We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

5,000
Open access books available

125,000
International authors and editors

140M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter

Securing the Deployment of Cloud-Hosted Services for Guaranteeing Multitenancy Isolation

Laud Charles Ochei

Abstract

Multitenancy introduces significant error and security challenges in the cloud depending on the location of the functionality to be shared and the required degree of isolation between the tenants. Existing approaches for securing the deployment of cloud-hosted services to serve multiple users have paid little attention to evaluating the effect of the varying degrees of multitenancy isolation on the security and access privilege of tenants (or components). In addition, approaches for securing the isolation of tenants (or components) are usually implemented at lower layers of the cloud stack and often apply to the entire system and not to individual tenants (or components). This study presents CLAMP (Cloud-based architectural approach for securing services through Multitenancy deployment Patterns) to securing the deployment of cloud-hosted services in a way that guarantees the required degree of isolation between the tenants. We evaluated the framework by applying it to a motivating cloud deployment problem. The findings show among other things that the framework can be used to select suitable deployment patterns, evaluate the effect of varying degrees of isolation on the cloud-hosted service, analyse the deployment requirements of cloud-hosted services and optimise the deployment of the cloud-hosted service to guarantee multitenancy isolation.

Keywords: security, cloud-hosted services, deployment, multitenancy, tenant isolation

1. Introduction

Applications on the cloud are accessed over the internet using standard internet protocols. In deciding to store data or host applications in the public cloud, an organisation loses its ability to access the servers that store its information. In this way, potentially sensitive data are at risk from insider attacks.

Therefore, cloud service providers must put in place security measures to physical access to the servers in the data center and frequently monitor data centers for suspicious activity. Security and privacy challenges deriving from the use of the internet are substantial and but no different from the security issues of the

applications not hosted in the cloud. The one significant security element introduced by the cloud is multitenancy [1].

Multitenancy is an essential cloud computing property. Multitenancy is a software architecture where one instance of a cloud offering is used to serve multiple tenants and/or components [2, 3]. Multitenancy means that your application is utilising a virtual machine on a physical computer that is hosting multiple virtual machines. There are many forms of attack utilising multitenancy- inadvertent data sharing, virtual machine escape, side channel attack, and denial of service attack.

Users can require varying or different degrees of isolation between components when implementing multitenancy. To avoid interference, a high degree of insulation between components may be required, but this usually results in high resource consumption and running costs per component. A low degree of isolation promotes sharing of components, resulting in low resource consumption and running costs, but with high performance impact when the workload changes and the application does not scale up/down.

The challenge therefore is how to: (i) ensure that there is isolation between multiple tenants accessing the service or components designed (or integrated) with the service; (ii) resolve the trade-offs between varying degrees of isolation between tenants or components.

Motivated by this problem, this study presents a framework, CLAMP (Cloud-based architectural approach for securing services through Multitenancy deployment Patterns) to securing the deployment of cloud-hosted services in a way that guarantees the isolation between tenants. The framework assumes that the issues of security are tackled from the perspective of the tenant owns software components and is responsible for configuring them to design and deploy its own cloud-hosted application on a shared cloud platform whose provider does not have control over these components.

We evaluated the framework by applying it to a motivating cloud deployment problem that requires securing several components of a cloud-hosted service while guaranteeing the required degree of isolation between tenants. The research question addressed in this study is: “How can we secure the deployment of cloud-hosted services in a way that guarantees isolation between tenants”.

The main contributions of this study are:

1. To develop a framework for securing the deployment of cloud-hosted services in a way that guarantees the isolation of tenants.

2. To evaluate the framework by applying it to a motivating cloud deployment problem.

3. To develop a cloud security checklist for guiding software architects in implementing the framework.

4. Present recommendations and best practice guidelines for securing the deployment of cloud-hosted services based on the framework.

Our findings show among other things that the framework can be used to select suitable deployment patterns, evaluate the effect of varying degrees of isolation on the cloud-hosted service, analyse the deployment requirements of cloud-hosted services and optimise the deployment of the cloud-hosted service to guarantee multitenancy isolation.

The rest of this chapter is organised as follows. Section 2 presents an overview of cloud computing and cloud security. Section 3 presents architectures for cloud-hosted services. Section 4 presents multitenancy in a cloud environment. Section 5 discusses related work on multitenancy and cloud security. Section 6 presents a framework for securing the deployment of cloud-hosted services for guaranteeing multitenant isolation, while Section 7 evaluates the framework by applying it to a motivating cloud deployment problem. Section 8 provides further discussion and recommendations for securing the deployment of cloud-hosted services based on the framework. Section 9 concludes the chapters with future work.

2. Cloud computing security

This section gives an overview of cloud computing and cloud security and multitenancy.

2.1 Cloud computing

According to Armbrust et al. [4], "cloud computing refers to both the applications delivered as a service over the Internet and the hardware and systems software in the data centers that provides those.
services."

The cloud includes hardware for the data centre as well as software. The cloud could either be a *public cloud* (that is, cloud that is provided to the general public in a prepaid manner), *private cloud* (that is, an organisation's internal IT infrastructure which is not available to the public at large), or a *hybrid cloud* (that is, a private cloud's computing capacity that is enhanced by the public cloud).

Although there are so many definitions that have been given for the term cloud computing, there is common agreement on the basic characteristics of a cloud computing environment. These include [3]—pay-per-use, elastic capacity and the illusion of infinite, self-service interface, and resources that are abstracted or virtualized.

There are three basic cloud service models:

i. *Software as a Service (SaaS):* In the SaaS model, cloud providers can install, operate and access their application software using a web browser. An example of a SaaS provider is Salesforce.com, which utilises the SaaS model to provide Customer Relationship Management (CRM) applications located on their server to customers. This eliminates the need for customers to run and install the application on their own computers.

ii. *Platform as a Service (PaaS):* In the PaaS model, cloud providers deliver cloud platforms which represent an environment for application developers to create and deploy their applications. A notable example of PaaS is the Google App Engine, which provides an environment for creating and deploying web-based applications written in specific programming languages.

iii. *Infrastructure as a Service (IaaS):* In the IaaS model, cloud providers offer physical (computers, storage) and virtualized computer resources. Examples of IaaS providers include: Amazon EC2, and Azure Services Platform.

2.2 Cloud security

Cloud security relates to a wide range of policies, techniques, applications, and controls used to safeguard virtualized IP, information, apps, services, and related infrastructure. Cloud security is very essential for companies making the shift to the cloud and also for customers who use the cloud for a range of personal services especially as security threats continue to evolve and become more advanced. Cloud security concerns fall into two wide classifications: (i) security concerns faced by cloud providers (businesses providing software, platform, or infrastructure-as - a-service organisations through the cloud); (ii) security concerns faced by their customers (businesses or organisations that host applications or store data in the cloud). However, the responsibility is shared. There are four (4) main forms of attack that use multitenancy: inadvertent information sharing, virtual machine escape, side-channel attack, denial of service attack. The focus of this study is mostly related to inadvertent information sharing where a tenant has a set of components/resources or services which are mapped to some physical resource on the cloud platform. Under this situation, data residing on the physical resource from one tenant may be leak to another tenant.

Cloud service suppliers often store more than one customer information on the same server in order to conserve resources (e.g., CPU, memory, storage space) reduce cost and maintain service level agreement. To handle such sensitive situations, cloud service providers usually put in place robust secure measures to ensure proper data isolation and logical storage segregation [5].

