Automating Security Enhancement for Cloud Services

Mohamed Oulaaffart

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Automating Security Enhancement for Cloud Services

Renforcement Automatisé de la Sécurité des Services Cloud

THÈSE

présentée et soutenue publiquement le 05 décembre 2023

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Mohamed OULAAFFART

Composition du jury

Président :	François Charoy	Professeur à l'Université de Lorraine, France
Rapporteurs :	Nora Boulahia Cuppens	Professeure titulaire à Polytechnique Montréal, Canada
	Marco Anisetti	Professeur titulaire à l'Université de Milan, Italie
Examinateur :	Véronique Legrand	Professeure titulaire au CNAM, France
Encadrants :	Rémi Badonnel	Professeur des Universités à l'Université de Lorraine, France
	Olivier Festor	Professeur des Universités à l'Université de Lorraine, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Acknowledgments

First and foremost, I am extremely grateful to my supervisors, Prof. Rémi Badonnel, and Prof. Olivier Festor, for their invaluable advice, continuous support and patience during my PhD study. Their immense knowledge and plentiful experience have encouraged me in all the time of my academic research.

I would like to thank all members of RESIST research team in Inria/Loria Nancy for the support you have offered me throughout my journey.

My appreciation also goes out to all my friends for their unwavering support during the past three years.

And my biggest thanks go to my family, my wife Ghita, my parents and my brother Jamal for their encouragement and support throughout my studies.

Résumé

Les avancées des techniques de virtualisation et la maturité des langages d'orchestration ont contribué à la conception et au déploiement de services cloud composites. Ces services cloud peuvent être sujets à des changements dans le temps, en raison de la migration de leurs ressources. Cela peut introduire de nouvelles vulnérabilités, qui compromettent un service cloud dans son ensemble. Dans ce contexte, cette thèse propose d'améliorer et d'automatiser la sécurité des services cloud composites selon trois axes principaux. Le premier axe consiste en un framework de sécurité automatique basé sur SMT pour prendre en charge les migrations dans les services cloud composites, comme ceux orchestrés avec le langage TOSCA (*Topology and Orchestration Specification for Cloud Application*s). Il s'appuie sur des techniques de vérification pour évaluer automatiquement les changements de configuration qui affectent les composants des services cloud pendant leurs migrations, déterminer les vulnérabilités potentielles et sélectionner des contre-mesures adéquates. Le deuxième axe étudie la conception d'un tiers de confiance inter-cloud, appelé C3S-TTP (*Composite Cloud Configuration Security-Trusted Third Party*). Celui-ci est capable de réaliser une évaluation précise et exhaustive des vulnérabilités, tout en limitant les informations partagées entre le fournisseur cloud et le client cloud. Le troisième axe est centré sur l'investigation d'une stratégie défensive par cible mouvante qui combine des algorithmes d'intelligence artificielle avec des techniques de vérification. Le but est de compromettre les activités de reconnaissance effectuées par les attaquants à travers une large exploration des états, tout en minimisant l'apparition de nouvelles vulnérabilités qui peuvent avoir un impact important sur la surface d'attaques des services cloud concernés.

Mots-clés: Cloud Computing, Sécurité Cloud, Automatisation, Migration dans le Cloud

Abstract

The advances in virtualization techniques and the maturity of orchestration languages have contributed to the design and deployment of cloud composite services. These cloud services may be subject to changes over time, due to the migration of their resources. This may introduce new vulnerabilities, that compromise the whole services. In that context, this thesis proposes to enhance and automate the security of cloud composite services, according to three main axes. The first axis consists in an automated SMT-based security framework for supporting migrations in cloud composite services, such as those orchestrated with the TOSCA (Topology and Orchestration Specification for Cloud Applications) language. It relies on verification techniques for automatically assessing the configuration changes that affect the components of cloud services during their migrations and determining adequate countermeasures. The second axis investigates the design of an inter-cloud trusted third party, called C3S-TTP (Composite Cloud Configuration Security-Trusted Third Party). This one is capable to perform a precise and exhaustive vulnerability assessment, without requiring the cloud provider and the cloud tenant to share critical configuration information between each other. The third axis is centered on the investigation of a moving target defense strategy which combines artificial intelligence algorithms together with verification techniques. The purpose is to deceive reconnaissance activities performed by attackers through a large exploration of states, while minimizing the occurrence of new vulnerabilities that may impact on the attack surface of cloud composite services.

Keywords: Cloud Computing, Cloud Security, Security Automation, Cloud Migration

Contents

Glossary

APT : Advanced Persistent Threat
AWS : Amazon Web Services
BPMN : Business Process Model and Notation
BTS : Bug Tracking Systems
CSP : Cloud Service Provider
CVE : Common Vulnerabilities and Exposures
CVSS : Common Vulnerability Scoring System
DDoS : Distributed Denial of Service
GCP : Google Cloud Platform
HIPAA : Health Insurance Portability and Accountability Act
HOT : Heat Orchestration Template
IaaS : Infrastructure as a Service
IAM : Identity and Access Management
IPS : Intrusion Prevention System
MDP : Markov Decision Processes
MTD : Moving Target Defense
NFV : Network Functions Virtualization
NIDS : Network Intrusion Detection Systems
NIST : National Institute of Standards and Technology
NVD : National Vulnerability Database
PCI DSS : Payment Card Industry Data Security Standard
OLB : OpenStack Launchpad Bug-tracker
OSVDB : Open Source Vulnerability Database
OVAL : Open Vulnerability and Assessment Language
RL : Reinforcement learning
SaaS : Software as a Service
SARSA : State-Action-Reward-State-Action
SCAP : Security Configuration Automation Protocol
SDN : Software Defined Network
SFC : Service Function Chain
SIEM : Security Information and Event Management
SLA : Service Level Agreements
SMT : Satisfiability Modulo Theories
SMT-LIB : Satisfiability Modulo Theories LIBrary
SOFIC : Security Ontology for the Inter-Cloud
SSO : Single Sign On
TOSCA : Topology and Orchestration Specification for Cloud Applications
TPM : Trust Platform Module
WAF : Web Application Firewall
XCDDF : Extensible Configuration Checklist Description Format
XSA : Xen Security Advisory

Chapter 1

General Introduction

1.1 Context

Contents

Cloud computing introduces a new paradigm for composing and deploying large-scale distributed services, such as platforms, databases, network devices, virtual machines, and software components, over multiple cloud providers. It leverages the ongoing advances in virtualization techniques that facilitate the sharing of resources and also the efforts made in the network area that allow access to the resources in an on-demand manner from any location. Cloud computing is based on the *pay-as-you-go* approach, which permits cloud consumers to control their IT costs, since they pay only for the resources they use and reduce the equipment maintenance and acquisition costs. Furthermore, cloud computing enables users to scale up and scale down computing resources on demand in order to meet changing and fluctuating business demands.

According to the National Institute of Standards and Technology (NIST) [1], cloud services may be classified into three main models corresponding to different levels of control. The *Infrastructure as a Service* (IaaS) model provides the fundamental computing resources, such as processing, storage, and networking, required to run applications and user software, including operating systems. The cloud tenants can then rent web-accessible virtual storage that can scale up and down based on the needs of the customer. The *Platform as a Service* (PaaS) model provides the development and run-time environments for developing, running, and managing applications without taking care of the underlying infrastructures. The last model remains the *Software as a Service* (SaaS) which takes a step forward with the provision of applications fully managed by the cloud provider. These three models allow cloud consumers to specify and determine the level of control they want in their cloud infrastructures.

Cloud services may also be classified according to their deployment models. There are four main deployment models that are considered by NIST, including the public deployment, the private deployment, the community deployment, and the hybrid deployment, that depend on the method of access and the class of users eligible to access a given service. In a *public cloud*, the infrastructure is typically owned by an external cloud provider, such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP). The one hosts the services of many tenants that share the same cloud infrastructure, which can be accessed from any location worldwide using web interfaces. Cloud resources can be hosted in a data center, or be distributed across more than one data center that can be in one or more regions around the world. The cloud service provider is responsible for the maintenance and monitoring of the service. The *private cloud* is typically an on-premise and proprietary cloud owned by an organization and placed within its internal infrastructure. It targets the storage of sensitive information, and may only be accessible through the intranet of this organization. The *community cloud* corresponds to a deployment with the cloud is shared amongst several organizations that share the same constraints and interests, such as security requirements, compliance considerations, and business goals. The community cloud infrastructure is only accessible to the participating organizations. Finally, the *hybrid cloud* stands for a combination of two or more of these deployment models [1]. For instance, an organization may deploy sensitive data on-premises while placing non-sensitive data in the public cloud.

In the meantime, the maturity of orchestration languages, such as *Topology and Orchestration Specification for Cloud Applications (TOSCA)* [2], as well as the evolution of application architectures from monolithic to micro-services enable the building and deployment of more complex cloud services elaborated from elementary resources that are more cloud-native and provide better scalability and manageability. The orchestration languages provide a better expressivity and extensibility to further respond to tenant requirements. Also, the advances in migration techniques that allow resources to be relocated across several cloud infrastructures of the same or different cloud providers contribute to achieve multiple performance goals. Cloud migration, whether hot or cold, comes with important benefits in the context of cloud computing, by facilitating multiple management activities, such server consolidation, load balancing, resource management, energy savings, and remediation of server fault situations [3]. The advantages of these cloud services, including rapid elasticity, resource pooling, measured services, cost reduction, should not let users lose sight of the important risks that can jeopardize the security of these resources. In fact, these resources face several security threats that may result in negative personal, ethical, and financial impacts. Furthermore, the dynamics of resources across cloud infrastructures generated by their migration induce several configuration changes. These changes may result in the occurrence of vulnerabilities that may compromise the security of the migrated cloud resource, or even the security of the whole cloud composite service. Moreover, migrating resources to several cloud providers involves the deployment of valuable assets and sensitive information in some cases. This requires adequate mechanisms that guarantee the required level of security [4].

1.2 Problem Statement

With the rapid growth of cloud computing as a lever to build elaborated services that may be deployed over distributed infrastructures, it becomes a major challenge to guarantee the security of these services and their resources. Migration operations that enable the transfer of one (or several) cloud resource(s) from one provider (or infrastructure) to another one may induce several configuration changes. This process is often motivated by performance and cost objectives with regard to cloud properties such as scalability, elasticity, server consolidation, energy savings, and on-demand self-service. However, such changes may directly impact the security of cloud services and increase their exposure to security attacks. The changes that affect the migrated resources and their dependencies may involuntarily result in vulnerabilities that are exploitable by malicious users to cause important damages, including disclosure of information, data loss, and data tampering.

In the meantime, the facilities of cloud computing have contributed to the multiplication of cloud providers, that support cloud tenants with a large variety of resources. These resources may be deployed over different cloud infrastructures, and may be reallocated from one cloud infrastructure to another one, in order to meet performance requirements. Cloud tenants have to make sure that theses changes do not introduce vulnerabilities prior initiating any resource migration. This supposes the sharing of configuration information between the cloud tenant and the cloud provider. However, these stakeholders may be reluctant to share information regarding respectively the considered resource and the migration environment. There is a need to elaborate inter-cloud solutions that perform configuration verification to guarantee the required level of security, while minimizing the sharing of configuration information. In the literature, important efforts have been performed to design and implement inter-cloud security solutions, but mainly focusing on federated access control policies and identity management.

In addition, static defense techniques have shown their limits to efficiently protect such highly-exposed services. Moving target defense (MTD) methods introduce dynamics and contribute to defeat reconnaissance activities performed by hackers against these services. They consist in periodically applying changes on the configuration of these services and their resources. These methods have been enhanced by the integration of artificial intelligence and machine learning algorithms, in order to automate the selection of movements to be applied on the resources. While these techniques allow to increase the exploration of configurations and thus reduce the prediction of the hackers regarding the states of a given cloud service, they may also place the service in a vulnerable configuration.

In that context, a key challenge is to propose automation approaches for enhancing the security of cloud composite services by exploiting verification techniques. These approaches should take advantage of the knowledge provided by orchestration languages. They should facilitate the sharing of configuration information amongst stakeholders. They should also be adequate for different defensive security paradigms, such as dynamic and static ones, involving endogenous and exogenous security mechanisms.

1.3 Our Contributions

We propose in this thesis three main contributions that aim to automate and enhance the security of cloud composite services. The first contribution is an **SMT-based security framework** for supporting migrations in these services. Its goal is to automatically assess the configuration changes that affect the resources of cloud services, and to determine potential vulnerabilities during their migration. We implemented a proof-of-concept prototype to evaluate the feasibility and performance of this approach. The framework detailed in the first contribution serves as the basis for the elaboration of the second contribution. The second contribution consists in an **inter-cloud trusted third-party**, called C3S-TTP, which serves as an intermediate to share configuration informations from the cloud tenant and the cloud provider. Its objective is to address the reluctance of such stakeholders to share information about their resources and their infrastructures. Assessing vulnerabilities with a partial view may significantly degrade the assessment performance. We have designed the architecture of this trusted third-party, specified the interactions amongst stakeholders using an extended version of the TOSCA language, and shown to what extend it contributes to a better vulnerability assessment. Finally, the third contribution extends our work to the case of moving target defense strategies. We propose with that respect the design of a **novel moving target defense strategy** coupling artificial intelligence methods with verification techniques. The purpose of this one is to find a balance between the exploration of states that enables to defeat reconnaissance activities and the prevention of vulnerable configurations that may be exploited by attackers against cloud services.

1.4 Thesis Overview

The remainder of this manuscript is structured as given below, with Figure 1.1 summarizing its overall organization and the relationships amongst the different chapters.
Chapter 2 provides the state of the art regarding approaches on the orchestration of cloud resources. This includes efforts on orchestration languages and on the migration of cloud resources to achieve multiple goals, especially load balancing, server consolidation, maintenance activities, and energy savings. The level of automation provided by these approaches is given special consideration.
Chapter 3 focuses on existing approaches related to the security of cloud resource migration. We describe security threats, attacks targeting cloud composite services, and security mechanisms for orchestrated migration. We also highlight the security automation aspects provided in these approaches and point out the gaps that still exist in this area.
In Chapter 4, we introduce our first contribution on automated SMT-based security framework for supporting the migration of resources in cloud composite services. The goal is to analyze the configuration of TOSCA-based services in order to detect vulnerabilities using OVAL vulnerability description datasets. In the case of vulnerable configurations, the proposed approach includes identifying the necessary security mechanisms to address them. We also present the developed proof-of-concept prototype, which aims to demonstrate the approach feasibility.
Chapter 5 presents our inter-cloud trusted third-party approach, called C3S-TTP, for supporting secure configurations in cloud composite services. The proposed architecture is based on the framework detailed in Chapter 4. In this new approach, we extend the TOSCA language

to ensure information exchange amongst the different stakeholders involved in cloud resource migrations. The proposed trusted third party is capable of performing a precise and exhaustive vulnerability assessment without requiring the cloud provider and the cloud tenant to share critical configuration information with each other.
Chapter 6 introduces a novel moving target defense strategy to protect cloud composite services. In order to minimize the attack surface of such services, this strategy employs reinforcement learning algorithms coupled with verification techniques. The migration of resources is performed periodically or based on specific events, in order to counter the reconnaissance activities of attackers, but also to minimize the occurrence of new vulnerabilities.
Chapter 7 details the overall conclusions of our contributions. We also discuss several perspectives on how these approaches can be enhanced and extended.

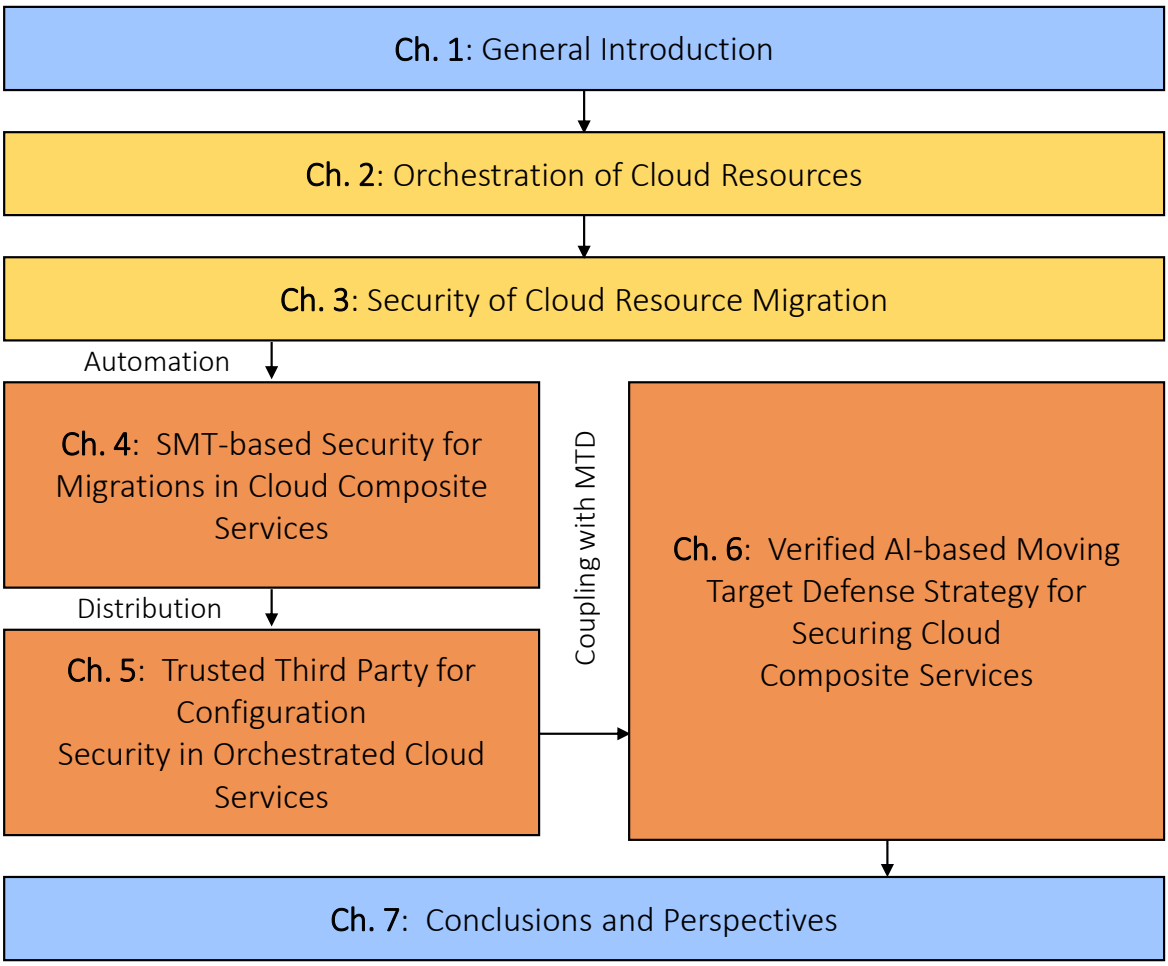


Figure 1.1: Thesis overview

1.5 Publications

The contributions of this thesis have led to the following papers:

- Mohamed Oulaaffart, Rémi Badonnel and Olivier Festor "Towards Automating Security Enhancement for Cloud Services." In the Proceeding of IFIP/IEEE International Symposium on Integrated Network Management (IFIP/IEEE IM), pp 692-696, IEEE, 2021.
- Mohamed Oulaaffart, Rémi Badonnel, Christophe Bianco. "An Automated SMT-based Security Framework for Supporting Migrations in Cloud Composite Services. " In the Proceeding of the IEEE/IFIP Network Operations and Management Symposium (IEEE/IFIP NOMS). pp 1-9, 2022, IEEE.
- Mohamed Oulaaffart, Rémi Badonnel and Olivier Festor "CMSec: A Vulnerability Prevention Tool for Supporting Migrations in Cloud Composite Services. " In the Proceeding of the IEEE International Conference on Cloud Networking (IEEE CloudNet). pp 1-9, 2022, IEEE.
- Mohamed Oulaaffart, Rémi Badonnel and Olivier Festor "C3S-TTP: A Trusted Third Party for Configuration Security in TOSCA-based Cloud Services", Submitted to Springer Journal of Network and Systems Management.
- Mohamed Oulaaffart, Rémi Badonnel and Olivier Festor ."A Verified AI-based Moving Target Defense Strategy for Orchestrated Cloud Services.", To be submitted to an international conference.

Chapter 2

Orchestration of cloud resources

Contents

2.1 Introduction

Cloud computing presents new models of distributed computing by offering the possibility to compose elaborate services based on complex hardware and software systems. These services are composed of heterogeneous resources and deployed across several infrastructures, and often exposing integration and interoperability dependencies. The interdependencies among cloud resources require complex configuration changes throughout their whole lifecycle, from the selection phase to the decommissioning one. This complexity makes the management of these resources hard and challenging, as these services are often made up of multiple individual components that need to work seamlessly together. These components may be provided by different vendors, each with their own interface, documentation, and support processes. Additionally, as these services

are often delivered over the internet, they may be subject to network latency and outages that can impact their performance. Moreover, composite services may need to be integrated with existing systems and applications, which can add further challenges to the management process. Therefore, the maturity of orchestration languages and advances in migration techniques enable the successful hosting and delivery of these services by meeting several performance goals.

Orchestration languages offer numerous performance advantages in a cloud environment. These languages enable the automation of complex processes, which can significantly improve the speed and precision of tasks that are generally performed manually. By automating these processes, orchestration languages can reduce the potential for human error, resulting in increased reliability and availability. In addition, orchestration languages enable efficient resource management, allowing organizations to optimize their usage of cloud services and reduce costs. This can be achieved through the dynamic allocation of resources to applications and services as their demands change. In addition, orchestration languages offer a high degree of scalability, which means that applications can adapt quickly and easily to changing workloads with no impact on performance.

When it comes to composite services deployed over cloud environments, migration techniques provide significant performance benefits. Migrations enable the movement of resources between physical infrastructures without interrupting their operation, allowing organizations to dynamically balance their workload and resources. These migration activities minimize downtime, which can improve overall service availability and reduce the risk of service disruption. Furthermore, resource relocation can help optimize resource utilization by allowing resources to be moved to hosts with available resources, improving performance and lowering the risk of overload.This technique also enables efficient server maintenance and upgrades, as resources can be seamlessly moved to alternate hosts without disruption, ensuring continued availability and reducing the impact of maintenance activities on performance. Overall, live migration techniques can significantly improve the performance and efficiency of cloud services, helping organizations better meet the needs of their customers and achieve their goals.

In this chapter, we first point out the challenges of managing cloud composite services. Then, we discuss different approaches that leverage the most known orchestration languages to achieve performance aspects and provide automation in the deployment of these resources. Finally, we present different migration techniques and their major roles in improving the performance of cloud resources and the underlying infrastructures.

The remainder of this chapter is therefore organized as follows. First, Section 2.2 details the most known orchestration languages and their role in improving the performance of cloud resources. Section 2.3 then describes different migration techniques and their importance for enhancing several performance goals. Section 2.4 discusses the optimization of migration techniques. Finally, Section2.5 provides a synthesis of the proposed approach.

