Requirements Eng
DOI 10.1007/s00766-014-0218-7

ORIGINAL ARTICLE

Building a security reference architecture for cloud systems

Eduardo B. Fernandez · Raul Monge · Keiko Hashizume

Received: 30 June 2014 / Accepted: 17 December 2014

Abstract Reference architectures (RAs) are useful tools to understand and build complex systems, and many cloud providers and software product vendors have developed versions of them. RAs describe at an abstract level (no implementation details) the main features of their cloud systems. Security is a fundamental concern in clouds and several cloud vendors provide security reference architectures (SRAs) to describe the security features of their services. A SRA is an abstract architecture describing a conceptual model of security for a cloud system and provides a way to specify security requirements for a wide range of concrete architectures. We propose here a method to build a SRA for clouds defined using UML models and patterns, which goes beyond existing models in providing a global view and a more precise description. We present a metamodel as well as security and misuse patterns for this purpose. We validate our approach by showing that it can describe more precisely existing models and that it has a variety of uses. We describe in detail one of these uses, a way of evaluating the security level of a SRA.

Keywords Security reference architecture · Security patterns · Reference architecture · Security requirements · Secure software development · Cloud computing · IaaS security

E. B. Fernandez (✉) · K. Hashizume
Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL, USA
e-mail: ed@cse.fau.edu

K. Hashizume
e-mail: akhp1@yahoo.com

R. Monge
Departament of Informatics, Universidad Técnica Federico Santa María, Valparaiso, Chile
e-mail: rmonge@inf.utfsm.cl

1 Introduction

Cloud computing systems involve a variety of devices connected to them, may require different deployment models, and provide a variety of services, all of which result in a highly complex system and create many security concerns. Many of the cloud security issues are also true for any kind of distributed system that uses web applications; however, cloud architectures bring new attacks [1, 2] and the result of a successful attack could be catastrophic because an attacker may compromise data from many users. Clouds may store large amounts of sensitive information, and there is a clear attraction for all kinds of attackers, from criminal groups looking for lucre to terrorists looking for disruption.

Cloud systems are very complex, and to study their security, we need to start from their global architectures. A *reference architecture* (RA) is an abstract architecture that describes functionality without getting into implementation details [3, 4]. RAs have become useful tools to understand and build complex systems, and many cloud providers and software product vendors have developed versions of them, e.g. [5–7]. A *security reference architecture* (SRA) is an RA where security mechanisms have been added in appropriate places to provide some degree of security [8]. Until now, however, outside of vendors, very few SRAs have appeared [8–10]. Almost all of them are either partial, have specific objectives, or use rather imprecise and ad hoc models, where implementation details are mixed with architectural aspects. We propose here an approach for building SRAs using patterns described by UML models, which we consider the first attempt to define a precise and

Published online: 06 January 2015

semiformal security cloud computing architecture for the complete cloud environment. We show that SRAs are useful to apply security to cloud systems, to evaluate their level of security, to define Service Level Agreements (SLAs), and for a variety of other purposes.

We developed a cloud reference architecture [11, 12], which we use as starting point for our SRA, and we enumerate its threats for which we find countermeasures in the form of *security patterns*. A security pattern encapsulates a defense to a threat in a concise and reusable way, and we have built a catalog of them as well as a methodology for their use [13]. By checking whether threats can be stopped or mitigated in the SRA, we can evaluate its level of security. We have done a systematic enumeration of cloud threats [2] and have started building a catalog of cloud misuse patterns [14]. A *misuse pattern* describes how an attack is performed to lead to a misuse; with a complete catalog of misuse patterns, we can apply them systematically and use the RA to find where we should add corresponding security patterns to stop them.

Our architecture is semi-concrete, classical, and multi-organization according to the classification of [15] (see Sect. 2), which is oriented to structural aspects; other architectures, e.g. TCI [16], emphasize functional aspects and are complementary. Our way of representing such SRA is semiformal, relying on an object-oriented approach using patterns that include UML models. UML models can be complemented with formal descriptions such as OCL [17], and we can make this architecture more formal if needed. We believe that a semiformal approach is the only practical approach given the complexity of the system we are considering; of course, parts of the architecture can be formally modeled. Purely formal methods are difficult for most practitioners [18]. Our approach is in tune with the idea that security needs a global and holistic approach [19, 20]. We have not strived for completeness; a SRA is a very large system, the NIST SRA [8] takes more than a hundred pages, but many of its functions are variations or repetitions of some idea and there is no need to show every case; we only want to show an approach for building such an architecture, demonstrate its value, and build some of its parts to illustrate our ideas. Since our architecture is built out of patterns, we show a security pattern and a misuse pattern to illustrate the approach. The most critical functions of security are in the IaaS level, and we concentrate on this level, but we show its relationship to the upper levels (PaaS and SaaS). Identity management aspects are left out since they are used also for other functions [21]. Other complementary aspects have been presented as patterns [13].

Our contributions include the following:

- A systematic approach to build SRAs for clouds which uses patterns and that produces architectures which are more abstract[1] and precise than the SRAs in the current literature. The SRA specifies security requirements for a range of cloud systems, starting from an abstract model.
- New patterns to add to the few existing security patterns for clouds and a list of other possible patterns.
- A new misuse pattern to add to the only three cloud misuse patterns published until now, as well as a list of possible misuse patterns for clouds.
- A metamodel to relate together the concepts of SRAs, which helps to unify concepts.
- A list of other possible uses of SRAs, which is a way to validate them and to justify the effort in building them.
- A way to enumerate threats to evaluate the security of cloud systems, which expands our own approach [22] and can have more general application.

Section 2 describes background and summarizes our previous work on this topic. Section 3 discusses related work describing several industrial and academic SRAs. Section 4 describes our approach for securing RAs and a metamodel to relate the concepts we use. Section 5 enumerates stakeholders (roles) and some use cases found in clouds. Section 6 considers how to enumerate threats and how to defend against them using security patterns. Section 7 talks about misuse patterns and shows a misuse pattern for malicious VM migration. Section 8 presents a partial view of the resulting SRA. Section 9 presents a security pattern for a virtual machine (VM) image repository and a model for security administration. Section 10 discusses validation aspects and provides a list of possible uses, which include how to evaluate the degree of security of a SRA and shows that it subsumes previous models. We end with conclusions in Sect. 11.

2 Background and our previous work

As indicated, we use patterns as building blocks for our SRA. A *pattern* is a solution to a recurrent software problem in a given context. This solution resolves a set of forces that constrain and define guidelines for it, e.g. "the solution must be transparent to the users". The solution is usually expressed using UML class, sequence, state, and activity diagrams (although we usually do not need all these models). OCL constraints can add formality if needed. A set of consequences indicate how well the forces were satisfied by the solution as well as the possible negative aspects of the solution. An implementation section provides hints on how to use the pattern in an application, indicating what steps are needed and possible realizations.

[1] with no implementation details.

A section on related patterns indicates other ones that complement the pattern or that provide alternative solutions. A pattern embodies the knowledge and experience of software developers and can be reused in new applications. Patterns are also good for communication between designers and to evaluate and reengineer existing systems [13]. A *security pattern* describes a mechanism or procedure to defend against an attack. *Abstract Security Patterns* (ASPs) describe conceptual security mechanisms that realize one or more security policies able to handle a threat or comply with a security-related regulation or institutional policy [23]. For building conceptual models (i.e., implementation independent), we developed a type of pattern called *Semantic Analysis Pattern* (SAP), which implements a small set of coherent use cases [24]. A *misuse pattern* describes how a misuse is performed from the point of view of the attacker. It defines the environment where the attack is performed, countermeasures to stop it, and provides forensic information in order to trace the attack once it happens [25, 26]. *Enterprise patterns* refer to problems at the enterprise level, including middleware and database aspects [27].

A *reference architecture* is a standardized, generic architecture, valid for a particular domain that does not contain implementation details [3, 4]. An RA provides a template-like solution that can be instantiated into a concrete software architecture by adding implementation-oriented aspects. There is no consensus about what an RA should contain; Avgeriou [3] describes an example and indicates what should be included in one, typically a class diagram and a set of use cases with their roles, but other authors include other aspects [28, 29]. Our work on Semantic Analysis Patterns [24], was a step toward building RAs out of patterns, an idea also used later in [29, 30] There are also several RAs for cloud computing, some of them are abstract, while others focus on specific areas or products [29, 31], and we survey later some of them which are often paired with corresponding SRAs.

As indicated, we use UML for describing the SRA. UML is a semiformal language whose syntax is formally defined using a metamodel [32]. It is widely used, many tools support its use, it is an industry standard, and it is familiar to a wide segment of practitioners. It can also be complemented with formal methods, and its standard defines an associated formal language, OCL [17]. Being a graphic language, it is highly intuitive and has a direct correspondence to code. There exists an extensive literature on design and security patterns, and the majority of them describe their solutions using UML.

Purely formal methods, while more precise and supporting formal proofs of properties, are much harder to use by practitioners, and none of their languages is prevalent, much less a standard. UML has limitations as an architectural language, and more specialized languages exist [33]. Again, none of these languages is a standard and for describing structural aspects UML appeared acceptable. In our work, we do not discard the use of formal and/or architectural languages for the applications of SRA described in Sect. 10; we just think UML is convenient.

We developed a systematic way of enumerating threats by analyzing each activity in each use case of a system and considering how it could be attacked [22]. This approach finds all the high-level threats as goals of the attacker and can be expanded at the lower architectural levels with threat catalogs, e.g. [34]. However, we do not need to predict all possible attacks; we can make the system able to preserve its critical assets, even if parts of the system have been compromised. This is the "submarine" approach, where if one compartment is flooded, critical functions can be protected. In order to use this idea, we need to define clear interfaces between units and validate their interactions.

In order to understand cloud security better, we performed a systematic review of their security issues, where we collected the main cloud threats and vulnerabilities found in the literature [2]. In that analysis, we presented a categorization of security issues focused in each service model (SaaS, PaaS, and IaaS), and we identified which service model can be affected by specific threats. Also, we described the relationship between these threats and vulnerabilities and provided possible countermeasures (security patterns) for each identified threat. For the preparation of this paper, we also looked at other analyses of cloud security issues [16, 35–37]. In particular, Kalloniatis et al. [38] was useful to provide a conceptual framework.

We described three specific cloud threats in the form of misuse patterns [14]: Resource Usage Monitoring Inference, Malicious Virtual Machine Creation, and Malicious Virtual Machine Migration. The Resource Usage Monitoring Inference misuse pattern describes how an attacker by colocating his virtual machine in the same server as the victim can infer some information. The Malicious Virtual Machine Creation misuse pattern depicts how an attacker can create a virtual machine image (VMI) which contains malicious code in order to infect other virtual machines that use this image. Malicious Virtual Machine Migration describes how a virtual machine can be compromised while being migrated to another server. We talk more about misuse patterns in Sect. 7.

We developed an RA that provides a conceptual view of cloud systems [11, 12]. Since clouds are complex systems, we presented each service model as a compound pattern[2] which describes its requirements, characteristics, main units, and the relationship between these units. We also

[2] A pattern composed of simpler patterns.

included some use cases that describe common functions for cloud services in general as well as for each service model. The model of Fig. 1 is the highest level view of the architecture, the cloud computing environment, called a cloud ecosystem in [8], and its main purpose is to provide a perspective of its components. Each component can be refined in the same way to define concrete architectures; for example, the network model can include firewalls or software-defined network (SDN) structures [39]. In Sect. 9, we show a refinement of its administrative structure. We build on that RA to define our SRA.

We summarize now this RA (Fig. 1). The *Portal* is the way to access cloud services. A *cloud* is composed of cloud services, infrastructure (cluster), and support services. Cloud physical resources can be located in different zones or *clusters*. A cluster is a collection of *nodes* that are located within a close physical proximity. A node is made up of a set of *Hardware* units (*Servers*, *Storage* and *Network*), virtual machines (*VMs*), and a virtual machine monitor (*VMM*). A VMM creates and manages virtual machines and makes direct access to the hardware on behalf of them. The fundamental cloud service levels are Software-as-a-Service (*SaaS*), Platform-as-a-Service (*PaaS*), and Infrastructure-as-a-Service (*IaaS*). The SaaS provides on-demand *Applications*, while the PaaS offers *Virtual Environments* such as *Development*, *Deployment*, and *Testing Environments*, which include programming languages, databases, libraries, and other tools. IaaS provides virtualized resources such as servers and storage that can be assigned to virtual machines. *Support services* are needed to provision the creation, implementation, and management of cloud services which we call collectively a Cloud Management Point, discussed in Sect. 9. *Business Support Services* provide centralized management of cloud resources, including metering, billing, reporting, and account administration. *Operational Support Services* are responsible for monitoring, provisioning, and other management functions such as configuration, upgrading, and installation of the system. *Non-Functional Services* include security, privacy, availability, reliability, interoperability, and possibly other quality factors.

