Two patterns for cloud computing: Secure Virtual Machine Image Repository and Cloud Policy Management Point

EDUARDO B. FERNANDEZ, Universidad Tecnica Federico Santa Maria, Chile
RAUL MONGE, Universidad Tecnica Federico Santa Maria, Chile
KEIKO HASHIZUME, Florida Atlantic University, USA

Two security patterns for clouds are presented: Secure Virtual Machine Image Repository, that controls the introduction or creation of malicious virtual machine images, and Cloud Policy Management Point, that defines a dashboard for the security administrator to control access to cloud resources.

Categories and Subject Descriptors: D.2.11 [**Software Engineering**]: Software Architecture—*Languages, Patterns*; H.1.2 [**Management of Computing and Information Systems**]: Miscellaneous—*Security*

General Terms: Security Reference Architecture

Additional Key Words and Phrases: Cloud Computing, Reference Architecture, Security patterns, Misuse patterns

ACM Reference Format:

Eduardo B. Fernandez, Raul Monge, and Keiko Hashizume, C. 2013. Two patterns for cloud computing: Secure Virtual Machine Image Repository and Cloud Policy Management Point. jn 2, 3, Article 1 (October 2013), 11 pages.

1. INTRODUCTION

There are many threats for cloud systems [Hashizume et. al. 2013a]. Some are old threats that apply to any distributed system using the Internet but a good number are new ones. However, few security patterns exist to control new threats. Two patterns for this purpose are presented here:

—**Secure Virtual Machine Image Repository**: Avoid the poisoning of VM images during creation and the leaking of sensitive information accidentally left in the VMI by enforcing access control to the repository.

—**Cloud Policy Management Point**: Provide an administrative dashboard for security functions, including authentication, authorization, cryptography, logging, and control of VM images.

These patterns are composite patterns and are shown in Figure 1. We describe these patterns using a modified POSA template [Buschmann et. al. 1996] and our audience are cloud architects, cloud system designers, and

This work is supported by the Chilean agency CONICYT, research contract 80120008, for the visit of Prof. Eduardo Fernandez, from Florida Atlantic University.
Author's addresses: Eduardo B. Fernandez (email: ed@cs.fau.ed, and Raul Monge (email: rmonge@inf.utfsm.cl), Departamento de Informática, Universidad Técnica Federico Santa, María, Avenida Espana 1680, Valparaiso, Chile; Keiko Hashizume, Florida Atlantic University, USA, email: keikoish79@gmail.com.