2.2 Cloud orchestration of composite cloud resources

The management of cloud resources poses a significant challenge due to the simultaneous provisioning, scaling, and maintenance of multiple resources. Within this context, orchestration plays a vital role by enabling the automation of workflows. Orchestrating cloud resources is

essential for ensuring the seamless collaboration of diverse elements, thereby facilitating their efficient management. To accomplish this, orchestration languages such as TOSCA (Topology and Orchestration Specification for Cloud Applications), HOT (Heat Orchestration Template), and AWS CloudFormation, provide standardized means to specify and manage cloud infrastructure and resources. These languages also offer the capability to automate the deployment and scaling of applications, while addressing the complexities arising from resource distribution and heterogeneity.

2.2.1 Complexity of cloud resources management

Large-scale and distributed cloud resources are deployed over heterogeneous and multi-tenant infrastructures, making their management highly complex since this process involves the coordination of multiple tasks. In addition, the advent of multi-cloud computing, which enables the proliferation of autonomous and heterogeneous cloud service providers, further exacerbates the challenges of managing cloud applications. The deployment of these resources over multiple environments, public, private, and hybrid clouds, requires the interconnection of multiple distributed components to make them work as a single entity [5]. Cloud resource management relies on operating complex cross-domain processes and provides the tools to combine software, hardware, and manual processes into a fully operational system.

The management of cloud resources faces several issues related mainly to portability, interoperability, and integration. The portability of cloud resources enables them to be executed in a variety of environments, which is the bottleneck for deploying cloud resources across heterogeneous running environments [6]. In addition, the interoperability of resources in the cloud still represents a real challenge for managing resources [7]. Interoperability enables the transfer of resources across clouds; it gives cloud users efficient control over their workloads and a wider range of service options. Integration issues between various technologies create difficulties when managing cloud resources; they can affect all layers of cloud applications [8]. In this context, cloud resource orchestration fills this void in order to fully realize the potential of cloud computing.

2.2.2 Role and importance of orchestration in the cloud

Cloud resource orchestration aims to automate different tasks needed to manage the operation of workloads in cloud environments in order to meet specific business functions. The goal is to ensure successful hosting and delivery of resources and provide the required QoS by maximizing availability and throughput while minimizing latency and avoiding overload. Cloud orchestration operates across all layers of a cloud stack by enabling automation through different cloud models. Orchestration provides several benefits. It allows the reduction of the number of needed IT resources by coordinating processes. It also simplifies the management of redundant tasks, reducing the amount of know-how, manual effort, and time needed to manage these resources. Moreover, cloud orchestration ensures more resilience by providing dynamic interconnections among heterogeneous devices [9, 10].

The automation of cloud resource deployment across heterogeneous environments remains a challenging issue, in particular in a multi-cloud context. Thus, many frameworks such as Puppet, Ubuntu Juju, Ansible, Amazon OpsWorks, and Chef enable several features of orchestrating these resources. Languages such as TOSCA, HOT, and AWS CloudFormation are the most

mature languages to describe such topologies and resources and represent the most possibilities for modeling cloud applications. They remain definitive ways and means to provide orchestration solutions and tools to ensure automation and orchestration of cloud resources.

2.2.3 Operations of cloud orchestration

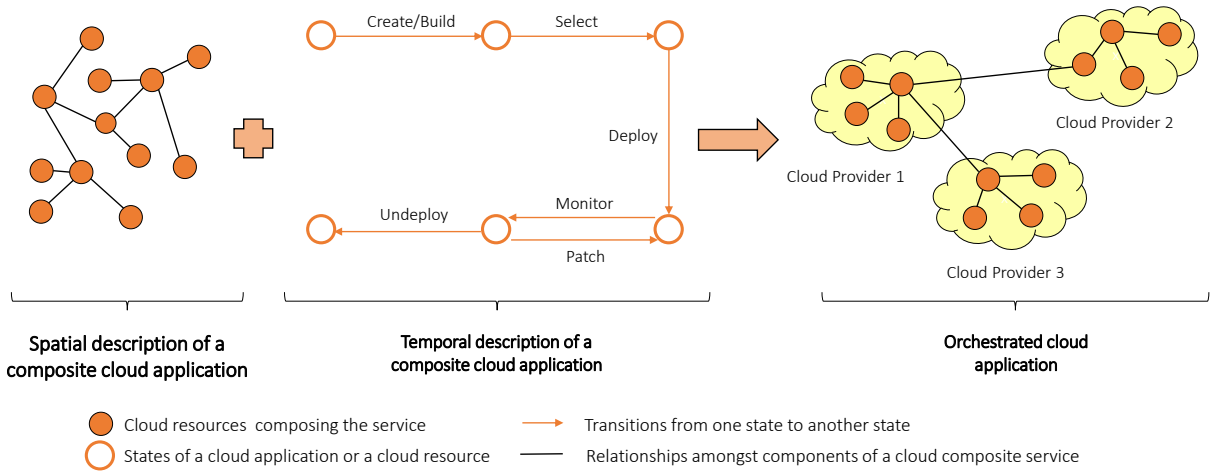


Figure 2.1: Orchestration of a composite cloud service

The orchestration of cloud composite services consists of several phases. Figure 2.1 showcases an overview of the two major descriptions of a composite cloud service: the temporal and the spatial description. The first one, on the left of the figure, represents a cloud service composed of several components denoted by orange nodes. The black lines that connect these different nodes symbolize the relationships and dependencies necessary to ensure interconnections and communications amongst these components in order to guarantee the operation of the whole cloud service. Let us consider a concrete example of a distributed e-commerce application involving a set of micro-services and back-end databases. In this case, the nodes correspond to micro-services implemented in the form of web applications and MySQL database instances. The web micro-services are linked to the database instances using two types of relationships named store and retrieve from. They allow to store consistent data in the database and retrieving the required data when requested.

The operation of a cloud service involves several phases that represent its overall lifecycle. The latter is clearly represented in the middle part of Figure 2.1. It consists of selecting, describing, configuring, deploying, monitoring, and controlling cloud resources. The selection phase aims to define resources to meet a specific need. These resources are then configured in order to establish different relationships and interconnections among them. This step also consists of defining a set of properties and attributes for each component. The resources are distributed across the appropriate cloud environment(s). Defined descriptions are interpreted into operations, and different components start running according to the specified configuration properties. When all cloud application components are connected to each other and begin providing the expected services,

monitoring operations are then performed regularly to ensure that resources are continuously operating according to requirements and prevent the violation of SLAs (Service Level Agreements). In some cases, deployed applications do not respond to the defined requirements, or some aspects of SLAs are violated. Thus, control actions should be taken in order to remediate such situations [5, 10]. The right part of Figure 2.1 represents the cloud service distributed over three cloud providers. The choice of such a configuration depends on several factors, such as privacy and the underlying costs.

2.2.4 Orchestration languages

Orchestration of cloud composite services represents a fastidious and challenging task due to the heterogeneity and multi-tenacy of cloud platforms and resource composition. It involves deploying, provisioning, managing cloud resources, and maintaining the desired level. Cloud orchestration languages provide answers to the issue of automating deployments of cloud resources over several cloud environments. They enable the creation of composite and distributed services from elementary components. These languages also permit the definition of interoperable and portable applications across cloud infrastructures. They ensure the scalability and automation of cloud resources. During our survey of the various existing works, we identified three main orchestration languages that present high level of maturity we have compared them based on several points: TOSCA, HOT and AWS CloudFormation.

- **Topology and Orchestration Specification for Cloud Applications**: It is an OASIS standard that consists of describing cloud applications using topology templates. TOSCA enables automated management of the whole life-cycle of cloud application components. It also allows for specifying the capabilities and requirements of each cloud resource and defining relationships and dependencies among cloud application components. TOSCA descriptions are interoperable and reusable regardless of the underlying platforms. TOSCA provides a system type to express different template entities such as node types, relationship types, requirement types, capability types, artifact types, and policy types. Recently, the TOSCA simple profile in YAML has been in specified. It contains a notable enhancement and provides a declarative method for defining cloud application topologies using TOSCA [2, 11, 12].

- **Heat Orchestration Template**: It is a template format supported by HEAT, the main project of the OpenStack Orchestration program. The templates of HOT are orchestration documents written in YAML to automate deployments of the whole life-cycle of cloud applications across multiple cloud providers. The main elements of HOT are presented as a collection of resources that could be virtual machines, networks, security groups, servers, floating IPs, and volumes. These resources are grouped into logical elements called stacks. HOT also allows to specify functions inside templates by means of properties and parameters. HOT is compliant with the CloudFormation Template defined by Amazon, and such a function offers the possibility to execute existing CloudFormation templates on OpenStack [13, 14].

- **Amazon CloudFormation**: It is an Amazon-developed orchestration language. It is used to model and automatically provision AWS and third-party cloud application resources.

AWS templates are used to model and provision AWS resources and any associated dependencies or runtime parameters required to run applications, using JSON or YAML. Moreover, in AWS CloudFormation, cloud resources are managed as a single unit named a stack, which is a set of software and hardware resources. Amazon can provide an auto-scaling group, an elastic load balancer, and an Amazon relational database service after the stack cloud applications are up and running to avoid manually configuring resources to work together [11, 15].

Table 2.1 provides a detailed comparison amongst TOSCA, HOT and AWS CloudFormation. In this table, there are 4 levels (not covered, low, medium, and high) to compare the composability, extensibility, interoperability, portability, scalability, and automation of these languages. Not covered means the corresponding characteristic is not discussed in the official documentation or approaches using this language. The low level means that the considered criteria are barely mentioned in the documentation. The medium level signifies that characteristics are discussed in the documentation or the approaches that leverage such languages, but they still need to be developed. The last level (high) stands for the highest level of maturity regarding this characteristic.

Table 2.1: Comparison amongst the orchestration languages OASIS TOSCA, HOT and AWS CloudFormation.

Orchestration Language	**OASIS TOSCA**	**OpenStack HOT**	**AWS CloudFormation**
Service Model Coverage	IaaS, PaaS and SaaS Model	IaaS Model only	IaaS, PaaS and SaaS Model
Language Nature	Open and Standardized	Open and Standardized	Proprietary
Language Type	Declarative	Declarative	Declarative
Support Language	XML/YAML	YAML	JSON/YAML
Edition Tools	Winery Modelling Tool	Horizon Dashboard	AWS CloudFormation Designer

Continued on next page

Table 2.1 – *Continued from previous page*

Orchestration Language	**OASIS TOSCA**	**OpenStack HOT**	**AWS CloudFormation**
Resources	Cloud Resources (Network, Storage, Containers, Clusters, Micro services)	Cloud Resources(Compute Resources, Storage Resources,Networking Resources, Security Groups, Scaling Rules, and Custom Resources)	AWS Resources (Single Amazon EC2 Instances, Complex Multi-tier Applications, Multi-region Applications)
Resources Description	Template Topologies (Service Templates)	Templates	Templates
Templates Types	Node Templates and Relationship Templates that together define the topology model of a service as a directed graph	A Unique Type for All the Resources	A Unique Type for All the Resources
Relationships	Specific Templates Types are Used to Define Relationships Between Nodes (such as: HostedOn, ConnectsTO)	Templates Specify the Relationships Between Resources (e.g. This Volume is Connected to This Server)	Templates Specify the Relationships Between Resources
Runtime Objects	Instances	Stacks	Stacks
Expressivity	●	◐	◐
Composability	●	◐	◐
Extensibility	●	◐	○
Interoperability	◐	◐	○
Portability	◐	◐	○
Scalability	●	●	●

Continued on next page

Table 2.1 – *Continued from previous page*

Orchestration Language	**OASIS TOSCA**	**OpenStack HOT**	**AWS CloudFormation**
Automation	●	◐	●
○ : Not covered	◔ : Low	◐ : Medium	● : High

2.2.4.1 Expressivity

Many aspects are provided by orchestration languages, allowing cloud clients to express various requirements. They allow to describe the whole structure of a given cloud service, including components, dependencies, relationships, attributes, and parameters. They provide means to manage the deployment and configuration of individual components composing cloud services, guaranteeing the required QoS and ensuring a high level of communication amongst different components and services [16]. These languages present some similar aspects and different features in terms of expressivity. For instance, TOSCA and HOT are standards, whereas AWS CloudFormation is a proprietary format from AWS. Furthermore, OASIS TOSCA and AWS CloudFormation address the IaaS, PaaS and SaaS layers, allowing the automation of any production process, while HOT addresses the IaaS layer only. Although TOSCA, HOT, and AWS CloudFormation are declarative, their adopted languages are different. The first version of TOSCA was XML-based, but lately it has provided a YAML-based language. AWS CloudFormation templates are either in the JSON or YAML format, and HOT is a YAML-based language.

Concerning the edition tools, TOSCA relies on Winery, a web-based environment that enables the modeling of the whole service template. For HOT templates, a graphical tool is provided by the Horizon dashboard, which offers a panel for the heat orchestration project. AWS CloudFormation offers a modeling tool called AWS CloudFormation Designer, which is used for creating, viewing, and modifying templates. It allows to design template resources using a drag-and-drop interface and then edit their details using an integrated JSON and YAML editor. Even though these languages rely on customizable templates to describe cloud applications, their structure remains different. Actually, OASIS TOSCA introduces two types of templates: Node templates and relationship templates. They are structured into components and allow to define the capabilities and requirements of each resource. Unlike AWS CloudFormation and HOT, which only allow to represent the entire resource, dependencies between nodes in TOSCA can be customized and defined separately using the Relationship type templates. However, in HOT and AWS CloudFormation, the dependencies are expressed in the same template as the other resources. HOT and AWS CloudFormation run the defined resources in a stack that refers to a collection of resources that can be managed in a single unit. A stack can have all the resources such as web servers and databases, while TOSCA considers the notion of instances.

2.2.4.2 Composability and extensibility

With the introduction of service-oriented architectures that provide valuable services to cloud consumers, orchestration languages have evolved to provide the ability to compose reliable services from elementary components. This paradigm enables the integration of distributed and heterogeneous services over the internet in order to consolidate business applications. In this context, the standardizing service templates of OASIS TOSCA facilitate the composition of a service from components, even if those components are hosted on different providers, including the local IT departments, or in different automation environments, often built with technology from different suppliers. For instance, large organizations could use automation products from different suppliers for different data centers [17]. In the same context, the AWS CloudFormation documentation lists a set of supported resources that can be composed using this language. In the same way, Heat Orchestration Templates enables the building of reliable cloud applications from elementary components.

Orchestration languages provide extensible and reusable elements that allow users to define new features to cover a specific need. They enable defining cloud applications that permit meeting specific users requirements and help to avoid redundant tasks. In this regard, according to the official documentation of the TOSCA language, it is considered extensible and supports a wide range of standards and policies. This mechanism allows attributes and elements from any other name-space to be integrated and appear on any TOSCA element that do not contradict the semantics of TOSCA elements [17]. AWS CloudFormation also includes AWS CloudFormation Registry, which enables the modeling and management of cloud applications. Similarly, input parameters defined in the parameters section of the HOT templates allow users to customize a template during deployments. This makes templates more easily reusable by avoiding hard-coded assumptions.

2.2.4.3 Portability and interoperability

Portability and interoperability in the cloud refer to the ability to build reliable systems from reusable components that will interact and work together in heterogeneous infrastructures. These features ensure the migration of resources across multiple cloud service providers and guarantee the same QoS for cloud applications regardless of the underlying environment. They cover several aspects, including data portability, application portability, platform portability, application interoperability, and platform interoperability. These two properties ensure the re-use of their elementary components across heterogeneous computing platforms.

In this context, OASIS TOSCA allows for the definition of an interoperable descriptions of cloud services, including their components, relationships, dependencies, requirements, and capabilities. This enables the portability of cloud applications and their management across cloud providers, regardless of the underlying platforms or infrastructures. In addition, OASIS TOSCA offers a standardized way to describe the topology of cloud composite services. It also addresses management portability using the portability of workflow languages used to describe deployment and management plans [18]. TOSCA allows for the description of the components of complex cloud applications that can be combined independently of the vendor(s) supplying them. Regarding HOT, it permits defining portable applications only across multiple OpenStack providers. HEAT also provides compatibility with the AWS CloudFormation template format. It

offers the possibility to launch existing CloudFormation templates in OpenStack environments. Concerning AWS CloudFormation, it guarantees the portability of AWS cloud resources across regions using AWS Lambda functions. In fact, its proprietary and non-standard aspects make it very specific and limit its impact on the interoperability and portability.

2.2.4.4 Scalability and automation

Cloud infrastructures ensure the scalability of resources, including data storage capacity, processing power, networking, and cloud application instances. This property allows to increase or decrease the number of running instances in order to meet changing demands. Scalability in cloud environments can be achieved in two ways: vertically or horizontally. The vertical scaling of resources consists of adding or subtracting power from an existing cloud server's storage capacity, memory, or processing power, whereas the horizontal scaling of resources aims to add more resources like servers to the system to spread out the workload across machines, which in turn increases performance and storage capacity. For this, cloud providers offer numerous solutions to respond appropriately and cost-effectively to the increase in workload demands. However, scaling up or down applications should be defined by the user proactively in the descriptions of cloud applications. Thus, cloud orchestration languages offer the possibility of specifying elastic parameters and policies. For instance, the TOSCA language offers a specific type named tosca.capabilities.Scalable to express the scalability of resources in its templates. This element defines the maximum, the minimum, and the default number of instances of a node. Also, AWS CloudFormation and Heat allow the creation of auto-scaling stacks. For this, metrics need to be retrieved from resources, and some actions need to be triggered when a specified event occurs on these metrics. In Heat, the functionality of auto-scaling is provided by the Auto-Scaling group, which is defined in the templates. Regarding AWS CloudFormation, templates allow to define auto-scaling policies that permit to add or remove instances from the auto-scaling group when the defined thresholds are exceeded.The scalability features are optimized by the automation provided by orchestration languages. This aspect of automation covers many aspects, including scaling resources, failure recovery, dependency management, and numerous other tasks. In order to achieve this objective, orchestration languages such as OASIS TOSCA, HOT and AWS CloudFormation allow for the automation of tasks related to the cloud environment and reduce manual interventions while provisioning and managing cloud computing composite applications. These languages also offer new methods for managing dependencies between tasks, as well as automatic failure recovery when hazardous faults occur or something goes wrong in the process. The standardization aspect of TOSCA enables automated management of cloud applications across heterogeneous cloud provider platforms. Despite the fact that AWS CloudFormation and HOT enable automated orchestration of cloud resources in limited types of infrastructure, some approaches extend orchestration languages to reduce time consumption and the technical know-how required to deploy cloud applications in order to improve automation. For example, the authors of [19] propose a framework for automatically completing TOSCA templates. They suggest that the user focus only on defining application component templates without giving any attention to platform and infrastructure templates. The latter will be provided by the proposed algorithm. The TOSCA descriptions are completed by adding middleware and infrastructure components until the topology is complete and can be processed by a provisioning engine. The

proposed solution enables the user to specify which components should be added to the topology. Other initiatives go one step further and propose solutions that combine orchestration languages with other tools or languages in order to achieve automation goals. As an example, [20] presents a solution that aims to build up a cloud service orchestrator capable of automating the execution of tasks and operations required for the provisioning of a cloud application. The approach grounds on the TOSCA simple profile, which provides a meta model written in YAML. The designed orchestrator takes a YAML model and transforms it into a BPMN (Business Process Model and Notation) model, which is then fed to an engine that instantiates and coordinates the related process. The solution is composed of three components, a TOSCA parser that parses and validates TOSCA templates, the BPMN-generator that creates a provisioning plan leveraging TOSCA templates, and the BPMN-validator that validates the generated plan.

2.3 Resources migration in the cloud

Advances in virtualization techniques and the composability of cloud services contribute to the migration across cloud environments. Migrations involve transferring one or more elementary components of a cloud service from one cloud provider (infrastructure) to another. The migration of resources aims at achieving numerous performance goals, such as load balancing, maintenance operations, and energy savings. Migration activities, whether hot or cold, impact the performance of the affected resources.

2.3.1 Importance and role of cloud migration

Cloud service providers take advantage of the benefits provided by virtualization techniques in order to exploit their infrastructures wisely. However, the growing number of cloud resources deployed in such infrastructures makes their management hard and requires thoughtful strategies. Otherwise, manipulations of these resources in data centers, such as creating and destroying instances, and their exploitation by cloud consumers, may result in cloud infrastructure misuse. Thus, techniques of migration introduce new mechanisms to overcome such an issue. They aim to relocate one or several resources of a given cloud service from one cloud provider to another with respect to the required performance goals.

Figure 2.2 showcases an example of a migration process. It represents a cloud composite service distributed across three cloud providers (cloud provider 1, cloud provider 2, and cloud provider 3). This service is composed of 23 elementary components symbolized by the orange nodes (8 in the first cloud provider, 8 in the second cloud provider, and 7 in the third cloud provider). The components are linked to each other using either horizontal dependencies (black lines) when they belong to the same layer or vertical dependencies (red lines) when they are in different layers. The elementary component represented by the blue node is subject to a migration from cloud provider 1 to cloud provider 2. This operation generates modifications to the overall configuration of the considered cloud service, including its dependencies. The initial dependencies are deleted, and new ones are created (discontinuous black lines). Other components of the application may also be migrated depending on the context of the application (workloads, maintenance objectives, server states).

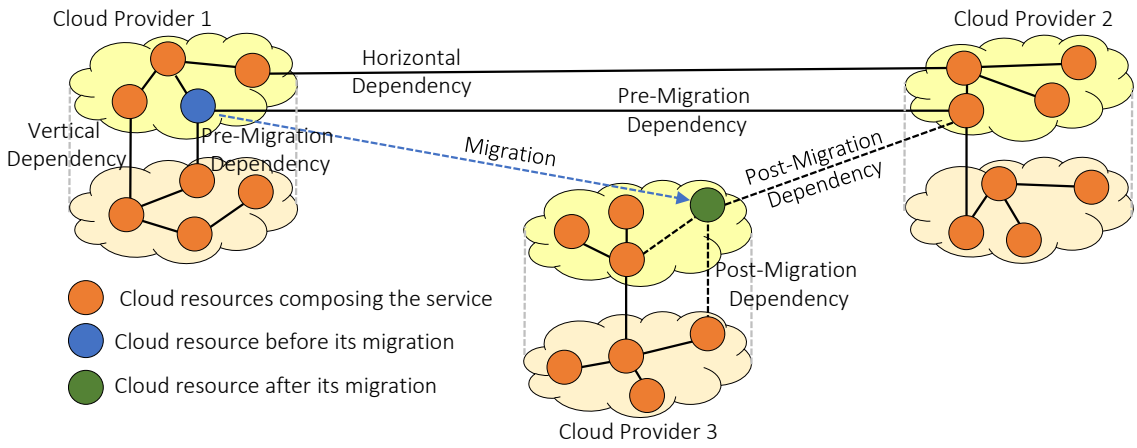


Figure 2.2: Migration of elementary components of a cloud composite service

2.3.1.1 Live and offline migration

Regarding the state of resources (running or switched off), we consider two types of resource migration: non-live migration, also known as cold migration or offline migration, and live migration, also known as hot migration. In offline migration, resources are powered off, and then all the memory and storage data migrate before the resource restarts at the destination site. This type of migration does not require keeping resources running while they are moved from the source site to the target one. Such a migration is characterized by downtime and the unavailability of services for a certain period of time. This makes it unsuitable for critical resource migration and rarely used in data centers. Furthermore, all network connection are cut during the relocation of resources; however, they are rebuilt after resuming migrated resources [21, 22, 23]. Unlike offline migration, live migration allows the continuity of services during the relocation of resources. This benefit enables seamless connectivity, avoiding service level agreement violations and allowing for optimal resource utilization. However, this type of migration is used only when the running services cannot be interrupted. Furthermore, it has to deal with many issues, mainly memory data migration and network connection continuity. In addition, storage issues must be resolved if the source and the destination sites do not share the same storage resources [21, 22, 23]. The service running in the migrated resource is significantly disrupted by offline migration. Hence, its use is limited since most applications need to run continuously. Therefore, in the next sections, we will focus on live migration, as it is widely used in several applications and well developed in the literature.

