assistant for Randomized Monitoring over Routes (ARMOR) at the Los Angeles International Airport (LAX) since 2007. They also proposed: Intelligent Randomization in Scheduling (IRIS) for the U.S. Federal Air Marshal services, Game-theoretic Unpredictable and Randomly Deployed Security (GUARDS) for the U.S. Transportation Security Administration, and Port Resilience Operational/Tactical Enforcement to Combat Terrorism (PROTECT) for the U.S. Cost Guard. In cloud computing systems, we have administrators and customers. The administrators set the security of the cloud environment and wait for new customers to enroll into their services. We have a perfect case of leader (cloud administrator) and follower (customer). The research question is how we can design a security system that follows the principle of leader-follower that is stable enough that once the cloud administrator fixes the security of the cloud system, the malicious customer will not be able to hack into it. We believe that Stackelberg security games appeals to the cloud because in cloud computing, administrators attempt to secure the system with limited resources: Stackelberg security games has that property.

5.5 Deep cloud security: on using artificial intelligence to design a secure cloud platform

In recent years, there is growing worry among thinkers on the role of artificial intelligence in our life. We have self-driving cars, computers that can outperform human beings in some specific tasks that involve cleverness and so on. People are worrying that the machines will take over as they are becoming more and more autonomous. We do not believe that the machines will ever take over the world but we do believe that it is time for cyber security to benefit of artificial intelligence. The idea is to have a system that will be able to recognize the security vulnerabilities and mitigate them by itself. One way to do it would be to look at all the vulnerabilities that have been discovered, to look exactly the causes of those vulnerabilities and to generate a machine-learning algorithm that will learn all security bugs so that it can detect future security bugs and mitigate them. It sounds difficult but it shouldn't be if we follow the footsteps of our colleagues from other fields. The basic methodology used by our colleagues from computer vision and natural language processing is to feed algorithm with very large datasets. Through iteration and other neural network techniques, the algorithm learns to identify and describe images or texts without any human intervention. We can adopt the same methodology in cyber security as we have big datasets at our disposal. We can mimic their algorithms, adapt them to fit the cyber security requirements, feed them with them with security bugs data and observe whether they can recognize security bugs automatically.

5.6 Towards the real cloud computing architecture

When I first heard about cloud computing, I literally thought that the computing was coming from the cloud. Later on, after dive deeply into the literature, I finally understood the technical part of cloud computing. I understood that this enigmatic technology was coined "cloud because the users data are off-premises; the data is not in the users hand anymore and is managed by cloud administrators that run the datacenters of the different cloud service providers. We believe that it is a fairly good advancement in term of the development of technology. But some people have accused the evangelists of cloud computing as usurpers because they believe cloud computing is not new, the architecture has been proposed a while ago, and that the development of the Internet helped bring back those architectures. Therefore they see cloud computing more as an evolution than a revolution. In order to (re)establish cloud computing as an evolution, we propose a new architecture where there will be no datacenters. In fact we believe that, instead of the data being stored in datacenters, it should be kept moving on the Internet. This idea might look superfluous but it is doable. As instance, when we browse the Internet and click on a photo, the photo is downloaded from the server where it is hosted, transported through the Internet and then delivered to our web client. The TCP/IP model gives a description of how the Internet works currently. We contend that regarding the amount of data we have in the internet now, the cost to maintenance the datacenter and so on, the TCP/IP model should be amended for a better future Internet. This is a great challenge and will sure come with great obstacles both technical and social. But, with the breadth of technological advancements such software defined network (SDN) and content centric network (CCN), we believe that it is feasible.

Overall, we try to provide new solutions to facilitate decision-making and provide an all around secure cloud platforms. We find that they are independent organizations like the FIDO alliance that are doing excellent work in this field. We plan to find solutions to incorporate their work into ours towards better and more secure technologies.