3 Related work

A number of SRAs have appeared, most of them from cloud vendors; we discuss them below. Some papers propose architectures for specific defenses against one type (or a few types) of attack. Chonka et al. [35] describe a mechanism to detect and filter DoS attacks against clouds. Prolexias products also defend against DoS [40], and a survey of IDS techniques for clouds is given in [41]. Juels

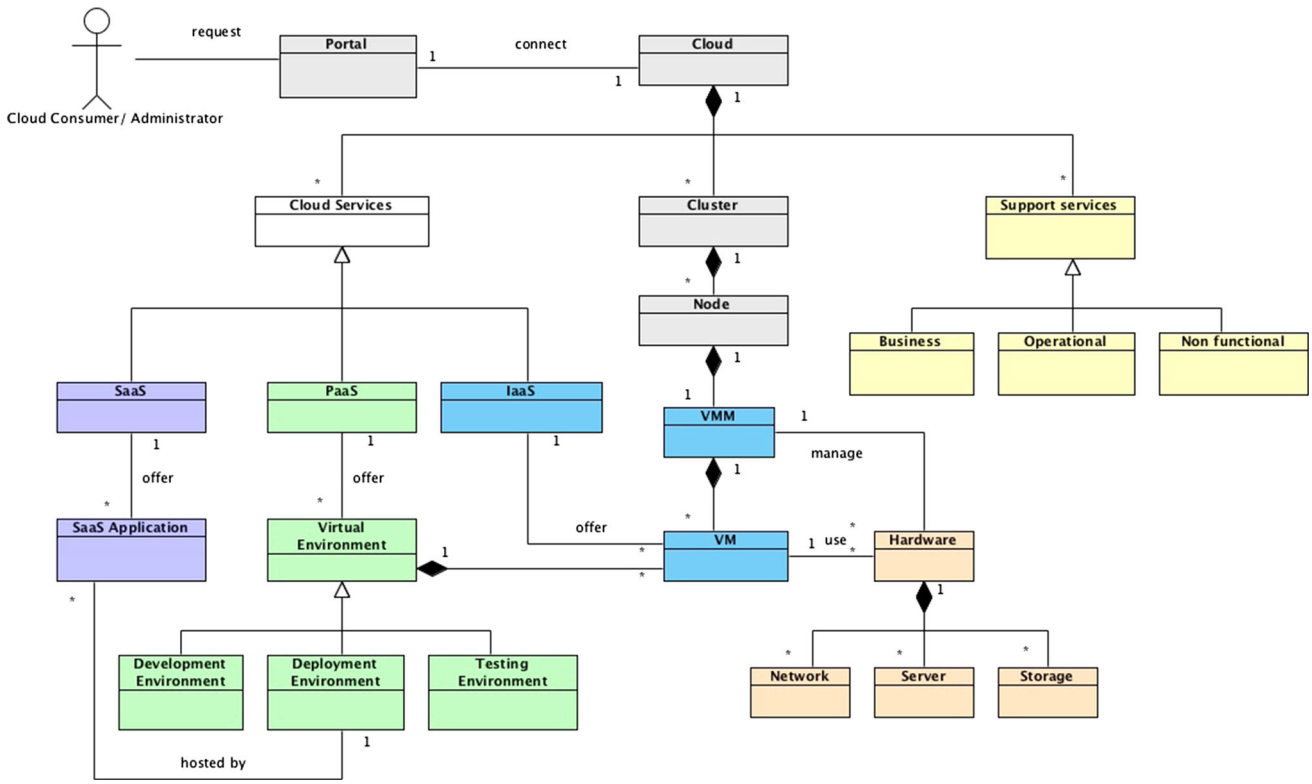


Fig. 1 Class diagram for a cloud computing environment

and Oprea [42] include an authentication structure, an auditing approach, and an availability mechanism and focus on the integrity of the data stored in the cloud. Data risks in the cloud are discussed in [1]. EMA [43] emphasizes the need to protect the administrative functions. Although they are useful works, their scopes are too narrow to define SRAs. Some ontologies have also been proposed, e.g., [44, 45], but they mostly define terms used in cloud computing, to be used in cloud selection and recommender systems. Others discuss general security issues in clouds [36, 38].

Other papers consider secure architectures oriented to some specific objectives. Lombardi and Di Pietro [46] start by analyzing the origin of attacks and propose a secure architecture[3] for IaaS. Their approach assumes a Trusted Computing Base that provides trusted VMs. They define the requirements for a monitoring system that watches for modifications to the kernel data and code. Their architecture includes logging and periodic checksums of executable files and libraries as well as analysis of performance overhead, and they tested it using some known attacks. Campbell and his group are building middleware for assured clouds [9]; their system handles security and reliability through a set of agents. Neither paper uses a global architectural model of the system to define these functions. Ruth Breu's group proposed an RA for the security services of the SaaS level [10]. They use enterprise patterns but no security patterns. Their SRA is basically a deployment model. They do not try to relate SaaS security to IaaS security, which may result in redundant mechanisms and services. A theoretical model of a few specific cloud platforms uses a Petri net model with a simplified architecture and no security [47].

A few security architectures consider security functional aspects, as opposed to ours which addresses structural aspects [16, 48]. The best known of these is the Trusted Cloud Initiative (TCI), which was proposed by the Cloud Security Alliance (CSA) [16], and presents a set of functional layers: presentation, application, information, and infrastructure. These approaches are complementary to our SRA's approach.

Several SRAs come from industry:

- IBMs reference architecture includes business support services (accounting, billing), a SLA model, customer management, operator support services, virtualization management, monitoring, event management, and image lifecycle management [6, 49]. Its version 3.0 has also a section on security [50]. They use LDAP-based authentication, role-based access control (RBAC), encryption, and other mechanisms as well as following the OAuth standard [51]. All this is described in an informal way using lists and block diagrams.
- Microsoft uses role-based access control (RBAC) applying a need-to-know policy to access resources, as well as multifactor authentication, and logging/auditing functions [7]. Interesting features include protection of network DNSs using ACLs and secure lifecycle development for applications. They apply a defense-in-depth strategy and perform penetration testing. They also mention the use of code patterns and tool-based validation.
- Okuhara et al. [52] describe Fujitsu's security architecture which emphasizes the logical separation of computational environments, source code reviews, authentication, and identity management (using WS-Federation). They also bring up the need for transparency from cloud providers, the use of a security dashboard for visualization of security functions, and the need to separate logging and monitoring functions.
- Amazon describes in [53] the security aspects of their web services which include their cloud services. They comply with the Payment Card Industry (PCI) Data Security Standard (DSS) [54], and control configuration management as well as standard authentication and authorization functions. The paper also describes the security of their virtual private clouds, of their MapReduce database, and the handling of their firewall protection. Their security features are presented as a list, and no architecture relating them is given.
- VMware describe their SRA in terms of their hardware units with a few details of their functional aspects and its mapping to the PCI architecture [55].
- Oracle describes their SRA in [56]. They have three versions of it, oriented to data security, fraud detection, and compliance. These architectures are mapped to their products.
- Cisco has a SRA called SAFE [57]. They claim to apply principles like defense in depth, modular design, and best practices; but they do not offer much detail of their SRA, although they claim to have more detailed descriptions.
- Juniper Metafabric architecture [58] emphasizes network aspects based on SDN.
- Trend Micro provides a centralized management interface for physical, cloud, and virtual end point security tools, while integrating the policies for end point security into their SecureCloud offering [59]. Their SRA is used to integrate their security mechanisms.
- Other companies have published reference architectures—e.g., Eucalyptus [60], Hewlett Packard [5]—but in them security is barely mentioned. Some patterns for cloud computing—including a few security patterns—

[3] A secure architecture is a specific architecture with some security properties, while a SRA is a generic model representing the security features of any architecture.

are assembled in the form of a SRA in [61], but they do not provide much detail.

Two SRAs come from standards organizations:

- A complete and specialized SRA is the PCI-compliant cloud reference architecture, which defines a basic framework for building clouds that are compliant with the PCI DSS standard [54]. This SRA defines a cloud architecture using products such as VMware, Cisco, Trend Micro, and HyTrust. It shows how the security controls obtained through these products can meet the PCI DSS requirements. This architecture consists of four fundamental layers: The cloud application layer represents the external interface for user access to cloud services. The business orchestration layer consists of the configuration of the cloud entities and the governance policies for controlling the cloud deployment. The service orchestration layer coordinates services, and the infrastructure layer defines the platforms to be used. This architecture attempts to provide security requirements that cloud providers should follow in order to meet the PCI DSS standards using specific products. However, this standard is not based on threat analysis and uses specific products of the companies of the authors of the architecture.
- The most comprehensive work on a cloud SRA comes from NIST, which has published a report that describes in great detail the aspects of such an architecture [8]. It also describes the functions of the Broker, the Auditor, and the Communications provider that we consider part of the cloud computing environment. This architecture is more general than those of vendors and also more abstract. However, its model is not very precise and uses only block diagrams. They do not consider threats either.

The SRAs from commercial sources emphasize the use of their own products and are not general or formal enough to be used for research or even to select a specific architecture; Muller and van der Laar emphasize that RAs should not be system or product-line specific; the NIST SRA is more abstract, but still imprecise [30]. In all of these architectures, the lack of more precise or rigorous models is a clear weakness. They typically use block diagrams which show the involved components, but not the way they associate with each other; for example, the fact that an association between components is one to many is not shown in the models.

It is not clear in those models where the security mechanisms should be attached to specific functional units; saying that authentication is used in the system is not enough. Because they do not use patterns, it is difficult to see commonalities between subsystems; every subsystem is an ad hoc unit. Some academic papers are more rigorous, but they only focus on specific mechanisms and lack a global view. Except for [46], none of these SRAs considers threats (at least not explicitly), to determine what security mechanisms should be included. Our work attempts to improve this situation, and it subsumes all these models in that all their specific architectures can be produced as specializations of the SRA.

The closest work to our paper we have found also uses patterns, but their objectives are different: They are establishing a cloud-specific security management system [62]. The functions of such a system include determining assets, consideration of regulations, policy definition, and privacy. Their model is oriented to fulfill the ISO 27000 security regulations.

4 Securing a cloud reference architecture

We show now how to build a SRA and present a metamodel to relate the concepts we use. Subsequent sections provide details of the steps. Remember that our objective is to show how to produce a SRA, not to present a complete one.

4.1 Procedure to define needed security services

We show a set of steps to find out what security services we need and where to insert them in the functional architecture. It is not a methodology to build secure applications as the ones surveyed in [63]. The steps, described in Fig. 2, include the following:

- We start from typical cloud use cases and their associated roles. Lists of cloud use cases and roles are shown in [28, 64] (see Sect. 5).
- We analyze each use case looking for vulnerabilities and threats as in [22]. This implies checking each activity in the activity diagram of each use case to see how it can be attacked. This approach results in a systematic enumeration of threats. We use the list of threats from [2] to confirm these threats and to find possible further vulnerabilities and threats (see Sect. 6).
- These threats are then expressed in the form of misuse patterns. We developed some misuse patterns for cloud computing in [14], and we consider more of them here (see Sect. 7).
- We apply policies to handle the threats, and we identify security patterns to realize the policies. There are some defenses that come from best practices and others that handle specific threats. There are also regulatory policies which are realized as security patterns. We use an example of cloud administration (see Sect. 9).

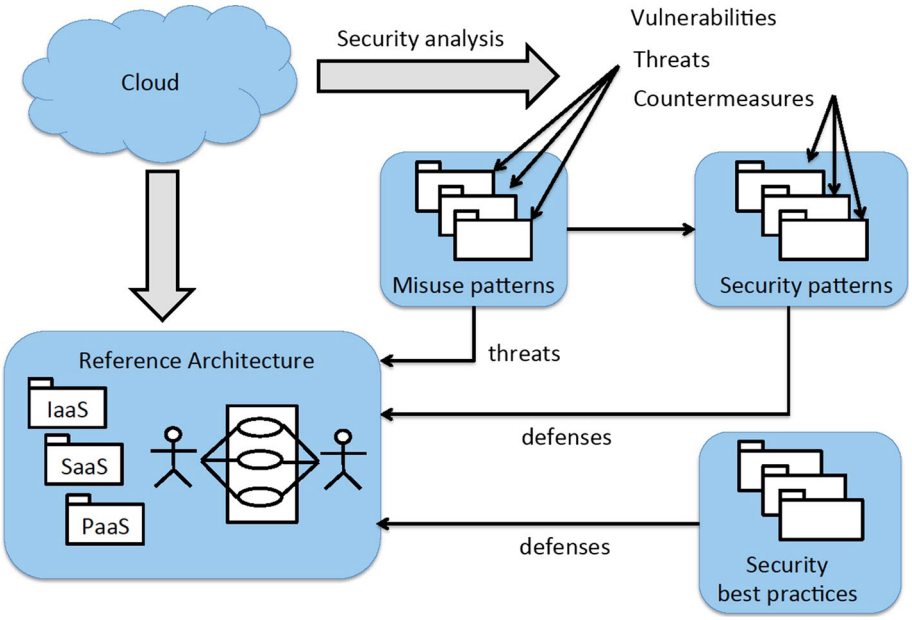


Fig. 2 Securing a cloud reference architecture

- We refine sections of the architecture and secure them in similar fashion to get to the final model (see Sect. 8).

The justification of these steps is based on the fact that use cases define all possible interactions with the system if we leave out the possibility that the attacker can have physical access to the cloud. If we analyze each activity in each use case, we can identify all threats for which we can later find defenses. We show in the next sections all the steps above in more details, but first, we present a metamodel to relate our concepts.

4.2 A metamodel for securing clouds

Figure 3 relates our security concepts to each other. *Threats* take advantage of *vulnerabilities* that can exist in any cloud service level. A vulnerability is a flaw in the system implementation or in its configuration and use. A SRA is not concerned with vulnerabilities, but with the use of them in its concrete instances by attackers to reach their goals (threats). Threats come from analysis of *use cases* (Sect. 6.1) or from published *threat lists* [2, 34]. Each use case has a set of *roles* that describe the participants in the use case. We can stop a threat by removing the corresponding vulnerability or by controlling its propagation (by removing other vulnerabilities) through the use of a *Security Pattern*. The security pattern to use can be selected from the countermeasures defined in the *misuse pattern* which describes the threat (see Sect. 7). As indicated, we can also select security patterns to apply from the list of threats, but it is more economic to select only the security patterns needed to stop the identified misuses. In other words, there could be a threat that may not lead to any significant misuse, and we do not need to prevent it. Security patterns can also be selected from some methodology based on patterns [63], even if those methodologies are oriented to build specific types of applications. Threats that lead to misuses are the goals of the attacker and are performed through low-level threats in the threat list or directly through a use case operation.

Some threats apply to all service levels. For example, buffer overflow is a language problem and allows escalation of privilege by the attacker operating at any level. Other threats are specific to the level; for example, a financial application can be attacked by taking advantage of lack of proper authentication in remote access to accounts. If the threat takes advantage of a flaw in an application, it may compromise the security of that application. If the threat affects the IaaS level, it affects all the cloud computations; and if it happens at the PaaS level, it can affect all the applications developed or deployed in the cloud.