2.3.2 Objectives of live migration

Live migration aims to support performance goals by preventing the violation of the Service Level Agreement (SLA) and guaranteeing the required the Quality of Service (QoS). Itcan be performedbetween edge servers, physical hosts in the LAN network, or different data centers through theWAN [24]. It aims to achieve several objectives, such as load balancing of the work-

loads on cloud infrastructures, online maintenance of resources, and server consolidation. Load balancing in cloud infrastructures helps to improve performance by distributing the workload across a set of cooperating hosts. It allows for the avoidance of congestion on individual servers and the extension of their lifespan. On one hand, it permits to alleviate overloaded servers in order to optimize performances and guarantee the required QoS, and on the other hand it assigns new tasks to the under loaded servers in order to save energy [25, 26]. In addition, servers need operations maintenance and hardware or software updates. In some cases, servers would also be switched off in order to mitigate software crushes, perform software upgrades, or investigate hazardous or eventual faults. Hence, these servers can be replaced with new ones by migrating, seamlessly, all the resources deployed on them [27]. Live migration helps also to achieve server consolidation techniques. It consists of aggregating running resources in several servers into a reduced number of more powerful servers. Through these operations, the goal is to reduce energy consumption and use the infrastructure appropriately [28, 29]. The aforementioned objectives are common to all types of cloud infrastructures. However, cloud providers with multiple sites can go beyond and leverage live migration to attain other goals and obtain lower costs. In this context, a strategy named *Follow the Sun* proposes an approach based on migrating resources to the nearest site to the client in order to minimize the network latency. Also, the work detailed in [30] proposes an approach named *Follow the Moon* and recommends moving resources at night. Finally, techniques of live migration are widely used in federated clouds [31, 32] and hybrid ones [33, 34].

2.3.3 Types of resources migrated in the cloud

Migration techniques of cloud resources can be classified into many categories according to several criteria. We will discuss various migration techniques for memory data migration. Then we will explain the data storage migration and give the set of strategies that facilitate it. Finally, we will point out different types of migration that cover the network aspects of migration.

The service running in the migrated resource is significantly disrupted by offline migration. Hence, its use is limited since most applications need to run continuously. Therefore, in the next sections, we will focus on live migration, as it is widely used in several applications and well developed in the literature.

2.3.3.1 Memory migration

Memory data migration has a major role in the live migration process. Indeed, the memory of a running cloud resource includes CPU states and the memory states of both the guest operating system and all running processes within the cloud resource. Its process of migration is performed in three main phases: the push phase, the stop-and-copy phase, and the pull phase. As a result, all of these states are moved to the new site in real time to minimize service disruption. Therefore, several efforts have been made in order to reduce the cost of relocating cloud resources. Numerous methods and techniques have been developed to reduce resource exploitation during such processes. In this regard, we will detail the main techniques used to migrate the memory of a cloud resource, namely: pre-copy, post-copy, and hybrid-copy. We will also discuss approaches that provide algorithms to improve each technique [35]. The pre-copy method is widely used in resource live migration, and it remains the most common approach that we encounter in

the literature. This technique is performed in two main phases. Firstly, it migrates all the memory pages to the destination server in the first phase. During this operation, the memory pages that may change are called *dirty pages.* The latter are transferred in many iterations during the next phase of the process. The transfer is done until the predefined thresholds (dirty rate) of dirty pages are met; the cloud resource is then shut down on the source site. This technique is known for its consistency and robustness. In fact, during migration, the source site keeps all the data until the end of the operation [36, 37]. As a result, if the migration is incomplete due to some reason(s), for instance if VM fails or some troubles in the network happened, the resource can continue to run in the source server. This advantage, nonetheless is paired with some challenges. Indeed, when the memory pages are dirtied faster than the available network bandwidth, the dirty rate would never be satisfied and the migration process cannot be convergent such issues is known as the migration convergence problem [38, 39, 40]. The number of dirty pages remains the bottleneck of the pre-copy migration technique. Therefore, several innovations have been proposed in order to overcome this issue and optimize this technique. As a result, compression continues to be one of the important mechanisms used to improve the pre-copy migration technique [41, 42].It consists in reducing the amount of transferred data during the migration process. Indeed, memory pages are compressed into fewer bytes of data in the source host before being transferred to the destination server [43]. Moreover, several approaches like [44, 45] leverage the advantages of check pointing /Recovery and Trace/Reply technology to support pre-copy migration. It introduces a novel migration system named *CR/TR-Motion.* Furthermore, the data deduplication method is designed to speed up the migration of resources. It focuses on eliminating the duplicate and identical pages that may exist within a resource before its migration. This method is used in many approaches, such as [46, 46]. These approaches design a migration system called *Shrinker* which aims at reducing the number of duplicate pages between the migrated resource and the running resource on the destination server. In a similar vein, the authors of [47, 48, 49] present solutions that reduce redundant memory data by employing novel algorithms and combining various methods. Finally, [50] presents a dynamic page transfer reordering approach that consists of postponing the migration of frequently updated pages. This helps to lower the page retransmission during the iterative copying phase.Contrary to pre-copy techniques, post-copy consists, first and foremost, of suspending the resource cloud at the initial site. Then the whole resource is transferred to the destination site. Finally, the resource is resumed. The whole memory is copied only once in the post-copy method, which makes the migration time predictable. After restarting the resource, the rest of the pages are pushed to the destination server using the pre-paging method. Post-copy sends memory pages one at a time over the network, avoiding the duplicate transmission overhead of the pre-copy method [51]. However, it is considered unreliable, and any unexpected error during the transfer would lead to the loss of data. This makes the process of recovering the migrated resource impossible. [39, 40]. In order to improve this technique, several research initiatives like [52, 53] introduce approaches to optimize this mechanism. The table 2.2 highlights the important information related to these approaches. Other directions of research combine the strong points of the pre-copy and post-copy techniques and introduce the hybrid technique. The latter copies, first, memory data with a limited number of iterations, and then hands over the execution of the VM to the destination site. The rest of the memory pages using the mechanisms of the post-copy technique. The vast majority of the memory pages are transferred during the pre-memory phase in order to limit the

number of pages remotely accessed by the destination server. Such a method, in fact, achieves less of a trade-off between reliability and the utilized network bandwidth [54, 55, 56, 57].

2.3.3.2 Storage migration

Migration of data storage concerns the relocation of the disk image of the cloud resource from the source to the destination cloud environment. Such an operation is performed when the source and the destination environments do not share the same storage pool. It has some similarities with memory migration in that they both deal with moving data from one location to another. Hence, some techniques used in memory data migration are adopted to migrate resource disk images, such as data deduplication. Storage data migration faces a set of issues related mainly to the imbalance between the large amount of data to migrate and the available bandwidth of the network. To overcome such issues, optimization methods have been introduced in the literature. Three main strategies are proposed for VMware ESX to migrate data storage namely: Snapshotting, Dirty Block Tracking (DBT) and IO Mirroring [58]. Indeed, snap-shooting consists of creating a read-only snapshot of the desk image and transferring it to the new server. Simultaneously, all new changes that impact this disk are redirected to a new file. Then, snapshots are created and copied to the destination in many iterations. Regarding the DBT technique, it adopts the same mechanism as the pre-copy memory data migration. First, the entire disk image is transferred to the destination site. After that, the dirty blocks are tracked and transferred iteratively until the predefined threshold is satisfied. In contrast, the process of IO mirroring is based on a VMKernel data mover that consists of copying all the data in bulk. In [59] ,for instance, IO mirroring is integrated with the pre-copy memory migration mechanism to implement a live migration system named XvMotion. Another technique known as replication is being developed in several approaches in the context of migrating storage of cloud resources. It consists of two parts: bulk transfer and I/O redirection. Like IO Mirroring, this method relies on bulk transfers that move the original disk of the resource. However, the I/O redirection sends the new changes (writes) synchronously or asynchronously to the destination site. The synchronous replication guarantees consistency and helps avoid losing data. Also, in contrast to the asynchronous method, it can lead to the degradation of the application's performance. To take advantage of its robustness, the authors of [60] have introduced an approach that combines the two aforementioned ways of performing replication migration. Similarly, [44, 45] leverage the Copy-On-Write (COW) technology to store the data of a resource in two layers: base image and COW image. Moreover, data storage migration leverages the strong points of data deduplication used in the memory data migration section. It consists of cutting the migrated resource image into blocks and calculating the fingerprint of each block. Then blocks are relocated to the new site. Identifying blocks by their fingerprints allows us to transfer them once and avoid redundancies during this process [61, 62].

2.3.3.3 Networking migration

According to the scope of the migration, resources can be migrated over Local Area Network (LAN) or over a Wide Area Network (WAN). Migrations over LANs consist of performing the migration in the same datacenter. In this case, the source and destination servers share the same subnet and the same pool of storage. Because data memory is used during such operations,

this type of migration is simple and has little impact on performance. Besides, it does not require any modification concerning the storage of data or the network connectivity. Relocating resources in a WAN, on the other hand, necessitates more effort and ingenuity. In this case, each data center has its own storage pool and specific network configurations. As a result, migrating resources across a WAN necessitates using the appropriate method to relocate both data memory and data storage, as well as the appropriate method to restore network connectivity. Consequently, it requires achieving the best combination among memory migration techniques and the storage data migration mechanism, in order to accelerate the process of migration and avoid the degradation of performance [63]. Classification of the introduced migration types according to the state of the resource, the memory data migration, the disk data migration, and the network scope of migration seek, in different ways and methods to achieve performance objectives. Table 2.2 presents different approaches that deal with these strategies. In this table, we define the type of migration, either live or cold, and we determine if it is memory, storage, or network that is covered by the considered approach. We also define four levels of automation and the use of orchestration languages in these approaches. The first level (not covered) signifies that there is no use of orchestration languages to provide automation in the considered approach. The low level refers to a slight use of some features of orchestration languages, and automation is provided slightly in the approach. The medium level corresponds to the use of orchestration languages aspects to provide automation, but they still need to be improved. The highest level means the controlled aspect of the migration is fully automated, or the used orchestration language is fully leveraged.

Approach	**Type**	**Memory**	**Storage**	**Network**	**Energy Savings**		**Availability**		**Bandwidth Optimization**	
					Auto.	*Orch.*	*Auto.*	*Orch.*	*Auto.*	*Orch.*
Alshathri et al. [64]	Live	N.D.	N.D.	N.D.	◐	◐	○	○	○	○
Guazzone et al. [65]	N.D.	N.D.	N.D.	N.D.	◐	●	○	○	○	○
Cerroni et al. [66]	Live	Pre-Copy	N.D.	LAN	○	○	◐	◐	○	○
Narantuya et al. [67]	Offline & Live	N.D.	N.D.	WAN	○	○	◐	◐	○	○
Zhang et al. [68]	Live	N.D.	N.D.	WAN	○	○	○	○	◐	◐
Zhang et al. [49]	Live	Data Deduplication	N.D.	LAN	○	○	○	○	◐	◐
Al-Kiswany et al. [69]	Live	VMFlockMS	VMFlockMS	WAN	○	○	○	○	◐	◐
Hacking et al. [70]	Live	Delta Compression	N.D.	WAN	○	○	○	○	◐	○
Svard et al. [42]	Live	Delta Compression	N.D.	WAN	○	○	○	○	◐	○
Liu et al. [45]	Live	Checkpointing	N.D.	WAN	◐	○	○	○	◐	○
Riteau et al. [46]	Live	Shrinker	N.D.	WAN	○	○	○	○	◐	○
Roemer et al. [71]	Live	Compression	Compression	N.D.	◐	○	○	○	◐	○
Hines et al. [72]	Live	Post-Copy	N.D.	LAN	◐	○	◐	○	◐	○
Hines et al. [51]	Live	Post-Copy	N.D.	LAN	○	○	◐	○	◐	○
Hirofuchi et al. [39]	Live	Advanced Live Migration	N.D.	N.D.	○	○	◐	○	◐	○
Kim et al. [40]	Live	Guided-copy	N.D.	WAN	◐	○	◐	○	◐	○
Deshpande et al. [55]	Live	Hybrid	N.D.	N.D.	○	○	◐	○	◐	○
Deshpande et al. [56]	Live	Scatter-Gather	N.D.	N.D.	○	○	◐	○	◐	○
Gupta et al. [73]	Offline & Live	N.D.	N.D.	N.D.	◐	◐	◐	○	○	○
○ : Not covered	◐ : Low	◐ : Medium	● : High	N.D. : Not Documented Orch.: Orchestration Auto.: Automation						

Table 2.2: Migration types and levels of orchestration and automation in migration-based approaches.

2.4 Optimization and performance of cloud resources migration

Although resource migration enables the achievement of several performance goals, it is not without cost, particularly when the relocation of resources is performed using live migration. For this, many efforts provide techniques that deal with moving data memory and data storage to cloud resources in order to reduce costs. These techniques have shown their importance in ensuring the trade-off between the performance degradation aspect and the necessity of relocating resources. However, the operations performed during the process of migration result in the consumption of a huge amount of energy and bandwidth and may cause the unavailability of resources in some situations.

2.4.1 Power-management and energy saving

Migration of cloud resources aims to achieve many purposes such as load balancing, system maintenance, and fault tolerance; however, the process of migration leads to high energy consumption. Several approaches introduce methods for reducing power consumption and improving the trade-off between energy consumption and resource performance. In this regard, the authors of [74] present an approach that extends P-TOSCA, a proposed model and proven concept based on the TOSCA language, in order to improve energy efficiency. The objective is to extend the node description by adding another property to the template by extending the ServerProperties element and adding a new property called EnergyEfficient. The designed model is based on an intelligent agent that selects the suitable target server that has enough resources to receive migrated instances from an underutilized source server. In order to save energy, the latter is turned off. In addition, [64] proposes an algorithm that reduces the energy consumption in a datacenter. The approach leverages two characteristics: the similarity in the number of requests delivered to the cloud by the user and task migration to reduce bandwidth utilization and transmission time for file trasfert between different resources. [65] provides a framework for automatically managing the resources of cloud infrastructures in order to reduce energy consumption and conserve QoS by minimizing service level agreement violations. The core of the proposed solution remains the source manager. It monitors deployed resources and ensures respect for desired performance objectives. To achieve a trade-off between energy consumption and performance preservation, the source manager leverages virtualization technologies and control-theoretic techniques that provide a method for enabling computing systems to automatically manage performance and power consumption without human intervention. Furthermore, the solution contains the Migration Manager, a component that computes and determines the suitable placements for VMs, allowing energy savings and performance conservation. This computation is done regularly and automatically according to changes related to VMs that may occur.

2.4.2 Availability, resilience and fault tolerance

The high availability and resilience are important properties that users seek to conserve when relocating resources. Therefore, numerous approaches provide responses to avoid service interruptions that may impact the availability and increase resilience, especially during the simultaneous migration of several resources. In this regard, the authors of [66] present an approach that deals with the live migration of multiple VMs. During such an operation, the availability of VMs

must be assured, so the consistency of the network, memory, and storage should be guaranteed throughout the whole operation. The authors have shown by experiment that the parameters that quantify the performance of live migration are downtime and time of migration. These parameters tend to have opposite behaviors. To achieve a trade-off between them, a geometric model is presented. Results of solving this program have illustrated that live migration must be used when transferring multiple VMs and that a few dirty page transfer rounds are often enough to minimize the total migration time, thereby enhancing availability. In addition, [67] introduces an approach aiming to reduce the downtime of cloud migration. In fact, the authors employ graph theory to analyze VM dependency and develop a migration strategy. This approach consists of migrating VMs in groups in order to reduce downtime. These groups are defined by dependent VMs according to the data traffic between them. In order to measure the efficiency of their approach, the authors use the downtime metric.

2.4.3 Network and bandwidth optimization

The migration of cloud resources entails moving massive workloads across multiple and disparate platforms. This process requires considering the needed network resources, especially the bandwidth, in order to ensure a successful migration. In this context, several approaches come with techniques allowing for the optimization of the use of bandwidth during the relocation of resources. In fact, the authors of [68] present an approach that deals with media clouds. The latter exchange a large amount of data, resulting in massive internal traffic in DCN. Consequently, internal bandwidth becomes the bottleneck. The approach proposes a cluster-aware collaborative VM migration scheme to satisfy resource constraints and reduce the internal traffic of DCN in the media cloud. This model includes a clustering algorithm that identifies clusters and a placement algorithm that carries out the migration process. These algorithms do not migrate VMs, but they decide the objects and destinations of migration, considering the migration cost. The proposed algorithms integrate clustering placement and a dynamic migration process. Furthermore, [49] emphasizes the importance of data deduplication that optimizes the bandwidth usage, which is a technique for migrating either memory or storage data from a cloud resource. It consists of eliminating redundant memory data during migration by using several means, including Run Length Encode. The obtained results have shown that the proposed technique reduces total data transferred by 56.60%, contributing to bandwidth optimization. Similarly, [69] presents VMFlock, a solution for reducing memory size during cloud resource migration and improving network performance through deduplication techniques and the acceleration of application instantiation at the destination host. VMFlock contains a VM profiler and the VM Launch Pad to identify the VM memory area necessary to boot the VM image and launch the application.

Orchestration languages are used in cloud environments in order to achieve many performance goals. In fact, many approaches and research directions have explored the reduction of energy consumption, the optimization of the bandwidth utilization, and enhancement of the availability and resilience of resources without violating SLAs or degrading cloud application performance during migration. A number of these approaches leverage orchestration techniques, languages, and mechanisms to attain these goals. For example, in [74], a TOSCA-based efficient management model is used to reduce energy consumption during cloud resource migration. The solution presented in [65] provide improvements in energy consumption, by determining automatically

the adequate placement for VMs. Furthermore, [67] offers a framework that serves for automated migration. It relies on a strategy that plans and executes the migration of the dependent VMs in a predefined order to decrease the service downtime. Other approaches deal with the same problem by tackling the optimization of bandwidth during migration. [49, 68, 69] are approaches that present techniques and methods for optimizing memory migration while maintaining the desired level of performance.

Continuous migrations of elementary components of composite cloud applications are performed to achieve maintenance goals, load balancing, resource savings, and increased efficiency. Techniques aiming at supporting this research direction have been introduced in many approaches. However, relocating cloud resources may introduce changes to the context and environment. Such a result can lead to security flaws that may compromise the security properties of these applications. Therefore, the next chapter will be devoted to exploring and analyzing approaches that exploit a few orchestration languages capabilities in order to deal with the security of cloud resources.

2.5 Synthesis

The orchestration of cloud resources, in particular in the context of composite services, is a challenging task. This pertains particularly to several factors, namely multi-tenancy, resource sharing, and the heterogeneity of cloud environments. These services require guaranteeing continuous operation of resources regardless of the underlying platforms. Therefore, orchestration languages, such as TOSCA, HOT, and AWS CloudFormation, enable the automation of different phases of the lifecycle of these services. The growing maturity of these languages allows the composition of flexible and portable services in adequation with cloud tenant requirements. Resource migrations contribute to a better cloud management. They consist of relocating elementary resources across cloud infrastructures, in order to achieve several performance goals, such as server consolidation, traffic management, and maintenance activities. However, they may also introduce overheads to the involved stakeholders, including the migrated resources themselves, the source host, the destination host, and co-located resources. Optimization techniques covering the availability of resources, energy savings and resiliency have been introduced to improve performance aspects of resource migrations. Regardless of the conveniences provided by the orchestration languages and virtualization techniques in terms of automation, the migration of cloud resources faces multiple security threats, that may compromise one or several resources of a given cloud service. The next chapter will be dedicated to presenting different approaches that aim to secure resources, with a special focus on migrated resources.

Chapter 3

Security of cloud resource migration

Contents

3.1 Introduction

The approaches previously presented in Chapter 2 leverage orchestration languages and dynamics induced by migration of cloud resources to improve several performance goals. In addition, virtualization techniques supported by the composition of cloud resources meet the increasing demands of relocating resources across heterogeneous cloud infrastructures. This helps to successfully achieve various resource management objectives such as load balancing, energy savings, fault tolerance, and system maintenance. However, cloud composite services face several security issues that may compromise their confidentiality, availability and integrity [4].

Cloud computing faces several security issues that pose significant challenges for both consumers and cloud providers. In recent years, many threats targeting cloud-based environments have been identified. In addition, potential malicious entities, which can be either internal users, external users, or even cloud service providers, perform attacks on cloud infrastructures that may cause significant damage to these infrastructures. They exploit not only the traditional security flaws but also the features provided by cloud infrastructures such as virtualization, multi-tenancy, and resource sharing. This is exacerbated by the live of offline migrations of cloud resources across heterogeneous environments that contribute to the propagation of these attacks and the generation of new vulnerabilities.

In order to overcome these concerns, several approaches propose different solutions and strategies to deal with these different threats, in particular during migration. These security countermeasures may be exogenous or endogenous. In fact, endogenous security mechanisms are internal mechanisms applied directly to the envisaged resources. They cover the generation of lightweight resources, the analysis of the configuration of cloud resources, and the ongoing certification of those resources. Whereas, exogenous mechanisms involve applying external security countermeasures to the resource in question. They focus on virtualizing security functions, building and monitoring security chains, and leveraging encryption mechanisms to ensure reliable migration and audit of cloud platforms.

The remainder of this chapter is therefore organized as follows. First, Section 3.2 gives an overview of different security threats and concerns regarding cloud resources and the underlying platforms. Section 3.3 then details the most common attacks targeting cloud resources with a particular focus on those exploiting cloud features mainly live migration, multi-tenancy and virtualization. Section 3.4 discusses endogenous and exogenous security mechanisms used to ensure security of cloud resources, pointing out those that leverage orchestration, automation an migration aspects. Finally, Section 3.5 provides a synthesis and details different shortcomings in the existing approaches.