Chapter 6

Conclusion

In this dissertation, we investigated and addressed fundamental problems of security issues and the absence of a suitable authorization mechanism for cloud computing. In Chapter I, we set the basements of our thesis by listing the security issues of cloud computing and the scope of our research. We used the many survey papers about cloud computing to establish a list. In Chapter 2, we gave an in-depth explanation of cloud computing. We explained its main services, its technical and economic aspects. But most importantly, we succeeded to prove, through a vulnerability analysis of OpenStack vulnerabilities, that all the technical security issues encountered in cloud computing were represented in form of vulnerabilities. We concluded that then the all-in-one solution for cloud security must be a security quantification based on the vulnerabilities, which we tackle in Chapter 3. We developed our security quantification process. We set our scopes to be all types of security in cloud computing specially multi-tenancy and the security of cloud management stacks. We built our method by using fault tree analysis as a blueprint. 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: $R = L \times I$. 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. We want to reiterate that the main conclusion of this chapter is the fact that most technical vulnerabilities in cloud are represented in form of vulnerabilities and that our security quantification allows to alleviate the risk related to the aforementioned vulnerabilities. In Chapter 4, after we proposed a solution that we contend can solve most cloud security issues available in the different survey papers, we attacked a different problem: the dynamicity of cloud computing versus the static traditional access control mechanisms that are used to enforce the security of access request decisions in cloud computing. The

problem of a non-suitable access control for dynamic systems is not new. In fact, in the early 2000s the JASON report [15] first highlighted the necessity to built adequate access controls for those types of systems. Cloud computing just aggravates the issue. We propose a Risk-Adaptive Authorization Mechanism (RAdAM) to solve the issue. We considered different situations in cloud computing: simple user, user in collaboration, and federation mode. We were able to evaluate our authorization mechanisms by using Fuzzy Inference Systems (FIS). We proposed a Vulnerability-Based Authorization Mechanism (VBAM) to complement RAdAM or any other type of authorization mechanisms. In Chapter 5 we discuss about the future work. We consider that there are many interesting avenues for research in cloud security: deep security, Elo security rating, Stackelberg security games in cloud.

Appendix A

List of Publications

A.1 Journal

1. **Doudou Fall**, Takeshi Okuda, Youki Kadobayashi, and Suguru Yamaguchi. Risk-Adaptive Authorization Mechanism in Cloud Computing. Journal of Information Processing (JIP), Special Issue of "Internet and operation technologies in the era of cloud." (conditional acceptance). Chapter 4.
2. **Doudou Fall**, Takeshi Okuda, Youki Kadobayashi, and Suguru Yamaguchi. Security Risk Quantification Mechanism for Infrastructure as a Service Cloud Computing Platforms. Journal of Information Processing (JIP), Special Issue: Applications and the Internet in Conjunction with Main Topics of COMPSAC 2014, Vol. 23, No. 4. July 2015. Chapter 3.

A.2 International Conferences

1. **Doudou Fall**, Takeshi Okuda, Youki Kadobayashi, and Suguru Yamaguchi. Towards A Vulnerability Tree Security Evaluation of OpenStack's Logical Architecture. In Thorsten Holz and Sotiris Ioannidis, editors, The 7th International Conference on Trust & Trustworthy Computing (TRUST2014), Vol. 8564 of Lecture Notes in Computer Science, pp. 127-142. Springer International Publishing, July 2014. Chapter 3.

2. **Doudou Fall**, Noppawat Chaisamran, Takeshi Okuda, Youki Kadobayashi, and Suguru Yamaguchi. Security Quantification of Complex Attacks in Infrastructure as a Service Cloud Computing. In 3rd International Conference on Cloud Computing and Services Science (CLOSER 2013) poster presentation, May 2013. Chapter 3.

3. Takeshi Takahashi, Gregory Blanc, **Doudou Fall**, Youki Kadobayashi, Hiroaki Hazeyama, and Shin'ichiro Matsuo. Enabling Secure Multitenancy in Cloud Computing: Challenges and Approaches. In Proceedings of 2nd Baltic Congress on Future Internet Communications (BCFIC), April 2012.

4. **Doudou Fall**, Gregory Blanc, Takeshi Okuda, Youki Kadobayashi, and Suguru Yamaguchi. Toward Quantified Risk-Adaptive Access Control for Multi-tenant Cloud Computing. In Proceedings of the 6th Joint Workshop on Information Security (JWIS2011), October 2011. Chapter 4.