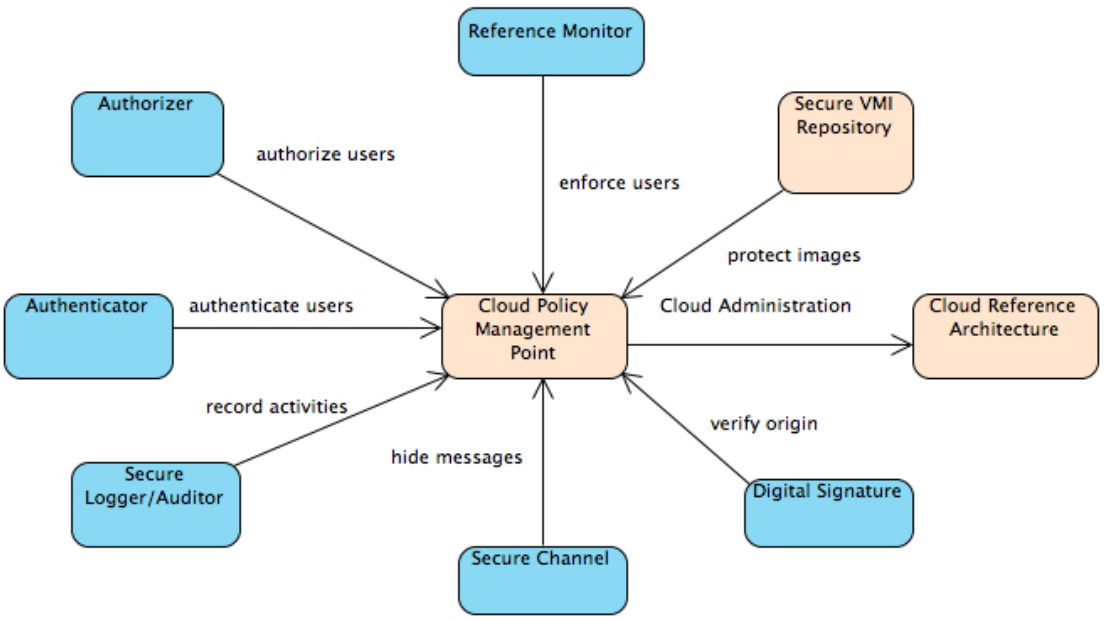

Fig. 1. Pattern diagram for the Cloud Policy Management Point and the Secure VMI Repository

cloud application developers. These patterns are part of a catalog of cloud security patterns that complement a Cloud Reference Architecture (RA) [Hashizume et. al. 2013c], in order to build a Security Reference Architecture (SRA). They have, of course, value on their own.

2. SECURE VIRTUAL MACHINE IMAGE REPOSITORY

2.1 Intent

Avoid the poisoning of VM images during their creation and the leaking of sensitive information accidentally left in the VMI by enforcing access control to the repository.

2.2 Context

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

2.3 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 Virtual Machine Image (VMI) could contain malware and we need to provide the consumers with clean images. An infected Virtual Machine (VM) could misuse customer data or attack other VMs. We need to clean the VMIs before their use.

—*Data leakage*. User may leave accidentally 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 way to control who can add, remove, or modify images in the repository.

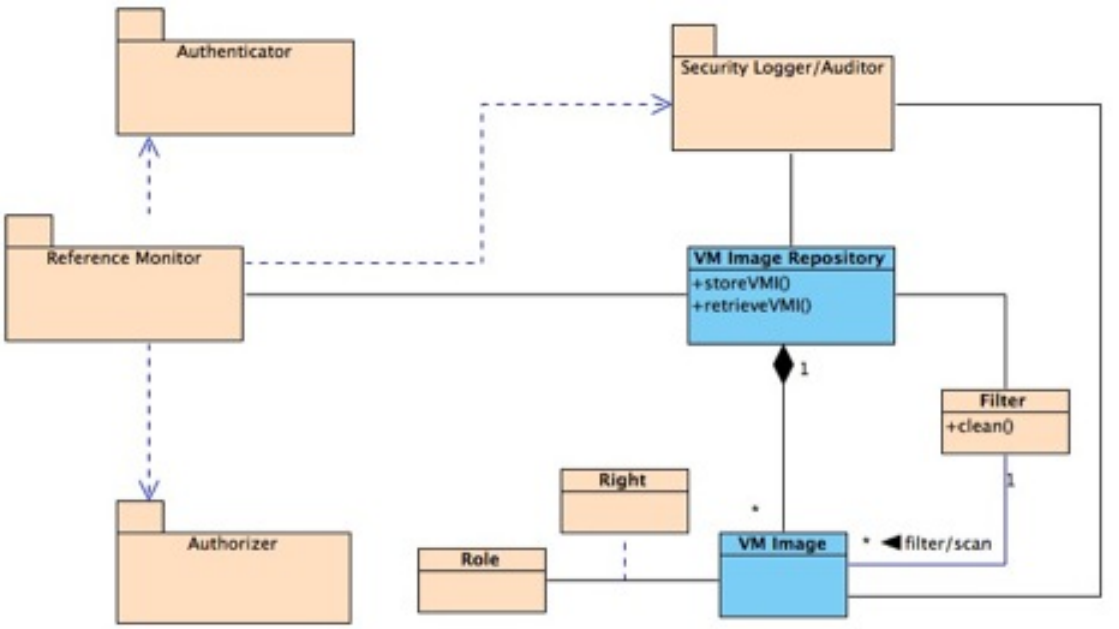

Fig. 2. Class diagram of the "Secure VMI Repository"

—*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.