Cloud security is the protection of data, applications, and infrastructures involved in cloud computing. Cloud security concerns can be grouped in various ways. Gartner listed seven (7) categories of cloud security. In the "data segregation" category, which is the closest to the focus of our study, the cloud is typically in a shared environment alongside data from other customers [6]. The Cloud Security Alliance identified 12 areas of concern [7]. In "Abuse and Nefarious Use of Cloud Services" category, which is the closet to our study, the focus is on the use of poorly secured cloud service deployments, free cloud service trials and fraudulent account sign-ups via payment instrument fraud expose cloud computing models such as IaaS, PaaS, and SaaS to malicious attacks.

3. Architectures for cloud-hosted services

The architectures or cloud patterns used to deploy cloud-hosted services to the cloud are of great importance to software architects because they determine whether or not the system's essential quality attributes (e.g., performance) will be exhibited [1, 8, 9].

3.1 Architectural patterns

Architectural and design patterns have long been used to provide known solutions to many common problems a distributed system face [1, 10]. A system/application architecture decides whether or not it will show its necessary quality attributes (e.g., performance, availability, and security) [1, 8].

Definition 2.3: **Architectural Pattern.** Architectural patterns are compositions of architectural elements that provide bundled solutions to solve recurring problems a system faces [1].

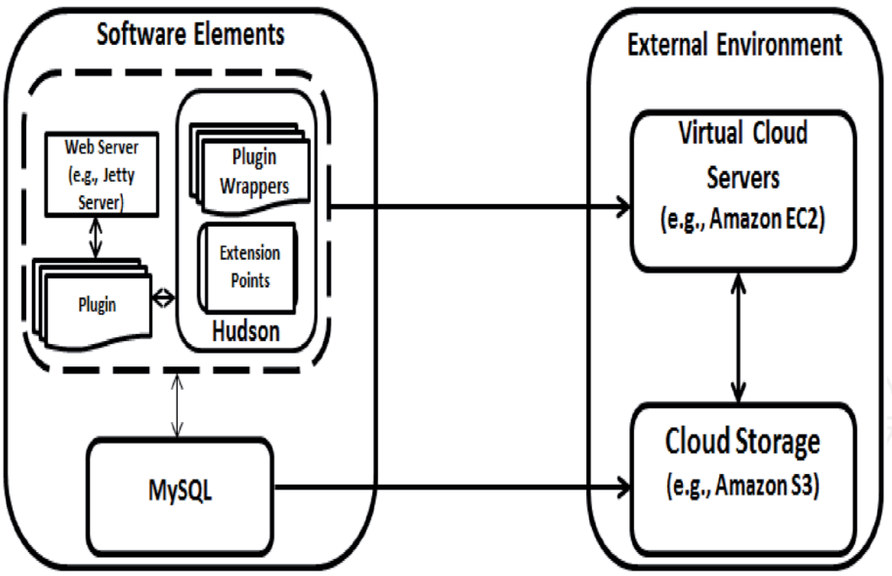

Figure 1.
Mapping elements of a cloud-hosted service to the external environment.

A cloud pattern in the cloud computing environment represents a well-defined format for explaining an appropriate solution to a cloud-related problem [11]. There are several cloud problems, such as: (i) selecting an appropriate cloud type for hosting applications; (ii) selecting a cloud service delivery approach; (iii) deploying a multi-tenant service in a way that ensures tenant isolation.

Cloud deployment architects are using cloud patterns as a reference guide to document best practice on how to plan, develop and deploy cloud-based applications.

Definition 2.4: **Cloud Deployment Pattern.** A "Cloud deployment pattern" is defined as a type of architectural pattern, which embodies decisions as to how elements of the cloud application will be assigned to the cloud environment where the application is executed.

Our definition of cloud deployment pattern is similar to the concept of design patterns [10], (architectural) deployment patterns [1], collaboration architectures [8], cloud computing patterns [11], cloud architecture patterns [12], and cloud design patterns [13].

One of a cloud deployment architect's main duty is to assign cloud application elements to the hardware elements (e.g. processor, filesystems) and communication elements (e.g. protocols, message queues) in the cloud environment so that the necessary quality attributes can be achieved.

Figure 1 demonstrates how elements of Hudson (a typical of Global Software Development tool) are mapped to the elements of the cloud environment. Hudson operates on an Amazon EC2 instance while periodically extracts and stores the data it produces on separate cloud storage (e.g., Amazon S3).

4. Multitenancy in a cloud environment

Multitenancy is an essential cloud computing property where a single instance of a cloud offering is used to serve multiple tenants and/or components [14, 15]. One of the challenges of implementing multitenancy on the cloud is how to enable the required degree of isolation between multiple components of a cloud-hosted

application (or tenants accessing a cloud-hosted application). We refer to this as *multitenancy isolation*.

Definition 1: **Multitenancy isolation**. The term "Multitenancy Isolation" refers to an approach to ensuring that one tenant's performance, stored data volume, and access rights do not impact other tenants accessing the shared application component or its functionality. Multitenancy isolation can be represented in three main cloud multitenancy patterns [11]:

1. Shared component: Tenants use the same instance of a resource and may not be aware that other tenants are using it.

2. Tenant-isolated component: Tenants share the same resource instance but are assured of their isolation. This pattern allows for the tenant-specific configuration of the functionality or resource offered.

3. Dedicated component: Tenants do not share resource instance. That is, each tenant is associated with one instance (or a certain number of instances) of the resource.

4.1 Degrees of multitenancy isolation

The degree of isolation between tenants accessing a shared component of an application can be expressed in the three multitenancy patterns (i.e., shared component, tenant-isolated component and dedicated component). The shared component reflects the lowest degree of isolation between tenants whilst the highest is the dedicated component.

The three key areas where tenant isolation can be addressed in a system are: performance, stored data volume and access privileges. For example, in performance isolation, other tenants should not be affected by the workload created by other tenants. For example, other tenants should not be impacted by the workload generated by other tenants when considering performance isolation.

Guo et al. [16] evaluated different isolation capabilities related to authentication, information protection, faults, administration etc.

Different isolation capabilities related to faults, information protection, authentication, administration, etc., have been evaluated by Guo et al. [16]. Bauer and Adams [17] have studied how to virtualization can be used to ensure that the failure of one tenant instance does not spread into other tenant instances.

A high degree of isolation can be achieved by deploying an application component exclusively for one tenant. This would ensure that there is little or no performance interference between the components when workload changes. The deployment of an application component specifically for one tenant can achieve a high degree of insulation. This ensures that when workload changes, there is little or no performance impact between the components.

Nevertheless, since components are not shared (e.g. in a situation where some strict laws and regulations prohibit them from being shared), this means duplicating the components for each tenant, resulting in high resource consumption and running costs. In general, this would restrict the number of requests to access the components.

It may also be that a component requires a low degree of isolation, for example, to facilitate sharing of the functionality, data, and resources of the component. This would minimise resource consumption and running costs, but other

component's performance might be affected if one of the components experiences a change in workload.

The challenge for a cloud deployment architect would therefore be how to overcome the trade-offs between the required performance, system resources and access privileges at different levels of an application when selecting one (or combinations) of multitenancy patterns to deploy software tools in the cloud. Resolving the trade-off involving access privileges of users at different levels of an application depending on the type of multitenancy deployment pattern that is being used is one of the strategies for providing security for cloud-hosted services deployed based on multitenancy architecture.