Bibliography

[1] Robert Dukaric and Matjaz B Juric. Towards a unified taxonomy and architecture of cloud frameworks. *Future Generation Computer Systems*, 29(5):1196–1210, 2013. (document), 2.6, 2.5, 5.2

[2] S Özkan. Cve details: The ultimate security vulnerability datasourcevulnerabilities by type. (document), 2.6

[3] Douglas F Parkhill. Challenge of the computer utility. 1966. 1.1

[4] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011. 1.1, 2.1.1, 2.1.4, 3.1, 4.1

[5] Hassan Takabi, James BD Joshi, and Gail-Joon Ahn. Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, 8(6):24–31, 2010. 1.2, 3.1

[6] Luis M Vaquero, Luis Rodero-Merino, and Daniel Morán. Locking the sky: a survey on iaas cloud security. *Computing*, 91(1):93–118, 2011. 1.2, 3.1

[7] Mohamed Almorsy, John Grundy, and Ingo Muller. An analysis of the cloud computing security problem. In *Proceedings of APSEC 2010 Cloud Workshop, Sydney, Australia, 30th Nov*, 2010. 1.2, 3.1

[8] Daniele Catteddu. *Cloud Computing: benefits, risks and recommendations for information security*. Springer, 2010. 1.2, 3.1

[9] Minqi Zhou, Rong Zhang, Wei Xie, Weining Qian, and Aoying Zhou. Security and privacy in cloud computing: A survey. In *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on*, pages 105–112. IEEE, 2010. 1.2, 3.1

[10] Siani Pearson and Azzedine Benameur. Privacy, security and trust issues arising from cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 693–702. IEEE, 2010. 1.2, 3.1

[11] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 199–212. ACM, 2009. 1.2, 2.5, 3.1, 3.3.1

[12] Richard Chow, Philippe Golle, Markus Jakobsson, Elaine Shi, Jessica Staddon, Ryusuke Masuoka, and Jesus Molina. Controlling data in the cloud: outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 85–90. ACM, 2009. 1.2, 3.1

[13] Cloud Security Alliance. 1.2

[14] Top Threats Working Group et al. The notorious nine: cloud computing top threats in 2013. *Cloud Security Alliance*, 2013. 1.2

[15] Program Office JASON. Horizontal integration: Broader access models for realising information dominance. Technical report, Report JSR-04-132, The MITRE Corporation, JASON Program Office, December 2004. Available at http://fas.org/irp/agency/dod/jason/classpol.pdf, 2004. 1.2, 4.1, 4.2, 6

[16] R McGraw. Risk adaptive access control (radac). In *Proceedings of NIST & NSA privilege management workshop*, 2009. 1.3, 4.1, 4.2, 4.3

[17] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965. 1.3, 4.1, 4.5

[18] D Elliott Bell and Leonard J LaPadula. Secure computer systems: Mathematical foundations. Technical report, DTIC Document, 1973. 1.3, 4.1, 4.3

[19] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009. 2.1, 2.1.1

[20] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010. 2.1.1

[21] Nikolas Roman Herbst, Samuel Kounev, and Ralf Reussner. Elasticity in cloud computing: What it is, and what it is not. 2.4

[22] Federico Etro. The economics of cloud computing, 2011. 2.4

[23] OpenStack Community. Openstack cloud software. 3.1

[24] OpenNebula Community. Opennebula cloud. 3.1

[25] CloudStack Community. Apache cloudstack software. 3.1

[26] Eucalyptus Community. Eucalyptus cloud. 3.1

[27] Computer Security Division. National vulnerability database. 3.1, 4.6.2

[28] Peter Mell, Karen Scarfone, and Sasha Romanosky. A complete guide to the common vulnerability scoring system version 2.0. In *Published by FIRST-Forum of Incident Response and Security Teams*, pages 1–23, 2007. 3.1, 3.3.3, 4.6.2

[29] OpenStack Community. Openstack logical architecture. 3.1, 3.6.1

[30] Doudou Fall, Noppawat Chaisamran, Takeshi Okuda, Youki Kadobayashi, and Suguru Yamaguchi. Security quantification of complex attacks in infrastructure as a service cloud computing. In *Proceedings of the 3rd International Conference on Cloud Computing and Services Science*, 2013. 3.2

[31] Ennan Zhai, David Isaac Wolinsky, Hongda Xiao, Hongqiang Liu, Xueyuan Su, and Bryan Ford. Auditing the structural reliability of the clouds. Technical report, Technical Report YALEU/DCS/TR-1479, Department of Computer Science, Yale University, 2013. Available at http://www.cs.yale.edu/homes/zhai-ennan/sra.pdf, 2013. 3.2

