A Software-Defined Security Strategy for Supporting Autonomic Security Enforcement in Distributed Cloud

Maxime Compastié, Rémi Badonnel, Olivier Festor, Ruan He, Mohamed Kassi-Lahlou

A Software-Defined Security Strategy for Supporting Autonomic Security Enforcement in Distributed Cloud

Maxime Compastié*†, Rémi Badonnel*, Olivier Festor*, Ruan He† and Mohamed Kassi-Lahlou†
*MADYNES Research Team - LORIA/INRIA, France. {remi.badonnel,olivier.festor}@loria.fr
†Orange Labs France.{maxime.compastie,ruan.he,mohamed.kassilahlou}@orange.com

***Abstract*—We propose in this paper a software-defined security framework, for supporting the enforcement of security policies in distributed cloud environments. These ones require security mechanisms able to cope with their multi-tenancy and multi-cloud properties. This framework relies on the autonomic paradigm to dynamically configure and adjust these mechanisms to distributed cloud constraints, and exploit the software-defined logic to express and propagate security policies to the considered cloud resources. The proposed framework is evaluated through a set of validation scenarios corresponding to a realistic use cases including cloud resource allocation/deallocation, cloud resource state change, and dynamic access control.**

***Index Terms*—computer security, network and service management, cloud computing, autonomic computing, policy based management**

I. Introduction

Cloud computing defines a new architectural model for building ellaborated infrastructures and applications based on multiple computing resources, such as virtual machines, network devices, software components, provided as a service that can be easily deployed through the Internet. According to the NIST Institute [1], this model is mainly characterized by the following features: *on-demand self-service*, *broad network access*, *resource pooling*, *rapid elasticity*, and *measured service*. It supports an as a service scheme that permits a transparent access to resources and the outsourcing of part of the management to the cloud provider. This separation enables optimizing the resource allocation and usage, but may also introduce management complexity due to its distributed nature. In particular, the cloud infrastructure and its applications may typically be divided into isolated sets of resources called *tenants*, corresponding to different ownerships and requirements, defining the *multi-tenancy* property. Another property comes to the facts that the resources may be distributed among several infrastructures, as each of them may be specialized in a dedicated processing. Distributed cloud can be defined by the conjuction of the *multi-tenancy* and *multi-cloud* properties.

In this context, security management has become a major challenge. The dynamics of cloud infrastructures induced by their *on-demand self-service*, *rapid elasticity* and distribution has outrun traditional security management, while the ubiquity and high availability of cloud resources make them attractive targets for attackers [2]. The paper introduces a software-defined security strategy for supporting autonomic security enforcement in distributed cloud. It investigates how autonomics and programmability mechanisms could support such a security management. Autonomic computing permits to address the scalability issues generated by large and distributed cloud infrastructure resources, by delegating part of the management tasks to the environment itself. In addition, network programmability has already showed its benefit for software-defined networking by separating the network infrastructure into two separate planes, i.e. data-plane and control plane, and contributing to its dynamic configuration and adaptation.

Similarly, there is an important need for supporting *software-defined security* in distributed cloud. We propose in that context a dedicated framework to specify security policies at this scale, and permit their autonomic enforcement in a multi-tenant multi-cloud environment. Security mechanisms should be dynamically aligned and adjusted based on changes that may occur in the distributed cloud. We detail the corresponding architecture and confront it to a set of validation scenarios corresponding to a realistic use-case.

The following of this paper is organized as follow: an overview of existing works and their limits is given in Section II. The proposed architecture and its components are described in Section III. They are evaluated through a set of validation scenarios in Section IV. Finally, Section V concludes the paper and points out future research efforts.