4.2 Implementation of multitenancy isolation

Multitenancy isolation can be implemented both at the process levels (i.e., based on the processes that interacts with the system) and data levels (i.e., based data that is being generated or manipulated by the system) of a cloud-hosted service. **Figure 2** shows an architecture that can be used to implement multitenancy isolation at the data level. This implementation represents an application that logs each operation into a database by relying on an automated build verification and testing in response to a specific event such as detecting changes in a file.

A specific example of an implementation shown in **Figure 2** is to use Hudson's Files Found-Trigger plugin to poll one or more directories and start a build if there are certain files in those directories [18]. Hudson is an open source tool and so can be easily modified by adding a Java class that accepts a filename as argument into the plugin. The plugin is loaded into a separate class loader during execution, to avoid interfering with the core functionality of Hudson.

Definition 2: **Application Component.** This refers to an encapsulation of a functionality or resource that is shared between multiple tenants. A component of an application could be a data handling component (e.g. database), communication

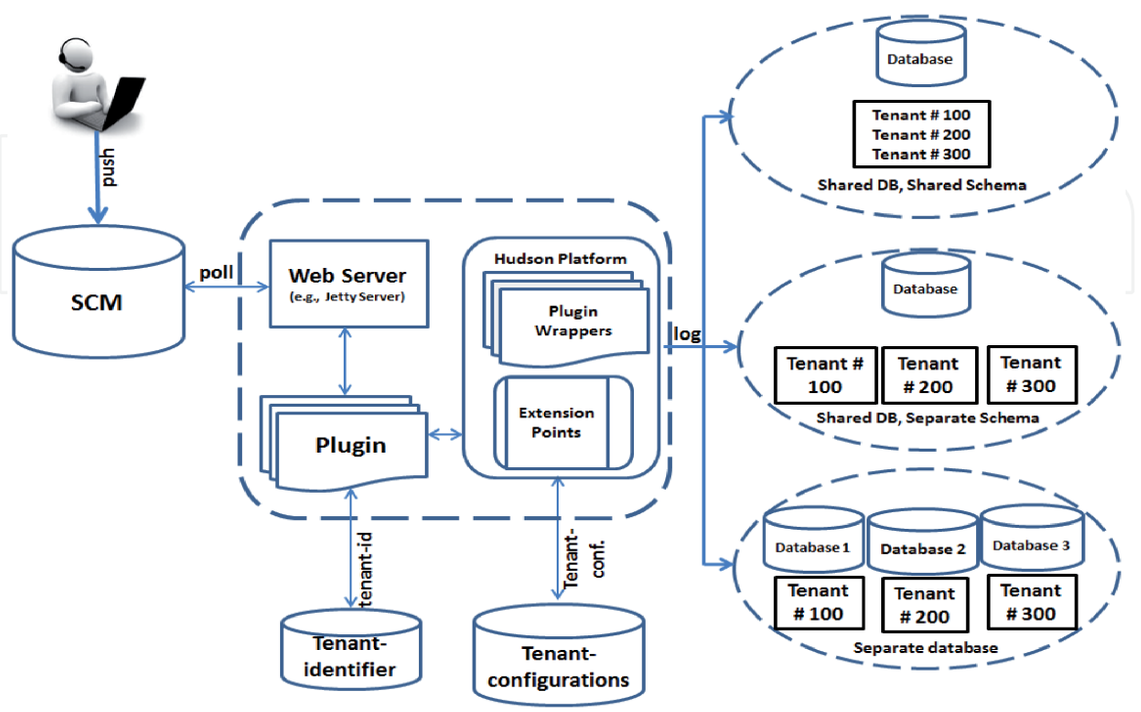

Figure 2.
Multitenancy isolation architecture for cloud-hosted applications.

component (e.g. message queue), user interface component (e.g. AJAX) or processing component (e.g. load balancer).

There are several solutions to multitenancy implementation which have been widely discussed in the literature. Multitenancy can be introduced at different cloud stack layers: application layer [16], middleware layer [19], and data layer [20, 21].

It has been suggested that customization is the solution to addressing the hidden constraints on multitenancy such as complexities, security, scalability and flexibility [22]. Furthermore, integrating a plugin into a cloud-based service can provide a workaround for true multitenancy. Again, most of the solutions available to incorporate multitenancy require a re-engineering of the cloud service to some degree [17, 23].

Other research work on multitenancy isolation include: [24–30].

5. Related work on cloud security

Apart from the general research on best practices in securing the cloud against various forms of attacks, there is little research on approaches to secure cloud services against attacks arising from implementing multitenancy architectures. There is also little research on approaches for securing the deployment of cloud-hosted services in a way that guarantees varying degrees of isolation between tenants.

According to Bass et al., one of the significant security challenges introduced in the cloud is multitenancy [1]. Implementing multitenancy means that your cloud-hosted services are utilising the virtual machine on a physical machine that host multiple virtual machines. Much of literature on multitenancy and cloud security has established that the obvious approach to addressing the problem is for cloud providers to allow users to reserve entire virtual machines for their use. Although this defeats some of the economic benefits of using the cloud, it is nevertheless a mechanism to prevent multitenancy attacks [1–3].

Previous research has looked at this problem from the perspective of the cloud providers, for instance, autoscaling algorithms and supporting security-based strategies provided by IaaS providers such as Amazon and optimization frameworks suggested for use by SaaS providers such as Salesforce.com.

This study, however, looks at the issue from the tenant's viewpoint, who owns software components and is responsible for configuring them to build and deploy their own cloud-hosted application on a shared cloud platform where the cloud provider has no control over the software components. The focus of this chapter is to provide a framework for securing the deployment of cloud-hosted services in a way that guarantees multitenancy isolation.

The work by [31] is one of the most detailed studies on cloud security. The author explores different aspects of security and the possible solutions that have been considered by different authors. The author did not consider approaches for securing the deployment of cloud-hosted services in a way that guarantees varying degrees of isolation between tenants.

6. Framework for securing the deployment of cloud-hosted services for guaranteeing multitenant isolation

The section discusses the framework for securing the deployment of cloud-hosted services for guaranteeing multitenant isolation.

6.1 Developing the CLAMP framework

The study presents a robust framework, CLAMP, for securing the deployment of cloud-hosted services for guaranteeing multitenancy isolation. The framework, CLAMP (Cloud-based architectural approach for securing services through Multitenancy deployment Patterns), is basically a framework for guiding software architects in securing the deployment of cloud-hosted services in a way that guarantees the required degree of isolation between other tenants when one of the tenants (or components) experiences a high workload or security breach.

The CLAMP framework is illustrated as a layered architecture in **Figure 3**. It shows how the components of the framework work together to support the task of securing the deployment of components of a cloud-hosted service for guaranteeing multitenancy isolation. The development of CLAMP was inspired by the well understood architectural structure/pattern called layered pattern [1]. A layer is an abstract "virtual machine" that provides a cohesive set services through a managed interface. In a strictly layered system, a layer can only use the services of the layer immediately below it. This structure is used to imbue a system with portability, the ability to change the underlying computing platform.

The different components of the CLAMP framework are described as follows.