2.4 Solution

Provide a mechanism to control access to VMIs in order to prevent atatckers from placing poisoned images or modifying existing images. Before placing a new image or using an existing image clean it by scanning and filtering.

Structure. Figure 2 shows a class model for the secure VM images repository system. The **Virtual Machine Image Repository** holds a set of **Virtual Machine Images** (VMI) 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.

Dynamics. Use cases include: Publish a VMI (Add a VMI to the repository), retrieve a VMI from the repository, modify a VMI, and others. Figure 3 shows the use case "Publish a VMI". In this use case a user in the role of Publisher logs in and authenticates himself. Once he is authenticated, he publishes a VMI (stores it in the VMI Repository) after being checked by the Reference Monitor.

2.5 Implementation

We can use different authorization models, e.g. access matrix, RBAC, or multilevel [Gollman 2006] to control access to the VMI Repository. If we use Role-Based Access control (RBAC) to control access we need to assign

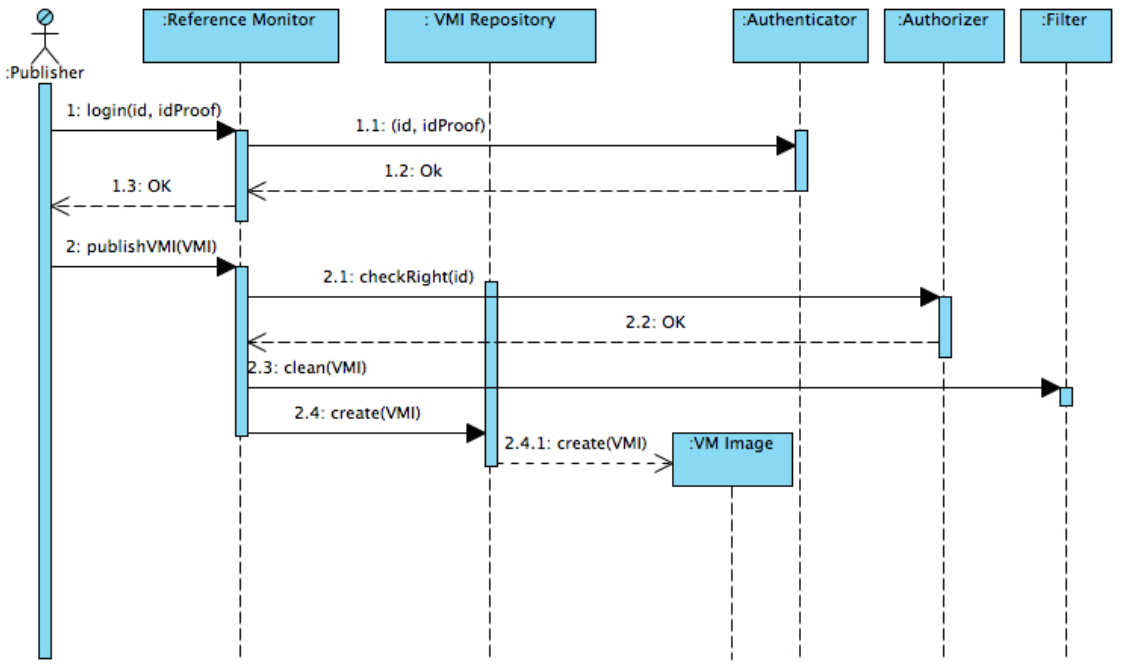

Fig. 3. Use case "Publish a VMI"

cloud consumers to roles. Only some roles will be able to add/modify VMIs. Other roles will only be able to retrieve images.