II. Related Work

The security of cloud infrastructures has already been largely explored in the literature. In particular [3] highlights several challenges related to policy-based security management, such as the specification of a cloud security policy, the assurance of the security decisions, as well as the the certification of security components in that context. In the same manner, the *TCloud framework* [4] proposes to enforce a security policy with a hardened cloud stack. This one provides infrastructure-level and platform-level security components, that might be compatible with multi-cloud environment, with a hardened build of OpenStack environment. However, these solutions do not specifically address self-configuration mechanisms, nor the management issues generated by multi-cloud and multi-tenancy properties. The author of [5] proposes also a cloud management framework able to deal with multi-tenancy, but this one is limited to access control policies and cannot

support other security mechanisms. The proposed architecture is independent from the available security mechanisms and adresses their self-configuration in a distributed cloud.

In the area of programmability, *software-defined networking (SDN)* permits to separate the *control plane* making decisions about where the traffic should be sent from the *data plane* forwarding of packets. This paradigm enables a dynamic and adaptive policy enforcement. It may also serve as a support for chaining security functions. For instance, the *Flowtags* framework described in [6] enables the integration of middleboxes whose composition is supported by SDN controller. [7] propose a framework for enforceing a network security policy through a set of middleboxes. But, this solution only considers middleboxs for instanciating security mechanisms. IETF is also working on SDN-based security services using interface to network security functions [8]. Such approaches the advantages of SDN with respect to security policy enforcement. We provide a *software-defined* framework, dedicated to security management and not limited to the networks enforcement field.

The autonomic computing paradigm defines a lifecycle featuring *self-management* and details the related architecture [9]. It includes *self-configuration*, *self-optimization*, *self-protection* and *self-healing* activities. Although it does not bring a formal distributed cloud support, it may introduce the negotiation among independent components. This approach may deal with exhaustive enforcement issues, as autonomic components can continuously enforce the security policy and adapt to the changes in their action perimeters. Even if the two previous paradygms do not directly deal with distributed cloud issues, they provide important building blocks for supporting security policy enforcement and defining a security management architecture in that context and in our framework.

With respect to security policies, the OASIS consortium introduces two standardized languages: *XACML (eXtensible Access Control Markup Language)* for representing and exchanging security policies [10] and SAML (*Security Assertion Markup Language*) for specifying security statements [11]. However, they do not handle any modifications of cloud policies nor its evolution propagation to enforcers. This approach remains relevant as the XACML defines modular components for security enforcement. Besides, an architecture and use-cases featuring XACML and SAML in distributed environment have been detailled in [12]. The latter validates the usability of XACML in distributed systems, underlining some limitations such as the need for a high granularity of sub-policies and the difficulty of maintaining an encoded security policy. The languages and formats introduced by SCAP protocol constitutes also an interesting support, as they cover many complementary specifications, such as vulnerability descriptions and scorings, that are exploitable for automating security in distributed cloud [13]. Those standards are usable in our architecture.

III. Software-Defined Security Strategy for Distributed Cloud Environments

Our approach introduces a software-defined security (SDSec) strategy for supporting distributed cloud, and addressing its multi-tenant and multi-cloud properties. It relies on *software-defined* scheme to provide a global security policy specification interface and exploit autonomic mechanisms within distributed cloud infrastructures to enable cloud resources to be dynamically and exhaustively protected according to this policy. More precisely, first consists in a *global security policy (GSP)* which formaly defines at a business level the security objectives of cloud resources and is then translated into several *tenant-level security policies (TLSP)*, providing security statements that must be verified by specified resources at the tenant level within the distributed cloud.