6.1.1 Layer one: selection of a suitable architectural pattern

This layer addresses the selection of a suitable architectural pattern. In order to secure the deployment of cloud-hosted services for guaranteeing multitenancy isolation, it may be very difficult if not impossible to use one cloud pattern to deploy the service to the cloud due to the different requirements of the service including accessibility of the service to a wider audience and a combined assurance for security and privacy. For instance, the architect would require a combination of

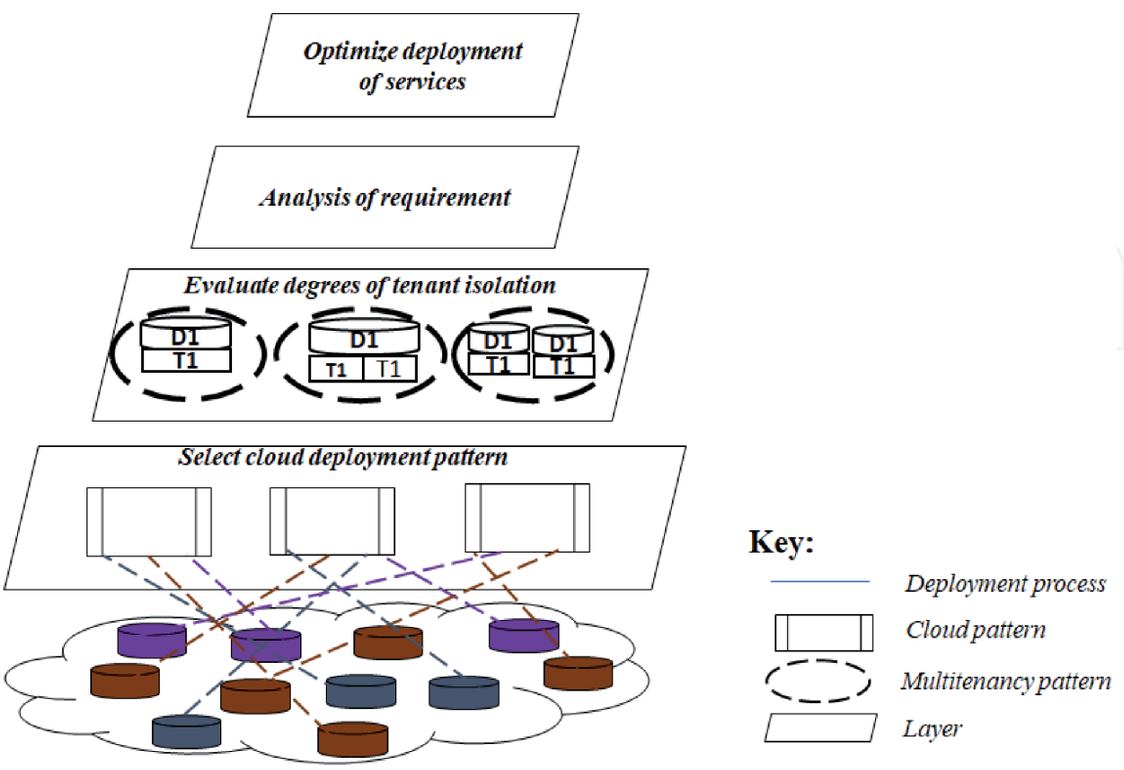

Figure 3.
A layered architecture for securing the deployment of cloud-services for guaranteeing multitenancy isolation.

several deployment patterns together with supporting technologies for archiving components of the cloud-hosted service (i.e., in a hybrid fashion) to integrate components located in a different cloud environment to form one cloud solution. Again, if communication is required internally to exchange messages between application components, then a message-oriented middleware technology would also be needed. Therefore, the challenge is that of selecting a suitable pattern (together with the supporting technologies) or a combination of patterns in order to secure the deployment of cloud-hosted services for guaranteeing multitenancy isolation. It is assumed that there is a repository of cloud deployment patterns from where a software architect can select a suitable pattern (s) to address the business requirements of the company/user.

6.1.2 Layer two: evaluation of the required degree of isolation between tenants

The layer addresses the evaluation of the required degree of isolation between tenants. There are varying degrees of isolation between tenants that are accessing a cloud-hosted service. Some of the tenants would require a higher or different degree of isolation than others. Tenants would be able to share application components as much as possible at the very basic degree of multitenancy, which translates into increases use of underlying resources.

At the very basic degree of multitenancy, tenants would be able to share application components as much as possible which translates to increased utilisation of underlying resources. While some components of the application may benefit from a low degree of isolation between tenants, other components may require a higher degree of isolation because the component may be either too sensitive or cannot be shared as a result of certain corporate legislation and regulation.

There is increasing evidence, for example, that many cloud providers are reluctant to set up data centres in mainland Europe due to stricter legal requirements that prohibit data processing outside Europe [32, 33]. This requirement will traverse down to the IaaS level, and customers must take this into consideration if intending to host applications outsourced to such cloud providers [11] that host customers data outside Europe. Therefore, evaluating the required degree of isolation between the tenants will allow for the appropriate mapping of security requirements during the deployment of cloud-hosted services onto cloud provider's infrastructure.

6.1.3 Layer three: analysis of the deployment requirements of the cloud-hosted service

Layer three addresses the analysis of the deployment requirements of the cloud-hosted service. This involves two main activities: (i) mapping tenant isolation to key process of the cloud-hosted services, cloud resources required to support the service and layers of the cloud stack on the which the service will be executed; (ii) analysing the trade-offs that should be considered when implementing the required degree of tenant isolation.

The mapping is rooted in the framework of a typical architectural deployment system that has two main components: the cloud application (that is, the component or service to be deployed) and the cloud environment (that is, the environment in which the process/service is performed) [1]. This mapping also captures the link between a process associated with a cloud-hosted service (e.g., continuous integration process), being used in a hybrid deployment scenario by utilising a cloud-hosted environment (e.g., SaaS and PaaS deployment environment).

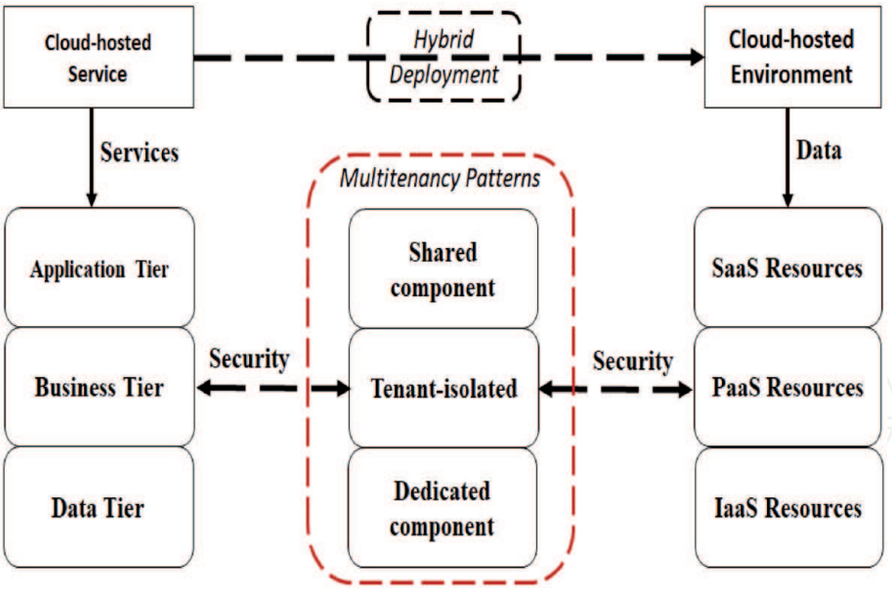

Figure 4.
Mapping of degrees of tenant isolation to cloud-hosted services and resources.