Some implementations e.g. [Wei et. al. 2009] also keep track of the origin of images and provide repository maintenance services. In some cases, an Update Checker identifies outdated software packages in VMIs [Schwarzkopf et. al. 2012].

2.6 Known uses

—[Wei et. al. 2009] proposes an architecture for a Secure Repository for VMIs, based on an analysis of threats. Their approach to address the threats is to control access to images, and to provide efficient image filters and scanners that can detect potential threats and repair the related vulnerabilities.

—[Schwarzkopf et. al. 2012] addresses the problem of patching software to keep it up to date, especially when vulnerabilities are patched. An Update Checker identifies outdated software packages in virtual machine images.

—[EMA 2010] indicates the importance of controlling access to images and suggests that administrative privileges are needed to define, retrieve, and manage VM images.

—The NIST Security Reference Architecture discusses requirements for VMIs [NIST 2013].

2.7 Consequences

The solution presents the following **advantages**:

—*Clean Images*. Scan and filter VMI to check for malware and remove it if present.

—*Data leakage*. Check new VMI for accidentally-left sensitive data and remove it.

—*Repository access*. We can control access by first authenticating users and then applying RBAC or a similar model.

—*Overhead*. Access to images is not a very frequent action so the overhead should be low.
—*Records*. We can log accesses and audit them later using the Security Logger/Auditor [Fernandez et. al. 2011].

Its **liabilities** include:

—*Extra work for the administrator*. Now the administrator has to keep track of images and their handling. A usable interface is important.
—*Cost and complexity*. The new mechanisms have to be bought or designed and add some complexity to the architecture.
—*Restrictive access*. We can also allow trusted users to publish VMIs.

2.8 Related patterns

—*Cloud Reference Architecture* [Hashizume et. al. 2013c]. Can be used to decide where to put security controls in the repository.
—*Security Logger/Auditor* [Fernandez 2013a]. How can we keep track of user's actions in order to determine who did what and when? Log all security-sensitive actions performed by users and provide controlled access to records for Audit purposes.
—*Authenticator, Authorizer, and Reference Monitor*. They control access to the VMI repository [Fernandez 2013a].
—*Malicious Virtual Machine Creation* [Hashizume et. al. 2013b]. This misuse pattern may not happen if access to the images repository is carefully controlled.
—*Malicious Virtual Machine Migration Process* (misuse pattern) [Hashizume et. al. 2013b]. The content of the transferred VM may have malicious code and compromise the receiving node. This malware may have been introduced by compromising images.
—*Cloud Policy Management Point* (in this paper). It includes the Secure VMI Repository as part of the security controls of the cloud administrator.

3. CLOUD POLICY MANAGEMENT POINT

3.1 Intent

Provide an administrative dashboard for security functions, including authentication, authorization, cryptography, logging, and control of VM images.

AKA

Security Dashboard, Security Administration Point.

3.2 Context

Cloud computing systems, but most of the functions apply to any distributed system.

3.3 Problem

Security administration requires controlling who can enter the system, who can access what resources, and related functions. How can we decide if a user should be allowed to enter our system? How can we define which resources the user is allowed to access? How can we protect information stored in the cloud and the data of our users?

The solution is constrained by the following forces:

—*Expressiveness*. We should be able to represent any policies or constraints to decide access.
—*Security*. Access and identity information should only be manipulated by authorized people.
—*Usability*. The information about access should be presented to the security administrator in a clear and systematic way.