3.2 Cloud security issues and threats in the cloud

Cloud computing facilitates access to software, platforms, and infrastructure services. Due to its performance benefits, users and organizations transfer their applications, data, and services to cloud storage servers. Regardless of its advantages, the transformation from local to remote computing has brought numerous security problems and challenges for the consumer and the provider. In addition, numerous cloud services are provided by third parties, which raises new security threats. The cloud provider delivers its services over the Internet and uses many web technologies that raise new security concerns. These different security threats remain a serious issue that may compromise the availability, integrity, and confidentiality of cloud resources. They can lead to potential attacks on cloud platforms and infrastructures. In this regard, security threats that target cloud resources are discussed in several papers. In this section, we discuss the most important efforts that analyze the security challenges facing cloud resources.

3.2.1 Cloud storage and data security

The growth of cloud computing has brought numerous benefits, but it has also introduced new security threats to organizations storing sensitive data in the cloud. The security of data and storage systems in the cloud is constantly under attack from a variety of threats. Thus, several approaches discuss such an issue. For instance, the work detailed in [75] presents a study detailing the data security issues in cloud computing. Concerning cloud data computing and storage security, the authors investigate several issues that are associated with online services, including authentication, authorization, availability, and accountability. In order to achieve secure storage and privacy for distant computation and data storage, the authors derive, for each attribute, a set of strategies and processes. In addition, the authors of [76] cover security issues and risks in both public and private clouds. They review further security-related challenges, such as service availability, multi-tenant service issues, data storage issues, identification, and access control. Other approaches concentrate on threats and risks affecting cloud infrastructures that use and manage sensitive data in particular areas. For instance, [77] studies various data security concerns in the healthcare field. The authors draw attention to the risk associated with data centralization, which gives hackers a one-stop honeypot to steal data, intercept data in motion, and give cloud service providers ownership of the data.

3.2.2 Virtualization and networking security

Virtualization and networking in the cloud have revolutionized the way organizations operate, but they also present new security challenges. Threat actors have become increasingly sophisticated and are targeting virtual environments with the goal of compromising sensitive information and disrupting operations. These threats are extremely diversified, from malicious insiders and advanced persistent threats (APTs) to sophisticated malware and cloud-native security threats. Such security issues are detailed in-depth in several approaches in the literature. The work detailed in [78] provides a thorough analysis of security issues related to virtualization. The authors outline the fundamentals of cloud virtualization before presenting a framework for cloud system virtualization. The study covered secure system isolation and presented problems caused by weak core virtualization implementation and strong virtualization attributes. In the same way, authors of [79] examine a number of cloud security concerns, including resource sharing, virtualization and hypervisors, heterogeneity, access control and authorization, trust and secure service management, outsourcing data and applications, and data storage security and privacy. However, the authors do not provide any solutions to prevent these threats. In contrast, the approach presented in [80] draws attention to the most important vulnerabilities that are associated with hardware aspects of virtualization and common attacks and threats in this field. A Bayesian attack graph model is provided in order to evaluate the risks that are connected to the threats that have been found. An in-depth overview of the numerous countermeasures that have been proposed to mitigate the identified security flaws is presented, along with an inventory of the difficulties associated with their implementation.

3.2.3 Privacy in cloud environments

Privacy threats in the cloud refer to unauthorized access, use, disclosure, alteration, or destruction of personal or confidential information stored or processed in cloud computing environments. These threats can come from various sources, including cybercriminals, state-sponsored actors, insiders, and even cloud service providers themselves. It is important to be aware of these threats and to implement appropriate measures to protect sensitive information in the cloud. In recent years, numerous research approaches have detailed users privacy threats in cloud computing involving personal or confidential information stored or processed in cloud computing environments. These studies typically focus on access restriction, encryption, trust, and reputation aspects. In this regard, [81] presents a survey focusing on privacy and security for cloud service providers. The authors define security and privacy in cloud environments clearly and independently. They detail commonly used privacy methodologies before presenting different security aspects, including confidentiality, integrity, access control, availability, and auditing characteristics. In the same context, the authors of [82] describe various privacy strategies used in cloud computing environments and list the drawbacks of each that could pose security risks. They also provide a thorough assessment of privacy and security issues based on reputation and trust. They go into great detail about how to extend and combine various elements, including trust, reputation, access control, multi-tenancy, and hierarchical keys, to guarantee users privacy. They also suggest a strategy that aims to implement many layers of security controls to mitigate targeted attacks. In addition, [83] presents a deep analysis of cloud security concerns and pinpoints prospective threats related to the cloud computing context. In addition, it details a taxonomy of suggested solutions for data security and privacy based on a study of research publications over the past few years. The objective is to point out different security threats and risks that target privacy in cloud infrastructures.

Cloud security threats pose a significant risk to organizations as they target several critical aspects of cloud computing, including virtualization, authorization, privacy, networking, data, and storage. Virtualization vulnerabilities can be exploited to compromise the underlying infrastructure, while weaknesses in authorization can result in unauthorized access to sensitive information. Privacy concerns arise from the potential exposure of sensitive data and the lack of control over its use. Networking threats can lead to eavesdropping and man-in-the-middle attacks. Data breaches can result in the loss or theft of sensitive information, and storage threats can result in the corruption or loss of important data.

3.3 Attacks targeting cloud composite services

Cloud-based attacks can come from a variety of sources, including external attackers and internal users. The attacks can range from simple unauthorized access to complex vector attacks that exploit multiple vulnerabilities. Attackers may come from a variety of sources, including malicious actors, hacktivists, government actors, and even insiders. The type of attacker can vary depending on the motivation, resources, and skills of the attacker. They use a variety of methods to target cloud systems, including exploiting vulnerabilities in cloud software, using phishing scams to gain access to sensitive information, and conducting Distributed Denial of Service (DDoS) attacks to disrupt services. These attacks can be particularly devastating as

they allow attackers to take control of the cloud system and then move to other systems to steal sensitive information or break into them.

3.3.1 Attackers targeting cloud resources

In comparison to resources deployed in traditional infrastructures, cloud composite services have numerous security flaws, making their unavailability considerably harmful. Sharing infrastructure amongst numerous users enables the achievement of various performance objectives, but it also represents a potential threat. The authors of [84, 85] demonstrate how attackers can access the physical hosts where other nearby cloud resources are housed by taking advantage of vulnerabilities in the hypervisor architecture. In this kind of attack, a malicious user targets a resource deployed over a hypervisor. The impacted resources then negotiate a compromise with the hypervisor. The attackers then take the opportunity to intensify their attacks against all the resources that are currently using the compromised hypervisor. As a result, under this hypervisor, all resources will see gradual changes. One of the security issues in cloud computing is the availability of cloud resources. DoS and DDoS attacks on cloud infrastructures actually result in the unavailability of resources, which may have an impact on many cloud users. When these cyberattacks occur, they do more harm than they would to traditional infrastructures, since when the workload increases with respect to a specific service, the cloud environment provides additional computational power to that service [86, 87]. Another problem that affects cloud users is data leakage. Through these activities, sensitive data from many databases stored in cloud infrastructures is exfiltrated. Furthermore, attackers have the ability to engage in actions that result in account or service traffic hijacking. These types of security breaches provide malicious users with access to key parts of a service that has been deployed, thereby compromising the confidentiality, integrity, and availability of such services [88]. Attackers who take advantage of such security flaws can be divided into three categories: outsider attackers, insider and attackers.

3.3.1.1 Outsider attackers

In the area of cloud computing, an outsider attacker refers to an external entity that attempts to gain unauthorized access to cloud resources or the underlying infrastructures from outside the considered cloud environment. In order to attack a cloud service provider, a cloud customer, or a third-party supporting organization, the attacker leverages technical, operational, process, and social engineering vulnerabilities [89]. This allows for further access and the spreading of attacks against the confidentiality, integrity, and availability of the targeted resources. Using remote software and hardware and taking advantage of the human weaknesses of cloud users, external attackers can launch numerous attacks on cloud infrastructure throughout the network.

3.3.1.2 Insider attackers

An insider (internal) attacker is an entity with authorized access to cloud infrastructures. In general, these attackers can be malicious users of cloud service providers, malicious clients of cloud services, or malicious third parties with access to the providers infrastructure. Insider attackers have a significant advantage over external ones because they have authorized access to the system and may also be familiar with the network architecture and system policies and

procedures. In addition, there may be fewer security mechanisms against internal attackers because many organizations tend to focus on protection against external attacks. As a result, they use these privileges to get additional access or support third parties to conduct attacks on targeted cloud infrastructures.

Regardless of the benefits of cloud computing in terms of performance and the composition of services from several fundamental elementary resources, it is still subject to attacks and vulnerabilities that might jeopardize the security of these resources. Attacks on traditional systems target cloud infrastructures. We will discuss the most common attacks targeting cloud services and the underlying infrastructure, specifically when these resources undergo migrations.

3.3.2 Virtualization-based attacks

The software layer known as the hypervisor, which stands between cloud resources and actual hardware, is in charge of covering the underlying architecture. It is essential to ensuring the multi-tenancy and resource sharing functions of cloud computing. It allocates physical resources such as memory, CPU, and ports to the guest resources. Since they have the highest level of privilege and may execute any command from this point, hypervisors should be regarded as the layer in the cloud stack that needs to be protected the most from security threats. If an attacker modifies or compromises the hypervisor, they can gain complete control of all host system resources.

The majority of attacks on the hypervisors are performed by internal users. They begin by compromising a hosted cloud resource over the hypervisor, eventually landing on another resource or the hardware layer via the hypervisor layer. In addition, when the host operating system is compromised, it can affect I/O and networking devices. This can offer potential attackers the advantage of compromising the resources running such software. A compromised hypervisor can easily host arbitrary malicious software that allows malicious users to gain access to the whole system. In addition, the multi-tenancy feature, which enables sharing resources among multiple users, relying on virtualization capabilities, represents a real weakness for the security of cloud infrastructures. Indeed, multi-tenancy can be exploited in the form of co-tenancy, co-location, and co-residence attacks in order to gain access to valuable data located at the same physical location. In this regard, the authors of [90] conduct a deep assessment to identify existing vulnerabilities in the main virtualization platforms. Their study is based on the Xen Security Advisory (XSA), Common Vulnerabilities and Exposures (CVE), and National Vulnerability Database (NVD). They also categorize them in different groups in order to accurately distinguish those that may be exploited to perform attacks regarding cloud environments. In addition, the approach detailed in [91] presents different malware-based attacks that target the IaaS layer. The authors focus on attacks performed against virtual machines. The objective is to provide the necessary knowledge about these attacks and the different threats surrounding the actors involved in cloud environments. The work discussed in [92] draws up a detailed inventory of all the attacks that may exploit virtualization and multi-tenancy aspects of cloud infrastructures. The authors point out the impact of insecure hypervisors on the resources running over them. They also insist on security attacks related to the resources sharing, mainly the prepackaged images in open-source repositories. In fact, these images provide an easy method for deploying and restoring virtual systems efficiently and quickly across numerous cloud infrastructures. However, their integrity

is an essential security requirement for services provided by cloud computing. A compromised image may be exploited to affect other virtual resources and propagate the damage during the migrations of these resources.

3.3.3 Network-based attacks

One of the primary attack vectors for targeting resources running on cloud platforms is the network. Some of these attacks were designed after the invention of cloud computing technology, but the majority are conducted against traditional on-premise infrastructures. The malicious entities perform these attacks with the purpose of degrading the resources and services provided to the user. In this type of attack, servers and network resources are targeted by many computers, which results in a flood of messages, corrupted packets, and connection requests that cause a denial of service. These attacks overwhelm the system with requests, wasting computational power, performance time, and cryptographic operations. These attacks can even be extended and launched in a distributed manner (DDoS) from several sources to increase the damage and lower the detection of their signatures by the defense mechanisms. The impact of such attacks on cloud infrastructures is greater compared to traditional infrastructure-based systems. Attackers can use cloud features such as auto-scaling to cause more damage to compromised systems. In fact, when an attack is launched on a given cloud infrastructure, its resource consumption starts increasing very rapidly. As a result of the auto-scaling, more resources are allocated to meet these demands. This process of resource allocation continues and causes a heavy loss if the attack is not detected and mitigated. In addition, several resources that run in and share the same cloud environment are affected if one of them is compromised by an attack. Similarly, if it migrates to another infrastructure or server, it will spread and cause the same damage.

Several approaches discuss denial-of-service attacks and their impact on cloud infrastructures, with a particular focus on migration aspects. For instance, the authors of [93], show that a denial-of-service attack named the migrant attack can lead to excessive live migrations using the dynamic resource allocation features of cloud infrastructures. In fact, the attacker can vary the computational resource usage of a compromised virtual machine and trigger several migrations, even if the numbers of migrations are fixed by dynamic allocation schemes that are designed to minimize their number. The proposed method also demonstrated that even if VMs on the same deployed cloud infrastructure or physical machine are perfectly isolated via virtualization, a malicious VM can still affect the availability of the co-located VMs. In addition, the work detailed in [94] demonstrates that cloud infrastructures are vulnerable to networking attacks, including denial of service, which can lead to failures and anomalies during live migrations. The authors show that using the abortion of a TCP connection, the attackers can cause unrecoverable memory inconsistency for post-copy, leading to a significant increase in downtime and performance degradation of the running cloud resources.

3.3.4 Hardware-based attacks

The multi-tenant features of cloud computing that allow sharing resources from the same infrastructure constitute a vulnerable point that might be exploited by insider attackers. When a malicious entity gains unauthorized access to physical resources, it can retrieve and exfiltrate sensitive information and even compromise the underlying infrastructure. Attackers use general

side channel attacks to get access to such resources. These attacks permit obtaining sensitive information by observing the execution time or power consumption variations generated via cache hits and misses. Three main types of side-channel attacks can be defined based on the information that is leaked: time-driven, trace-driven, and access-driven. The most common attacks are time-driven, as they allow the capture of an important amount of information. The time-driven attack tracks the overall number of cache hits and misses by observing the aggregate profile [95]. It can be passive if the system is not directly accessible by the attackers or active if the attackers physically have access to the considered system.

3.4 Security mechanisms for orchestrated migration

Several security risks may compromise their confidentiality, availability, and integrity. In particular, these issues may occur while relocating the elementary components of cloud applications from one infrastructure to another. To highlight these security issues and the mechanisms put in place to address them, we consider Figure 3.1. This figure represents a composite cloud application distributed over three cloud providers. While the application is running, one of its elementary components has undergone a migration from one cloud provider to another while relocating the elementary components of cloud applications from one infrastructure to another. To highlight these security issues and the mechanisms put in place to address them, we consider Figure A. This figure represents a composite cloud application distributed over three cloud providers. While the application is running, one of its elementary components has undergone a migration from one cloud provider to another. This operation may lead to a security issue due to contextual configuration changes. This situation requires putting in place security mechanisms that can be either exogenous, as denoted by the blue triangles, or endogenous, as symbolized by the discontinuous green circle. In some cases and according to the level of risk, the combination of several mechanisms would be necessary to protect the cloud resource.

3.4.1 Approaches based on endogenous mechanisms

The security of cloud resources based on endogenous mechanisms refers to the application of internal mechanisms used to secure these resources. We detail different approaches to leveraging such mechanisms to protect resources from potential security flaws. For this, we will first introduce the approaches that deal with resource vulnerability management. Then we will consider solutions that propose to modify the composition of resources and keep only the necessary packages in order to reduce their attack surface. Finally, we will look at approaches that leverage certification techniques in order to guarantee the required level of trust in cloud applications.

3.4.1.1 Analysis of resource configuration

The continuous growth of cloud resources migrated across diversified and heterogeneous environments exposes them to frequent configuration changes. The migration makes the elementary components composing a given cloud service interact with new ones and with new configuration elements related to the cloud infrastructures. These changes may lead to or even generate security threats. When an elementary component is compromised, its operation, abilities, and function

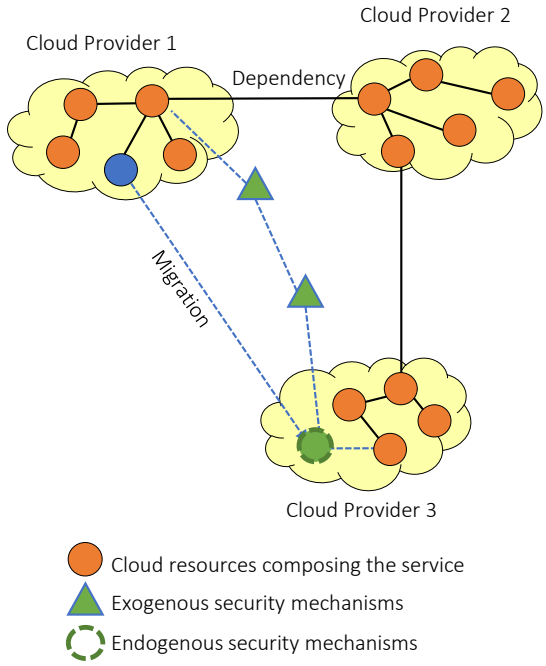


Figure 3.1: Automated security enhancement during the migration of cloud resources

become untrustworthy and eventually disabled; thus, the whole cloud service becomes compromised as well. Therefore, the continuous analysis of such configurations remains a necessary task that requires automated processes in order to identify and remediate potential vulnerabilities and security flaws that may compromise their security. In this regard, [96] proposes a vulnerability management approach that aims to support vulnerability awareness in autonomic networks. The authors leverage OVAL vulnerability descriptions and the Cfengine tool. The objective is to detect vulnerable configurations in large-scale environments based on high-level policies by integrating OVAL descriptions into the management plane. According to the considered environment, the proposed solution retrieves different vulnerability descriptions from the official OVAL repositories. Using the Cfengine tool, they are translated dynamically to security policies. The latter are distributed by the Cfengine Server to different agents that trigger alerts when a defined vulnerability state is encountered. The authors propose also using the Ovalyser in order to compute the theoretical model. It is a plug-in-based OVAL to Cfengine translator, and it operates in several steps from the introduction of the Oval vulnerability description file until the generation of the Cfengine policy. In order to demonstrate the effectiveness of their solution, the authors provide a real-world use case based on the IOS platform for Cisco devices. In this case, the framework generates Cfengine policy rules capable of analyzing and detecting vulnerabilities across such platforms, thereby increasing vulnerability awareness autonomously. Similarly, the approach detailed in [97] suggests automatically correcting known security vulnerabilities and protecting the system, formalizing the change decision problem as a satisfiability or

SAT problem. The objective of the SAT problem, which is provided with a boolean expression, is to locate an assignment for the variables in such a way that the formula will evaluate to true. The proposed framework fixes those system features that cannot change and frees those variables for which modifications are available since it specifies our vulnerability knowledge source as a propositional logical formula. The SAT solving engine is set to work in order to figure out what changes need to be made in order to make the system more secure. The authors suggest the idea of a future state as a way to provide both proactive and reactive solutions. This specifies how a system will appear when a particular change has been applied to it. These descriptions can be utilized for the purpose of conducting an analysis of the impact that modifications will have on system security without actually modifying the system. The authors of [98] propose, an approach that deals with the same issues. They conduct a deep assessment of OpenStack's vulnerability lifecycle and discover severe risk levels resulting from prolonged patch release durations. Then, they propose an approach that seeks to find new methods and strategies to mitigate vulnerabilities, including 0-Days. To mitigate these risks, they propose a solution that aims to gather and correlate information from several security knowledge sources, including malware signature repositories and bug tracking systems. In contrast to the traditional scanner and bug tracking systems that use the National Vulnerability Database (NVD) and Open Source Vulnerability Database (OSVDB) to gather information, the provided approach goes further and uses the OpenStack Launchpad Bug-tracker (OLB), Exploits Database (EDB) and Symantec Database SymDB, which increases the quality of the datasets. The proposed framework is composed of five components used to collect information from vulnerability databases, correlate, structure, and exploit them.

3.4.1.2 Generation of constrained resources

The heterogeneity of the cloud platforms, the complexity of cloud applications (dynamic, flexible, often migrated or replaced), and the growing number of cyber threats that target the security properties of cloud applications increase the risk of compromising them. To tackle this thorny problem, initiatives propose to modify the internal configuration of resources in order to protect them from potential security threats. The approach detailed in [99] proposes a solution based on the TOSCA language that aims to generate protected Unikernels. It leverages the capabilities of TOSCA in order to drive the integration and configuration of security mechanisms within cloud resources in an automatic manner, according to different security levels. The objective is to reduce the attack surface of these resources. This approach uses Unikernel techniques to create lightweight cloud resources with a small set of libraries. The adopted solution is based on two extensions: Unitosca and Sectosca. The Unitosca allows the refinement of the TOSCA level description in the context of services implemented using Unikernels, and the Sectosca allows the specification of security constraints. The proposed framework handles the creation of protected resources, then allows for the management of security mechanisms while adhering to the security requirements, and supports adaptation to contextual changes. Similarly, to ensure the security of lightweight resources, The approach detailed in [100] proposes to monitor the security of a lightweight cloud infrastructure that exploits remote attestation to verify the software integrity of cloud applications during their whole life-cycle. The proposed solution, named Docker Integrity Verification Engine (DIVE), targets the Docker container engine and covers both the host and

the services in the containers. The proposed solution focuses only on the integrity of the running cloud services on the host and in the containers. The authors use containers because they are a standard for containerizing applications on a Linux platform and are widely supported by vendors and production-oriented cloud environments. However, they consider that the proposed solution can be generalized to cover a wide spectrum of lightweight virtualization technologies. Similar to this, the authors of [101] propose a solution named Confine that aims to reduce the attack surface of Docker containers. The proposed approach uses static code analysis to inspect the containerized application and all its dependencies, thereby limiting the exposed interface of the underlying kernel that can be compromised. The proposed framework operates automatically; it takes a container image as its input and generates a customized system call policy. Containers, once initialized, run a single application for their entire execution time.

3.4.1.3 Certification of resources

Certification techniques are mechanisms that allow the non-functional properties of services, primarily security functions, to be proven. However, the application of these mechanisms to composite cloud services remains very limited, and only a few approaches have explored this research direction. These services are continuously designed, released, and deployed over heterogeneous infrastructures. Their dynamic and flexible aspects make the classical certification techniques unsuitable and hard to apply to these services; thus, guaranteeing continuous certification of such resources remains challenging and requires adequate solutions. In fact, security certification of composite services is crucial for a wide diffusion of the service paradigm in critical cloud environments where security is a required priority. In this context, the approach detailed in [102] provides a strategy that aims to evaluate a set of security properties for composite services. The proposed solution is built on Business Process Execution Language (BPEL) [103], a programming language for representing, modeling, and composing services. It also allows to specify the order in which service operations are invoked, the data that is exchanged at each step of the composition, and the conditions under which a service is selected and integrated into the business process. The authors propose a test-based certification scheme that generates a virtual certificate for a given composite service based on the security certificates of its elementary components. The proposed approach relies on five entities: a certification authority, service providers, a process owner, a service discovery, and a customer. In order to showcase the effectiveness of the proposed solution, the authors provide a use case using different components of a flight reservation service (eFlight) that allows the customers to compare different offers, book a flight, and pay for it over the Internet. The authors demonstrate through this use case that their solution guarantees a given level of security assurance for the whole composite service. These same authors have proposed a rigorous certification technique to define a trusted cloud ecosystem in [104] as a further step in their ongoing work. In contrast to conventional assurance mechanisms, which rely on static verification techniques and are therefore irrelevant in the context of composite applications, this method is an assurance technique designed to meet the needs of cloud tenants in terms of trust in cloud environments and the applications deployed within them. In this method, certification of the system is initiated in a controlled lab environment. After the service has been certified in a lab setting, it is moved to a production environment, where it must again pass a set of tests to show that it supports the same properties. The last stage consists of setting up a continuous

certification process to monitor the status of the certification process during its operation. In addition, [105] proposes a distributed strategy for composing services securely with information flow control for composite cloud services. The goal of this proposed approach is to ensure the security of continuous information flow. In fact, in the context of multi-cloud computing, which involves the interaction of several composite services, data may be processed by several service components from multiple clouds. Access control cannot detect the information leakage caused by the subsequent operations in other service components. Therefore, the proposed solution relies on model checking techniques; the elementary components of a given cloud service are first verified, and then a compositional verification procedure is executed to ensure the information flow security along with the composition of these services. The authors demonstrate that their approach can reduce the cost of verification compared to global verification.