5 Stakeholders and use cases

As indicated, use cases define all the interactions of users with a system. If we enumerate use cases and look at all their activities (steps of the use case), we can find all the places where an attacker can compromise the system. Determination of threats is based on postulating the possible ways in which an attacker could get some gain. We look first at the stakeholders and then at the use cases of a cloud system.

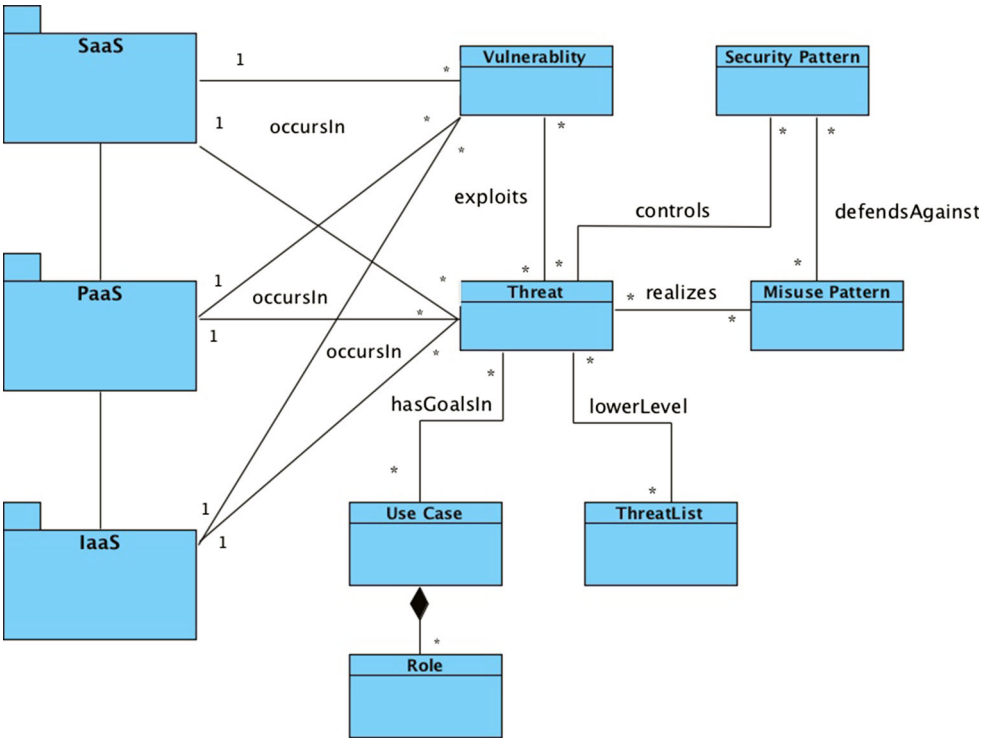


Fig. 3 Metamodel of the concepts used to secure clouds

5.1 Stakeholders (actors)

We need to look at the actors (defined by their roles) which participate in the operation of a cloud system. We identify here the stakeholders involved in the use and operation of the cloud, which include the following:

- *Service Consumer* (SC), or customer—this can be an individual or an institution, collectively denoted as Party,[4] including different roles such as end user, developer, IT manager, and others. They can use services at any level.
- *Service Provider* (SP)—it is a company or institution providing a set of cloud services, a complete level of services (e.g., PaaS), or a specific set of services (possibly through a cloud broker).
- *Cloud Administrator*—a person or group of people in charge of cloud management. One of them is the Security Administrator. There may be also a Resource Administrator, a Virtualization Administrator, and others [49].
- *Cluster Administrator*—clusters correspond to geographic zones and each zone may have a local administrator.
- *Cloud Auditor*—person who inspects compliance with regulations or standards.
- *Service Broker* (or cloud broker)—trusted parties that combine services from different SPs and enforce regulations and practices [8, 66].
- *Cloud Builder*—the team who sets up the operations of the SP [67].
- *Cloud Application Builder*—those who build applications to execute in the SP or use services from the SP.

There are others, all of which correspond to roles or actors in some use cases. Sets of stakeholders, developers using PaaS, are given in [49, 68]. These roles participate in one or more use cases, which define their interactions with the cloud system. We show in the next section some uses cases for security administration. Similarly, we should develop sets of use cases for virtual machine management and other functions.

5.2 Use cases for security administration

Security administration is a fundamental function of the SP. We need a complete and usable structure to let administrators define an effective security structure. This structure should be part of a more encompassing service management unit. A typical realization is in the form of a Policy Management Point (PMP), a type of security dashboard for security administrators to analyze the security status of the system and where administrators can define access rules [52, 69].

We show here some use cases for the functions of the security administrator. We also need to add security

[4] Party is also a pattern [65].

functions to every use case for the cloud customers; e.g., most use cases require at least login, which usually implies authentication and in order to access resources users need authorization and logging.

Without loss of generality, we assume role-based access control (RBAC) as authorization model. In RBAC, a user is assigned to roles, and rights are assigned to a role [70]. There are some security functions that can be applied to all cloud models as well as other types of systems. This is just an illustrative list; it is necessary to define a structured governance function from which security functions and policies can be derived.

- *Login* Provides entrance to a portal for users to access cloud services (usually an "include" use case).
- *Create user* A user can be a single individual or an institution (collection of users), described as a Party pattern.
- *Delete user* Once a Service Provider employee leaves the company or a customer closes his account, the account that corresponds to him is erased.
- *Create role* A role defines a task that a person performs in his job. Examples of roles were shown in Sect. 5.1. Rights are assigned to roles.
- *Delete role* A role can be deleted if there are no users associated to it or it is not considered useful.
- *Assign rights to a role* Rights define the functions that a role can perform in the system. For instance, a cluster administrator can migrate virtual machines within his jurisdiction.
- *Assign roles to a user* Users must be given roles before interacting with the system.
- *Set up security options* Security administrators define what cryptographic measures will be implemented across all cloud layers. They also set up the authorization model and the authentication methods.

Once we know the roles involved and the way they interact with the cloud, we can find the threats to the system.

6 Identifying and controlling threats

In order to define the required defenses, we first identify threats to this system and then consider how to control them. We apply here our threat enumeration and control approach [22].

6.1 Identifying threats

Some authors, e.g., [71], define security mechanisms based on security attributes derived from institution policies or previous security analysis. General measures result in excess of security mechanisms, which is costly and reduces performance. Security mechanisms should be added to the system in order to control specific threats. We can enumerate threats systematically by considering each activity in each use case and analyzing its possible threats.[5] The approach considers all the activities where attacks can occur. For illustration, we use an example of a VMI Repository, which stores VM images for use by service consumers and which is part of the administrative functions of the cloud. We apply to each action in this repository the STRIDE attacks [73]; e.g., read a VMI (confidentiality attack), or tamper with a VMI (integrity attack). We leave out of the list Escalation attacks because they are not attackers' goals, they are just means to their goals. The specific types of attacks may not be exhaustive, and we complement them with the analysis of security threats in [2], which lists cloud threats described in the literature, including the OWASP list [34]. We do not show all the identified threats, e.g., DoS, because they would make the diagram hard to read, we use a table as in [22].

Figure 4 describes an activity diagram that corresponds to a sequence of use cases: Create VMI and Publish VMI. For example, when a consumer creates a VMI, she can inject in it malicious code (see Sect. 7). Table 1 summarizes an analysis of each action in the activity diagram according to the security attributes which may be compromised, the source of the threat, and the assets that can be compromised. The affected security attributes are confidentiality (CO), integrity (IN), availability (AV), and accountability (AC). The source of the threat can be an authorized insider (AIn), an unauthorized insider (UIn), or an outsider (Out). A1–A4 represent the normal flows, the rest are malicious flows of information. T_{ij} denotes the jth threat in Activity A_i.

6.2 Cloud defenses

We have developed a variety of security patterns for all the architectural levels of a system, including some for middleware distribution concerns [13]. Several of these apply directly to clouds, but others may need to be adapted for cloud environments. We also need new patterns. In this section, we provide a list of some security patterns for clouds and provide some examples, which can be used as general guidance. We use again the VMI Repository as example.

After we have identified threats by analyzing the activities of use cases, we can find security patterns that mitigate or stop these threats. We can now evaluate whether these security patterns (defenses) cover the threats. For instance, as shown in Table 2, we can apply existing security patterns to mitigate or stop some of the threats identified in Table 1. The Authenticator and Authorizer

[5] Note that this is more precise than using misuse cases [72].

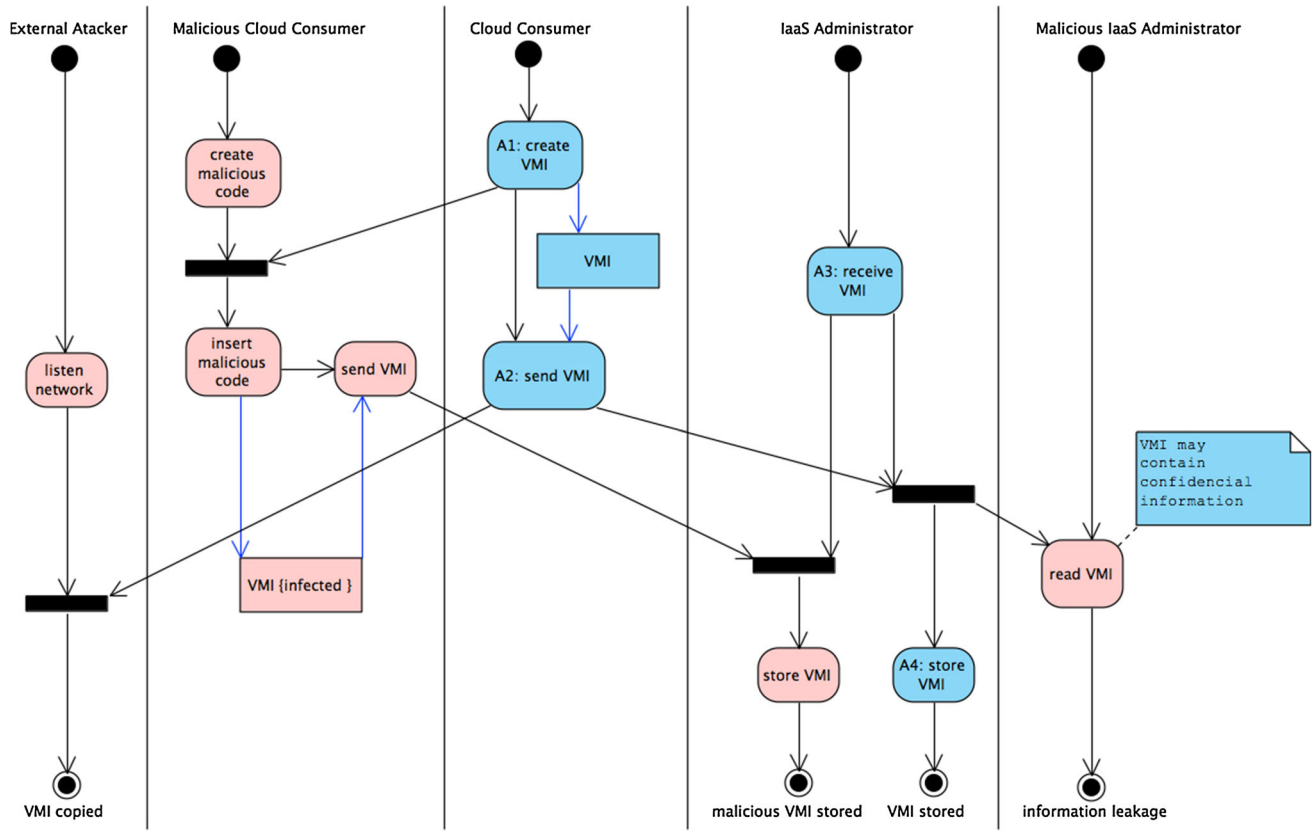


Fig. 4 Activity diagram for use cases "Create VMI" and "Publish VMI"

Table 1 Misuse activities analysis

Actor	Action	Misuse activity					
		ID	Security attr. CO/ IN/AV/AC	Source AIn/ UIn/Out	Description	Attacker	Asset
Cloud consumer	A1: Create VMI	T11	IN	Out	Insert malicious code in the image	Malicious consumer	VMI
Cloud consumer	A2: Send VMI	T21	CO	Out	VMI may be read and copied while being transmitted	Extern	VMI
		T22	IN	Out	VMI may be modified while in transit	Extern	VMI
		T23	AC	Out	Disavows sending a VMI	Malicious Consumer	VMI
IaaS administrator	A3: Receive VMI	T31	CO	AIn/UIn	Collects sensitive information from VMI	Malicious admin	VMI
		T32	AV	AIn	Disavows receiving a VMI	Malicious admin	VMI
		T33	IN	UIn/AIn	Insert malicious code in the image	Malicious admin	VMI
IaaS administrator	A4: Store VMI	T41	IN	UIn/AIn	Store poisoned VMI	Malicious admin or consumer	VMI

patterns [13] can stop T11, T31, T33, and T41, because now only authenticated and authorized users can publish a VMI. The Filter component mitigates threat T41.[6] Any authorized user can store his VMI in the public repository, but it will be scanned to remove any malicious code before being stored, although this action does not guarantee that malware will be eradicated. The Secure Channel mitigates T21 and T22, which provides integrity and confidentiality while the VMI is being transmitted. These patterns will

[6] As there is no pattern for this function, we can consider it a "best practice".

Table 2 Threat List versus defenses

ID	Threats	Defenses
T11	The cloud consumer is malicious and inserts malicious code into the VMI	Authenticator–Authorizer
T21	An external attacker listens to the network to obtain information about the VMI	Secure Channel
T22	VMI may be modified while in transit	Secure Channel
T23	Disavows sending a VMI	Security Logger/Auditor
T31	The IaaS administrator is malicious and collects information within the VMI	Authenticator–Authorizer
T32	The IaaS disavows receiving a VMI	Security Logger/Auditor
T33	Insert malicious code in the image	Authenticator–Authorizer
T41	The IaaS administrator stores a malicious VMI	Authenticator, authorizer, filter

become components of the Secure VMI Repository pattern as well as of the SRA. In general, there are several systematic methodologies to use patterns to stop threats [31].