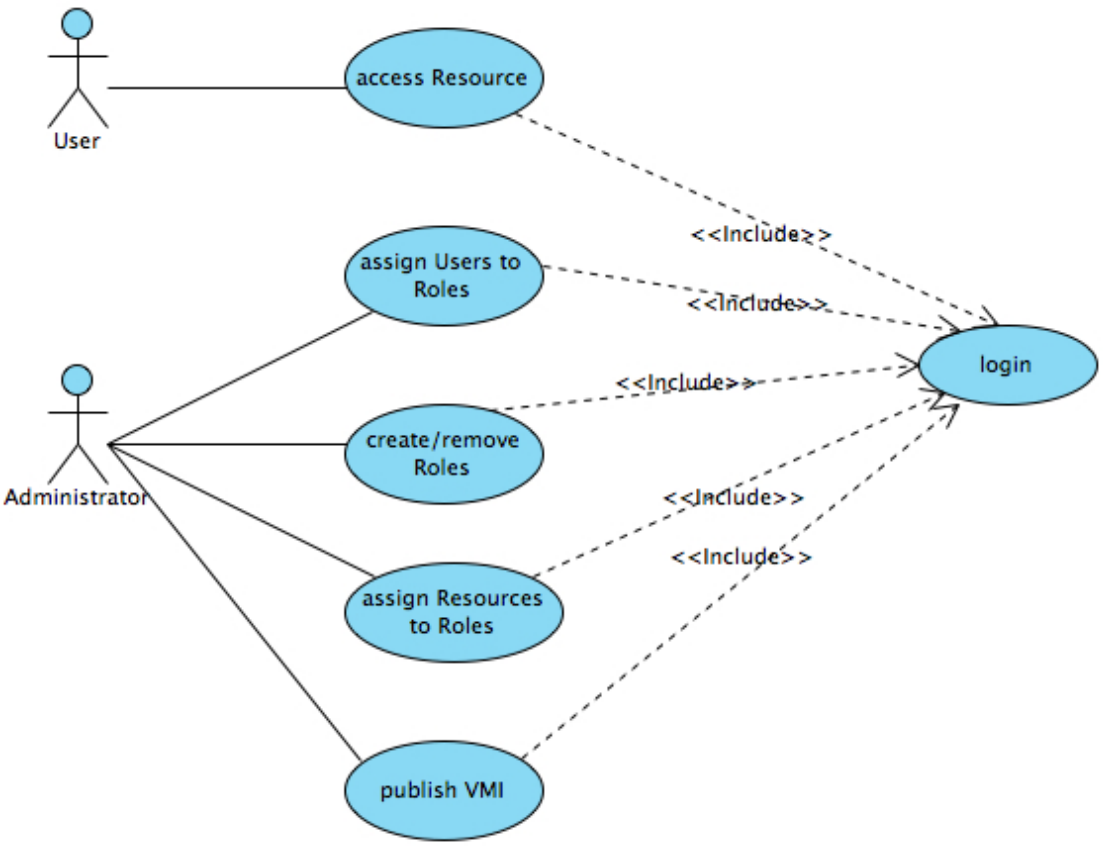

Fig. 4. Use cases for the Policy Management Point

—*Scalability*. The number of users and the number of roles should be able to be increased conveniently.

—*Extensibility*. We should be able to add new management functions in a convenient way.

—*Flexibility*. It should be easy to change policies about security.

3.4 Solution

Describe each conceptual element of authentication, authorization, and logging separately and with appropriate operations. Provide data to implement the use cases of Figure 4.

Structure. In Figure 5 the **Cloud Policy Management Point** (PMP) manipulates information about an RBAC authorization model, including **Role**, **Resource**, and **Right** (RBAC pattern). A Role includes a collection of **Users**, of which some are **Administrative Roles**. The **Authenticator** controls **Authentication Information** used to authenticate users (Authenticator pattern). A **Security Logger**/**Auditor** pattern includes the relevant functions. A **Cryptographic Module** includes the relevant cryptographic functions (**Encryptor** and **Signer**/**Verifier**) required by the PMP.

Dynamics. Figure 6 shows the use case: "Create a role and its rights". After the security administrator logs in he gets a token to create a session where he can create a role and assign rights to it.