[32] Hongda Xiao, Bryan Ford, and Joan Feigenbaum. Structural cloud audits that protect private information. In *Proceedings of the 2013 ACM workshop on Cloud computing security workshop*, pages 101–112. ACM, 2013. 3.2

[33] Rasib Hassan Khan, Jukka Ylitalo, and Abu Shohel Ahmed. Openid authentication as a service in openstack. In *Information Assurance and Security (IAS), 2011 7th International Conference on*, pages 372–377. IEEE, 2011. 3.2

[34] Sasko Ristov, Marjan Gusev, and Aleksandar Donevski. Openstack cloud security vulnerabilities from inside and outside. In *CLOUD COMPUTING 2013, The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 101–107, 2013. 3.2

[35] Aleksandar Donevski, Sasko Ristov, and Marjan Gusev. Security assessment of virtual machines in open source clouds. In *Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on*, pages 1094–1099. IEEE, 2013. 3.2

[36] Aryan TaheriMonfared and Martin Gilje Jaatun. As strong as the weakest link: Handling compromised compoenents in openstack. In *Proceedings of the third IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2011. 3.2

[37] Xinming Ou, Wayne F Boyer, and Miles A McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 336–345. ACM, 2006. 3.2

[38] Xinming Ou, Sudhakar Govindavajhala, and Andrew W Appel. Mulval: A logic-based network security analyzer. 3.2

[39] Michael Lyle Artz. *Netspa: A network security planning architecture*. PhD thesis, Massachusetts Institute of Technology, 2002. 3.2

[40] Sushil Jajodia, Steven Noel, and Brian OBerry. Topological analysis of network attack vulnerability. In *Managing Cyber Threats*, pages 247–266. Springer, 2005. 3.2

[41] Tenable Network Security. Nessus vulnerability scanner. 3.2

[42] BeyondTrust Corporation. Retina network security scanner. 3.2

[43] Tripwire Company. Tripwire ip360 vulnerability & risk management. 3.2

[44] Skybox Corp. Risk analytics for cyber security. 3.2

[45] ReadSeal Inc. Read seal systems. 3.2

[46] Mohamed Ben-Daya. *Failure Mode and Effect Analysis*. Springer, 2009. 3.2

[47] James J Rooney and Lee N Vanden Heuvel. Root cause analysis for beginners. *Quality progress*, 37(7):45–56, 2004. 3.2

[48] William E Vesely, Francine F Goldberg, Norman H Roberts, and David F Haasl. Fault tree handbook. Technical report, DTIC Document, 1981. 3.2, 3.3.2

[49] Sandra M. E. Wint. An overview of risk. 3.3

[50] Lindy Ellis. Reliability of systems, equipment and components - guide to fault tree analysis. 3.3

[51] Gary Stoneburner, Alice Goguen, and Alexis Feringa. Risk management guide for information technology systems. *Nist special publication*, 800(30):800–30, 2002. 3.3

[52] Tim Bedford and Roger Cooke. *Probabilistic risk analysis: foundations and methods*. Cambridge University Press, 2001. 3.3.2, 3.3.2

[53] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2010. 3.5

[54] Saurabh Kumar Garg, Steven Versteeg, and Rajkumar Buyya. Smicloud: A framework for comparing and ranking cloud services. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 210–218. IEEE, 2011. 3.5

[55] Hangwei Qian, Hualong Zu, Chenghua Cao, and Qixin Wang. Css: Facilitate the cloud service selection in iaas platforms. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, pages 347–354. IEEE, 2013. 3.5

[56] Jonas Repschlaeger, Stefan Wind, Ruediger Zarnekow, and Klaus Turowski. A reference guide to cloud computing dimensions: Infrastructure as a service classification framework. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 2178–2188. IEEE, 2012. 3.5

[57] Shira Ovide. A price war erupts in cloud services. 3.5

[58] Brant A Cheikes, David Waltermire, and Karen Scarfone. Common platform enumeration: Naming specification version 2.3. *NIST Interagency Report 7695, NIST-IR*, 7695, 2011. 3.7

[59] Andrew Buttner and Neal Ziring. Common platform enumeration (cpe) - specification version 2.2. *The MITRE Corporation - National Security Agency*, 2009. 3.7