3.4.2 Approaches based on exogenous mechanisms

In this section, we will detail approaches that provide solutions that use exogenous mechanisms to secure cloud resources. They consist of applying, externally, some mechanisms to the considered resources. For this, we will, firstly, consider approaches that outsource security functions by leveraging network and virtualization capabilities. Then, we will look at approaches that rely on building security chains. We will also look at approaches based on hardware encryption. Finally, we will go over the checks that are performed on destination platforms to determine their level of trust and, as a result, enable or disallow resource migration.

3.4.2.1 Virtualization of security functions

The continued use of third-party providers for the execution of necessary security functions is a mechanism of major importance for enhancing the security of a wide range of cloud resources and significantly reducing the attack surface of these resources. The construction of these security functions facilitates the use of the capabilities of software-defined networking and virtualization. These security functions are then implemented in the form of security middle boxes [106]. This paradigm enables security tests to be run in the cloud regardless of resource availability. Their operation is carried out in an undetectable manner to the users' benefit. As a consequence, a number of initiatives have concentrated their attention on this paradigm in order to provide support for the security of cloud computing resources. For instance, the authors of [107] propose a solution that aims to maintain a database of the overall security information of various running services in a given cloud environment. In traditional networks, administrators can make changes with relative ease. However, in cloud computing, resources are allocated on the fly, which makes performing network interventions more difficult. As a result, the proposed strategy makes use of the capabilities provided by Software-Defined Networks (SDN) and the information stored in the database in order to define and implement various security functions in an autonomous manner. The solution leverages the existing frameworks and methods in traditional networks, such as vulnerability scanning tools, intrusion detection systems (IDS), network policy enforcement, and federated identity management, in order to collect and gather information related to the security of all the running services. The proposed framework uses this information to define security policies and automatically reconfigure the network in order to meet all of the necessary

security requirements while also ensuring that resources will remain accessible even during attacks. In addition, the authors of [108] propose an access control solution that aims to build high-level security policies. The approach extends the TOSCA language in order to elaborate a security orchestrator on top of the Network Functions Virtualization (NFV) MANO orchestrator. The designed framework consists of extracting security attributes from TOSCA files, generating dynamic access control policies, and enforcing the created policy rules at the appropriate enforcement points. The solution covers distributed resources over several providers as it defines security policies for different tenant domains, which may use virtual resources (e.g., VMs, VNFs), in different geographically distributed data centers, resulting in flexible and scalable protection coverage that is across different NFV layers and multiple data centers. The architecture of the proposed framework is composed by a NFV orchestrator that builds the cloud service topology using different TOSCA files, which interact with the NFV orchestrator to provide monitoring, resource access control, and security policy enforcement to the deployed components. However, the main building block of the architecture remains the access control engine that specifies different security policies and models for different tenants. The authors develop a prototype to showcase the effectiveness of this approach. The obtained results show that the creation of an on-demand, tenant-specific, dynamic access control policy in the clouds has become possible thanks to their software-defined access control engine. Interestingly, the throughput of the security orchestrator cloud is always maintained at a satisfactory level, despite the increasing number of tenants and users. In a similar perspective, the method described in [109] suggests a security service function tree architecture that tries to consolidate several security functions in order to minimize the resource requirements for allocating virtual security functions. The approach relies on SDN and NFV capabilities for the virtualization of security services, specifically the Service Function Chain (SFC), which enables the building of an ordered list of service functions and dynamically direct network traffic through different service function paths. As a result, an SFC can be created to gradually integrate the virtual security functions and offer security services to cloud tenants. However, numerous security service function chains are necessary to guarantee the security needs of all tenants and prevent all forms of attack. As a result, the authors used the decision tree classification technique to construct a distributed model for the SecSFT that links decision rules to the associated NFV nodes. Each of the NFV nodes collects and evaluates the network flow attribute values. By comparing the network flows to the decision rules, the flows are identified and divided, and suspicious network flows are either identified and filtered at the present node or transmitted to the next nodes for more accurate division and detection.

3.4.2.2 Verification of security chains

The verification of security chains is one of the research directions developed in the literature in order to improve the security of cloud applications. In this regard, the approach described in [110, 111] proposes a strategy to support the formal verification of security chains in software-defined networks. The proposed solution leverages the virtualized aspects of security mechanisms provided by SDN coupled with verification techniques, namely Satisfiability Modulo Theories (SMT) or model checking, in order to ensure consistent security chains are outsourced and deployed in the network. The solution is extended to cover both the control and data planes of the critical security chains that combine different security functions, such as firewalls, intrusion

detection systems, and services for preventing data leakage. The proposed framework extends the Pyretic network programming language for specifying data plane configurations and verifying the control plane. It also uses Kinetic to provide the required functions for verifying the control plane and the synaptic checker in order to verify the data plane. To demonstrate the feasibility and effectiveness of the proposed solution, the authors present an automated scenario of control and data plane applied to smartphones. During the exchange of information between a device and remote destination, the security manager activates the appropriate security functions by pushing the SDN rules according to the risks and the context of the given scenario. In addition, [112] deals with the issue of identifying anomalies that could be present or arise in a single or multi-firewall system. These anomalies are a result of inconsistencies between various established filtering rules. The approach proposes a solution that also aims to analyze, cleanse, and confirm the accuracy of written firewall rules. The authors identify various anomalies that could occur in a distributed or centralized firewall system using formal and verification methods. In fact, a number of anomalies, including shadowing anomalies, correlation anomalies, generalization anomalies, redundancy anomalies, and irrelevance anomalies, may happen within a single firewall as well as amongst several firewalls. They also provided algorithms for finding discrepancies between filtering rules in two or more linked firewalls. When there are numerous firewalls, the inter-firewall anomaly discovery process should be performed on every firewall in the network path that connects any two subdomains. Users are notified when anomalies are found so they can correct them as necessary. Since the authors consider that only administrators should have the final say in these situations, the correction of the anomalies is done manually. In the same way, the authors of [113] provide a real-time verification method with the goal of ensuring network state consistency in a cloud context. To assess consistency, the scheme employs a two-stage process of network update requests and responses. The first step is to separate the network state metadata and security policies into constraint space and security space. The network request is then quickly checked against the constraint space to uncover any fraudulent requests and guarantee that the controller is maintaining the correct global network view. When the verification is successful, the scheme checks the request against the security space to make sure the flow rules that the controller sends to the switches are in compliance with the security policies. Additionally, the SDN group table adds the label to the packet header during the network update response phase in order to identify the forwarding path. To determine the correct forwarding path, the controller actively transmits a tailored packet. With little communication overhead, the technique can check the packet forwarding path and find the aberrant path.

3.4.2.3 Hardware-based encryption mechanisms

Trusted computing technologies provide hardware and software support for secure storage and software integrity protection. This paradigm aims to enforce the integrity of a system when it interacts with other ones. This paradigm, when combined with encryption mechanisms, enables the source provider who needs to perform a migration to ensure that the destination system is trustworthy. Therefore, several approaches support this research direction. For instance, the authors of [114] provide an approach designed to maintain trust when migrating cloud resources between various cloud environments. The suggested solution leverages trusted computing to strengthen the trustworthiness of the federation among heterogeneous clouds. To improve the

security of the computing environment across a wide range of computing platforms and implementations, the TC actually combines hardware and software security. The major objectives of trusted computing are to enforce a system's integrity when it interacts with other systems and to provide security that is more robust than conventional software-based security systems. The suggested technique allows for the detection of unauthorized hardware and software alterations by gathering data on their current configuration. In the same perspective, the approach in [115] provides a solution that supports trust amongst cloud platforms that rely on the Trusted Token Protocol. The goal is to perform virtual machine migrations without compromising the host cloud platform's security or user-defined trust level. The solution ensures that the user VM is only migrated to a reliable cloud platform and leverages the Trust Platform Module (TPM) capabilities to secure the user VM during migration. To avoid relying on third parties to issue certificates, the authors propose creating a platform trust credential for all involved cloud platforms. As a result, the user can specify the acceptable level of trust in the VM migration policy that the cloud provider complies with while the VM is migrated. The suggested method enables the user to audit the relocated resources and maintain some level of resource traceability, guaranteeing that resources never migrate to an unreliable, malicious, or susceptible platform. In addition, the work described in[116] offers a solution for virtual machines that use vTPM and are mostly used in private clouds. First, the authors specify different requirements that a secure migration must meet. They then suggest a vTPM structure suitable for virtual machine migrations. This structure is used to build a secure VM-vTPM migration protocol that minimizes key generation and ensures platform integrity during migration. The authors compare their strategy in depth to those of other approaches in order to demonstrate how effective it is, and they come to the conclusion that their solution offers stronger security. To demonstrate the feasibility of their strategy, they also develop a prototype based on the Xen hypervisor, and they affirm that it may be extended to other virtualization platforms.

3.4.2.4 Compliance of cloud destination platforms

Ensuring security assurance is a major issue that is challenging for cloud providers. Thus, cloud security audits and the compliance of these infrastructures with the users requirements remain of major importance. Indeed, it allows cloud consumers to be aware of the security level of these platforms. As a result, approaches provide frameworks and solutions for auditing, checking, and evaluating cloud service provider offerings prior to and after resource migration. In this context, the authors of [117] introduce a solution that aims to evaluate the compliance of cloud environments prior to migrations. The proposed approach is designed to reduce human intervention and eventually reduce cost and time by verifying compliance automatically. It also allows cloud vendors to be more transparent with cloud customers, which increases trust in cloud vendors. For this, the evidence may typically be collected from the cloud service providers to determine the level of trust with respect to the infrastructure of a given provider and its capability to comply with the expected security policies. This approach presents a tool that involves collecting information from the cloud and then transmitting it to a dedicated engine that performs the assessment. According to the output of this engine, a decision is made: either the system is compliant or not. Similarly, the work presented in [118] proposes an audit approach that aims to evaluate cloud service provider (CSP) platforms. It relies on predefined

goals (operational, strategic, and technical) and conditions that respond to user expectations in order to evaluate these infrastructures. The proposed approach insists on the assessment of destination CSP platforms before migrations, relying on some criteria that respect the predefined goals and conditions. The authors extend their solution to cover the monitoring of resources after migration in order to maintain the desired levels of assurance. The monitoring activities also allow for the consolidation of several services that enable cloud tenants to continually monitor and validate the status of security controls after migrations. They focus on particular areas of cloud operations such as security operations and processes, alerts on security incidents, and breaches of privacy for entities. In order to demonstrate the applicability of their approach, the authors provide a use case of a real cloud migration, covering the different aspects of the proposed approach. In the same vein, [119] proposes a methodology for providing security transparency through auditing called the Security Transparency and Audit Tool. The goal is to gather and examine data from cloud service providers to assess compliance with regulations and to specify corrective measures. The suggested approach takes crucial security concepts like audit (evaluation of evidence) and transparency (accessibility to information) into account, integrating them to create a unified framework. It also includes a systematic process that enables the assurance of security and transparency through a wide range of operations. The operation of the framework enables the collection and assessment of evidence regarding CSPs conformance to a predefined set of organizational requirements, imposes remedial actions to address flaws, and tracks the implementation of those actions.

3.4.3 Compatibility of cloud security approaches with orchestration, migration and automation

The approaches dealing with the security of cloud resources detailed in the previous sections cover some aspects of orchestration languages in order to provide a certain level of automation for securing cloud resources. Other ones focus their efforts on protecting these resources during their migration across heterogeneous cloud infrastructures. Table 3.1 details the level (High, Medium, Low) and the coverage of automation, orchestration, and migration in each approach. The first level (not covered) means that there is no use of orchestration languages to provide automation, and the migration is not covered in the considered approach. The low level refers to a slight use of some features of orchestration languages, and automation is provided slightly to perform migration of single components of the given composite service. The medium level corresponds to the use of orchestration languages to provide automation, but they still need to be improved. This level also covers the use of migration to relocate several components of a service. The highest level means the considered process of the migration is fully automated, or the used orchestration language is fully leveraged.

3.4.3.1 Orchestration-based approaches

Many approaches dealing with the security of cloud resources leverage orchestration languages and techniques. For instance, in [96, 97], some orchestration aspects are used to perform vulnerability assessments in autonomous networks. The authors develop their solution relying on the VMANS framework, an autonomous framework for assessing and mitigating vulnerabilities, that relies on a specific orchestration strategy to deal with the assessment process. The proposed

Approach	Nature	Category	Orch	Auto	Mig
Barrere et al. [96]	Endo Mec	Vulnerability Assessment	●	●	○
Barrere et al. [97]	Endo Mec	Vulnerability Assessment	●	●	○
Torkura et al. [98]	Endo Mec	Vulnerability Assessment	○	●	○
Anisetti et al. [102]	Endo Mec	Certification and Trust	◐	●	○
Anisetti et al. [104]	Endo Mec	Certification and Trust	◐	●	●
Kerschbaum et al. [103]	Endo Mec	Certification and Trust	◐	●	○
Ning et al. [105]	Endo Mec	Certification and Trust	◐	●	○
Compastié et al.[99]	Endo Mec	Generation of Resources	●	●	○
De Benedictis et al.[100]	Endo Mec	Generation of Resources	●	●	○
Ghavamnia et al.[101]	Endo Mec	Generation of Resources	◐	●	○
Seeber et al. [107]	Exo Mec	Network Function Virtualization	○	◐	○
Pattaranantakul et al. [108]	Exo Mec	Network Function Virtualization	●	●	○
Hurel et al. [106]	Exo Mec	Network Function Virtualization	●	●	○
Luo et al. [109]	Exo Mec	Network Function Virtualization	●	●	○
Schnepf et al. [110]	Exo Mec	Security Chaining	●	●	○
Al-Shaer et al. [112]	Exo Mec	Security Chaining	○	◐	○
Wang et al. [113]	Exo Mec	Security Chaining	◐	◐	○
Ullah et al. [117]	Exo Mec	Compliance and Checking	○	●	○
Ismail et al. [118]	Exo Mec	Compliance and Checking	○	◐	●
Mukhtar et al. [119]	Exo Mec	Compliance and Checking	○	●	●
Celesti et al. [114]	Exo Mec	Platforms and Encryption	○	◐	●
Aslam et al. [115]	Exo Mec	Platforms and Encryption	○	○	●
Danev et al.[116]	Exo Mec	Platforms and Encryption	○	○	●
Sighom et al. [120]	Exo Mec	Platforms and Encryption	○	○	●
○ : Not covered ◐ : Low ◐ : Medium ● : High Orch: Orchestration Auto: Automation Mig: Migration Endo Mec: Endogenous Mechanism Exo Mec: Exogenous Mechanism					

Table 3.1: Compatibility with automation, orchestration and migration in security approaches.

architecture operates through two main processes: the first monitors different vulnerability descriptions, while the second, which is orchestrated by the vulnerability manager, identifies and mitigates potential vulnerabilities that may compromise the security of the whole system. In addition, the work detailed in [108] extends the TOSCA orchestration language to define automatically accessible control policies at the cloud scale. The authors develop a TOSCA-based security orchestrator that allows dynamically generating access control models and policies for different tenant domains, resulting in flexible and scalable protection coverage across different NFV layers and multiple cloud environments. The evaluation of the solution shows that it achieves the desired level of automation, throughput, scalability, and adaptability regardless of the variable number of tenants and users. The authors of [99] extend the TOSCA language as well in order to create a software-defined security framework that generates lightweight components with the required packages. The objective is to drive the integration and configuration of security mechanisms within cloud resources in an automatic manner. The protected components corresponding to the different orchestrated security levels can be generated in a proactive manner and are compatible with the elasticity and on-demand properties of cloud resources. Moreover, the authors of [110] propose an automated strategy that aims to support the verification of security chains, relying mainly on SMT and model checking techniques. The proposed solution addresses both the data and control planes, where the data plane corresponds to the forwarding switches with the security functions that are currently deployed in the network, and the control plane describes the orchestration that allows the data plane to be automatically reconfigured. The architecture of the solution is composed of several building blocks, including the security manager. The latter is supported by an SDN controller that orchestrates security chains hosted in a given cloud provider's infrastructure using the Pyretic network programming language.

3.4.3.2 Automation-based approaches

The complexity and growth of cloud services make manual security management and compliance challenging. In fact, manual security might be slow to identify and remediate new threats or errors. This may expose these services to attacks or result in compliance breaches. Security automation can increase detection and remediation significantly by integrating with all applicable policies, procedures, applications, and infrastructure. In order to reduce human intervention during the execution and run-time phases of these resources, several approaches addressing the security automation features of cloud composite services are provided in the previous section. For instance, the approaches detailed in [96, 97] propose solutions that deal with vulnerability management in an autonomous manner. The objective of the authors is to ensure safe configurations and reduce the exposure of the considered systems. To do this, the proposed frameworks consist of conducting the configuration assessment automatically, identifying the presence of vulnerable states, and performing the required maintenance operations. These approaches take advantage of several tools, mainly the automation provided by Cfengine which can operate in large-scale environments. In the same area, the work detailed in [98] proposes a framework that aims to mitigate vulnerabilities. Contrary to traditional scanners that use public vulnerabilities, the proposed solution gathers and collects information from several knowledge sources, including exploit databases, malware signature repositories, and bug tracking systems. The information is then used to automatically generate plugins that exploit the current information about exploits and

unknown vulnerabilities to take adequate corrective actions and avoid compromise of the considered systems. In addition, the approach developed in [108] proposes to manage access control policies using virtualized network functions to protect resources at cloud scale. The authors of [99] also propose to automate the generation of specific cloud resources, with limited attack surface using nikernel technologies, in order to match security requirements. Aspects of automation are also addressed in the work detailed in [110]. The authors propose a solution that uses SMT techniques and model checking approaches to automatically and rigorously verify security chains in software-defined networks. They also develop various algorithms for translating security chain specifications into formal models, which are then automatically verified using SMT solving and model checking. The authors of [102, 104] consider another issue related to the continuous certification of the elementary resources of cloud composite services. In fact, the dynamic and flexible aspects of cloud composite services make the applicability of the traditional certification solution difficult and challenging. Therefore, they propose a semi-automatic assurance strategy based on certification techniques, toward the definition of transparent and trusted cloud ecosystems. The approach aims mainly to guarantee a constant behavior of resources according to the pre-defined requirements. The proposed solution allows to increase confidence amongst different stakeholders in the cloud context, by ensuring continuous verification and consistency between requirements and models, which is at the basis of the chain of trust supported by the proposed certification scheme. The solution covers the continuous certification of the overall life cycle management process, including both certificate issuing and its adaptation to address contextual changes. The work discussed in [119] provides a solution to automatically audit cloud environments. The developed solution, called Security Transparency and Audit Tool (STAT), enables the gathering and assessment of information regarding cloud service providers compliance with a predefined set of organizational requirements. When the tool detects inconsistencies, it imposes corrective measures to mitigate them and keeps track of how they are addressed.

3.4.3.3 Migration-based approaches

The approaches discussed, which deal with exogenous and endogenous security mechanisms, clearly address orchestration and automation features in order to automate the security of composite cloud services. However, the elementary components of these services are continuously subject to migrations over multiple cloud environments to guarantee performance goals. Such relocations may introduce events and changes that impact the configuration of the considered resources and the underlying environments. We are therefore focusing on approaches that point out the security of migrated resources. The authors of [102, 104] present a cost-effective strategy that ensures continuous certification of components of a cloud service composite while they undergo migrations. The proposed approaches are based on portable certification and can be used to evaluate QoS during service migration and relocation. They enable the elaboration of a trusted ecosystem for cloud resources. They include the use of certificates to ensure resource behavior. These approaches are initiated with the certification of cloud resources in a controlled environment. After the deployment of cloud services, the resources are continuously tested in order to control their behavior and maintain the validity of certificates. These security mechanisms are restricted by the operations that are allowed on the considered resources, such as the availability of security patches to be applied. This may considerably reduce the possible countermeasures

with respect to a given resource. In addition, the authors of [117, 118] present approaches that propose solutions to audit cloud environments. They aim to assess cloud service providers both before and after the migration of resources, as well as during resource operations. In order to determine the level of trust in a specific provider's infrastructure and its ability to comply with the expected security policy, evidence is often gathered from cloud service providers. In the same way, the approach presented in [119] allows to improve transparency by performing some audit activities prior to migrations. The proposed framework allows for the collection and analysis of evidence from cloud service providers in order to verify conformity with the predefined requirements. Other initiatives, such as [97] consider safe configuration methods to ensure the security of cloud resources before their migration. The approach consists of analyzing and modifying the current configuration of a given resource to maintain it in a state compliant with the security policy, based on best practices or vulnerability descriptions. This enables the identification of potential unsafe configuration states and the selection of corrective operations, if available, to modify the configuration of the concerned cloud resources.

3.5 Synthesis

Cloud composite services are exposed to multiple security attacks that may cause important damages with respect to their confidentiality, availability and integrity. In particular, it is important to design and implement security mechanisms to support the migration of resources. These security mechanisms may be endogenous or exogenous to the considered resources. Endogenous mechanisms consist in applying internal security features to secure cloud resources. They include analyzing the internal configuration of the migrated resources. Others approaches are interested in modifying the composition of resources and keeping only the necessary packages in order to reduce the attack surface. On the other hand, exogenous mechanisms rely on applying some mechanisms externally to protect the resources under consideration. They may outsource security functions by leveraging network and virtualization capabilities, and building security chains. They may also exploit hardware and encryption mechanisms, or introduce frameworks that consist of checking and auditing the destination cloud platforms to determine their level of trust. Some of these approaches leverage aspects of orchestration languages to ensure a given degree of automation. Securing cloud migrations remains a major challenge. Most of the proposed solutions do not exploit the knowledge provided by orchestration languages and their specification to automate the security of the considered cloud services. Existing approaches do not consider the protection of resources during the overall phases of relocating resources, before, during, and after the migration process across several cloud infrastructures. There is also an important lack of consideration given to the complementarity of endogenous and exogenous mechanisms to ensure the cost-effective security of these resources.