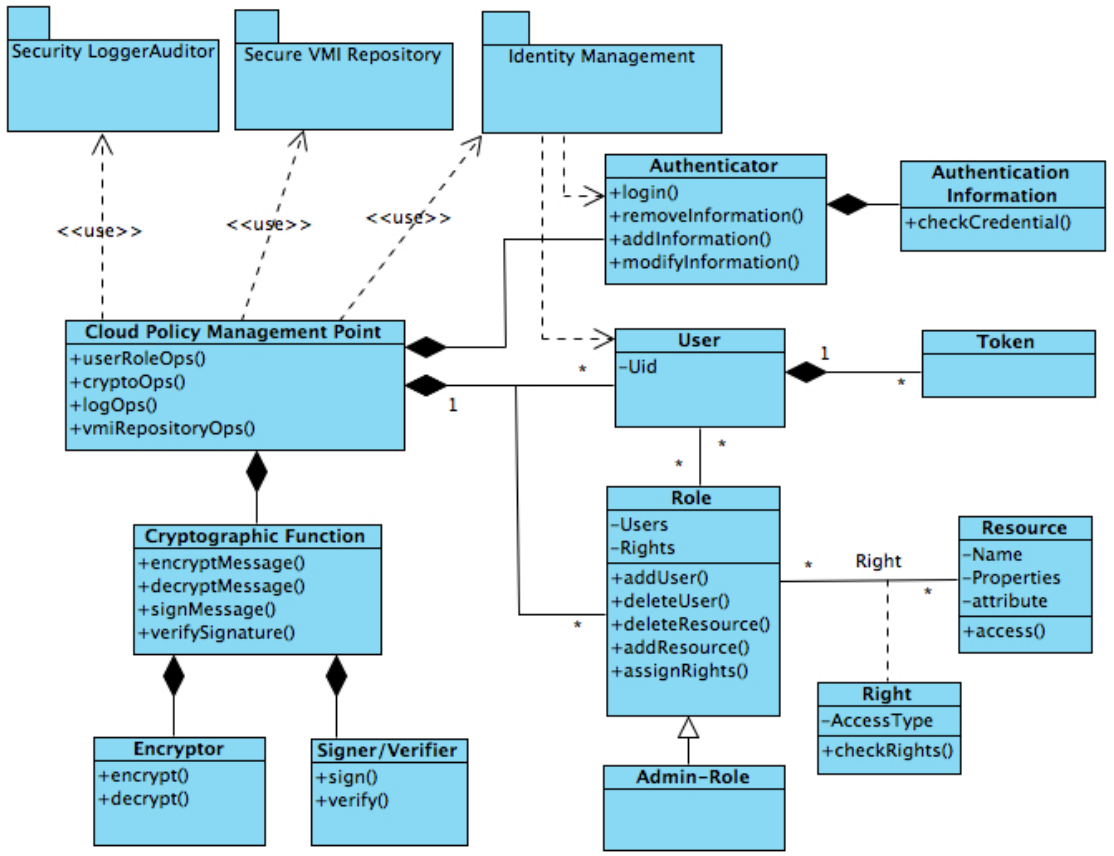

Fig. 5. Class diagram for the Policy Management Point

3.5 Implementation

All of the functions of Figure 5, except the VMI Repository, are replicated in the PaaS and SaaS levels, to be used by the corresponding administrators.

3.6 Known uses

—[Joosen et. al. 2011] describes a security dashboard using XACML to control the security of a cloud middleware system.

—Amazon's AWS [Amazon 2013].

—The Oracle Cloud Reference Architecture includes all the functions of this pattern [Oracle 2012].

—Fujitsu's cloud includes a security dashboard with similar functions [Okuhara et. al. 2010].

3.7 Consequences

The pattern has the following **advantages**:

—*Expresiveness*. We can represent any policies or constraints to decide access by describing them using roles and rights. Instead of RBAC we may use an Access Matrix to define authorizations.