Moreover, since web services are commonly used in clouds, we can apply security web services standards to secure cloud environments by using concrete versions of the patterns used in Table 2. We have developed some patterns for security web services standards such as WS-Security, XML Encryption, and XML Signature, most of which are surveyed in [74, 75] and described in [13].

7 Misuse patterns for SRAs

7.1 Applying misuse patterns

Misuse patterns start from the goals of the attacker and describe the ways the attacker accomplishes her goals in a specific architecture [25, 26]. A misuse pattern describes the sequence of messages that the attacker sends to different components of the architecture in order to reach her goals. The involved components are some of the components of the RA (or from refinements of some component), which act as guidelines to locate the actions of the pattern. The misuse pattern also indicates in which units we can collect evidence of an attack; again the units of the RA are used as guidelines. This implies the following:

- Misuse patterns cannot be used at the requirements stage because at that moment the architecture is not yet defined, we need an RA as a guideline.
- There may be several variations of the misuse pattern that correspond to the different ways that an attacker can accomplish her objectives.
- We can use the misuse patterns to evaluate the security of the final model, as shown in Sect. 10.2.

For illustration purposes, we show a misuse pattern in the next section (Sect. 7.2). This work can be extended by completing the catalog of misuse patterns to include those threats identified in [14]:

- *Covert channels in clouds* covert channels allow inter-VM communication bypassing the security rules of the hypervisor.
- *Virtual machine escape* it describes how to exploit the hypervisor in order to take control of the underlying platform.
- *Virtual machine hopping* it describes how a virtual machine can access other virtual machines, for example, by exploiting the hypervisor.
- *Sniffing virtual networks* it describes how a virtual machine can listen to the virtual network traffic in order to get confidential information.
- *Spoofing virtual networks* it describes how a malicious virtual machine can intercept information in the virtual network with the purpose of altering its routing function.

7.2 Malicious virtual machine migration process (misuse pattern)

7.2.1 Intent

The attacker tries to provoke leakage of sensitive information or modify the VM content while it is in transit.

7.2.2 Context

Cloud environments rely on virtualization. The virtual machine monitor (VMM) provides the foundation for virtualization management. One important use of a VMM is the migration of a VM from one VMM to another for reasons of availability, hardware maintenance, fault tolerance, and load balancing [76].

7.2.3 Problem

To perform some types of misuse, it is necessary to be able to monitor and intercept the transfer of a VM from one server to another.

The attack can be performed by taking advantage of the following vulnerabilities:

- The migration process involves transferring the VM content across a network that can be insecure such as the Internet.
- The VM may be transferred in clear text. Thus, its information can be captured or modified by an attacker.
- The VMM module that handles migration operations can be compromised and the VM directed to the attackers' node.
- A VM can be transferred to a compromised host where its contents can be accessed by the attacker.
- The content of the transferred VM may have malicious code and compromise the receiving node.

7.2.4 Solution

When a VM is transferred from one server to another, an attacker can monitor the network and obtain some confidential information or manipulate the VM content while it is in transit. Also, the attacker can intercept the VM transference and modify some sensitive information or inject malicious code into the transferred VM (Man-in-the-Middle attack). Finally, the attacker can compromise the VMM and gain full control of the migration process.

Structure

Figure 5 shows a class diagram for VM Migration Process. A *Party* can be either a *User* or an *Institution* (set of users). A Party can have several *Accounts*. A Party makes requests to the *Cloud* Controller via a *Portal*. A cloud is composed of *clusters*, where each cluster comprises a set of nodes. A node is a collection of *Hardware* (*Servers*, *Network*, and *Storage*), a *VMM*, and a set of *VMs*. A VMM creates and manages VMs. A VM is a software implementation of a machine that executes programs. Its kernel operations are performed by calls to the VMM. VMMs assign instances of the virtual machine to a physical server, which includes other hardware resources. The VMM manages the migration process of a VM from one server (VMM) to another.

Dynamics

UC1: *Man-in-the-middle attack during VM migration process* (Fig. 6)

Summary: An attacker listens to the network during a migration process to obtain some confidential data.

Actor: Attacker, Cloud Manager.

Precondition: The attacker has impersonated both the source and the destination VMM

Description: The use case considers following actions.

(a) The attacker starts monitoring the network traffic.
(b) The Cloud Manager requests to migrate a VM from one VMM to another one. The request will be forwarded from Cloud Controller to Cluster Controller, and then to the Node Controller, and finally the VMM will perform the migration process.
(c) The source VMM requests VM migration to the destination VMM.
(d) The source VMM starts transferring the VM to the destination VMM.
(e) The attacker captures the traffic being transmitted.
(f) The attacker modifies or injects malicious software to the VM.
(g) The destination VMM receives the VM and starts the new VM.

Postcondition: The attacker captured and modified the VM, which can lead to future attacks.

UC2: *DoS by migrating many VMs to a victim VMM* (Fig. 7)

Summary: A large number of VMs are transferred from a compromised VMM (VMMc) to the victim (VMMd).

Actor: Attacker.

Precondition: The source VMM has been compromised and the attacker has gained control of the migration module.

Description: The use case considers following actions.

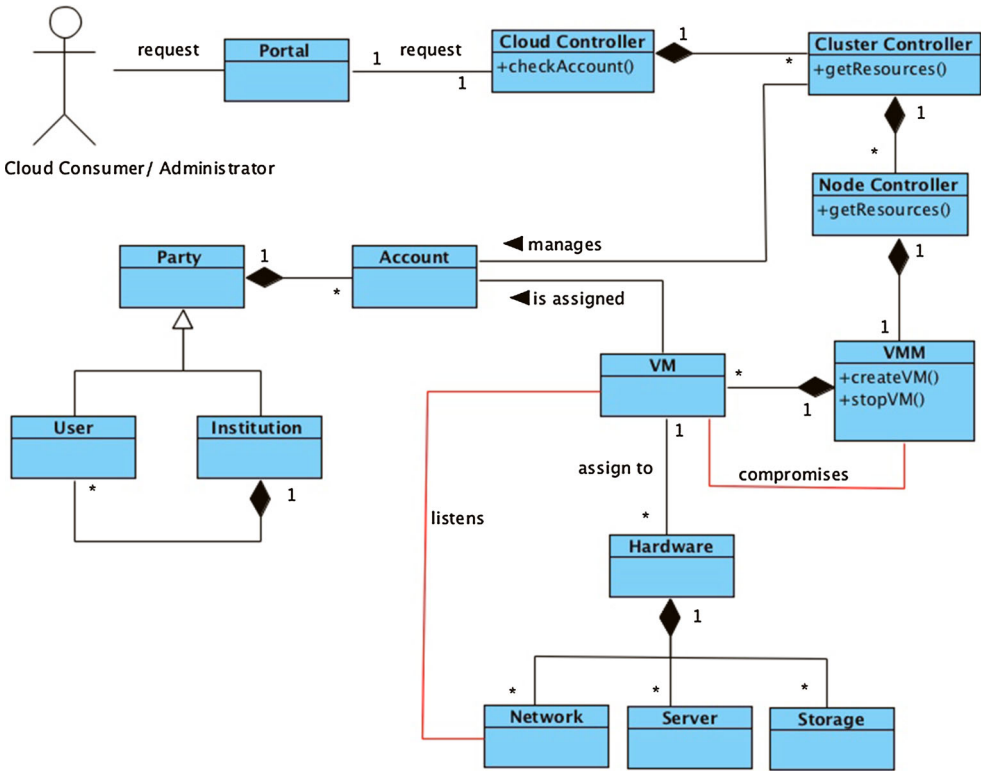
(a) The attacker requests to migrate a list of VMs to the victim VMM. The request will be forwarded from a Cloud Controller to a Cluster Controller, then to the Node Controller, and finally the VMM will perform the migration process.
(b) The compromised VMM requests VM migration to the victim VMM (VMMd).
(c) The destination VMM accepts the request.
(d) The compromised VMM starts transferring several VMs.
(e) The victim VMM receives the VMs and starts the new VMs.

Postcondition: The attacker has overwhelmed the victim machine by migrating a large number of VMs to the destination machine.

7.2.5 Consequences

Some benefits of the misuse pattern are the following:

- The attacker can intercept the VM and obtain some confidential information, or he can modify the content of the VM while it is crossing the network.
- After the attacker compromises a VMM, he may send a large number of VMs to a victims' machine, causing disruptions or denial of service.
- A compromised VMM can transfer the victims' VM to the attackers' machine, gaining full control of the VM.



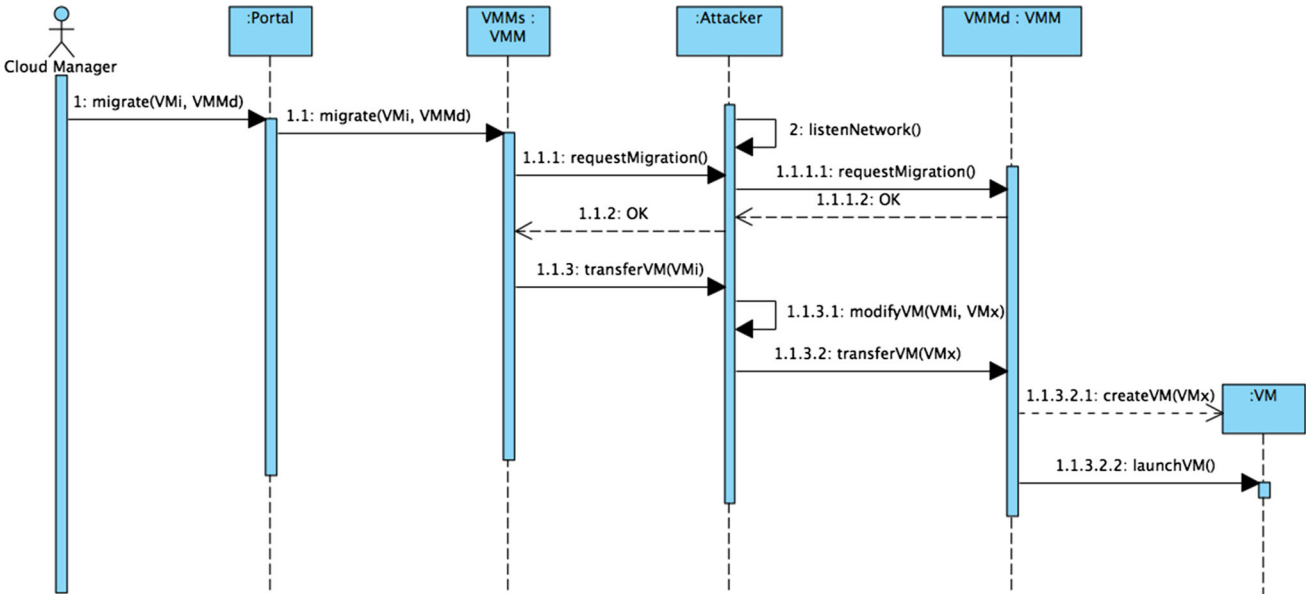


Fig. 5 Class diagram for malicious VM migration process

Fig. 6 Sequence diagram for the use case "man-in-the-middle attack during VM migration process"

- A transferred VM may contain malicious code that can infect other VMs that are under the control of the target VMM.

Possible sources of failure include the following:

- When the attacker eavesdrops on the communication channel, he may not get all the necessary data.
- Some defenses described in the next section can stop this attack.

7.2.6 Countermeasures

Insecure VM Migration can be stopped by the following countermeasures:

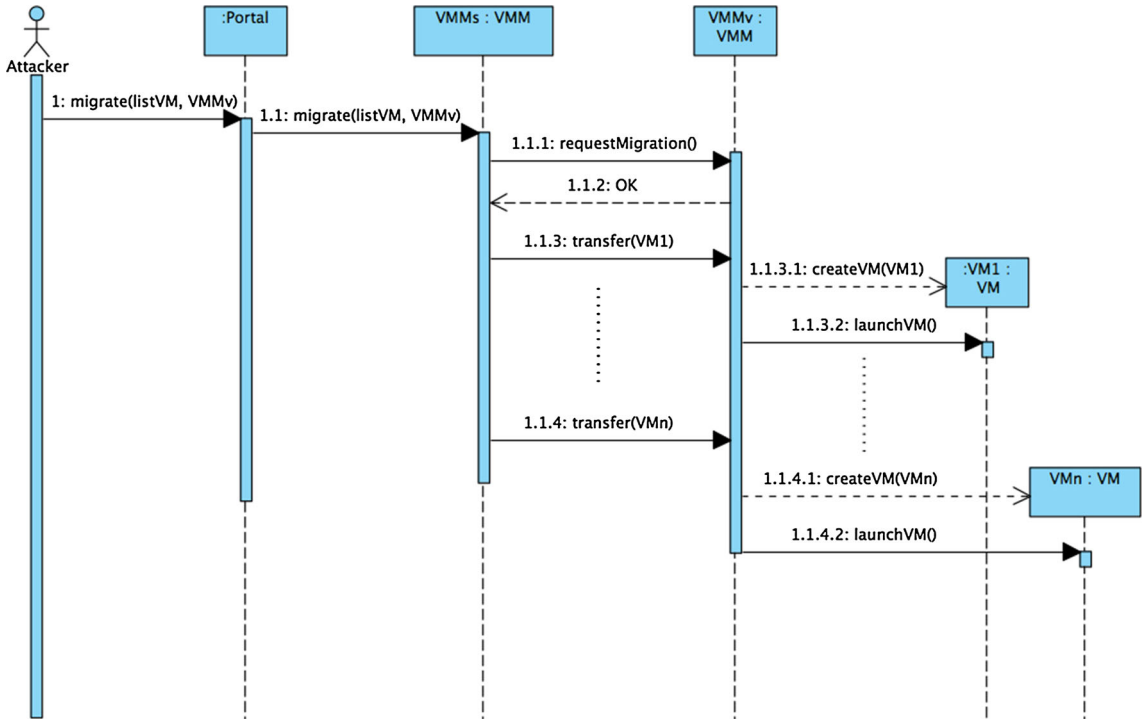


Fig. 7 Sequence diagram for the use case "migrate many VMs to a victim VMM

- Santos et al. [77] proposes a Trusted Cloud Computing Platform (TCCP) that provides confidential execution of guest virtual machines. It provides secure VM launch and migration operations.
- Zhang et al. [78] proposes a secure migration system that provides VM live migration capabilities under the condition that a VMM-protected system is present and active.
- The connection between the source and the destination VMMs should be authenticated and encrypted during the migration process.
- Isolate VM migration traffic to prevent eavesdropping attacks.
- Danev et al. [79] proposes a virtualization of TPM (Trusted Platform Module) that secures the VM migration process by protecting private information and detecting malicious software.
- Protect the VMIs such that a compromised node cannot produce poisoned VMs. This defense is described by the security pattern Secure VMI Repository (See Sect. 9.1).