In our previous research, we provided an explanatory framework for (i) mapping tenant isolation to different software processes, cloud resources and application stack layers (ii) illustrating the different trade-offs for consideration in order to achieve optimal deployment of components in a way that guarantees the required degree of tenant isolation [34] (see **Figure 4**).

Issues relating to security, privacy, trust and regulatory compliance can mostly be tackled in a hybrid fashion. For example, data / bugs created from a bug tracking system could be stored at some location to comply with privacy and legal regulations, while the architecture of the bug tracking system could be changed to limit the access of certain data to users residing in regions not deemed to be of interest to those who own the hosted data. Securing cloud-hosted services deployed with the goal of guaranteeing varying degrees of multitenancy isolation can best be tackled using a hybrid approach.

The second aspect of the analysis involves analysing the key trade-offs for consideration when implementing the required degree of tenant isolation for cloud-hosted software processes. There are six key aspects of the trade-offs that have to be considered when implementing security for multi-tenant cloud-hosted software services. These trade-off include tenant isolation versus (resource sharing, the number of users/requests, customizability, the size of generated data, the scope of control of the cloud application stack and business constraints). **Table 1** shows the trade-offs and the key decision that have to be main when considering the trade-offs.

6.1.4 Layer four: optimisation of the deployment of the cloud-hosted services

This layer deals with the optimization of the components of a cloud-hosted service. In a cloud environment, varying degrees of tenant isolation are possible, depending on the type of component being shared, the process supported by the component and the location of the component on the cloud application stack (i.e., application level, platform level, or infrastructure level).

In a cloud environment, depending on the type of component being shared, the processes enabled by the component, and the location of the component on the cloud application stack (i.e. application level, platform level, or network level),

Category	Checklist
Selection of a suitable architectural pattern	What are classes of cloud patterns available, what are the tools and processes to support the selection of suitable cloud patterns.
Evaluation of the required degree of isolation between tenants	What are the data and processes of the cloud-hosted service that require security? What is the required degree of isolation between tenants accessing the components of the cloud-hosted services?
Analysis of the deployment requirements of the cloud-hosted	How can you map the key resources of the cloud-service (e.g., store for the archive data) to the cloud provider's platform? What are the trade-offs to consider when securing the deployment of cloud-hosted services? (e.g., customizability, scope of control, business requirements)
Optimisation of the deployment of the cloud-hosted services	What are the components (or tenants) that are required to design (or integrate) with the cloud-hosted services? How feasible is it to tag components or whole system?

Table 1.
Security checklist for evaluating the framework.

varying degrees of tenant isolation are possible. Therefore, it is important for software architects to be able to able to control the required degree of isolation between tenants sharing components of a cloud-hosted application.

For instance, the deployment of an application component specifically for one tenant will achieve a high degree of isolation. This would make sure that when workload changes, there is little or no performance impact between the components.

However, because components are not shared it implies duplicating the components for each tenant, which leads to high resource consumption and running cost. Overall, this will limit the number of requests allowed to access the components. A low degree of isolation would allow sharing of the component's functionality, data and resources. This would reduce resource consumption and running cost, but the performance of other components may be affected when one of experiences a change in workload.

This is a decision-making challenge that requires an appropriate decision to be made to address the trade-off between a lower degree of isolation versus the possible influence that can occur between components or a high degree of isolation versus the difficulty of high resource usage and component running costs.

In a nutshell, the procedure for implementing the framework can be summaries with following four steps: (i) Select suitable deployment patterns (one or combination of several patterns), (ii) Evaluate the effect of varying degrees of isolation on the cloud-hosted service, (iii) Analyse the deployment requirements of cloud-hosted services and (iv) optimise the deployment of the cloud-hosted service to guarantee multitenancy isolation.

6.2 Developing a security checklist for deployment of cloud-hosted services

In addition to the framework, CLAMP, we develop a security checklist to guide software architects in securing the deployment of cloud hosted services. The layers of the frameworks are used to develop the categories of the checklist. Many of the items in the checklist may seem obvious but the purpose of a checklist is help ensure the completeness of the security design while implementing the CLAMP framework.

In using the security checklist, the software architect should think about how to review the security of the cloud-hosted services and figure out how well it satisfies security in each of the categories of the framework. In other words, what questions

would you ask a software architect to evaluate how the framework satisfies the requirements for securing the deployment of cloud-hosted services for guaranteeing multitenancy isolation. This is the basis for the security checklist.

7. Evaluation of framework for securing the deployment of cloud-hosted services

This section presents a simple case study of a cloud deployment problem to illustrate how to use the proposed framework to secure the deployment of a cloud-hosted services in a way that guarantees multitenancy isolation. The following scenario explains our motivation.

7.1 Motivating scenario

Let us assume that there are multiple components of a cloud service (e.g., data-handling component) hosted on the same or different cloud infrastructure. These components which are of various types and sizes are required to design (or integrate with) a cloud-hosted service (e.g., continuous integration system such as Hudson or Jenkins) and their supporting processes for deployment to multiple tenants. Tenants, in this case, may be multiple users, departments of a company or different companies. The laws and regulations of the company make it liable to share and archive data generated from the component (e.g., builds of source code) and keep it accessible for auditing purposes. However, access to some components or some aspects of the archived data will be provided solely to particular groups of tenants for security reasons. The question is: in a resource-constrained environment, how can we secure the deployment of components of this cloud-hosted service in a way that guarantees the required degree of isolation between other tenants when one of the tenants (or components) experiences a high workload or security breach (**Table 2**).

7.2 Applying the CLAMP framework

This section explains how to apply the proposed framework, CLAMP, to secure the deployment of this cloud-hosted service in a way that guarantees the required degree of isolation between other tenants. Each component of the framework has

Category	Analysis
Selection of a suitable architectural pattern	The problem requires a hybrid-related deployment pattern, namely, integrating data stored in multiple clouds
Evaluation of the required degree of Isolation between tenants	The requirement to allow a particular group of users to access some components for security reasons means that the company requires the highest degree of isolation between tenants
Analysis of the deployment requirements of the cloud-hosted	Map the tenant isolation to key processes associated with the cloud-hosted service, cloud resources and layers of the cloud stack. Analyse the trade-offs required for optimal deployment
Optimisation of the deployment of the cloud-hosted services	Tag each component. Analyse the trade-off involved, namely, achieving a high degree of isolation versus resource sharing. To address this trade-off, an optimization model is recommended to be used to select optimal components for deployment to the cloud

Table 2.
Summary of how problem was analysed per layer of the framework.

a part to play in securing the deployment of components of a cloud-hosted service. The structure of evaluating the framework, CLAMP, in its textual form, is specified as a string consisting of three sections-(i) Context; (ii) Problem; and (iii) Solution. In a more general sense, the string can be represented as: [CONTEXT; PROBLEM; SOLUTION]. Each layer of the framework maps to the step required to provide a solution to the cloud deployment problem. **Table 2** summaries how the problem was evaluated based each layer of the framework.