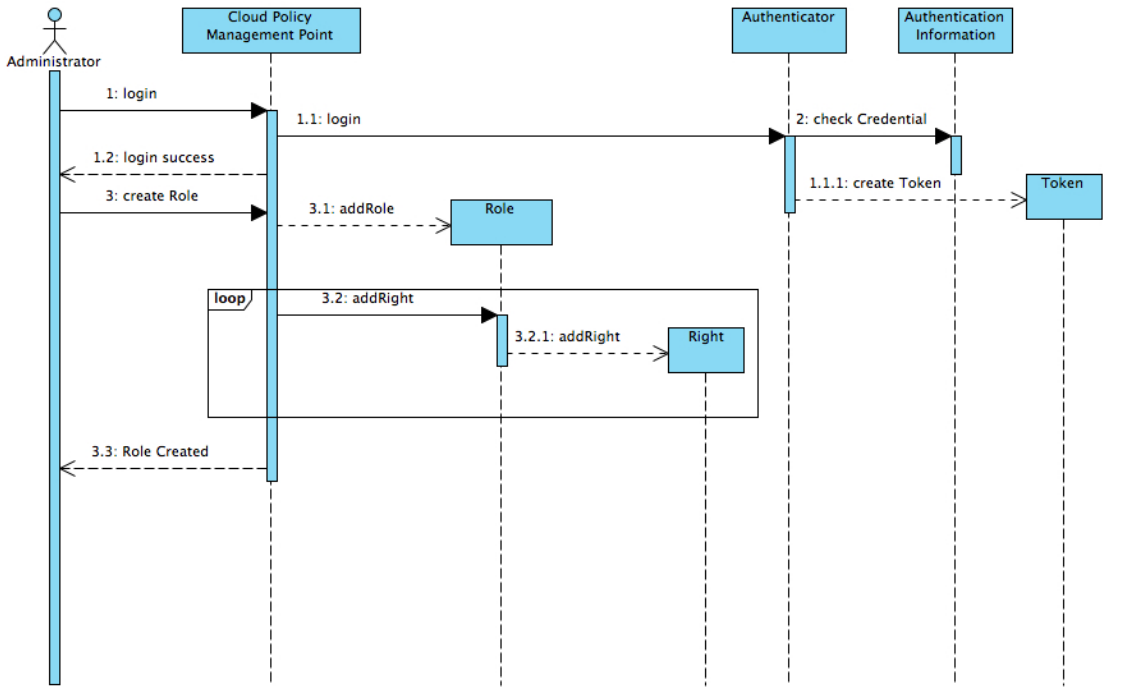

Fig. 6. Creating a role and its rights

—*Security*. Administrators are special roles and access to administrative information is controlled as for any other user.

—*Usability*. The classes correspond to logical units of access and other security functions and their use cases define a logical view of security.

—*Scalability*. We can increase the number of users and the number of roles just by creating objects in the respective classes.

—*Extensibility*. We can add new management functions by adding classes or operations in the model of Figure 5.

—*Flexibility*. Changing policies about security only requires to add/delete classes or operations in these classes.

Its **liabilities** include:

—*Complexity*. We need a good number of extra classes. The usability aspects are now very important.

3.8 Related patterns

—*RBAC* [Fernandez 2013a]. Describe how to assign rights based on the functions or tasks of people in an environment in which control of access to computing resources is required and where there is a large number of users, information types, or a large variety of resources.

—*Security Logger/Auditor* [Fernandez 2013a]. How can we keep track of user's actions in order to determine who did what and when? Log all security-sensitive actions performed by users and provide controlled access to records for Audit purposes.

—*Authenticator* [Fernandez 2013a]. When a user or system (subject) identifies itself to the system, how do we verify that the subject intending to access the system is who it says it is?

—*Policy-based Access Control* [Fernandez 2013a]. Decide if a subject is authorized to access an object according to policies defined in a central policy repository.