7.2.7 *Forensics*

Where can we find evidence of this attack?

- The provider can keep logs of the VMs that are transferred from one machine to another. Also, it can store information about the source and destination VMMs. Finally, we can also add detection of malicious software and failure to check correctly information integrity.

7.2.8 *Related patterns*

- Secure VMI Repository (See Sect. 9.1)

8 Secure reference architecture

As indicated earlier, the identified threats can be neutralized by applying appropriate security patterns. As an example, Table 2 in Sect. 6.2 shows how each threat in Table 1 can be controlled by a corresponding security pattern. Once security patterns are selected, we apply them into the RA in order to stop or mitigate threats. Security mechanisms are added to the basic RA, including Authenticator, Authorizer, Security Logger/Auditor and others that mitigate specific threats. To avoid impostors, we can use the Authenticator so that every action with the cloud is authenticated. The Security Logger/Auditor is used to log all activities that can be used for auditing at a later time. For authorization, we use role-based access control (RBAC), or a similar model, so only authorized users can perform some actions to assets. To avoid storing infected VMI, they are scanned and filtered before storing them in the VMI Repository.

Figure 8 shows the class diagram of the resulting secure IaaS architecture pattern, which is the most important level of the SRA. In this model, the package Authenticator is an instance of the Authenticator pattern [13] and enables the Cloud Controller to authenticate Cloud Consumers/Administrators. Instances of the Security Logger/Auditor pattern are used to keep track of any access to cloud resources such as VMs, VMMs, and VMIs. The Reference Monitor enforces authorization rights defined by the RBAC instances [13]. The Filter scans created virtual machines in order to remove malicious code. At this moment, we have secured the VMI administration which is part of the cloud. We continue in the same way to secure the rest. For example, the Authorizer controls access to the Cloud, Cluster, and Node Controllers; the Security Logger/Auditor logs the security-related activities of the VMM, and so on.

At the SaaS level the responsibility for security is in the hands of the corresponding SP; that is, if I run a travel agency in this level, I need to provide for my clients: authentication, authorization, encryption, etc. These security services must be supported at the IaaS level, including security administration (see Sect. 9.2). It is most convenient and more unified to base these services on the IaaS services, and we will not show here a model for PaaS security functions; for example, a SP in the SaaS level can map its authorization model into the IaaS authorization model. Another possibility is to use a specialized RA for this level [10], which can then be mapped to the IaaS level to avoid unnecessary redundancy. The same situation

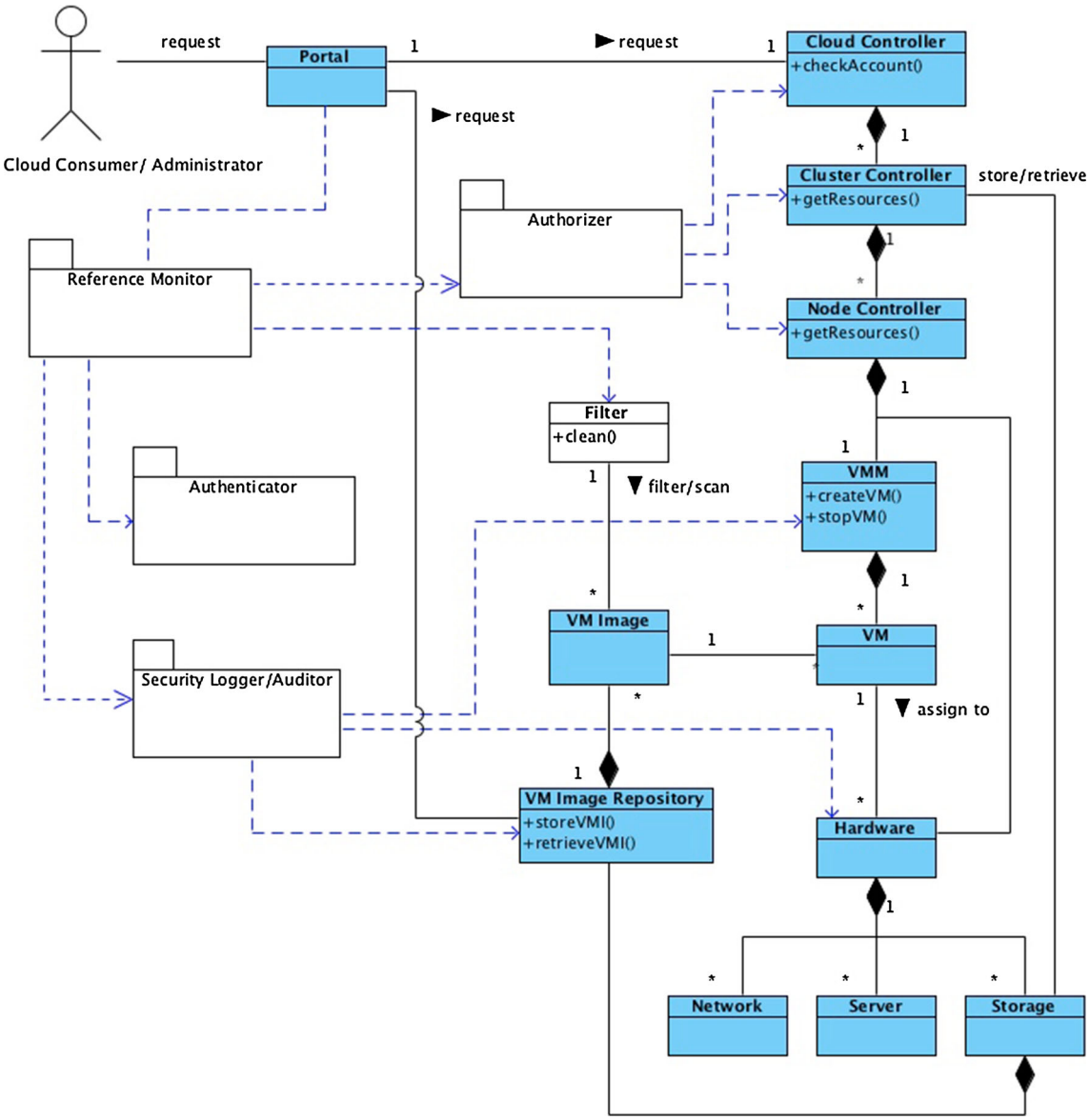


Fig. 8 Class diagram of the secure IaaS pattern

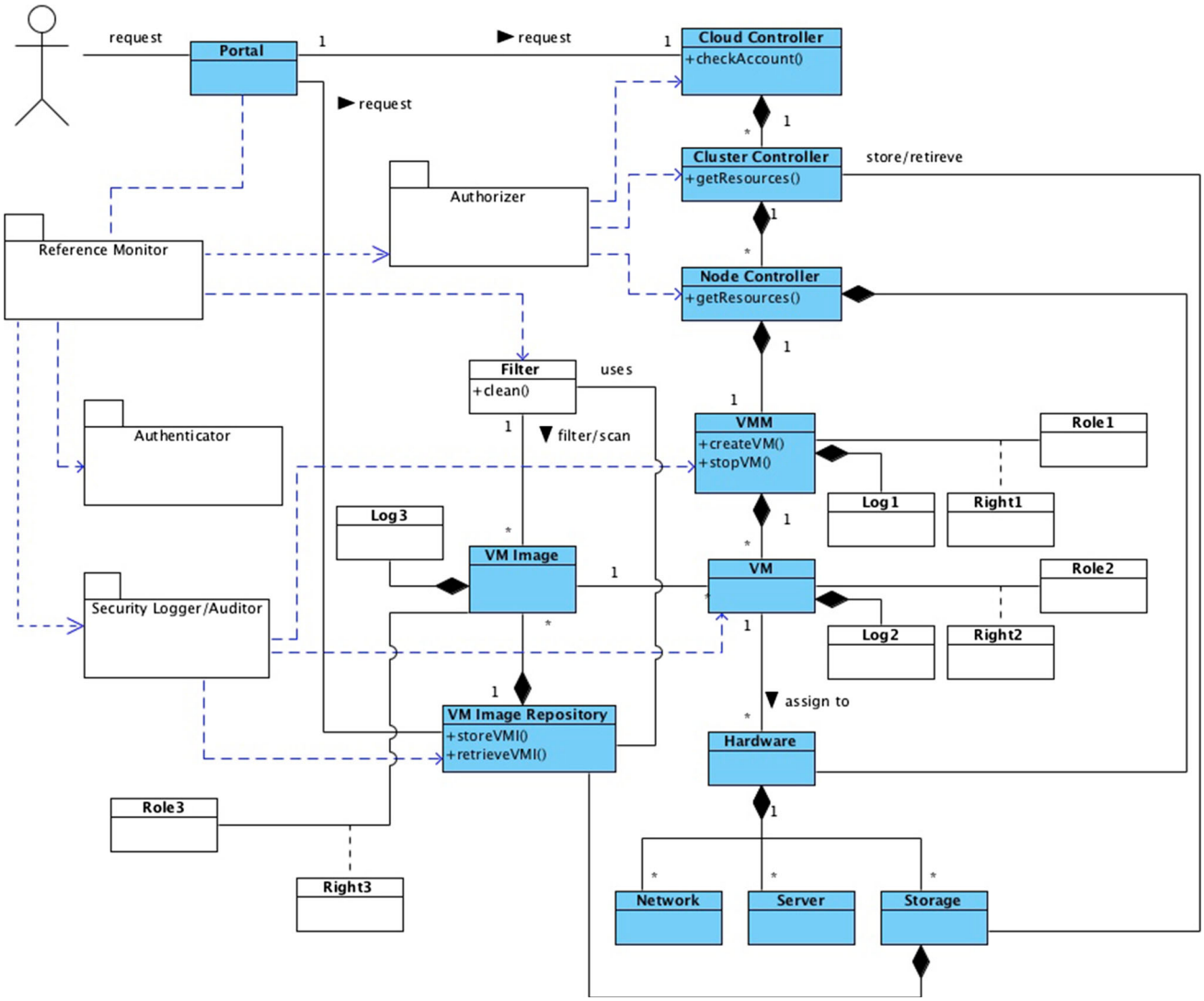


Fig. 9 Complete secure IaaS pattern

occurs at the PaaS level, where the corresponding SP must provide control of the components at this level. Again, these functions should be based on those of the IaaS level. Putting together the models for the three levels, described as patterns, we would have a complete SRA, where its units can be refined as needed. We do not show the models for the PaaS and SaaS levels for simplicity, their unsecured models can be found in [11].

The rest of the architecture is built similarly. Figure 9 shows the complete secure IaaS architecture pattern. In this model, the subsystem *Authenticator* is an instance of the Authenticator pattern and allows the *Cloud Controller* to authenticate *Cloud Consumers/Administrators*, or other components of the architecture that could be important. Log indicates instances of the logging pattern, and it is used to keep track of any access to a cloud resource such as *VM*, *VMM*, and *VM image repository*. The *Reference Monitor* reinforces authorization rights defined by the RBAC instances. The *Filter* scans all virtual machines in order to remove malicious code.

9 Security patterns applied to cloud security administration

After analyzing threats, we considered possible security patterns to handle them. We show now how we can build secure units of the SRA by adding security patterns. Specifically, we show the development of a secure administration unit, starting from a pattern for a Secure VMI Repository. We have produced a security pattern to describe the functions of this model [80] and we partially showed it here as illustration.[7] Note that this pattern and all

[7] It lacks sections Example, Implementation, Known Uses, Consequences, and Related Patterns.

patterns are not plug-ins, but are used by instantiation and where we can tailor them according to the application needs. We describe both security and misuse patterns using the pattern template of [81].

9.1 Secure virtual machine image repository (security pattern)

9.1.1 Intent

Avoid the poisoning of VM images (VMIs) during their creation and the leaking of sensitive information accidentally left in the VMI by enforcing access control and filtering in the image repository.

9.1.2 Context

Cloud computing providers publish VMIs in order to let consumers (clients) instantiate VMs. In some cloud systems consumers are also allowed to store VMIs for public use.

9.1.3 Problem

VMIs are necessary for creating VMs, but an attacker may place in the VMI repository images with malware that could infect virtual machines that are created using the poisoned images.

The following forces will affect the solution:

- *Clean Images* A VMI could contain malware and we need to provide the consumers with clean images before they use them. An infected VM could misuse customer data or attack other VMs.
- *Data leakage prevention* Users may accidentally leave sensitive data in the VMI and we need to prevent that leakage.
- *Repository access* Having an open repository is sometimes convenient, but it may allow malicious actions. We need some form of access control for images.
- *Overhead* The security controls should not significantly affect the performance of the system or the users will be hindered in their work.
- *Records* The use of a VMI is important for security, billing, and statistics. We should record this activity.

9.1.4 Solution

Provide a mechanism to control access to VMIs in order to prevent attackers from placing or producing poisoned images. Before placing a new image or using an existing image clean it by scanning and filtering. Keep a log of the repository use.