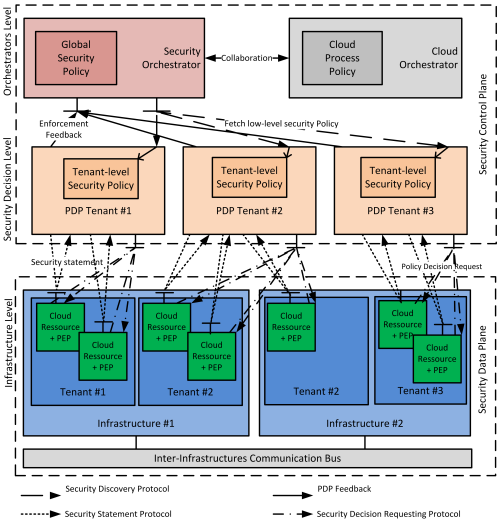

Figure 1. Proposed architecture for SDSec in distributed cloud

These *security statements* are then enforced on cloud resources, i.e. virtualized infrastructures and software products. They aim at altering the behavior of these components and protecting them based on countermeasures available with distributed cloud. This application can be active if its application requires negotiation with a decisional instance. The enforcement should be performed dynamically, more precisely in an adaptive (it adapts to any change in the enforced resource state or in the infrastructure), automatic (no operator interventions are needed for it), and self-configured manner (policy decisions for it are automatically made according to several criteria including the security requirements).

The different components parts of this SDSec architecture for distributed cloud are as follows:

- **The security orchestrator** hosts a GSP specified by the system administrator, exposes TLSP, and receives enforcement feedbacks from PDP to adapt them. It is reactive as exposed TLSP are modulated by such feedbacks and the cloud orchestrator interactions.
- **The policy decision point** parses its local TLSP to address security statements to the security enforcement stack and responds its security decisional requests. Its decisions can be influenced by third party components providing external security attributes.
- **The security discovery protocol** enables the PDP to

identify the security orchestrator and to fetch its corresponding TLSP. It works on the top of a middleware joining PDP and security orchestrator.

- **The policy enforcement point** is a component executing the security statements. It is dedicated to the enforcement on one type of cloud resources. It executes PDP security statements and exposes hooks for security decisions.
- **The enforcement discovery protocol** allows the PDP to identify and locate the PEPs in its security perimeter and identify their capabilities. It works between the PDP and the corresponding PEPs.
- **The statement protocol** sends the security statement from the PDP to the PEP, and returns the feedback. It is asynchrone, and is based on existing standards for security statements encoding.
- **The decision requesting protocol** enables the PEP to ask the PDP a security decision for a reactive enforcement in a synchronous manner.

Consequently, this architecture brings several benefits with respect to cloud security:

a) Distributed cloud protection enablement: this architecture is compliant with the *distributed cloud* constraint as it matches the multi-tenancy and multi-cloud properties. This compatibility is promoted by the GSP which is localy translated into a TLSP and adapted to a local resource enforcement. Moreover, the multi-cloud-aware discovery and communication protocols assist the multi-cloud security enforcement.

b) Software-defined management: this architecture implements the *software-defined* paradigm to configure the security mechanisms. This approach is inspired from the *software-defined networking (SDN)* but applied to the security field. Actually, in SDN, the control-plane corresponds to the SDN controllers and the data-plane matches the administrated network components. In our architecture, the control-plane is composed by the security orchestrator and all instanciated PDPs while all the instanciated PEPs and enforced cloud resources constitute the data-plane. The proposed architecture covers several areas of security policy enforcement. Also, this paradigm allows autonomic reaction to security state changes.

c) Self-configuration: this architecture relies on the *autonomic computing* concepts which propose a lifecycle for handling a dynamic and local management. Besides, this management is highly adapted to specified components. This property is mainly brought by the PEPs, as they configure and monitor the components they apply the security policy on. Moreover, they are specifically conceived for the components they are in charge of. Another interesting aspect of this self-configuration is the lower coupling with respect to the orchestration. Instead of the traditional orchestration model addressing a request and expecting a feedback, the security orchestrator adopts a passive approach by exposing security requirements, and refines the security requirement for different PDPs in different contexts.