—*Secure VMI Repository* (this paper).

4. CONCLUSIONS

These patterns add to our catalog of cloud patterns, which will include patterns for SDN, OAuth, OVF, Secure Migration, and others.

Acknowledgements

We thank our shepherd Claudius Link for his useful comments that significantly improved this paper and the Chilean agency CONICYT for supporting the visit of Prof. Eduardo Fernandez at Universidad Tecnica Federico Santa Maria.

REFERENCES

AMAZON 2013. Amazon Web Services: Overview of security processes. June 2013. On-line: http://aws.amazon.com/security.

F. BUSCHMANN, R. MEUNIER, H. ROHNERT, P. SOMMERLAND, AND M. STAL 1996. *Pattern-Oriented Software Architecture – Volume 1: A System of Patterns*. Volume 1, Wiley, 1996.

ENTERPRISE MANAGEMENT ASSOCS.(EMA) 2010. Securing the administration of virtualization. *Market research report*, March 2010.

E. B. FERNANDEZ, NOBUKAZU YOSHIOKA, HIRONORI WASHIZAKI, AND MICHAEL VANHILST 1996. An approach to model-based development of secure and reliable systems. In *Sixth International Conference on Availability, Reliability and Security (ARES 2011)*, August 22-26, Vienna, Austria.

E. B. FERNANDEZ 2013a. *Security patterns in practice – Designing secure architectures using software patterns*. Wiley Series on Software Design Patterns, 2013.

E. B. FERNANDEZ, R. MONGE, AND K. HASHIZUME 2013b. A Security Reference Architecture. Submitted for publication.

D. GOLLMANN 2006. *Computer security* (2nd Ed.), Wiley, 2006.

KEIKO HASHIZUME, DAVID G. ROSADO, EDUARDO FERNANDEZ-MEDINA, AND EDUARDO B. FERNANDEZ 2013a. An Analysis of Security issues for Cloud Computing. Accepted for the *Journal of Internet Services and Applications*, Springer.

K. HASHIZUME, NOBUKAZU YOSHIOKA, AND E. B. FERNANDEZ 2013b. Three Misuse Patterns for Cloud Computing. In *Security Engineering for Cloud Computing: Approaches and Tools*, D. G. Rosado, D. Mellado, E. Fernandez-Medina, and M. Piattini (Eds.), IGI Global, 2013, 36–53.

K. HASHIZUME, E. B. FERNANDEZ, AND M. M. LARRONDO-PETRIE 2013c. A Reference Architecture for Cloud Computing. Sent for publication.

W. JOOSEN, B. LAGAISSE, AND E. TROYEN 2011. Towards application driven security dashboards in future middleware. In *Internet Serv. Appl.*, Nov. 2011, DOI 10.1007/s13174–011–0047–6.

NIST 2011. Cloud Computing Reference Architecture. 2011. On-Line: http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505.

NIST 2013. NIST Cloud Computing Security Reference Architecture. 2013. On-Line: http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/CloudSecurity/NIST_Security_Reference_Architecture_01.16.2013-clean.pdf

M. OKUHARA, T. SHIOZAKI, AND T. SUZUKI 2010. Security architectures for cloud computing. *Fujitsu Sci. Tech. Journal*, vol. 46, No 4, October 2010, 397–402.

ORACLE CORP. 2013. Cloud Reference Architecture. November 2012.

R. SCHWARZKOPF, M. SCHMIDT, CH. STRACK, S. MARTIN, AND B. FREISLEBEN 2012. Increasing virtual machine security in cloud environments. *Journal of Cloud Computing: Advances, Systems and Applications*, Springer, 2012, 1–12.

J. WEI, X. ZHANG, G. AMMONS, V. BALA, AND P. NING 2009. Managing security of virtual machine images in a cloud environment. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security (CCSW09)*, Chicago Illinois, USA, ACM 2009, 91–96.