[60] Hakan Lindqvist. Mandatory access control. *Master's Thesis in Computing Science, Umea University, Department of Computing Science, SE-901*, 87, 2006. 4.2

[61] Deborah D Downs, Jerzy R Rub, Kenneth C Kung, and Carole S Jordan. Issues in discretionary access control. In *2012 IEEE Symposium on Security and Privacy*, pages 208–208. IEEE Computer Society, 1985. 4.2

[62] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996. 4.2

[63] Vincent C Hu, D Richard Kuhn, and David F Ferraiolo. Attribute-based access control. *Computer*, (2):85–88, 2015. 4.2

[64] Doudou Fall, Takeshi Okuda, Youki Kadobayashi, and Suguru Yamaguchi. Toward quantified risk-adaptive access control for multi-tenant cloud computing. pages 1–14, 2011. 4.3

[65] Daniel Ricardo dos Santos, Carla Merkle Westphall, and Carlos Becker Westphall. Risk-based dynamic access control for a highly scalable cloud federation. In *SECURWARE 2013, The Seventh International Conference on Emerging Security Information, Systems and Technologies*, pages 8–13, 2013. 4.3

[66] Daniel Ricardo dos Santos, Carla Merkle Westphall, and Carlos Becker Westphall. A dy-

namic risk-based access control architecture for cloud computing. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–9. IEEE, 2014. 4.3

[67] Erik Rissanen. extensible access control markup language (xacml) version 3.0, 2013. 4.3, 4.4.5

[68] Pau Chen Cheng, Pankaj Rohatgi, Claudia Keser, Paul A Karger, Grant M Wagner, and Angela Schuett Reninger. Fuzzy multi-level security: An experiment on quantified risk-adaptive access control. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 222–230. IEEE, 2007. 4.3

[69] Qun Ni, Elisa Bertino, and Jorge Lobo. Risk-based access control systems built on fuzzy inferences. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 250–260. ACM, 2010. 4.3, 4.7

[70] Chris Burnett, Liang Chen, Peter Edwards, and Timothy J Norman. Traac: Trust and risk aware access control. In *Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on*, pages 371–378. IEEE, 2014. 4.3

[71] Hemanth Khambhammettu, Sofiene Boulares, Kamel Adi, and Luigi Logrippo. A framework for risk assessment in access control systems. *Computers & Security*, 39:86–103, 2013. 4.3

[72] Riaz Ahmed Shaikh, Kamel Adi, and Luigi Logrippo. Dynamic risk-based decision methods for access control systems. *Computers & Security*, 31(4):447–464, 2012. 4.3

[73] J Stuart Hunter. The exponentially weighted moving average. *J. Quality Technol.*, 18(4):203–210, 1986. 4.3

[74] Antonio Celesti, Francesco Tusa, Massimo Villari, and Antonio Puliafito. How to enhance cloud architectures to enable cross-federation. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 337–345. IEEE, 2010. 4.4.3

[75] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Algorithms and architectures for parallel processing*, pages 13–31. Springer, 2010. 4.4.3

[76] Inigo Goiri, Jordi Guitart, and Jordi Torres. Characterizing cloud federation for enhancing providers' profit. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 123–130. IEEE, 2010. 4.4.3

[77] David Recordon and Drummond Reed. Openid 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management*, pages 11–16. ACM, 2006. 4.4.3

[78] Bo Tang, Qi Li, and Ravi Sandhu. A multi-tenant rbac model for collaborative cloud services. In *Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on*, pages 229–238. IEEE, 2013. 4.4.3

[79] Haryadi S Gunawi, Mingzhe Hao, Tanakorn Leesatapornwongsa, Tiratat Patana-anake, Thanh Do, Jeffry Adityatama, Kurnia J Eliazar, Agung Laksono, Jeffrey F Lukman, Vincentius Martin, et al. What bugs live in the cloud?: A study of 3000+ issues in cloud systems. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–14. ACM,

2014. 5.1

[80] Wolter Pieters, Sanne HG van der Ven, and Christian W Probst. A move in the security measurement stalemate: Elo-style ratings to quantify vulnerability. In *Proceedings of the 2012 workshop on New security paradigms*, pages 1–14. ACM, 2012. 5.3

[81] Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011. 5.4