7.2.1 Step one: selecting a suitable cloud deployment pattern

In order to address this challenge, this framework would recommend that the architect should reference some sort of a classification or taxonomy to guide in the selection of a suitable pattern together with the supporting technologies. In our previous work, we have developed a taxonomy and a process for guiding architect in selecting a suitable framework for cloud deployment [35]. In addition, a general process, CLIP (CLoud-based Identification process for deployment Patterns) has been developed for guiding architects in selecting applicable cloud deployment patterns (together with the supporting technologies) using the taxonomy for deploying services/application to the cloud we also discussed.

It is important to note that the company does not have direct access to the cloud IaaS. Therefore, the architect must select a deployment pattern that can be implemented at the application level to secure the deployment of the cloud-hosted services for guaranteeing multitenancy isolation. By making reference to the taxonomy of cloud-deployment patterns and the general process for selecting applicable deployment patterns based on the taxonomy, we would recommend that the architect should select a hybrid-related deployment pattern for addressing the requirements of the customer. It is assumed that the data archived by Hudson contains the source code and (possibly configuration files) that drives a critical function of an application used by the company.

The data stored by Hudson is presumed to contain the source code and (possibly configuration files) which drives a critical function of an application used by the company. Any unauthorised access to it may be devastating for the company. In this circumstance, the most appropriate multitenancy pattern to use is the hybrid backup deployment pattern. This pattern can be used to extract data to the cloud environment and archive it different cloud environments [11].

7.2.2 Step two: evaluating the varying degrees of isolation

This step involves evaluating the required degree of isolation between tenants and then select an appropriate multitenancy pattern or combination of patterns to support such a required degree of isolation. There are varying degrees of isolation between tenants that are accessing the cloud-hosted service and so some of the tenants would require a higher or different degree of isolation than others.

One of the key requirements of the company to provide access to some components or some aspects of the archived data solely to particular groups of tenants for security reasons. Based on this key requirement, we conclude that the company requires the highest degree of isolation between tenants.

The varying degrees of multitenancy isolation can be captured in three main cloud deployment patterns: shared component, tenant-isolated component and dedicated component. The shared component represents the lowest degree of isolation between tenants while the dedicated component represents the highest. In a dedicated component pattern, tenants do not share resources, though each tenant is associated with one instance or a certain number of instances of the resource.

7.2.3 Step three: analysis of the deployment requirements of the cloud-hosted service

The step involves analysing the deploying requirements of the cloud-hosted services. This analysis entails mapping tenant isolation to key processes associated with the cloud-hosted service, cloud resources required to support the service and layers of the cloud stack on the which the service will be executed. This analysis translates to using a hybrid approach to map the SaaS and PaaS level of the cloud provider to the cloud-hosted service which has a backup cloud storage. This type of cloud pattern is referred to as a hybrid backup pattern [3]. The archive data in a problem scenario can be stored in any location to comply with privacy and legal regulations of the company while the architecture of the cloud-hosted service could be modified to restrict exposure of certain data to users located in regions not considered to be of interest to the owners of the hosted data.

The second aspect of the analysis involves analysing the different trade-offs to be considered for optimal deployment of components with a guarantee of the required degree of tenant isolation. There are three main trade-offs that the company has to consider. The first trade-off relates to tenant isolation versus customizability. The higher the degree of isolation that is required, the easier it is to customise a cloud-hosted service to implement tenant isolation. However, because we assumed that the user has access to the application layer of the cloud stack, it would be more difficult to implement a higher degree of isolation at the application level in terms effort, time and skills set required to modify the source code. This raises issues of compatibility and interdependencies between the cloud-hosted services and required plugins and libraries. Each time a multitenant application or its deployment environment changes, then a tedious, complex and security maintenance process is also required.

The second trade-off relates to the "scope of control" of the cloud application stack. The architect has more flexibility to implement or support the implementation of the required degree of tenant isolation when there is greater "scope of control" of the cloud stack application. As the company requires a higher degree of isolation (e.g., based on the dedicated component), then the scope of control should extend beyond the higher level to the lower levels of the cloud stack (i.e., PaaS and IaaS) even as the cost of implementation of such a cloud security architecture will certainly increase.

The third trade-off relates to the trade-off between tenant isolation and business (or legal) requirements of the company. A key legal requirement of the company is that access to some components or some aspects of the archived data will be provided solely to particular groups of tenants for security reasons. The dedicated component which offers a high degree of isolation can be used to handle the legal requirements Such legal restriction, for example, legal restrictions and the location and configuration of the cloud infrastructure are usually difficult to compensate for at the application level. For example, a legal requirement can state that data that a specific cloud provider has hosted in Europe cannot be stored elsewhere (e.g., in the USA). Therefore an architect would have to map this form of requirement to a cloud infrastructure that specifically meets this requirement.

7.2.4 Step four: optimisation of the deployment of the cloud-hosted services

The key task in step four is to optimise the deployment of components of the cloud-hosted service. Some requirements cannot be fully satisfied and so there has to be some optimisation to ensure that the cloud deployment is carried out in way that does not compromise the security of the components of the cloud-hosted service. This entails tagging the components (or tenants) associated with the cloud-hosted service so that the software architects can be have more leverage to

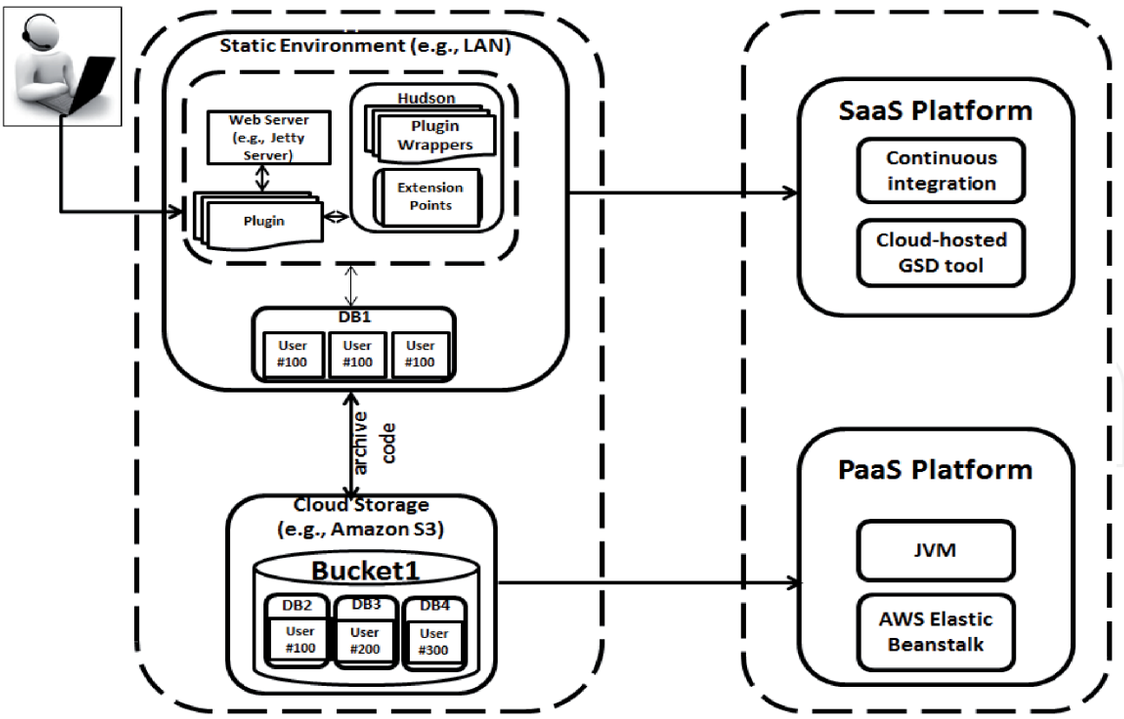

Figure 5.
Mapping a continuous integration system to cloud stack based on hybrid backup pattern.

implement the required degree of isolation between tenants. In [36] an implementation of the model-based algorithm was presented for providing optimal solutions for deploying components designed to use (or be integrated with) a cloud-hosted application in a way that guarantees multitenancy isolation (**Figure 5**).