Chapter 4

SMT-based Security for Migrations in Cloud Composite Services

Contents

4.1 Introduction

Cloud infrastructures enable the building of elaborated services through the composition and configuration of multiple computing resources, such as virtual machines, network devices, and software components. These elementary resources may be deployed across different infrastructures supported by one or several cloud provider(s), and are subject to changes over time [121]. The resulting dynamics increase the complexity of management tasks and may lead to vulnerabilities that can compromise the resources, or even the entire cloud composite service. In particular, the cold and hot migrations of cloud resources are currently facilitated by recent advances in

virtualization techniques[67], allowing to transfer the resource(s) of a cloud composite service from a given provider (or a given infrastructure) to another provider (or infrastructure) [122]. This process is often motivated by performance and cost objectives with regard to the cloud properties, namely high scalability, elasticity, resource pooling, and on-demand self-service [123]. However, it may directly impact the protection of cloud services and increase their exposure to security attacks. Indeed, the different changes that affect the migrated resources and their dependencies may involuntarily generate vulnerabilities that are then exploitable by malicious users to potentially cause important damages, including loss, disclosure, and even tampering of data [124].

We propose in this chapter an automated SMT-based security framework for supporting migrations in cloud composite services, such as those orchestrated with the Topology and Orchestration Specification for Cloud Applications (TOSCA) [2]. The objective is to automate the security of cloud resources by assessing the configuration changes that affect the resources during their migration and determining adequate security countermeasures to mitigate identified vulnerabilities, leveraging Satisfiability Modulo Theories (SMT). After overviewing the considered strategy and the underlying challenges, we will describe the proposed SMT-based security framework, its main components, and their interactions for enabling security automation based on verification techniques. It relies on three main phases. The first phase establishes a projection of the resource configuration induced by the migration, considering the changes that affect the migrated resource and its context. It then assesses this configuration by exploiting the knowledge provided by vulnerability descriptions. A vulnerability description can be seen as a logical combination of tests to be performed on a given configuration, with each test associating a configuration parameter with an expected state. The third phase consists in determining potential countermeasures to be executed on the resource itself or activated in its environment to prevent the observed vulnerabilities and maintain the overall cloud composite service security.

The remainder of the chapter is therefore organized as follows. First, Section 4.2 describes the considered security framework and illustrates its operation through a concrete example. Section 4.3 then formalizes the underlying security automation based on SMT solving. Section 4.4 details the proof-of-concept prototype implemented on top of the CVC4 open-source solver, and the performance evaluation based on an extensive series of experiments. Finally, Section 4.5 provides a synthesis of the proposed approach.

4.2 Security Automation Approach

In this section, we will describe the proposed security automation approach for supporting the migration of cloud resources that may affect cloud composite services. After a brief overview of the challenges related to this automation, we will present the considered framework, detail its main building blocks, and illustrate its operation based on a practical example.

4.2.1 Automation challenges

The migration of cloud composite services is motivated by several resource optimization strategies. efficient usage of servers, or resource co-location to minimize the latency amongst cloud infrastructures. These migrations are currently leveraging advances in virtualization techniques

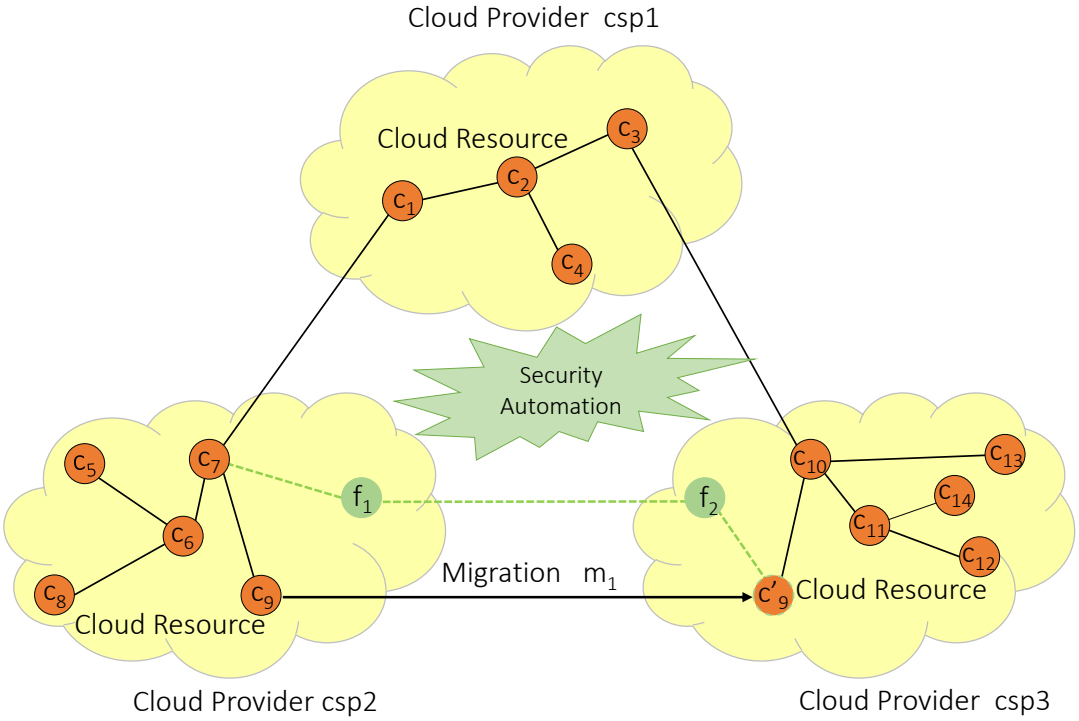


Figure 4.1: Illustrative example of security automation for supporting the migration of a resource, noted c_9, in a cloud composite service, using two exogenous security functions, noted f_1 and f_2

in order to improve runtime performance without requiring the instantiation of a new cloud resource. However, these migrations may introduce vulnerabilities and, therefore, introduce new security automation challenges in that respect. Figure 4.1 depicts the case of a cloud composite service hosted over three distinct cloud service providers, noted $csp1$, $csp2$, and $csp3$. This composite service is built from a set of elementary resources, noted c_i, and represented by orange nodes with a continuous border. For instance, the resources c_5 to c_9 are deployed over the second cloud service provider $csp2$. Let consider that the cloud resource c_9 that undergoes a migration m_1 (represented by the black arrow) from the cloud provider $csp2$ to the cloud provider $csp3$, motivated by performance considerations. After migration, the resource is represented by the orange node c'_9 with a dotted border. The considered migration supposes a contextual change for the resource. This change may increase its exposure to security attacks. It may imply dedicated countermeasures, including exogenous security functions represented in Figure 4.1 by nodes f_1 and f_2, that correspond in this case to firewalls and corresponding here to firewalls. In that context, security automation is required to support the migration of such a resource into cloud composite services. Amongst the main challenges related to this automation, it should first benefit from the specification of cloud composite services, in particular the orchestration language, which provides important knowledge on the context of the migrated resource and also on the dependencies that may exist amongst the migrated resource and the other resources that compose the service. The changes affecting the resource may impact the security of the whole

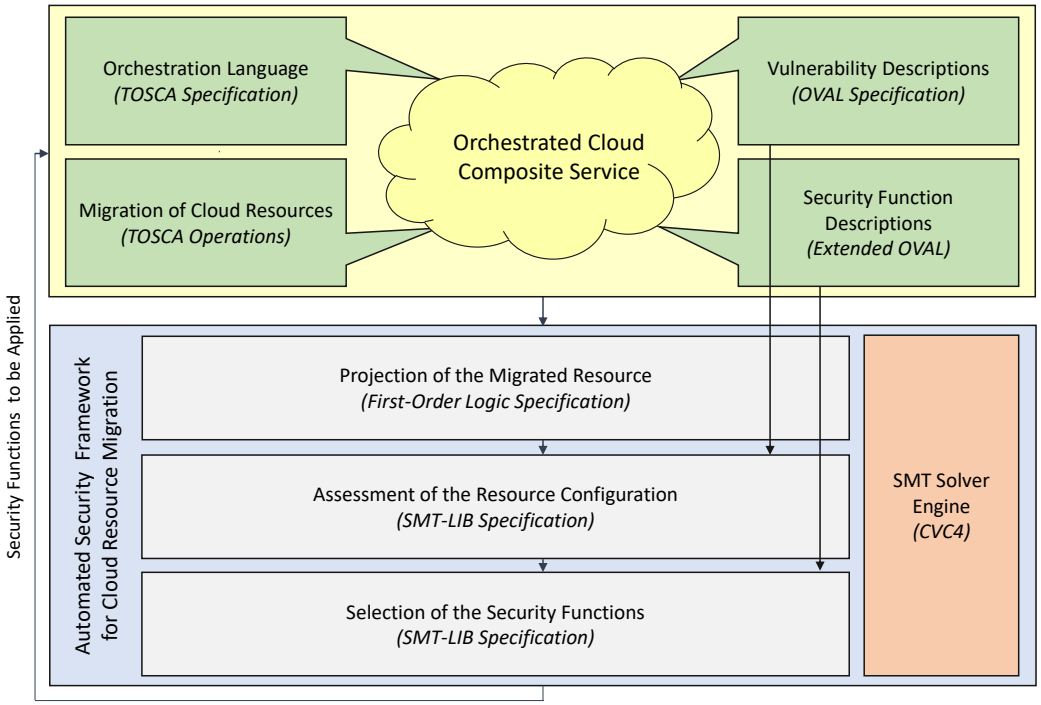


Figure 4.2: Main building blocks of the considered security framework for cloud migration

composite service. Second, this automation should exploit the knowledge sources corresponding to vulnerability descriptions in order to automatically assess the resource configuration and identify potential configuration vulnerabilities along with their severity. Third, it should take into account both the endogenous mechanisms that may be available for the considered resource (such as an update of the resource) and the exogenous mechanisms that may be supported by cloud service providers in order to reduce the attack surface.

4.2.2 Security framework for cloud migration

In order to address these automation challenges, we have designed a security framework for supporting resource migrations in cloud composite services. The different building blocks of this framework and their interactions are illustrated on Figure 4.2. The framework is triggered when a resource of the composite service undergoes a migration. This composite service, represented at the top of the figure, is orchestrated using the TOSCA orchestration language, which describes its elementary resources as well as their relationships. Concretely speaking, the TOSCA language specifies a cloud service as a TOSCA topology which defines a set of TOSCA nodes (e.g. a web micro-service node) that are interconnected by a set of TOSCA relationships (e.g. the interconnection to a MySQL database node). An extract of such a TOSCA specification with different nodes (in orange color) and relationships (in blue color) is detailed in Figure 4.4, which corresponds to a high-level representation without versions and parameters. As previously

mentioned, the TOSCA language is both open-source and characterized by high expressivity to support flexible and interoperable applications. Our framework also relies on complementary knowledge sources serving as inputs. These are vulnerability descriptions, that specify vulnerable configurations using the Open Vulnerability and Assessment Language (OVAL) [125], and security function descriptions that reuse the same specification to define the countermeasures that can be applied on the cloud resource in an endogenous or exogenous manner. The OVAL language, part of the Security Content Automation Protocol (SCAP) [126] developed by NIST, has become the de-facto standard for describing configuration and security information, such as vulnerabilities, descriptions of software configuration issues, and patches, in a machine-readable manner. Complementing Common Vulnerabilities and Exposures (CVE) definitions that only provide literal descriptions, it specifies each vulnerability as a logical combination of tests/conditions that, if observed on the target system, cause the security problem described by that vulnerability to be present on that system. This language is maintained by the OVAL Community, which works to promote the use of OVAL for improving the management of information security. OVAL provides a framework for expressing security content and a set of definitions that enable accurate and consistent representation of the content. It is supported by the OVAL community, which is working to promote the use of OVAL to improve information security management. OVAL provides a framework for expressing security content and a set of definitions that enable accurate and consistent representation of the content. As described in Figure 4.3, each OVAL definition corresponds to a criterion that logically combines a set of OVAL tests. Each OVAL test in turn examines an OVAL object (e.g. an Apache Tomcat web server) looking for a specific OVAL state (e.g. a given version for this server). Components found in the system matching the OVAL object are called OVAL items. These items are compared against the specified OVAL state in order to build the OVAL test result. The overall result for the criterion specified in the OVAL definition is built using the results of each referenced OVAL test.

Our framework is structured into three main building blocks. The first one stands for the projection of the migrated resource. The objective is to determine the new configuration of the migrated resource, as well as the other impacted resources, using the specification of the cloud composite service. This projection is performed prior to the effective migration in order to prevent the occurrence of vulnerable configurations that may expose the composite service to security attacks. The analysis of dependencies permits to identify the other resources that may be impacted by the migration, considering the TOSCA specification. The second building block corresponds to the assessment of the resource configuration based on vulnerability descriptions. The analysis relies on the projection of the migrated resource. It permits comparing the resource configuration expected after the migration to a given set of vulnerability descriptions and determining whether the projection matches one or several of the specified vulnerable configurations. The severity of vulnerabilities can also be determined based on the Common Vulnerability Scoring System (CVSS) associated with OVAL vulnerability descriptions and a quantification of the ease and impact of exploits. The third building block corresponds to the selection of countermeasures. It only applies when the assessment block has identified a vulnerable configuration. In that case, the objective is to select adequate security functions to prevent the occurrence of the vulnerability when the migration is effective. The considered countermeasures include security functions that are provided by the cloud service providers or that can be directly executed on the resource itself. We have formalized the underlying security automation using SMT solving

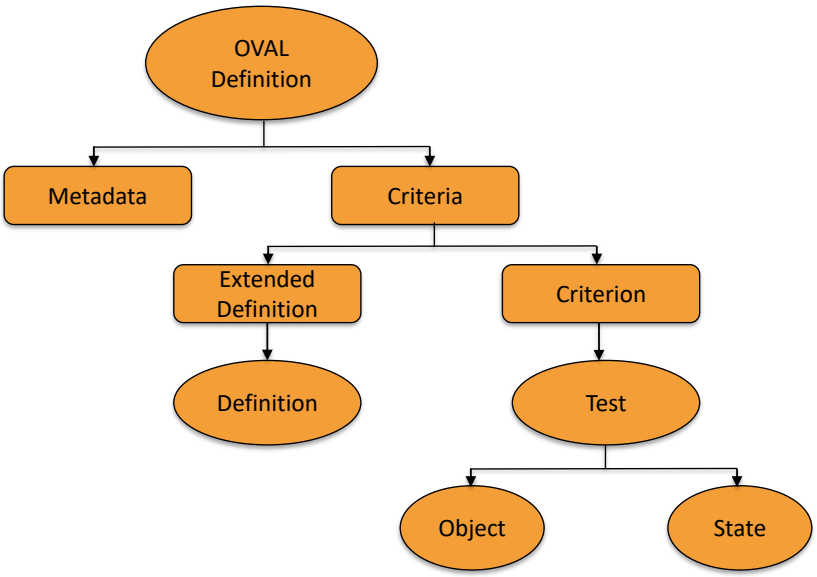


Figure 4.3: Description of the OVAL specification

to assess the resource configuration and determine adequate countermeasures. These techniques are used in formal verification and automated reasoning to determine the satisfiability of logical formulas that include variables and logical operators, as well as additional theories such as arithmetic, arrays, and uninterpreted functions. The two last building blocks exploit a SMT solver, namely the CVC4 solver, as a back-end service. According to recent benchmarking on open-source solvers [127], the CVC4 solver provides better overall performance than other SMT solvers, such as Z3 or VeriT. Such solvers operate by automatically checking the satisfiability of logical formulas that contain variables and logical operators, as well as additional theories such as arithmetic, arrays, or uninterpreted functions. The solver takes as input a formula, which is then transformed into a set of constraints that represent the relationships between the variables and the logical operators in the formula. The solver then uses various techniques, such as decision procedures, theory combination, and interpolation, to determine whether a solution exists that satisfies all the constraints. If a solution exists, the solver returns a model, which is an assignment of values to the variables that satisfies the constraints. If no solution exists, the solver returns an unsat result, which means that the formula is unsatisfiable. The operation of SMT solvers is automated, allowing for efficient and scalable verification of complex logical formulas in a wide range of application domains, such as software and hardware verification, security, and optimization.

4.2.3 Operation through an illustrative example

Let us consider a concrete example with an e-commerce cloud composite service involving a set of micro-services and backend databases, orchestrated with the TOSCA language. This com-

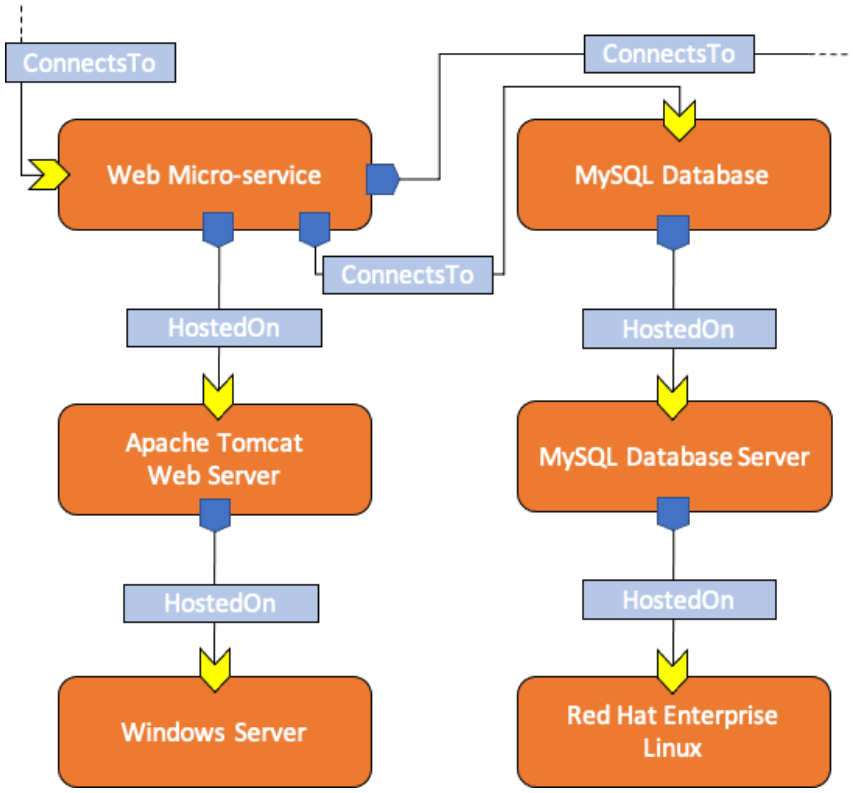


Figure 4.4: Extract of a TOSCA topology specification describing a cloud service composed of different nodes and relationships (high-level representation without versions and parameters)

posite service integrates a micro-service resource (corresponding to the c_9 node), implemented in the form of a web application built with the Struts open source framework, that extends the Java Servlet API. The micro-service interacts with other micro-services and backend MySQL databases, corresponding to nodes c_1 and c_6 in Figure 4.1. It is deployed over an Apache Tomcat web server 9.0.29 running on top of a virtualized Windows Server 2019 instance. Due to performance optimization, the micro-service undergoes a migration to a new cloud service provider that offers the same version of the operating system but supports version 9.0.18 of the Apache Tomcat server as well as version 2.3.31 of the Struts framework. Our security framework first established the projection of the migrated resource by considering the new versions of these resources. It then initiates an assessment of the projected resource configuration based on vulnerability descriptions. This assessment is automatically performed by translating the projected configuration and the vulnerability descriptions into first-order logic expressions that are then interpreted by the CVC4 solver to identify potential matches. It detects one configuration vulnerability, referring to CVE-2017-5638 for the cloud resource, and due to the new Struts framework version. This vulnerability can be exploited to perform remote code execution by allowing a remote attacker to inject operating system commands into the web application through the content-type header. As

the configuration is identified as vulnerable, the security framework then executes the selection of countermeasures, formalized as a satisfiability issue, to determine which corrective operations are possible, again using the SMT solver. Based on the security function descriptions, it identifies two potential countermeasures to address the considered vulnerability. The first one is endogenous, and corresponds to the execution of a software patch that updates the version of the Struts framework to version 2.5.26, while the second one is exogenous and corresponds to the activation of a web application firewall (WAF) with specific rules. As the cloud service provider does not allow clients to modify its WAF rules, a software patch is finally executed on the considered resource to update the Struts framework and prevent the occurrence of the vulnerability before the migration is performed.

4.3 Automation based on SMT solving

In order to support this framework, we formalize the underlying security automation based on SMT solving. The assessment of the migrated resource, and the selection of security functions to be applied, is modeled as a satisfiability problem. We consider a cloud service composed of $C = \{c_1, c_2, \dots\}$ standing for the set of resources (also called components). Each component c_i, such as a Struts micro-service, is in turn characterized by a set of properties $P = \{p_1, p_2, \dots\}$ that can be seen as unary predicates $p_i(c)$ defined for the considered component and its environment. For instance, a predicate can relate to the version of the Struts framework. These predicates permit to define the properties that the component possesses, as well as to specify the properties to be observed for the configuration assessment. We then introduce $S = \{s_1, s_2, \dots\}$ the set of component states describing in a compact manner a set of properties required to be observed over a cloud component, which can be defined as follows:

1. if $p_i \in P$, then $p_i \in S$ with $i \in \mathbb{N}$
2. if $\alpha, \beta \in S$, then $(\alpha \diamond \beta) \in S$ with $\diamond \in \{\wedge, \vee\}$
3. if $\alpha \in S$, then $(\neg\alpha) \in S$.

In the meantime, the function $state : C \rightarrow S$ takes a cloud component $c \in C$ as input and returns its current state $s \in S$. For instance, $state(c_i)$ can provide the different properties related to the Struts micro-service, such as the versions of the Struts framework and of the Apache Tomcat webserver.

4.3.1 Assessment of the migrated resource

The migration of the component c_i from a cloud service provider csp_i to another cloud service provider csp_j may lead to new configuration vulnerabilities. We introduce the following definitions related to the migration:

- $M = \{m_1, m_2, \dots\}$ denotes the set of migrations applied over a cloud composite service during its operation, they may concern one or several components c_i (such as the web micro-services in the illustrative example) of the given composite service.

- $impact : M \rightarrow S \equiv$ function that takes a migration $m \in M$ as input and returns a state $s \in S$ that projects the affected characteristics corresponding to the migration, independently from the considered component.
- $\Pi : S \times S \rightarrow S \equiv$ function that takes a projected state $s_1 \in S$ together with a component state $s_2 \in S$ as inputs and returns s_2 updated with the properties of $s1$.
- $migrate : C \times M \rightarrow S \equiv$ function that corresponds to a migration $m \in M$ applied on a component $c \in C$, and that returns the state of the component c after the operation of the given migration. It corresponds, in the illustrative example, to the migration of the web microservice from the source to the destination cloud infrastructure.