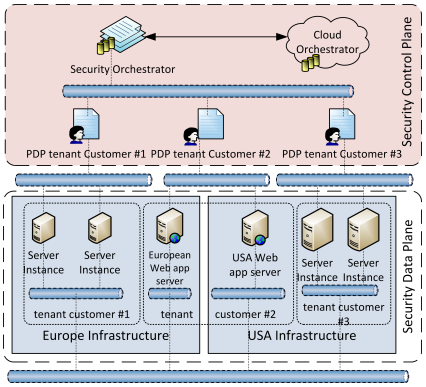

Figure 2. Use-case for supporting the validation of our SDSec architecture

IV. Architecture Validation

In order to evaluate the proposed approach, we confronted to several scenarios, based on the following use-case of a distributed cloud: a Cloud Service Provider (CSP) offers a *Platform-as-a-Service (PaaS)* solution to its customers, providing several infrastructures in different geographic areas. To protect its information system from the client security negligences (or deliberate malevolences), the CSP enforces a security policy (GSP) on its own infrastructures, and gather with each resource of its customers.

In this context, the multi-tenancy aspect of that use-case is promoted by the exploitation of the same infrastructure by several independent customers, and the multi-cloud aspect is promoted by the need for each infrastructure of the CSP to collaborate about the customers preferences, customers resource intercommunication and billing.

We will focus on the case of the *tenant customer #2* which regroups two virtual machines for hosting two web applications: one for a European version of his application and another one for the American version.

A. Resource allocation scenario

The CSP's client wants to set up a back-up server in his tenant for his web application. It must use SQL and FTP servers in a dedicated VM stored in the European infrastructure. The client requests the cloud orchestrator to create a VM instance from an image provided by the CSP and to deploy SQL and FTP servers. Each instanciated resource embeds a dedicated PEP. This deployment is notified to the security orchestrator.

Using the security discovery protocol, the PDP fetches the newly available TLSP from the security orchestrator. Then, using the enforcement discovery protocol, it identifies the new PEPs in its enforcement perimeter. As FTP and SQL services are newly deployed in the tenant, the security orchestrator assumes that the TLSP of the tenant's PDP is no more adapted and modifies the corresponding TLSP of this PDP. Those are again fetched by the security discovery protocol. Employing the security statement protocol, the PDP sends the new security statements to the PEP to enforce the TLSP, and fetches the

feedback. Finally, the PDP transmits a positive enforcement feedback to the security orchestrator.

B. Resource evolution scenario

The VM hosting the European instance of the web application is targeted by a *distributed denial of service (DDoS)* attack: its memory and CPU consumption indicators increase and exceed the alert threshold of the PEP, which was specified by the PDP. The PEP uses the security statement protocol to alert the PDP of the failing enforcement on the web server resources consumption. The PDP uses the security statement protocol to order the infrastructure PEP to increase temporarily the VM resource, and informs the security orchestrator of the non-enforcement of the GSP.

Then, the security orchestrator asks the network security enforcement to block attackers' IP addresses, and order the tenant's PDP to switch the affected VM into a fail-safe mode. Once the network security enforcement has blocked the DDoS attack, the security orchestrator reverts back the TLSP exposed to the *tenant customer #2* in order to restore the VM.

C. Access request scenario

The GSP imposes the credentials used for inter-cloud resource interactions to have an expiration delay. The verification of the validity is committed by the PDP using a tierce module. As the client has set-up back-up processes between the backup server hosted in the European infrastructure and the production server located in the american infrastructure, the related resources have to meet these requirements: when the production server connects to the back-up server, the connection is blocked and the related PEPs make decision requests to the PDP, providing hashes of the used credentials.

As the TLSP imposes the verification of the lifetime of the credentials as well, the PDP uses its third party module to check it. As this module has no precedent records of the hashes, it concludes that the transmitted credentials are newly created ones and are allowed to be used. The PDP responses to both security decision requests are positive. Thus, they are sent back to the two PEPs, allowing the two incoming connections.

D. Resource removal scenario

The client wants to update the VM supporting the American web application by proceeding to a fresh installation. To meet this objective, the client wants to completely remove it and reconfigure a new VM. He uses the cloud orchestrator to remove this VM, which is notified to the security orchestrator.