7.3 Applying the security checklist

In addition to applying the framework on the motivating problem, we also apply the security checklist to support design and analysis of process for securing the deployment of cloud-hosted services for guaranteeing multitenancy isolation. **Table 3** shows the result of the security checklist.

Category	Checklist
Selection of a suitable architectural pattern	The hybrid patterns are a class of cloud pattern that can be explored. The hybrid backup pattern is suitable for the problem. Tools and technologies such as cloud storage, and REST, and message exchange technologies can be implemented
Evaluation of the required degree of isolation between tenants	The highest degree of isolation would be required for isolate tenants. The data to secure include archive data- source code, configuration files. The key software process in this problem is the continuous integration process
Analysis of the Deployment requirements of the Cloud-hosted	The process supporting the cloud-hosted service (i.e., continuous integration) should be mapped to a cloud platform that allows data to be stored in multiple location without much restrictions. The key trade-offs in this problem are tenant isolation versus (customizability, scope of control, business requirements)
Optimisation of the deployment of the cloud-hosted services	The main components to optimise are—authorization/authentication data or database components, queue messages. The approach of tagging components can be done either manually or dynamically using a model/algorithm depending on the number of components and complexity of the processes involved

Table 3.
Applying the security checklist.

8. Discussions and recommendations

This section presents a general discussion of the key security issues that should be considered together with some recommendations that can be followed in order to secure the deployment of cloud-hosted services in a way that guarantees multitenancy isolation.

8.1 Assurance for compliance with legislation and regulatory requirements

One of the challenges of implementing cloud security is to provide assurance to cloud users who need to demonstrate compliance with various legislation and regulatory requirements. Our proposed framework addresses this challenge by providing guidance to the software architecture based on the a taxonomy of cloud deployment patterns to not only to select a suitable cloud deployment pattern but to also evaluate the requirements of the customer to select a cloud multitenancy pattern that guarantees the required degree of isolation between tenants.

For example, there is growing evidence that many cloud providers are unwilling to set data centres in mainland Europe because of tighter legal requirements that disallow the processing of data outside Europe (Hon & Millard 2017, Google 2017). This requirement will traverse down to the IaaS level, and customers must take this into consideration if intending to host applications outsourced to such cloud providers [11]. The challenge, therefore, for a cloud deployment architect is that there are no case studies to understand and evaluate the effect of the required degree of isolation on the performance, systems resources and access privileges at different levels of a cloud-hosted service when opting for one (or combinations) of a particular degree of isolation between tenants.

8.2 Customizability of the cloud-hosted services and supporting process

Customising a cloud-hosted GSD tool (or any cloud-hosted service) can be very challenging if the service has several components that are being shared. A service deployed on the cloud can have many inter-dependencies on different levels of the application itself and with other applications, plugins, libraries, etc., deployed with other cloud providers. This could impact the security of the cloud-hosted system in a way that we did not anticipate and thus the degree of tenant isolation that was needed. There is also a serious risk that incompatible plugins and libraries will be used to alter, configure and run these GSD tools. This could corrupt the GSD tool and stop other supporting programs/processes from running. A simple way to tackle this infrastructure problem is to move tenant isolation deployment down the lower levels of the cloud stack, where the architect can deploy the GSD framework on a PaaS platform, for example. Middleware issues and methods for SaaS device customizability were discussed in [37, 38].

8.3 Errors and sensitivity to workload interference

Multitenancy may pose significant error and security challenges in the cloud, particularly when different degrees of isolation are introduced between multiple tenants who share resources. When resources are shared between multiple tenants in a multitenant cloud-service, it is very possible to affect the performance and resource usage of other tenants due to errors associated with one tenant (e.g. due to overload of the tenant or inadequate resource allocated to the tenant).

The type of error associated with a cloud-hosted service is a pointer to the key resources to consider in achieving the required degree of tenant isolation.

For example, moving the VM image instance associated with a cloud hosted service whose file permission had been set on a local machine to the cloud infrastructure could cause affect the requires degree of tenant isolation and hence the security of other tenants during cloud deployment. Therefore, it is necessary to get repository ownership and permission right before deploying such a cloud-hosted service.

8.4 Tagging components with the required degree of isolation

One of the challenges of securing the deployment of a cloud-hosted service is how to handle such cloud-hosted services that several interdependencies with other services elements to which it interacts. Therefore, it is important that components designed to be used or incorporated with a cloud-hosted service should be tagged as much as possible when the necessary degree of tenant isolation is needed.

Tagging can be a complex and complicated process and may not even be feasible under certain circumstances (e.g. where the component is incorporated into other systems and is not under customer control). Therefore, this can also be predicted in a dynamic way instead of labelling each part with an insulation value as necessary.

In our previous work [39], we built an algorithm that dynamically learns the features of existing components in a repository and then uses this knowledge to associate each component with the appropriate degree of isolation. This information is critical to making key security decisions and optimising the resources consumed by the components, particularly in a dynamic or real-time environment.

9. Concluding remarks

The chapter presented CLAMP, a framework for securing the deployment of cloud-hosted services in a way that guarantees the isolation between tenants to contribute to the literature on multitenancy and cloud security. The framework is based on a layered architectural structure where the layers are allowed to use other layers in a strictly managed fashion; a layer is only allowed to use the layer immediately below.

The framework was evaluated by applying it to a motivating cloud deployment problem that requires securing several components of a cloud-hosted service while guaranteeing the required degree of isolation between tenants. The findings show among other things that the framework can be used to select suitable deployment patterns, evaluate the effect of varying degrees of isolation on the cloud-hosted service based on the requirements of the business, analyse the deployment requirements of cloud-hosted services and optimise the deployment of the cloud-hosted service to guarantee multitenancy isolation.

Future work would entail design an experimental procedure for automatically evaluating the framework (i.e., the layered-architectural structure) for securing the deployment of a real-life cloud-hosted service for guaranteeing isolation between tenants. Thereafter, this experimental design will incorporate into a simulator and testing tool for evaluating the layered-architecture for securing the cloud-hosted service for guaranteeing isolation between tenants. This approach has been discussed in [1] as a way to turn architectural parameters into constants, ranges and other that can be easily measured. This will allow software architects to determine the effect of each form of improvement or business requirements of the component or cloud-hosted service before deciding whether the service is secured enough to be deployed without compromising the required degree of isolation between tenants.