The assessment of the migrated resource is performed based on a set of vulnerability descriptions $V = \{v_1, v_2, \ldots, v_m\}$. For instance, one of these descriptions refers to the CVE-2017-5638 presented in the illustrative example. As each vulnerability description can be specified as a logical formula corresponding to a state $s \in S$, the vulnerability dataset can be seen as a disjunction of logical formulas given by $\phi = v_1 \vee v_2 \ldots \vee v_n = \bigvee(v_i)$ with $v_i \in V$. Considering these definitions, we specify an assessment function $\Phi : S \rightarrow Boolean$ that determines if a component $c \in C$ is vulnerable under V, meaning that the assessment of ϕ over the state of c is true. As a consequence, we assess the projection of the migrated resource under V using Equation 6.3, where the projection is obtained with the Π function applied to $impact(m_i)$ and $state(c)$.

$$\Phi(\Pi(impact(m_i), state(c)))with\ m_i \in M, c \in C \tag{4.1}$$

Depending on the results of this assessment, the resource migration can be effectively performed on the cloud composite service using the $migrate(c, m)$ function.

4.3.2 Selection of counter-measures

When vulnerabilities are identified, the security framework initiates the selection of counter-measures. We consider $F = F_{ex} \cup F_{en} = \{f_1, f_2, \ldots\}$, the set of available security functions, with F_{en} and F_{ex} standing respectively for the endogenous and exogenous ones. For instance, the web application firewall used to protect the Struts micro-service is specified as an exogenous security function $f \in F_{ex}$. We then consider the following definitions to support the selection:

- $future : F \rightarrow S \equiv$ function that takes a subset of security functions $F' \subset F$ as input and returns a state $s \in S$ that projects the affected characteristics after the application of the considered security functions.
- $activate : F \times C \rightarrow S \equiv$ function that corresponds to the effective application of security functions $F' \subset F$ on the component c and returns the new component state.

We then exploit the assessment function Φ to support the selection of counter-measures using Equation 4.2, where $\Pi(impact(m_i), state(c))$ provides the resource projection after migration, and the Π function is used a second time on this projection to assess the resource state after the application of the security functions using $future(F')$.

$$\Phi(\Pi(future(F'), \Pi(impact(m_i), state(c)))) \quad m_i \in M, F' \subset F, c \in C \tag{4.2}$$

Once the security functions are properly selected by the security framework, their application can be effectively performed using the $activate(F', c)$ function over the cloud composite service, in a proactive or reactive manner.

4.4 Performance evaluation

We have evaluated the performance of the proposed security framework through an extensive series of experiments. In that context, we have developed a proof-of-concept prototype written in Python, that supports the different blocks of the framework and is built on top of the open-source CVC4 SMT solver. The prototype internally generates a SMT-LIB (Satisfiability Modulo Theories LIBrary) specification, which can also be interpreted by other SMT solvers, such as Z3 and VeriT. The experiments were carried out on a standard laptop with a 2 GHz Intel Core i5 processor and 8 GB of RAM memory. We have considered an e-commerce cloud composite service orchestrated using the TOSCA language. It consists of a collection of microservices and back-end databases. The micro-services are implemented in the form of web applications built with the Struts open-source framework, which extends the Java Servlet API, and are running over Apache web servers deployed on virtualized infrastructures. Let us consider that one of these micro-services undergoes migrations across several cloud service providers that support different operating systems, namely Linux CentOS, Linux RedHat Enterprise, and Microsoft Windows Server, under different versions. The OVAL formalism is exploited for defining the tests that are used to characterize the configuration states and their properties, and for specifying the description of security functions that are applicable for the cloud resource, such as the application of patches or the activation of a web application firewall with specific security rules.

4.4.1 Different building blocks of the prototype

The developed evaluation prototype is composed of three main blocks on top of SMT solvers: the migrated resource projector, the resource configuration assessor and the security function selector. During our evaluations, we performed extensive experiments regarding several configurations; thus we provide an example of the assessment and the remediation scenarios.

4.4.1.1 Migrated Resource Projector

The first building block, called the migrated resource projector, is responsible for determining the new configuration of the migrated resource(s) as well as the other impacted resources using the specification of the cloud composite service. This projection is performed before effectively performing the migration in order to prevent the occurrence of vulnerable configurations that may expose the composite service to security attacks. Based on a first-order logic specification, this building block exploits the TOSCA specification of the cloud composite service, which describes its elementary resources and their relationships, together with the contextual changes induced by the resource migration (e.g. new environment offered by the destination cloud service provider or

by the destination cloud infrastructure), in order to infer the new configuration of the migrated resource(s). The analysis of dependencies allows for the identification of other resources that may be impacted by the considered migration.

4.4.1.2 Resource Configuration Assessor

The second block, called resource configuration assessor, is responsible for assessing the new configuration of the migrated resource(s) based on a set of OVAL vulnerability descriptions. The analysis relies on the projection of the migrated resource(s) provided by the first building block It compares the resource configuration expected after the migration corresponds to a given set of OVAL vulnerability descriptions, and it determines whether the projection matches one or several of the specified vulnerable configurations. Additionally, it is possible to quantify the severity and exploitability of identified vulnerabilities based on the Common Vulnerability ScoringSystem (CVSS) scoring associated with OVAL vulnerability descriptions. This second building block relies on a SMT solver serving as a back-end service to identify vulnerabilities using a SMT-LIB specification.

4.4.1.3 Security Function Selector

The third building block, called the security function selector, is responsible for selecting countermeasures in order to remediate detected vulnerable configurations. Based on a set of security function descriptions (expressed using the OVAL language), it aims at determining adequate countermeasures to prevent the occurrence of vulnerabilities when the migration is effective. These countermeasures, such as software patches or new internal parameters, can be directly applied to the migrated resource(s). They also may rely on security mechanisms that are hosted by the cloud service providers, such as virtualized network functions (VNF) that implement intrusion detection systems, firewalls, or data leakage prevention mechanisms. This building block exploits a SMT solver as a backend service, typically the same one as the resource configuration assessor block, in order to select security functions using a SMT-LIB specification.

4.4.1.4 Example of evaluation scenarios

In this example, we will consider two main scenarios (shown in Figure 4.5) illustrating, respectively, the assessment of a migrated resource and the selection of adequate countermeasures. We propose a concrete example with a real CVE, considering the two main cases of the assessment and the remediation of the identified vulnerability.

The first scenario, depicted in blue on Figure 4.5, illustrates the main steps of assessing the security of a migrated component for the orchestrated cloud composite service. It represents the migration of a web micro-service based on the Struts framework version 2.3.32 (denoted by c in Section 4.3), running over an Apache server version 2.4.54. During this migration, the source cloud environment supports Microsoft Windows Server 2016, while the destination cloud environment supports Microsoft Windows Server 2019. Both of them exploit the same version of the Struts framework and the same version of the Apache server. The cloud orchestrator, which manages the cloud service (*Arrow 1*) invokes our framework (*Arrow 2*) before performing the effective migration. It then establishes the projection of the migrated resource in the destination

cloud environment by collecting the necessary knowledge from the TOSCA specification of the considered cloud composite service (*Arrow 3*). Once established, the projection is sent to the assessor of the resource configuration (*Arrow 4*). This latter relies on a set of vulnerability descriptions from the official OVAL repository, which is represented by V in Section 4.3 (*Arrow 5*). The vulnerability assessment process is formalized as a satisfiability issue by considering the vulnerability descriptions together with the resource projection and generating an SMT-LIB specification. Our assessor building block then exploits the CVC4 SMT solver to interpret and solve this SMT-LIB specification and therefore determine whether the projection matches any vulnerable configurations, using the assessment function Φ. When no vulnerability is identified with regard to a known vulnerable configuration, the framework notifies the cloud orchestrator that the Struts web micro-service can be migrated, using the function $migrate(c, m)$, to the destination cloud environment (*Arrow 6*).

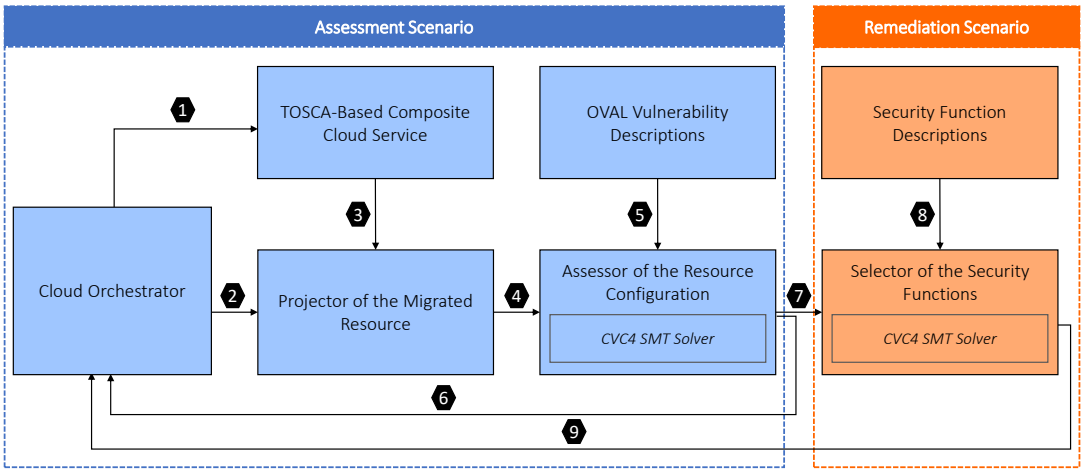


Figure 4.5: Two main evaluation scenarios corresponding respectively to the vulnerability assessment of a migrated cloud resource (in blue color) and the selection of countermeasures (in orange color) by the framework

The second main scenario, depicted in orange on Figure 4.5, illustrates the migration of another instance of the micro-service based on the Struts framework version 2.5.8. It appears that this version is characterized by a critical vulnerability, described in CVE-2017-5638 [128], allowing an attacker to execute arbitrary code remotely on a given web application. While the source cloud environment implements a firewall rule preventing this vulnerability to be exploited, the destination cloud environment does not implement any countermeasures at the time the migration is requested. First, the framework establishes the projection of the migrated resource over the destination cloud environment. After having collected related vulnerability descriptions, it starts the resource configuration assessor, which detects the previously mentioned vulnerability. The last building block is then activated for selecting adequate countermeasures (*Arrow 7*) relying on security function descriptions (*Arrow 8*). Two possible countermeasures, f_1 and f_2 are identified: the first one corresponds to the execution of an internal patch over the cloud resource, while

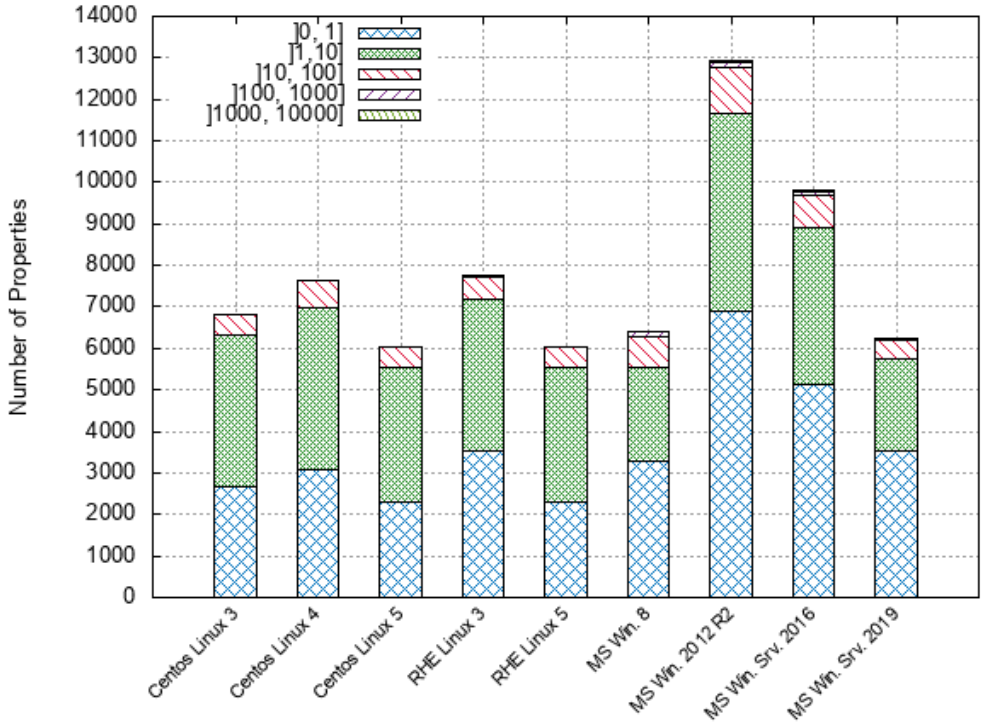


Figure 4.6: Distribution of properties specified by vulnerability descriptions for each considered vulnerability database

the second one stands for the activation of dedicated firewall rules using the ModSecurity web application firewall (WAF) in order to protect the considered cloud resource (*Arrow 9*). The activation of these mechanisms, using the function *activate* may be conditioned by several factors, such as portability, interoperability, and the operation of other resources using the same cloud platform.

4.4.2 Distribution of vulnerability properties

In a first preliminary series of experiments, we were interested in assessing the distribution of properties (or tests) amongst the considered vulnerability descriptions, with each OVAL test being associated with a given property. The objective was to quantify the redundancy of tests that are used to evaluate properties for the vulnerability datasets. For that, we have analyzed the OVAL datasets for the different operating systems and measured the occurrence of tests for each of them. Figure 4.6 describes this distribution per operating system, with the horizontal axis indicating the system and its version and the vertical axis corresponding to the number of properties. We considered five different occurrence categories using a logarithmic scale, with tests that are present respectively once, up to 10, up to 100, up to 1000, and up to 10000 times

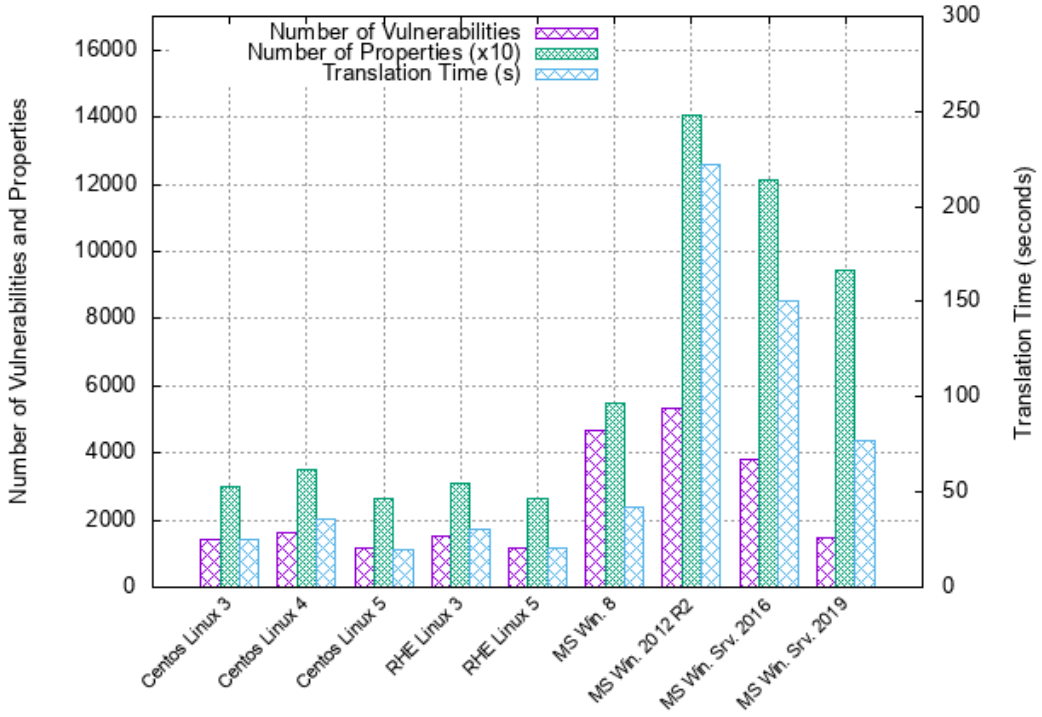


Figure 4.7: Translation time of vulnerability descriptions into logical specifications for each considered vulnerability dataset

in a vulnerability dataset. We have observed that a large proportion of tests (or properties) are present between 2 and 10 times, representing on average 44,15% for the different datasets. The other highest categories are less significant, with around 8,22% of tests that occur between 11 and 100 times, and less than 1% on average for tests occurring up to 1000 and 10000 times. The distributions are relatively homogeneous across operating systems, with the Microsoft Windows operating system having the highest number of occurrences. This distribution analysis shows that more than half of the tests or properties (around 53% on average) occur more than once in the different datasets, arguing in favor of our solution based on SMT solving, which is usually efficient in simplifying such satisfiability problems with redundant constraints.

4.4.3 Performance of migrated resource assessment

In the next series of experiments, we wanted to quantify the time required for translating the vulnerability descriptions into logical specifications and for assessing the projection of the cloud-migrated resource with respect to the different vulnerability datasets. Figure 4.7 describes the time required by the security framework to automatically translate a vulnerability dataset V into a logical specification with a conjunctive normal form that is then directly interpretable by the SMT solver. Our framework generates a conjunctive normal form by construction in order to prevent any additional pre-processing from being performed by the solver afterwards.

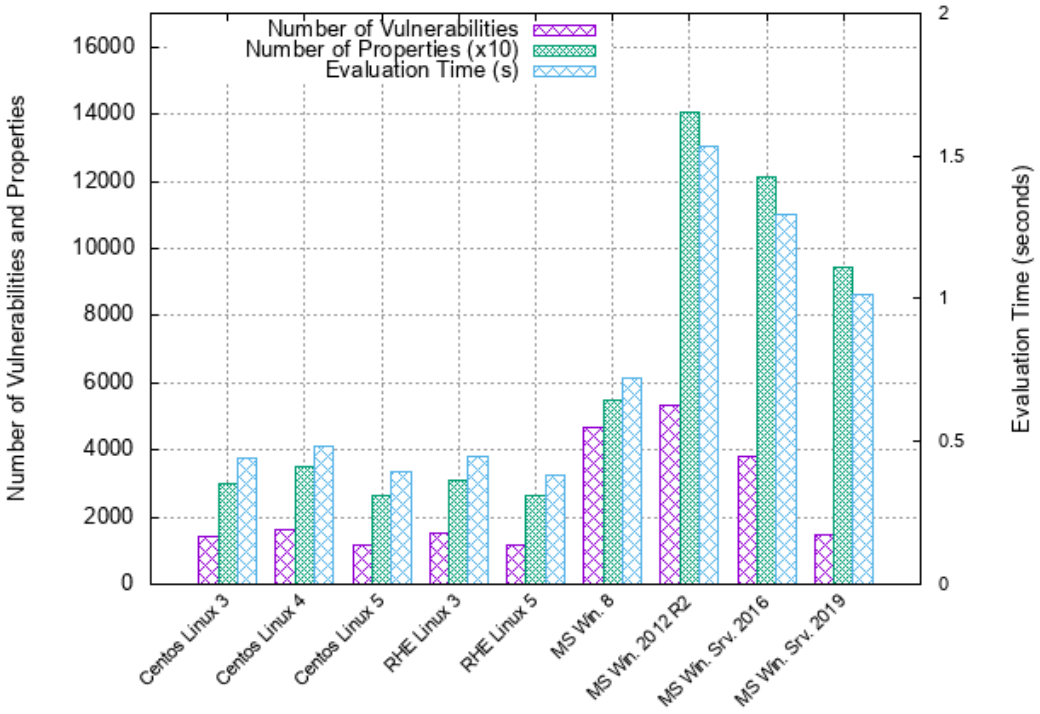


Figure 4.8: Evaluation time for assessing the projection of the migrated cloud resource with respect to each considered dataset

The figure details the translation time for each operating system dataset with regard to the number of vulnerabilities and properties. We can observe that this translation time remains below 225 seconds, corresponding to the largest dataset with Microsoft Windows 2012R2. It is mainly dependent on the number of properties, which relates itself to the number of vulnerability descriptions.

Figure 4.8 depicts the evaluation time required for assessing the projection of the migrated resource with respect to a given vulnerability dataset. The time includes the interpretation of the SMT-LIB specification together with the solving of the satisfiability issue. We can observe a linear behavior with respect to the number of tests (or properties). The minimal evaluation time has been obtained with RedHat Entreprise Linux 5, which corresponds to the smallest dataset, while the maximal time corresponds again to Microsoft Windows 2012R2, which is the largest dataset. The evaluation time is clearly smaller than the translation time and is kept under 1,6 seconds for all the vulnerability datasets, confirming the feasibility of our approach. The translation activity should rather be performed proactively, as soon as new vulnerability descriptions are published.

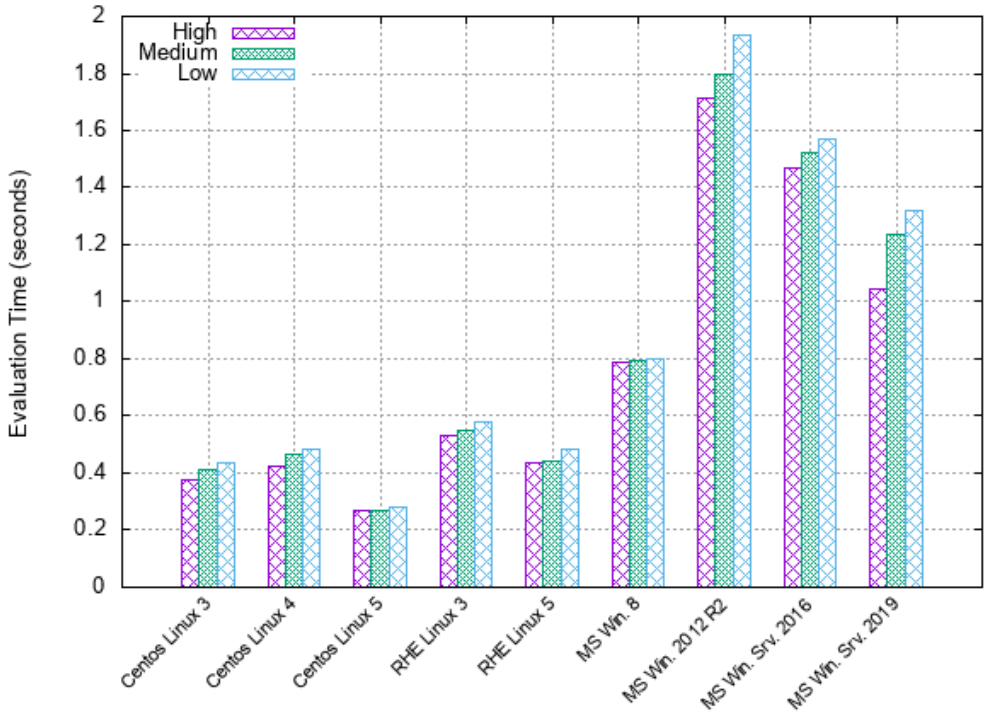


Figure 4.9: Evaluation time for selecting a security function depending on the frequency of impacted properties

4.4.4 Performance of counter-measure selection

In our last series of experiments, we wanted to quantify the assessment time required for selecting a given security function (or a set of security functions). In particular, we have looked at the frequency of the properties that are impacted by a given security function and how that influences the assessment time to determine whether the new configuration of the migrated resource is vulnerable or not with regard to this security function. This frequency corresponds to the number of occurrences of a given property in the considered dataset. In that context, we have considered three different scenarios, corresponding to security functions that impact respectively the most frequent properties (high scenario), the less frequent properties (low scenario), and properties with an averaged frequency (medium scenario) in the considered vulnerability datasets, as illustrated on Figure 4.9 describing the assessment time for selecting a security function with the different vulnerability datasets. These scenarios have been chosen to guarantee a homogeneous distribution of those considered amongst the less frequent, moderately frequent, and most frequent categories.