Structure Figure 10 shows a class model for the secure VM images repository system. The *virtual machine image repository* holds a set of VMIs that can be used to instantiate virtual machines. The *Reference Monitor* uses a *Filter* that scans all VM images before being published or retrieved. The *Authenticator* is an instance of the Authenticator Pattern that allows the Reference Monitor to authenticate the users that access the repository, who can publish or retrieve images if the Authorizer authorizes them. The Reference Monitor pattern enforces the authorization rights defined in the *Authorizer*. The *Security Logger/Auditor* keeps track of accesses to the repository.

9.2 Security administration model

Figure 10 includes the classes of the Secure VMI Repository pattern as an extension to two units of the cloud RA that implement some use cases of the cloud. In this way we can secure all units of the RA.

Some more security patterns for this purpose would be the following:

- *Secure migration process* it provides protection for live and offline migration.
- *Secure hypervisor* reinforces the security of the hypervisor to avoid some attacks.
- *Secure virtual network* it secures the communication among virtual machines.
- *Virtualized Trusted Platform* it provides a framework to determine whether the environment is secure before launching a virtual machine.
- *Web application scanner/filter* it scans web applications in order to identify security vulnerabilities and sensitive data.
- *Cloud data protection* it protects sensitive data while it is processed, stored or transferred (encryption, digital signature, fragmentation-redundancy-scattering, homomorphic encryption).
- *Secure DNS* where Access Control Lists (ACLs) are used to protect the DNS [7].
- *Security Group Firewall* it divides the firewall in customer groups that have similar filtering requirements [82].
- *Cloud-based Web Application Firewall (CWAF)*it controls access to web applications communicating through HTTP according to authorization rules with the objective of stopping XSS, SQL injection, and similar attacks [82].

10 Validation of the SRA

RAs are abstract models and cannot be evaluated with respect to security or performance through experimentation

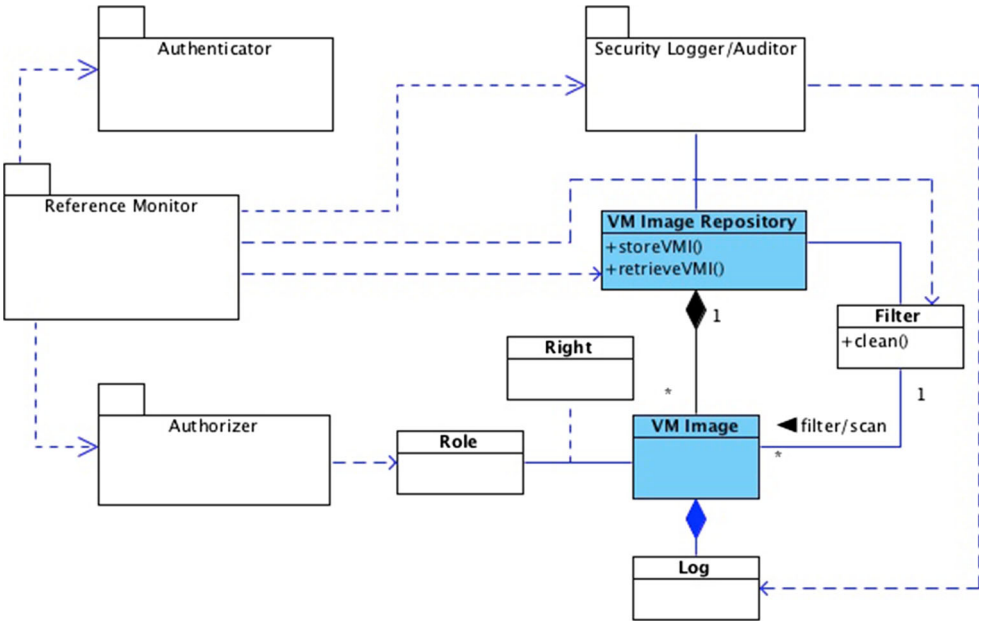


Fig. 10 Secure VMI repository system

or testing. An RA is similar to a pattern, and it has a similar use; it is a paradigm to guide implementation of new systems or evaluation of existing systems as well as other uses described in Sect. 10.1. Their evaluation must be based on how well they represent the relevant concepts of the systems they describe, how well they handle potential threats, how complete they are, how precise they are, how they can be applied to the design or evaluation of systems, and how useful they are for other relevant functions. Their final validation comes from practitioners who can find them useful and convenient to build concrete architectures.

In particular, we indicated that our models are "more precise" than those in the literature. All of these models are given in words accompanied by block diagrams. A UML representation in general and patterns in particular are more precise in the sense that UML syntax is well defined, stereotypes and profiles can add some semantics, OCL can add formalization, and patterns have well-defined templates, with specific information. Our approach is clearly more precise.

To show its completeness, a SRA should be able to represent all the security features in commercial SRAs. We have compared our SRA with the one proposed by NIST, and we have applied it to check if the industrial SRAs described earlier included features that we do not have. We found that our model could represent all the security features shown in the published models. In fact, none of the industrial models had all the security aspects of our SRA. Furthermore, their SRAs included implementation aspects and all of them were shown in block diagrams with undefined semantics. In other words, we can say that our SRA is more complete and precise than all the published models.

Another type of validation refers to the usefulness of the SRA, which is shown in the next section by enumerating some of the practical uses of a model like ours. One of those uses is developed in detail in Sect. 10.2.

10.1 Uses of cloud SRAs

Building a complete SRA that includes all use cases, components, and roles is lengthy and expensive. To justify this work, we need to find a good number of uses of value for the development or use of cloud systems. We show below such a list. This list can be considered a partial validation of our model, and it could be a way to validate any SRA, not just ours. We show one of these uses in detail, but most of them are future work and are part of our contribution. Section 10.2 shows details of one of them.

This list indicates possible advantages of having a SRA; however, there are also some threats to the practical value of the SRA: To get from a SRA to a practical concrete security architecture takes a good amount of work, time, and expertise. The applications of SRAs in the list need to be developed in detail to be proven useful; we have done this for a few of them, but for the rest it may turn out that they are not really useful or convenient to use. There is no absolute measure of completeness, so it is difficult to know if we left out some important aspect; for example, use cases are open-ended and threat enumeration may not include all possible threats.

A way to understand and decompose a complex system such as a secure cloud As such SRAs are good tools for teaching cloud security concepts and train personnel. They are more effective than textual descriptions or imprecise block diagrams.

A guide to orchestrate a secure cloud ecosystem The complete cloud environment needs the coordination of brokers, carriers, users, and several other stakeholders. The SRA is the reference point to align all these functions [8].

Refinements or remodeling of cloud systems We can describe more specific units or reconfigure them. MyCloud [83] describes a reconfigured VMM to improve its security; they did not use a SRA, but it is clear that it would have helped.

A holistic security view Several authors, e.g., [19, 20], emphasize the need to develop secure systems in a holistic way. Systems built piecemeal omit important interactions that may result in vulnerabilities. A SRA provides such a holistic view by indicating the places where security mechanisms can be attached and their effect in the functional parts of the architecture. As such a SRA can be useful for creating secure cloud development methodologies [63]. We can expand Fig. 9 by indicating all the points where threats are neutralized with corresponding security patterns. Holistic views are very important to combine quality factors such as safety or reliability with security [84].

A way to unify cloud terminology Different vendors have different ways to describe their security services and products. A SRA can be used as a framework to unify terms and descriptions. This is useful for selecting cloud providers. Some ontologies for this purpose exist, e.g., [44, 45], but they don't relate the terms to a RA, much less to a SRA.

A basis for concrete SRAs SRAs oriented to specific technologies or products can be derived from our abstract SRA. A new vendor of clouds systems can use a SRA to define its product. A SRA built using patterns can lead to specialized versions by changing adding, removing, or changing some patterns.

Evaluation of the security of a cloud As shown in Sect. 10.2 we can evaluate the security of a cloud we are building. If we are considering renting some SP services, we can use a SRA to evaluate its security by verifying that the SP includes the corresponding mechanisms.

Selection of cloud providers based on security requirements Mouratidis et al. [67] discusses an approach to select secure clouds, the use of a SRA would make their selection easier.

Compliance with standards and regulations An RA can be used to support security standards and regulations, which can be described as policies which in turn can be implemented as patterns and made part of the SRA. It helps architects or designers to identify what components of the cloud system are associated with the standard and can be used to comply with the specific rules of the standard. Applications derived from the SRA will automatically comply with the standards or regulations. Relating specific regulations to specific security mechanisms can be used to demonstrate compliance. Cloud standards, such as OVF [85], are easier to apply in a SRA.

Security Service Level Agreement (SSLA) An RA can provide a framework for defining the requirement of the provider with respect to the requirements of the consumer; the SRA can define the security mechanisms that the SP has or could have and the customer can then select them for the corresponding SSLA. In particular, a SSLA can include several levels and the SRA makes clear where the services belong. SSLAs require monitoring to assure that the SP fulfilled its contact; the SRA is useful to define where monitoring for this purpose is needed. We can build a SSLA which must be matched by the provider security mechanisms. The consumer can use the SRA to clarify her needs about security and to negotiate with the SP the quality of service required.

Reference for monitoring functions Monitoring requires mechanisms to obtain information about the system status. A SRA provides guidelines about the places where security events should be collected in order to fulfill SLAs requirements, for system administration, and for compliance. It may provide a guide for distribution of monitoring functions to make them more resistant to attacks [86]. There is already a commercial monitoring system which claims to be model driven [87].

Service certification Critical applications require the use of certified services. Even in non-critical applications, certification increases the trust of the consumer; for example, AWS and Microsoft claim ISO certification. Certification approaches may use ontologies or other formal models to describe certificates as well as monitoring requirements. A SRA can be used to guide the certification process. A cloud provider can show that his services can handle the corresponding threats which can increase customer trust.

Cloud setup The SRA provides a deployment template for IT teams to assist them in setting up a secure cloud [60]. The SRA provides the scope of resources they need and the recommended deployment model for specific use cases. It also provides design choices for the IaaS solution, including the physical resources required and a deployment topology.

Security administration The SRA can be used as a guideline to define the functions that are needed for administration. For example, when using RBAC, roles and rights can be defined with respect to the components of the SRA (See Sect. 9 for a discussion).

Reference for analyzing attacks to administrative, security, and monitoring services These functions are common targets because of their importance and require careful analysis of threats [86]. The SRA, by showing explicitly those services allows enumerating their use cases

and the threats against each activity in them, as we did in Sect. 6.

Forensics Because of distribution and virtualization, forensics is particularly hard in clouds. A SRA provides a framework where we can define what specific evidence can be collected after an attack and where we can collect it. We can also use the SRA to define specific points in the architecture where we can add special mechanisms to collect forensics.

Analysis of trust Trust can be evaluated using chains of trust where a component trusts another which in turn trusts another, and so on. Chains of trust can be used to define SLAs [77, 88]. The SRA provides a structure to assemble chains of trust.

Cloud Broker Cloud brokers are considered part of the SRA in some models [8]. In this case, we need to model them in the same way and include their security functions. Brokers are needed when consumers want access to multiple clouds or to specialized services not available in any SP. They use patterns such as Adapters and Enterprise Service Bus. We have written a secure version of the ESB that can be used in their architecture or in a lower view of a cloud [13].

Integration of a variety of devices The new trend toward Bring Your Own Device (BYOD) requires including mobile devices, sensors, and embedded systems as part of the cloud architecture. We can define where each device interacts with the cloud and the security controls they need for this interaction.

SDN Software-Defined Networking (SDN) to reconfigure services SDN lets applications manipulate the control software (which is separated from the data) of the network resources and devices. This approach is very suitable for clouds, since they have a dynamic environment where resources, users, and applications change along time. SDN can reconfigure networks in the presence of attacks [39]; for example, SDN switches can detect suspicious activity and react to it. SDN fits well with distributed policy enforcement [89] and can also be used to monitor SLAs and to assemble services from several clouds. New attacks are possible and need to be studied. The SRA provides a framework to define the use of SDN.

Hybrid clouds These are becoming more popular due to security and privacy issues associated with public clouds. This RA also provides general information for organizations wishing to integrate their existing IT processes and system with cloud infrastructure. Before migrating any process or system, organizations should refer to the cloud architecture to plan a strategy for integrating existing resources with clouds, to understand the inherent issues and limitations, and to think in terms of moving some processes and data to the cloud, as well as the security effects of such a migration.

Multicloud federations Federations of clouds are starting to appear. In order to integrate them, each cloud must indicate explicitly which resources it brings to the federation. To describe their structure and their security mechanisms, SRAs are very convenient as baselines of resources and to define user rights [90–92].

Framework for security testing We can relate the activities of the use cases to specific components in the architecture that may contain data assets. We can test if these assets receive their correct values when applying each use case.

Framework for distributing policies in the cloud Enforcement of a variety of policies can be more effective by distributing them to relevant local units [89]. A SRA provides a framework to guide the placement of policies.

Security as a Service There are several proposals for security as a service, e.g., [93]. The SRA shows clearly where we would want to buy or hire services of some type, e.g., cryptographic protection.

Understanding and comparison of research ideas Interesting security proposals such as those of [9, 46], can be understood better by seeing which parts of the SRA would be affected.

10.2 Evaluating the security of a cloud security using a reference architecture

We can use the SRA to evaluate the degree of security reached by a cloud built following the SRA as a guideline. If all the enumerated threats can be controlled, we can consider the new system secure. However, listing threats is not enough; we need a way to understand how an attack happens. We can use misuse patterns to test the security of the reference architecture. As discussed in Sect. 7, we have developed some misuse patterns including Malicious Virtual Machine Creation [14]. This misuse pattern describes how an attacker may create a virtual machine image which can contain malicious code so it can infect other users when they create their virtual machine. Figure 11 describes how an attacker can publish a virtual machine image that contains malicious code.