DOI: http://dx.doi.org/10.5772/intechopen.92142

IntechOpen

Author details

Laud Charles Ochei
Robert Gordon University, Aberdeen, United Kingdom

*Address all correspondence to: l.c.ochei@rgu.ac.uk

References

[1] Bass L, Clements P, Kazman R. Software Architecture in Practice, 3/E. United States: Pearson Education; 2013

[2] Bauer E, Adams R. Reliability and Availability of Cloud Computing. New Jersey: John Wiley & Sons; 2012

[3] Buyya R, Broberg J, Goscinski A. Cloud Computing: Principles and Paradigms. New Jersey, United States: John Wiley & Sons, Inc.; 2011. DOI: 10.1002/9780470940105

[4] Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, et al. A view of cloud computing. Communications of the ACM. 2010;**53**(4):50-58

[5] Srinivasan MK, Sarukesi K, Rodrigues P, Manoj MS, Revathy P. State-of-the-art cloud computing security taxonomies—A classification of security challenges in the present cloud computing environment. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics; ACM. 2012. pp. 470-476

[6] Brodkin J. Gartner—Seven Cloud-Computing Security Risks. 2019. Available from: https://www.infoworld.com/article/2652198/gartner–seven-cloudcomputing-security-risks.html [Accessed: 14 August 2019]

[7] Brook J-M, Field S, Shackleford D. Top threats to cloud computing plus: industry insights. 2019. Available from: https://cloudsecurityalliance.org/artifacts/top-threats-cloud-computing-plusindustry-insights/ [Accessed: 14 August 2019]

[8] Junuzovic S, Dewan P. Response times in n-user replicated, centralized, and proximity-based hybrid collaboration architectures. In: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work; ACM. 2006. pp. 129-138

[9] Stol K-J, Avgeriou P, Babar MA. Design and evaluation of a process for identifying architecture patterns in open source software. In: Ivica C, Volker G, Matthias B, editor. Software Architecture: 5th European Conference, ECSA 2011, Essen, Germany, September 13-16, 2011. Proceedings. Vol. 6903. London: Springer; 2011. pp. 147-163

[10] Vlissides J, Helm R, Johnson R, Gamma E. Design Patterns: Elements of Reusable Object-Oriented Software. Vol. 49. Boston, United States: Addison-Wesley; 1995. p. 120

[11] Fehling C, Leymann F, Retter R, Schupeck W, Arbitter P. Cloud Computing Patterns. London, England: Springer; 2014

[12] Wilder B. Cloud Architecture Patterns. 1st ed. Sebastopol, CA, United States: O'Reilly Media, Inc.; 2012

[13] Homer A, Sharp J, Brader L, Narumoto M, Swanson T. Cloud Design Patterns. Redmon, Washington, United States: Microsoft; 2014

[14] Krebs R, Momm C, Kounev S. Metrics and techniques for quantifying performance isolation in cloud environments. Science of Computer Programming. 2014;**90**:116-134

[15] Pearson S. Privacy, security and trust in cloud computing. In: Privacy and Security for Cloud Computing. London: Springer-Verlag; 2013. pp. 3-42

[16] Mehta A. Multi-tenancy for cloud architectures: Benefits and challenges. 2017. Available from: http://www.devx.

com/architect/Article/47798/ [Accessed: May 2020]

[17] Aiken L. Why multi-tenancy is key to successful and sustainable software-as-a-service (SaaS). 2017. Available from: http://www.cloudbook.net/resources/stories/. [Accessed: May 2020]

[18] Hudson. Apache Software Foundation. 2016. Available from: http://wiki.hudson-ci.org//display/HUDSON/Files+Found+Trigger [Accessed: May 2020]

[19] Strauch S, Andrikopoulos V, Leymann F, Muhler D. Esb mt: Enabling multi-tenancy in enterprise service buses. In: In 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom); IEEE. 2012. pp. 456-463

[20] Vengurlekar N. Isolation in private database clouds. 2012. Available from: http://www.oracle.com/technetwork/database/database-cloud/ [Accessed: May 2020]

[21] Wang ZH, Guo CJ, Gao B, Sun W, Zhang Z, An WH. A study and performance evaluation of the multi-tenant data tier design patterns for service oriented computing. In: IEEE International Conference on E-Business Engineering, 2008. ICEBE'08; IEEE. 2008. pp. 94-101

[22] Khan MF, Mirza AU, et al. An approach towards customized multi-tenancy. International Journal of Modern Education and Computer Science. 2012;**4**(9):39

[23] Momm C, Krebs R. A qualitative discussion of different approaches for implementing multi-tenant saas offerings. In: Software Engineering (Workshops). Vol. 11. 2011. pp. 139-150

[24] Mietzner R, Unger T, Titze R, Leymann F. Combining different multitenancy patterns in service-oriented applications. In: Enterprise Distributed Object Computing Conference, 2009. EDOC'09; IEEE International; IEEE. 2009. pp. 131-140

[25] Yusoh ZIM, Tang M. Composite saas placement and resource optimization in cloud computing using evolutionary algorithms. In: 2012 IEEE 5th International Conference on Cloud Computing (CLOUD); IEEE. 2012. pp. 590-597

[26] Shaikh F, Patil D. Multi-tenant e-commerce based on saas model to minimize it cost. In: In 2014 International Conference on Advances in Engineering and Technology Research (ICAETR); IEEE. 2014. pp. 1-4

[27] Westermann D, Momm C. Using software performance curves for dependable and cost-efficient service hosting. In: Proceedings of the 2nd International Workshop on the Quality of Service-Oriented Software Systems; ACM. 2010. p. 3

[28] Abbott ML, Fisher MT. The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise. Indiana, United States: Pearson Education; 2009

[29] Leymann F, Fehling C, Mietzner R, Nowak A, Dustdar S. Moving applications to the cloud: An approach based on application model enrichment. International Journal of Cooperative Information Systems. 2011;**20**(03):307-356

[30] Aldhalaan A, Menasc, e DA. Near-optimal allocation of VMS from iaas providers by saas providers. In: 2015 International Conference on Cloud and Autonomic Computing (ICCAC); IEEE. 2015. pp. 228-231

[31] Singh A, Chatterjee K. Cloud security issues and challenges: A survey. Journal of Network and Computer Applications. 2017;**79**:88-115

[32] Hon K, Millard C. Eu data protection law and the cloud. International Association of Privacy Professionals; 2020. Available from: https://iapp.org/resources/article/ [Accessed: February 2020]

[33] Google. Google cloud platform and the eu data protection directive. Google Inc.; 2020. Available from: https://cloud.google.com/security/compliance/eu-data-protection/ [Accessed: February 2020]

[34] Ochei LC, Bass J, Petrovski A. Degrees of tenant isolation for cloud-hosted software services: A cross-case analysis. Journal of Cloud Computing: Advances, Systems and Applications. 2018;**7**(22)

[35] Ochei LC, Bass JM, Petrovski A. A novel taxonomy of deployment patterns for cloud-hosted applications: A case study of global software development (gsd) tools and processes. International Journal on Advances in Software. 2015;**8**(3-4):420-434

[36] Ochei LC, Petrovski A, Bass JM. Optimal deployment of components of cloud-hosted application for guaranteeing multitenancy isolation. Journal of Cloud Computing. 2019;**8**(1):1

[37] Walraven S. Middleware and methods for customizable SaaS [PhD thesis]. KU Leuven: Department of Computer Science; 2014, 2014. p. 6

[38] Walraven S, Van Landuyt D, Truyen E, Handekyn K, Joosen W. Efficient customization of multi-tenant software-as-a-service applications with service lines. Journal of Systems and Software. 2014;**91**:48-62

[39] Ochei LC, Petrovski A, Bass J. An approach for achieving the required degree of multitenancy isolation for components of a cloud-hosted application. In: 4th International IBM Cloud Academy Conference (ICACON 2016). 2016