The security orchestrator updates its GSP to take into account the removal of the cloud resource and checks its consequences on the enforcement: the TLSP is updated. The PDP of *tenant customer #2* fetches the new TLSP, and stores it. The security orchestrator starts deallocating resources to the American VM and the PEP addresses a security decisional request to its PDP for allowing the removal. According to its TLSP, the PDP grants the request. The PEP lets the cloud orchestrator to complete the resource removal.

V. Conclusions

This paper proposes a software-defined security strategy for distributed cloud environments. It relies on the programmability of software-defined security, and exploits the autonomic paradigm for addressing the constraints induced by multi-tenancy and multi-cloud properties. The underlying architecture is composed of a security orchestrator, policy decision points (PDP) and policy enforcement point (PEP) interacting according to a dedicated set of protocols. Based on the specification of a security policy, it supports the dynamic configuration of security mechanisms to adjust to contextual changes, based on available resources and countermeasures. The proposed approach has been evaluated through a set of validation scenarios corresponding to a realistic use case, including cloud resource allocation/deallocation, cloud resource change, and dynamic access control.

The proposed architecture has raised several challenges with respect to the design of the considered components, and the specification of security policies in a multi-cloud and multi-tenant context. The PEPs will apply *model-driven scheme* to facilitate the inter-operability of heterogeneous enforcements. In the longer term, the security policy specification of distributed cloud, and the dedicated access mode will be investigated to complement the security orchestration.

References

[1] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing. 2011.

[2] Cloud Security Alliance. Top Threats to Cloud Computing v1. 0. *White Paper*, 2010.

[3] Adrian Waller, Ian Sandy, Eamonn Power, Efthimia Aivaloglou, Charalampos Skianis, Antonio Muñoz, and Antonio Maña. Policy Based Management for Security in Cloud Computing. In *FTRA International Conference on Secure and Trust Computing, Data Management, and Application*, pages 130–137. Springer, 2011.

[4] Alysson Bessani, Leucio A Cutillo, Gianluca Ramunno, Norbert Schirmer, and Paolo Smiraglia. The TClouds Platform: From the Concept to the Implementation of Benchmark Scenarios. *ACM SIGOPS Operating Systems Review*, 48(2):13–22, 2014.

[5] Olubisi Atinuke Runsewe. A Policy-Based Management Framework for Cloud Computing Security. Master's thesis, Université d'Ottawa/University of Ottawa, 2014.

[6] Seyed Kaveh Fayazbakhsh, Vyas Sekar, Minlan Yu, and Jeffrey C Mogul. Flowtags: Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pages 19–24. ACM, 2013.

[7] Tommy Koorevaar. Dynamic Enforcement of Security Policies in Multi-Tenant Cloud Networks. Master's thesis, École Polytechnique de Montréal, 2012.

[8] J. Park J. Jeong, H. Kim. Software-Defined Networking Based Security Services using Interface to Network Security Functions. Technical report, October 2015. draft-jeong-i2nsf-sdn-security-services-03.

[9] Jeffrey O Kephart and David M Chess. The Vision of Autonomic Computing. *Computer*, 36(1):41–50, 2003.

[10] Simon Godik, Tim Moses, A Anderson, B Parducci, C Adams, D Flinn, G Brose, H Lockhart, K Beznosov, M Kudo, et al. EXtensible Access Control Markup Language (XACML) version 1.0. 2003.

[11] Eve Maler et al. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML). *OASIS, September*, 2003.

[12] Jennifer Golbeck. Trust on the World Wide Web: a Survey. *Foundations and Trends in Web Science*, 1(2):131–197, 2006.

[13] David Waltermire, Stephen Quinn, Karen Scarfone, and Adam Halbardier. The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP version 1.2. *NIST Special Publication*, 800:126, 2011.