We can use a sequence diagram to show how the SRA can stop some threats described in the misuse pattern of Fig. 11. In this example, we can see if the attacker is not a valid user; the attack cannot go any further if we add an authentication pattern in its path. If the attacker is a valid user (or stole a valid credential), his attempt to publish an image will be intercepted by the reference monitor which will check whether he is authorized or not. Even if the attacker is authorized since he is a valid user and he has rights to publish an image, his image will be filtered to scan for malicious code or sensitive data. If the image contains

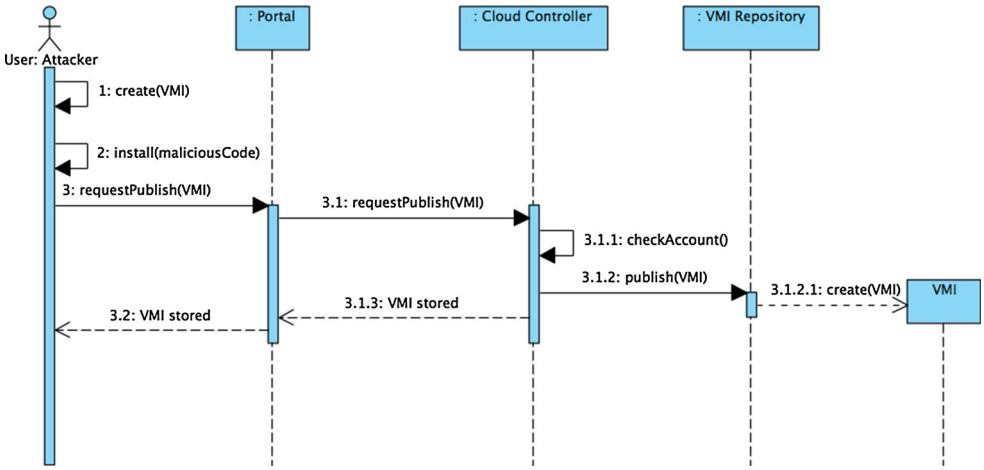


Fig. 11 Sequence diagram for the use case "Publish a Malicious VM Image"

malicious code, it will be removed before being stored. These actions can stop the misuse by providing a defense-in-depth barrier. Figure 12 shows the steps through the security-enhanced use case.

We evaluate the security level of the of the reference architecture by verifying that all misuse cases have been controlled by some security pattern. If we enumerate all the threats; for example, using the method of Sect. 6, we just need to verify that the architecture includes a security pattern that can neutralize all the threats. If $T = \{t_1 \ldots t_i \ldots\}$ is the set of threats, $SP = \{sp_1 \ldots sp_j \ldots\}$ is the set of security patterns, we have: $\forall t_i \in T \Rightarrow \exists sp_j \in SP$, where sp_j controls t_i. When the SRA is instantiated to define a specific type of cloud, each misuse pattern can be realized following the specific architecture components, which means that we may need further security patterns to stop them. If $MP = \{mp_1 \ldots mp_k \ldots\}$, where ap_i is an attack pattern[8] used by the MP, if $\exists sp_j \in SP : sp_j$ stops ap_i, the misuse case cannot succeed.

10.3 Applying the SRA to describe existing SRAs

As shown in Fig. 9, we can continue building the security aspects of the SRA by adding security controls (authorization) for access by administrators to the Hardware, Cloud Controller, the Cluster Controller, the Node Controller, and the VMM. Figure 9 also shows logging the VMM and the access to the VMI Repository. The way to deduce the need for these controllers would be based on a similar analysis to the one performed to the VMI Repository.

[8] An attack (threat) pattern describes a specific step leading to a misuse [94]; e.g., using a stolen credential to have access to a DBMS where we can perform a misuse by using SQL injection.

11 Conclusions and future work

We have shown an approach to build a SRA for clouds. We started by following a methodology similar to that in [95] to match the application requirements to the cloud security. In this case, the "application" includes the functional and administrative operations required to provide secure service to users. In order to develop a secure framework, we first identified threats by analyzing the activities of its use cases [22]. Identifying cloud threats is not enough; we need a way to describe how an attack is performed and what cloud units are compromised. We have developed misuse patterns that describe from the viewpoint of the attacker how an attack (misuse of information) is performed. We have started building a catalog of cloud misuse patterns [14], which can be used to verify if security patterns have been placed in the architecture to stop misuses of information. For this work, we developed new security and misuse patterns in order to demonstrate how they mitigate or stop identified threats and how a misuse happens.

Cloud computing systems are complex systems that leverage different technologies and can be deployed in different ways, as well as provide different types of services. All this implies that it can be a challenge to understand how to make a cloud secure. In this work, we have provided the following contributions:

- We showed how to secure an RA by applying a systematic methodology where we started from a cloud RA until we obtained a SRA which includes defenses against all identified threats.
- We introduced a simple metamodel to relate appropriate concepts.
- We have compared our SRA to several published proposals, and we found ours to be either more comprehensive, more formal, or more systematic than all of them.

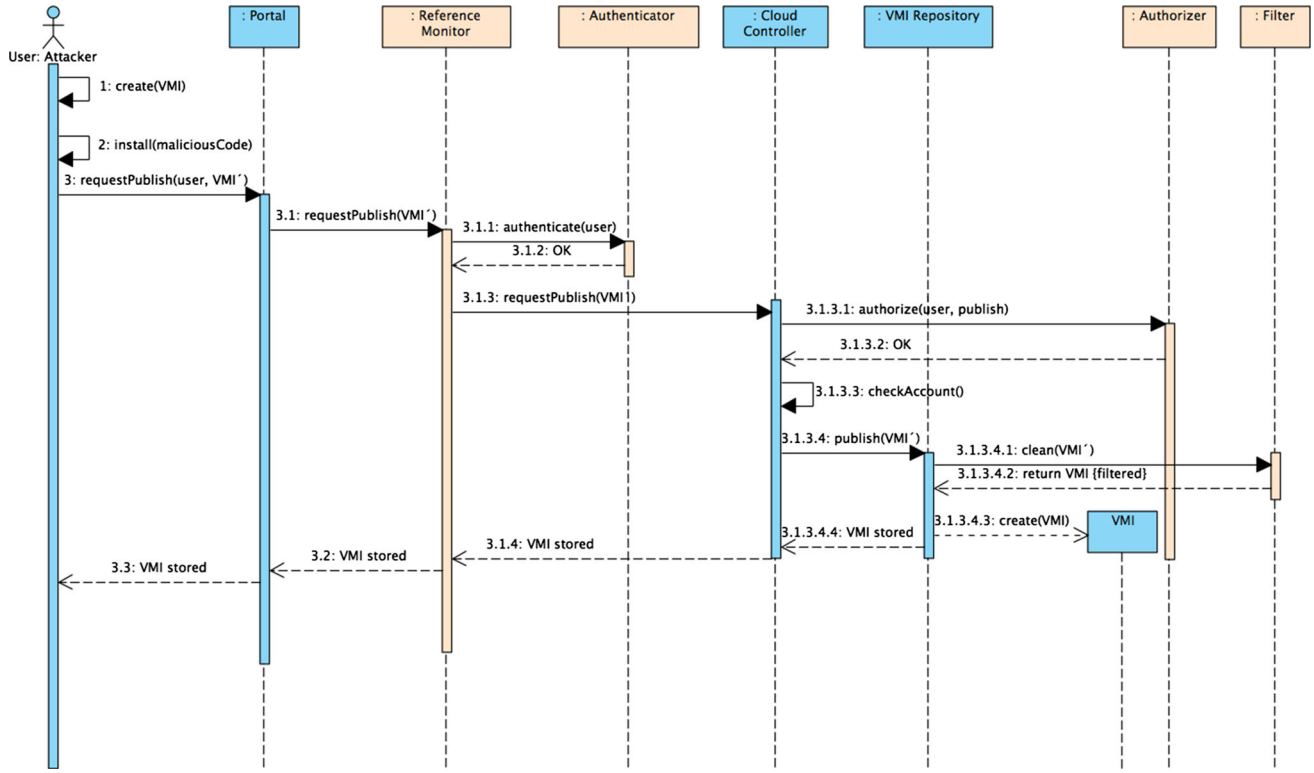


Fig. 12 Sequence diagram for the use case "Securely Publish VM Images"

- We developed an approach to evaluate the degree of security obtained in the SRA.
- We also provided a comprehensive list of situations where SRAs can be useful, which justifies the effort in their development.

Possible limitations of our architecture, which can be improved in future work, include the following:

- As indicated, UML has limitations to describe architectures and some useful and maybe important aspects may not be properly described. It may be worthwhile to recast the architecture using an ADL.
- It is not clear what level of detail is needed to make a SRA truly useful for practical purposes. Other than the main structural aspects of the architecture, we included a portion of the administrative subsystem. For example, we could similarly add details of the protection needed for the hypervisor and for data storage. As in all requirements, too little detail will require more work for building concrete architectures, but too much detail will restrict the freedom of the designer of the concrete architecture. In any case, we intend to develop in detail a few of these subsystems.

We have produced a good number of security patterns [13], but we still need to adjust them to be valid for cloud environments and to develop new security patterns that are specific for clouds. From [2], we identified some ideas for security and misuse patterns, which will provide a good amount of future work. Developing good catalogs for security and misuse patterns is very important to help designers, and architects use the SRA in many ways. The catalog of uses of Sect. 10.1 provides a source of future work to demonstrate the value of a SRA to fulfill those functions by developing these applications in detail. Producing a concrete architecture using XML web services and an Enterprise Service Bus [66] is a good demonstration of the value of our SRA.

Acknowledgements We thank the reviewers for their careful evaluation and their suggestions that significantly improved the paper. The work of Eduardo Fernandez was supported by the Chilean agency CONICYT, under research contract 80120008.

References

1. Clarke R (2013) Data risks in the cloud. J Theor Appl Electron Commer Res 8(3):59–73. doi:10.4067/S0718-18762013000300005, ISSN 0718-1876
2. Hashizume K, Rosado DG, Fernández-Medina E, Fernández EB (2013) An analysis of security issues for cloud computing. J Internet Serv Appl 4(1). doi:10.1186/1869-0238-4-5
3. Avgeriou P (2003) Describing, instantiating and evaluating a reference architecture: a case study. Enterp Archit J
4. Taylor RN, Medvidovic N, Dashofy EM (2009) Software architecture: foundations, theory, and practice. Wiley, London. ISBN 0470167742, 9780470167748

5. HP (2011) Understanding the HP CloudSystem Reference Architecture. White paper, Hewlett-Packard Development Company
6. IBM (2012) IBM SmartCloud. White paper, IBM Corporation
7. Microsoft Global Foundation Services (2009) Securing Microsoft's cloud infrastructure. Technical report, Microsoft
8. NIST Cloud Computing Security Working Group (2013) NIST cloud computing security reference architecture. Working document, NIST
9. Campbell RH, Montanari M, Farivar R (2012) A middleware for assured clouds. J Internet Serv Appl 3(1):87–94. doi:10.1007/s13174-011-0044-9
10. Hafner M, Memon M, Breu R (2009) SeAAS—a reference architecture for security services in SOA. J UCS 15(15): 2916–2936
11. Hashizume K, Fernandez EB, Larrondo-Petrie MM (2012) Cloud service model patterns. In: 19th international conference on pattern languages of programs (PLoP2012), Tucson, AZ
12. Hashizume K, Fernandez EB, Larrondo-Petrie M (2012) Cloud infrastructure pattern. In: First international symposium on software architecture and patterns. LACCEI, Panama City, Panama, pp 23–27
13. Fernandez EB (2013) Security patterns in practice: designing secure architectures using software patterns, 1st edn. Wiley, London. ISBN 1119998948
14. Hashizume K, Yoshioka N, Fernandez EB (2013) Three misuse patterns for cloud computing. In: Rosado DG, Mellado D, Fernandez-Medina E, Piattini MG (eds) Security engineering for cloud computing: approaches and tools. IGI Global, Hershey, pp 36–53. doi:10.4018/978-1-4666-2125-1.ch003
15. Angelov S, Grefen P, Greefhorst D (2012) A framework for analysis and design of software reference architectures. Inf Softw Technol 54(4):417–431. doi:10.1016/j.infsof.2011.11.009, ISSN 0950-5849
16. CSA (2011) Quick guide to the reference architecture TCI (trusted cloud initiative). Technical report, Cloud Security Alliance
17. Warmer J, Kleppe A (2003) The object constraint language: getting your models ready for MDA, 2nd edn. Addison-Wesley Longman, Boston. ISBN 0321179366
18. Garavel H, Graf S (2013) Formal methods for safe and secure computer systems. Technical report. BSI Study 875, Federal Office for Information Security, Bonn
19. Brown A, Apple B, Michael JB, Schumann MA (2012) Atomic-level security for web applications in a cloud environment. IEEE Comput 45(12):80–83. doi:10.1109/MC.2012.400
20. Fernández EB, Washizaki H, Yoshioka N, VanHilst M (2011) An approach to model-based development of secure and reliable systems. In: Sixth international conference on availability, reliability and security, ARES, pp 260–265, Vienna. doi:10.1109/ARES.2011.45
21. Delessy N, Fernandez EB, Larrondo-Petrie MM (2007) A pattern language for identity management. In: Proceedings of the international multi-conference on computing in the global information technology, ICCGI '07, p 31, IEEE Computer Society, Washington, DC. doi:10.1109/ICCGI.2007.5, ISBN 0-7695-2798-1
22. Braz FA, Fernández EB, VanHilst M (2008) Eliciting security requirements through misuse activities. In: 19th international workshop on database and expert systems applications (DEXA 2008), 1–5 Sept 2008, Turin, pp 328–333. doi:10.1109/DEXA.2008.101
23. Fernandez EB, Yoshioka N, Washizaki H, Yoder J (2014) Abstract security patterns for requirements specification and analysis of secure systems. In: WER 2014 conference, a track of the 17th Ibero-American conference on software engineering (CIbSE 2014), Pucon, Chile
24. Fernandez E, Yuan X (2000) Semantic analysis patterns. In: Laender A, Liddle S, Storey V (eds) Conceptual modeling—ER 2000, vol 1920 of lecture notes in computer science. Springer, Berlin, pp 183–195. doi:10.1007/3-540-45393-8_14, ISBN 978-3-540-41072-0
25. Fernandez E, Pelaez J, Larrondo-Petrie M (2007) Attack patterns: a new forensic and design tool. In: Craiger P, Shenoi S (eds) Advances in digital forensics III, vol 242 of IFIP—The International Federation for Information Processing. Springer, New York, pp 345–357. doi:10.1007/978-0-387-73742-3_24, ISBN 978-0-387-73741-6
26. Fernández EB, Yoshioka N, Washizaki H (2009) Modeling misuse patterns. In: Proceedings of the fourth international conference on availability, reliability and security, ARES 2009, 16–19 March, 2009, Fukuoka, pp 566–571. doi:10.1109/ARES.2009.139
27. Fowler M (2002) Patterns of enterprise application architecture. Addison-Wesley Longman, Boston. ISBN 0321127420
28. Liu F, Tong J, Mao J, Bohn R, Messina J, Badger L, Leaf D (2011) Cloud computing reference architecture. Special publication 500-292, NIST
29. Stricker V, Lauenroth K, Corte P, Gittler F, Panfilis SD, Pohl K (2010) Creating a reference architecture for service-based systems—a pattern-based approach. In: Towards the future internet—emerging trends from European research, pp 149–160. doi:10.3233/978-1-60750-539-6-149
30. Muller G, van de Laar P (2009) Researching reference architectures and their relationships with frameworks, methods, techniques, and tools. In: Kalawsky R, O'Brien J, Goonetilleke T, Grocott C (eds) 7th annual conference on systems engineering research (CSER 2009). Research School of Systems Engineering, Loughborough University, Loughborough
31. Uzunov AV, Fernandez EB, Falkner K (2012) Securing distributed systems using patterns: a survey. Comput Secur 31(5):681–703. doi:10.1016/j.cose.2012.04.005, ISSN 0167-4048
32. Object Management Group (2014) Unified Modeling Language™ (UML®) Tech. rep., Object Management Group Inc
33. Medvidovic N, Taylor R (2000) A classification and comparison framework for software architecture description languages. IEEE Trans Softw Eng 26(1):70–93. doi:10.1109/32.825767, ISSN 0098-5589
34. OWASP (2013) OWASP Top 10—2013: the ten most critical web application security risks. Technical report, The OWASP Foundation
35. Chonka A, Xiang Y, Zhou W, Bonti A (2011) Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks. J Netw Comput Appl 34(4):1097–1107. doi:10.1016/j.jnca.2010.06.004, ISSN 1084-8045
36. Fernandes D, Soares L, Gomes J, Freire M, Inácio P (2014) Security issues in cloud environments: a survey. IntJ Inf Secur 13(2):113–170. doi:10.1007/s10207-013-0208-7, ISSN 1615-5262
37. Ryan MD (2013) Cloud computing security: the scientific challenge, and a survey of solutions. J Syst Softw 86(9):2263–2268. doi:10.1016/j.jss.2012.12.025, ISSN 0164-1212
38. Kalloniatis C, Mouratidis H, Vassilis M, Islam S, Gritzalis S, Kavakli E (2014) Towards the design of secure and privacy-oriented information systems in the cloud: identifying the major concepts. Comput Stand Interfaces 36(4):75–759. doi:10.1016/j.csi.2013.12.010, ISSN 0920-5489
39. Tsugawa M, Matsunaga A, Fortes JA (2014) Cloud computing security: what changes with software-defined networking? In: Jajodia S, Kant K, Samarati P, Singhal A, Swarup V, Wang C (eds) Secure cloud computing. Springer, New York, pp 77–93. doi:10.1007/978-1-4614-9278-8_4, ISBN 978-1-4614-9277-1

40. Prolexic (2012) DDoS Denial of service protection and the cloud. White paper Prolexic Technologies Inc
41. Modi C, Patel D, Borisaniya B, Patel H, Patel A, Rajarajan M (2013) A survey of intrusion detection techniques in Cloud. J Netw Comput Appl 36(1):42–57. doi:10.1016/j.jnca.2012.05.003, ISSN 1084-8045
42. Juels A, Oprea A (2013) New approaches to security and availability for cloud data. Commun ACM 56(2):64–73. doi:10.1145/2408776.2408793, ISSN 0001-0782
43. EMA (2010) Securing the administration of virtualization. Market research report, Enterprise Management Associates
44. Moscato F, Aversa R, Di Martino B, Fortis T, Munteanu V (2011) An analysis of mOSAIC ontology for Cloud resources annotation. In: 2011 federated conference on computer science and information systems (FedCSIS), pp 973–980
45. Zhang M, Ranjan r, Haller A, Georgakopoulos D, Menzel M, Nepal S (2012) An ontology-based system for cloud infrastructure services' discovery. In: 2012 8th international conference on collaborative computing: networking, applications and worksharing (CollaborateCom), pp 524–530
46. Lombardi F, Pietro RD (2011) Secure virtualization for cloud computing. J Netw Comput Appl 34(4):1113–1122. doi:10.1016/j.jnca.2010.06.008, ISSN 1084-8045
47. Malik S, Khan S, Srinivasan S (2013) Modeling and analysis of state-of-the-art VM-based cloud management platforms. IEEE Trans Cloud Comput 1(1):1–1. doi:10.1109/TCC.2013.3, ISSN 2168-7161
48. Kalantari A, Esmaeli A, Ibrahim S (2012) A service-oriented security reference architecture. Int J Adv Comput Sci Inf Technol (IJACSIT) 1(1):25–31
49. Dodani M (2010) On 'cloud nine' through architecture. J Object Technol 9(3):31–39. doi:10.5381/jot.2010.9.3.c3, ISSN 1660-1769
50. IBM (2013) IBM cloud computing reference architecture 3.0—security. Technical report, IBM Developer Works, IBM Corporation
51. OAuth (2014) The OAuth 2.0 authorization framework. Web page, OAuth
52. Okuhara M, Shiozaki T, Suzuki T (2010) Security architectures for cloud computing. Fujitsu Sci Tech J (FSTJ) 46(4):397–402
53. Amazon Web Services (2014) Amazon Web Services: overview of security processes. Technical report, Amazon.com Inc.
54. Cisco HyTrust, VMware, Savvis, Coalfire (2011) PCI-compliant cloud reference architecture. White paper, Payment Card Industry Security Standard Council Data Security Standard
55. VMWare, SAVVIS (2009) Securing the cloud: a review of cloud computing, security implications and best practices. White paper, VMware Inc.
56. Wilkins M (2011) Oracle reference architecture: cloud foundation architecture, release 3.0. Technical report E24529–01, Oracle Corporation
57. Cisco (2009) Cisco SAFE: a security reference Architecture. White paper, Cisco Systems
58. Juniper Networks (2013) Juniper Networks metafabric architecture. White paper, Juniper Networks Inc.
59. Haletky E (2013) Trend Micro deep security reference architecture for the secure hybrid cloud. White paper, Trend Micro
60. E Systems (2014) Eucalyptus reference architectures. Technical report, Eucalyptus Systems
61. OSA (2014) SP-011: Cloud computing pattern. Technical repoer, OSA
62. Beckers K, Côté I, Faßbender S, Heisel M, Hofbauer S (2013) A pattern-based method for establishing a cloud-specific information security management system. Requir Eng 18(4):343–395. doi:10.1007/s00766-013-0174-7, ISSN 0947-3602
63. Uzunov AV, Fernandez EB, Falkner K (2012) Engineering security into distributed systems: a survey of methodologies. J Univers Comput Sci 18(20):2920–3006
64. Badger L, Bohn RB, Chandramouli R, Grance T, Karygiannis T, Patt-Corner R, Voas J (2010) Cloud computing use cases. Working document. NIST
65. Fowler M (1997) Analysis patterns: reusable objects models. Addison-Wesley Longman, Boston. ISBN 0-201-89542-0
66. Papazoglou M, van den Heuvel WJ (2007) Service oriented architectures: approaches, technologies and research issues. VLDB J 16(3):389–415. doi:10.1007/s00778-007-0044-3, ISSN 1066-8888
67. Mouratidis H, Islam S, Kalloniatis C, Gritzalis S (2013) A framework to support selection of cloud providers based on security and privacy requirements. J Syst Softw 86(9):2276–2293. doi:10.1016/j.jss.2013.03.011, ISSN 0164-1212
68. Chappelle D (2013) Security in depth reference architecture, release 3.0. White paper, Oracle Corporation, Redwood Shores
69. Joosen W, Lagaisse B, Truyen E, Handekyn K (2012) Towards application driven security dashboards in future middleware. J Internet Serv Appl 3(1):107–115. doi:10.1007/s13174-011-0047-6, ISSN 1867-4828
70. Gollmann D (2006) Computer security. Wiley, London
71. Harrison NB, Avgeriou P (2010) How do architecture patterns and tactics interact? A model and annotation. J Syst Softw 83(10):1735–1758. doi:10.1016/j.jss.2010.04.067, ISSN 0164-1212
72. Sindre G, Opdahl A (2005) Eliciting security requirements with misuse cases. Requir Eng 10(1):34–44. doi:10.1007/s00766-004-0194-4, ISSN 0947-3602
73. Howard M, Lipner S (2006) The security development lifecycle. Microsoft Press, Redmond. ISBN 0735622140
74. Fernandez EB, Hashizume K, Buckley I, Larrondo-Petrie MM, VanHilst M (2010) Web services security: standards and products. In: Gutierrez C, Fernandez-Medina E, Piattini M (eds) Web services security development and architecture: theoretical and practical issues, information science reference. Imprint of: IGI Publishing, Hershey. ISBN 1605669504, 9781605669502
75. Fernández EB, Ajaj O, Buckley I, Delessy-Gassant N, Hashizume K, Larrondo-Petrie MM (2012) A survey of patterns for web services security and reliability standards. Future Internet 4(2):430–450. doi:10.3390/fi4020430
76. Voorsluys W, Broberg J, Venugopal S, Buyya R (2009) Cost of virtual machine live migration in clouds: a performance evaluation. In: Proceedings of the 1st international conference on cloud computing, CloudCom '09. Springer, Berlin, pp 254–265. doi:10.1007/978-3-642-10665-1_23, ISBN 978-3-642-10664-4
77. Santos N, Gummadi KP, Rodrigues R (2009) Towards trusted cloud computing. In: Proceedings of the 2009 conference on hot topics in cloud computing, HotCloud'09, USENIX Association, Berkeley
78. Zhang F, Huang Y, Wang H, Chen H, Zang B, (2008) PALM: security preserving VM live migration for systems with VMM-enforced protection. In: Trusted infrastructure technologies conference, 2008. APTC '08. Third Asia-Pacific, pp 9–18. doi:10.1109/APTC.2008.15
79. Danev B, Masti RJ, Karame GO, Capkun S (2011) Enabling secure VM-vTPM migration in private clouds. In: Proceedings of the 27th annual computer security applications conference, ACSAC '11. ACM, New York, pp 187–196. doi:10.1145/2076732.2076759, ISBN 978-1-4503-0672-0
80. Fernandez EB, Monge R, Hashizume K, (2013) Two patterns for cloud computing: secure virtual machine image repository and cloud policy management point. In: 20th conference on pattern languages of programs (PLoP 2013), Monticello, IL
81. Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M (1996) Pattern-oriented software architecture: a system of patterns. Wiley, New York. ISBN 0-471-95869-7

82. Fernandez EB, Yoshioka N, Washizaki H (2014) Patterns for cloud firewalls. In: AsianPLoP (pattern languages of programs), Tokyo
83. Li M, Zang W, Bai K, Yu M, Liu P (2013) MyCloud: supporting user-configured privacy protection in cloud computing. In: Proceedings of the 29th annual computer security applications conference, ACSAC '13. ACM, New York, pp 59–68. doi:10.1145/2523649.2523680, ISBN 978-1-4503-2015-3
84. Young W, Leveson NG (2014) An integrated approach to safety and security based on systems theory. Commun ACM 57(2):31–35. doi:10.1145/2556938, ISSN 0001-0782
85. Hogan M, Liu F, Sokol A, Tong J (2011) NIST cloud computing standards roadmap. Special oublication 500-291, National Institute of Standards and Technology
86. Montanari M, Campbell R (2011) Attack-resilient compliance monitoring for large distributed infrastructure systems. In: 2011 5th international conference on network and system security (NSS), pp 192–199. doi:10.1109/ICNSS.2011.6060000
87. Zenoss (2014) Unified monitoring and event management. Technical report, Zenoss
88. Huang J, Nicol D (2013) Trust mechanisms for cloud computing. J Cloud Comput 2(1). doi:10.1186/2192-113X-2-9
89. Montanari M, Chan E, Larson K, Yoo W, Campbell RH (2013) Distributed security policy conformance. Comput Secur 33:28–40. doi:10.1016/j.cose.2012.11.007, ISSN 0167-4048
90. Bernstein D, Vij D (2010) Intercloud security considerations. In: 2010 IEEE second international conference on cloud computing technology and science (CloudCom), pp 537–544. doi:10.1109/CloudCom.82
91. Buyya R, Ranjan R, Calheiros RN (2009) Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: challenges and opportunities. In: 2009 international conference on high performance computing and simulation, HPCS 2009, Leipzig, 21–24 June 2009, pp 1–11. doi:10.1109/HPCSIM.2009.5192685
92. Kretzschmar M, Golling M (2011) Security management spectrum in future multi-provider Inter-Cloud environments: method to highlight necessary further development. In: 2011 5th international DMTF academic alliance workshop on systems and virtualization Management (SVM), pp 1–8. doi:10.1109/SVM.2011.6096462
93. Senk C (2013) Adoption of security as a service. J Internet Serv Appl 4(1):11. doi:10.1186/1869-0238-4-11, ISSN 1867-4828
94. Uzunov AV, Fernandez EB (2014) An extensible pattern-based library and taxonomy of security threats for distributed systems. Comput Stand Interfaces 36(4):734–747. doi:10.1016/j.csi.2013.12.008, ISSN 0920-5489
95. Fernandez EB, Larrondo-Petrie MM, Sorgente T, VanHilst M (2006) A methodology to develop secure systems using patterns. In: Mouratidis H, Giorgini P (eds) Integrating security and software engineering: advances and future vision. IGI Global, Hershey. ISBN 1599041472