We expected that the security functions corresponding to the scenario with the highest frequencies would provide the best assessment time for selecting security functions, which was

Table 4.1: Assessment and selection overhead considering different live migration baseline scenarios with a 1024 MB virtual machine

Migr. Link	Migr. Time	Assess. Overhead	Select. Overhead
100 Mbps	277,2s	0,25%	0,30%
1 Gbps	31,6s	2,14%	2,56%
10 Gbps	7,7s	8,22%	9,74%

indeed the case in our experiments performed with the different vulnerability datasets, when considering various cloud resource migrations. In addition, we can observe the influence of the dataset size. In particular, the differences amongst the three scenarios are the highest for the largest datasets with regard to the evaluation time. For instance, this difference is less than 8% for the smallest vulnerability datasets, while it reaches around 15% for the largest vulnerability datasets. The selection of security functions might therefore be prioritized with respect to the frequency of properties that are impacted by such functions, in addition to other performance criteria such as the cost induced by a given security function provided by a cloud provider. Considering the modeling and results presented in [129] on live migration times with a virtual machine of 1024 MB and different migration link speeds (from 100 Mbps to 10 Gbps), we quantified the average overhead for both the assessment and selection phases in comparison to these different baseline scenarios. While using a regular laptop, we observed overheads of less than 10%, as shown in Table 4.1 which depicts on average the migration times and overheads. This means that the time required for assessing a migrated resource and selecting the countermeasures has a low impact on the overall time for performing the live migration.

4.5 Synthesis

We have proposed a SMT-based security framework for supporting vulnerability management during the migration of resources in cloud composite services. It bridges the gap between the orchestration language that specifies the cloud composite services, and the vulnerability descriptions that define the configuration states that should be prevented for the migrated resource. We have described the different building blocks of the framework and their interactions. They operate according to three main phases, namely the projection of the migrated resource, the assessment of the corresponding configuration based on vulnerability datasets, and the selection of adequate security functions when this configuration appears to be vulnerable. We have shown the framework applicability based on an illustrative example corresponding to a Struts micro-service, part of a cloud composite service, undergoing a migration amongst cloud service providers. This security automation has been formalized using SMT solving and implemented into a proof-of-concept prototype written in Python on top of the CVC4 solver. Finally, we have performed a large series of experiments based on existing vulnerability descriptions coming from the official OVAL repository, and corresponding to different operating systems. This enables us to evaluate the benefits and limits of our framework with respect to various aspects, such as the formalization into the SMT-LIB format interpretable by the solver, the assessment time of the

projection of the migrated resource, and the automated selection of security functions that can be both endogenous and exogenous to the cloud resource. These different results compared to live migration baseline scenarios confirm the feasibility of our proposed solution. In the next chapter, we will study the possibility of leveraging the developed framework in distributed environments with multiple cloud service providers. The goal is to propose an inter-cloud trusted third party that ensures secure migrations amongst multiple service providers.

Chapter 5

Trusted Third Party for Configuration Security in Orchestrated Cloud Services

Contents

5.1 Introduction

Cloud resource migration may result in security issues that may compromise the security of these resources. In fact, cloud infrastructures may typically be characterized by heterogeneous configurations, such as the versions of virtualization and hosting platforms, the nature of the other services interacting with resources, and the security mechanisms implemented by the cloud provider. The configuration changes induced by the migration of cloud resources may introduce

new vulnerabilities that can be exploited by malicious entities to compromise the migrated resource or even the whole cloud composite service. The previous Chapter was dedicated to an approach that aims to automate the security of these services by relying on SMT techniques and leveraging OVAL vulnerability datasets.Therefore, it is important to exploit the previous framework and exploit the maturity of orchestration languages such as TOSCA in order to provide a trusted third party that guarantees secure migration amongst multiple cloud providers.

In addition, while several initiatives have been taken to increase the interoperability amongst cloud infrastructures and prevent vendor lock-in problems [130, 131], the stakeholders may be reluctant to share information related to security amongst each other. Important efforts have been performed to design and implement inter-cloud security solutions, but they are mainly focused on federated access control policies and identity management [132]. Our objective is to address configuration security by enabling the sharing of configuration information amongst cloud tenants and cloud providers, in order to properly assess and remediate vulnerabilities that may occur during the migration of cloud resources.

We propose in this chapter, an inter-cloud trusted third-party approach, called C3S-TTP[1], for supporting configuration security in Topology and Orchestration Specification for Cloud Applications (TOSCA) based cloud services, whose elementary resources may be subject to migration over time. The use of third parties have already shown its efficiency in several areas, such as the case of certification authorities. These third parties have proven to be highly effective for storing, signing, and issuing digital certificates. In our case, the migration of a cloud resource leads to configuration changes that may introduce new vulnerabilities that are then exploitable to perform security attacks. Analyzing configuration changes is challenged by the incompleteness and inconsistency of configuration information shared amongst stakeholders, namely cloud tenants and cloud providers. Maintaining the security of cloud services during the migration of their resources requires not only threat intelligence but also increased transparency with respect to configurations before performing these migrations. This knowledge sharing contributes to better security management, with the preparation of an adequate destination environment from the cloud provider side (i.e. activation of specific security rules) and the optimal hardening of the migrated resource from the cloud tenant side (i.e. application of a security patch). Cloud stakeholders, on the other hand, generally avoid sharing detailed technical configuration information about their resources and infrastructures, especially those related to or affecting security. Such disclosure may also be exploited by malicious stakeholders to initiate and perform security attacks against cloud services by facilitating reconnaissance activities with respect to vulnerabilities and existing security mechanisms.

Our solution aims at preventing or restricting such configuration information disclosure amongst cloud tenants and cloud providers while managing vulnerabilities that may occur during the migration of cloud resources. It relies on an inter-cloud trusted third-party architecture that exploits a third-party stakeholder for sharing configuration information amongst cloud tenants and cloud providers using an extended version of the TOSCA language, and assessing potential vulnerabilities. This trusted third party is triggered as soon as a cloud tenant requires the migration of one or more of its cloud resources. First, it collects configuration information related to the cloud resources that are candidates for migration. It then requests that cloud service

[1]Composite Cloud Configuration Security-Trusted Third Party

providers get configuration information regarding potential hosting environments, including the resource versioning and parameterization of these cloud environments, as well as information regarding security mechanisms that may be implemented by the cloud providers. Considering this overall knowledge and using our Open Vulnerability and Assessment Language (OVAL)-based vulnerability assessment framework detailed in [133], the third party determines configuration vulnerabilities that may appear during the migration of resources over potential hosting environments. These assessment results enable the trusted third party to recommend or decline the migration of a given resource back to the cloud tenant while reducing the disclosure of configuration information amongst involved cloud tenants and cloud providers.

The remainder of the chapter is organized as follows. Section 5.2 gives an overview of the background of inter-cloud management and security aspects. Section 5.3 describes the considered inter-cloud trusted-third-party architecture by detailing its main building blocks and the interactions amongst them to secure resource migrations. Section 5.4 depicts the proof-of-concept prototype implementing the proposed architecture on top of a configuration assessment framework based on a SMT solver engine, as well as providing the performance evaluation of our solution based on extensive experiments. Finally, Section 5.5 gives a synthesis of the approach and the main obtained results.

5.2 Background

While they are facilitated by recent advances in virtualization techniques, inter-cloud migrations are facing major issues in terms of interoperability and security management. In that context, several initiatives have been taken to facilitate the sharing and transfer of resources and data amongst distributed and heterogeneous cloud hosting platforms. Currently, the migration of resources amongst different cloud providers still often implies substantial costs, legal constraints, or even voluntary technical incompatibilities [134] that prevent an efficient management of cloud resources. The portability and interoperability properties are key enablers for the integration of these resources and the elaboration of cloud composite services. Research efforts, such as [135], contribute to the development of these properties by encouraging the collaboration amongst different cloud vendors. The authors, in particular, propose an inter-cloud framework that enables interoperability among heterogeneous cloud environments with the goal of dispatching and optimizing workloads to the most effective clouds available at run-time. The framework aims at preserving the required quality of service in accordance with the service level agreements (SLAs) established with a given cloud tenant that may own cloud composite services distributed over multiple cloud providers. Important approaches have also focused on semantic considerations, based on the statement that new cloud service providers have been introduced to the market with unique and proprietary application programming interfaces (API) for migration and collaboration amongst services. For instance, the authors of [136] have identified and structured common data models to improve the interoperability amongst different Infrastructure-as-a-Service (IaaS) providers and highlighted the benefits of standardizing unified APIs through service semantics in order to leverage the migration of cloud resources.

Managing security in the context of distributed and heterogeneous cloud services has also led to several inter-cloud solutions, particularly to address challenges related to identity man-

agement as well as authorization and access control. For instance, the authors of [137] introduce an inter-cloud identity management infrastructure for cloud environments. When a cloud user performs a request for a resource that cannot be directly handled by its home cloud or when a given cloud requires external resources to balance its workload, the home cloud initiates an authentication process using a trusted identity provider. This latter asserts that the user, which can be either a person or a software entity, has been authenticated and determines his rights. Once the identity has been verified, a specific request for an external resource is forwarded to the so-called foreign cloud. Similarly, the authors of [138, 139] describe basic models and architecture patterns for supporting federated access control in heterogeneous inter-cloud environments. They begin with an inter-cloud operation framework to enable resource integration and interoperability across multiple domains and heterogeneous cloud environments. They then extend it with a security layer to provide access control features with respect to these resources. Single Sign On (SSO) approaches, such as [139], have been designed and implemented for supporting more specifically cloud federation. The authors of [140] have extended a framework to model and evaluate cloud infrastructures with SSO features, considering multiple identity providers and cloud providers. They have also taken into account the security of data that is transferred amongst different entities in the cloud federation during the SSO mechanism and highlighted the benefits in large-scale environments. Ontology-based frameworks, such as the Security Ontology for the Inter-Cloud (SOFIC) solution [141], have been investigated with the objective of formally describing the security constraints that are expected to be specified in inter-cloud environments. They support the interoperability of security mechanisms based on security standards. The extensibility and adaptability enabled by such ontology-based frameworks are evaluated based on different inter-cloud scenarios. These different inter-cloud approaches contribute to improving security management in distributed and heterogeneous cloud infrastructures owned by different cloud providers. They are mainly focused on identity and access control considerations, while our approach is complementary by targeting configuration vulnerabilities that may occur independently from the implemented access control mechanisms.

Several other approaches are contributing to securing the migration of cloud resources. They may leverage security mechanisms that are endogenous to the resources, such as applying a patch, or security mechanisms that are exogenous to the resources, such as activating specific firewall rules. In order to reduce the attack surface of cloud resources, the authors of [99] propose an endogenous approach based on the generation of protected Unikernel images, which correspond to lightweight virtual machines containing only the strictly necessary packages and libraries. We also started to work on securing composite services based on vulnerability descriptions expressed with the OVAL language [97], but without considering information disclosure issues [133]. Certification techniques, such as [102, 104], have been considered for guaranteeing the behavior of cloud resources. They consist in checking the validity of certificates at run-time through a continuous verification of the resource correctness on the basis of certification activities against real and synthetic execution traces. Some other efforts are more endogenous to cloud resources. Solutions, such as [117, 118], aim at auditing cloud environments before the migration of considered resources. For this, they rely on the evidence collected from the cloud service providers to determine the level of trust with respect to the infrastructure of a given provider and quantify its capability to comply with the expected security policy. The authors of [142] also evaluate the risk associated with each cloud environment based on a quantification and classification of vul-

nerabilities inherent to the environment. Approaches, such as [110], target orchestrating security chains, including firewalls and intrusion detection systems, in order to protect cloud resources. For instance, formal verification methods are exploited to check these chains and detect any potential inconsistencies and redundancies that may occur in their rules. Complementary generation methods enable automatically synthesizing such security chains through the analysis of the network traffic related to these resources and the inference of behavioral models to characterize them.More generally, efforts such as [114, 115] exploit trusted computing methods to provide hardware-based, security-related functions in cloud infrastructures. For instance, they target integrating software trusted platform modules (TPM) into hypervisor environments in order to make TPM functions available to virtual machines. Also, we have shown in [143] how risk management methods can be applied to cloud infrastructures in order to automatically determine the counter-measures to be applied during resource migrations. Most of the existing techniques for inter-cloud security management are centered on identity management and access control and do not exploit the knowledge provided by the specification of cloud composite services. This may be explained by the fact that cloud stakeholders, including cloud tenants and cloud providers, may be reluctant to provide exhaustive configuration information regarding their resources and infrastructures.

Along with these approaches, commercial cloud service providers, such Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), offer dedicated security solutions to safeguard the cloud resources deployed in their environments. For instance, AWS offers an identity and access management (IAM) service for user identification and access control, in addition to supporting several compliance certifications, including Payment Card Industry Data Security Standard (PCI DSS) [144]. It also provides security groups to specify network firewall rules related to EC2 virtual machine instances [145]. Microsoft Azure also offers several security services, such as Azure Active Directory (AD) for identity management [146], and supports several compliance certifications, including the one related to the Health Insurance Portability and Accountability Act (HIPAA). Similarly, GCP manages identities based on a Google Cloud Identity and Access Management (IAM) service, supports major compliance certifications, and exploits artificial intelligence for security detection, with the Google Cloud Security AI Workbench [147].

5.3 C3S-TTP Third-Party Approach

We will describe in this section our trusted third-party approach for supporting composite cloud configuration security, during the migration of resources. The objective is to exploit a third party responsible for analyzing the configuration changes induced by migrated resources and preventing vulnerabilities in cloud composite services. We will detail the different building blocks of the underlying architecture as well as their interactions. In particular, we will show how the TOSCA orchestration language can be extended to support these interactions. We will also formalize the assessment process performed by the trusted third party by considering vulnerability descriptions extracted from the official OVAL repository and configuration information provided by the considered cloud tenants and providers.

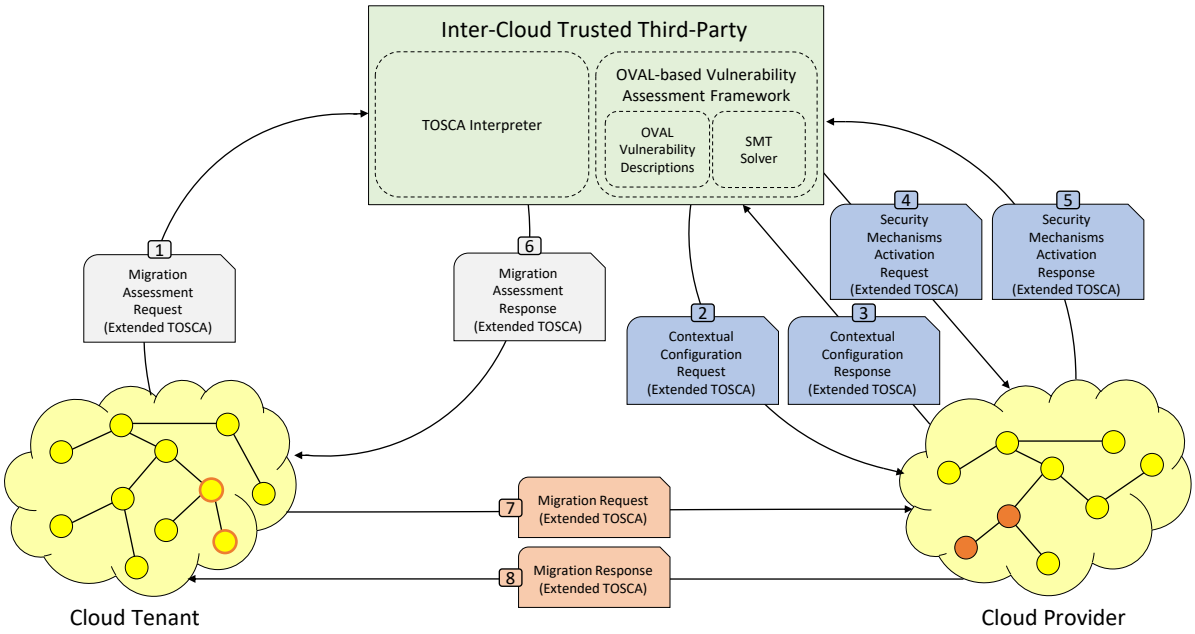


Figure 5.1: Main building blocks of our inter-cloud trusted third-party architecture

5.3.1 Architecture and Main Building-Blocks

The architecture of our C3S-TTP third-party approach is described in Figure 5.1, with the different involved building blocks and their interactions. It mainly relies on an inter-cloud trusted third party (top part of the figure) to support the interactions amongst the cloud tenant (left part of the figure) and the cloud provider (right part of the figure) during the migration of cloud resources. Migrating a cloud resource across different providers is facilitated by the recent advances in the virtualization area but poses important security challenges, in particular with respect to configuration vulnerabilities. These latter may potentially lead to a wide spectrum of negative and unwanted issues, such as instability, unavailability, and confidentiality problems. Usually, the risk level of a system is based on three main combined factors, namely the potentiality of a threat, the exposure of the system to such a threat, and the impact that a successful attack related to this threat may have on that system. The exposure of the system is in turn directly related to the vulnerabilities present in such a system. Preventing security attacks requires minimizing configuration vulnerabilities in cloud composite services. In the meantime, this vulnerability management activity itself should not disclose configuration information that may increase risks.

In a regular scenario, without considering a trusted third party, the migration of cloud resources requires the sharing of configuration information between the cloud tenant and the potential cloud providers in order to perform such vulnerability assessments, typically prior to the migration operations. The cloud tenant requires detailed configuration information regarding the hosting environment given by the cloud providers, including information about the security mechanisms that may be provided or implemented by them. In the same manner, if this as-

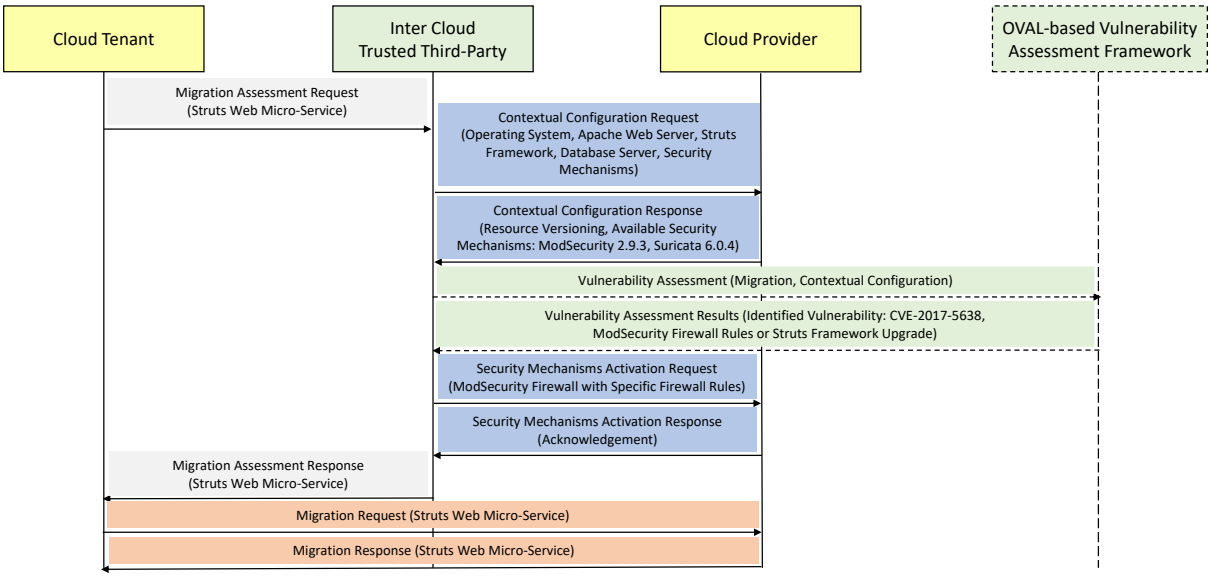


Figure 5.2: Sequence diagram of the TOSCA-based interactions amongst the architectural building blocks

sessment is done more by the provider, the cloud tenant should provide detailed configuration information about the cloud composite service and its resources in order to enable an efficient assessment. The lack of this detailed knowledge, i.e. having only a partial view of configurations, may typically have two main consequences. The first consequence is not properly controlling the attack surface of cloud services, which may result in vulnerabilities. The second consequence is a high level of security overload caused by provisioning in a preventive manner with security mechanisms that do not contribute to securing the services. For instance, we may activate additional firewall rules that are not required by a given resource in its current configuration.

We introduce an inter-cloud trusted third party to support the interactions between the cloud tenant and the cloud provider(s), during the migration of resources in TOSCA-based cloud composite services. TOSCA constitutes an open-source cloud orchestration language, in comparison to proprietary solutions such as AWS CloudFormation [148], and is characterized by a more advanced expressivity than OpenStack HOT [149], in order to build and operate flexible and interoperable services. A cloud service is described by this language as a composition (also called topology) of resources (also called nodes) that are interconnected among them through links (also called relationships). An illustration of such a topology is given in yellow on the left part of the figure 5.1, with a topology composed of more than 10 nodes. It is then possible to specify orchestration procedures over this topology, such asstarting, shutting down a node, or changing a relationship. Each element (node or relationship) derives benefits from inheritance and template mechanisms offered by the language. Such specifications, owned by the cloud tenant, provide detailed configuration knowledge regarding the resources of the cloud composite service and their dependencies. This knowledge should be taken into account to prevent vulnerabilities during resource migrations, but the cloud tenant may refuse to share it with potential cloud providers

due to security and intellectual property constraints. The objective of the trusted third party is to facilitate the sharing of configuration details to perform an efficient vulnerability assessment while preventing critical data disclosures with respect to cloud tenants and/or cloud providers. As shown in the figure, it serves as a trusted intermediary between the cloud tenant and the cloud provider and is responsible for performing vulnerability assessment activities. It relies on two main internal components that correspond to a TOSCA interpreter and an OVAL-based vulnerability assessment framework. These latter are used to determine the configuration resulting from a cloud resource migration and then to compare it to a set of vulnerability descriptions coming from the official OVAL repository. OVAL is an open standardized language, supported by MITRE, for describing vulnerabilities and specifying how to assess and report upon a system state. A vulnerability is typically considered a logical combination of conditions that, if observed on a target system, would cause the security problem described by the vulnerability to be present on the system. The OVAL language follows the same concept by considering a vulnerability description as an OVAL definition. An OVAL definition specifies a criterion that logically combines a set of OVAL tests. Each OVAL test, in turn, represents the process by which a specific condition or property is assessed on the target system. Each OVAL test examines an OVAL object (e.g. an Apache web server) looking for a specific OVAL state (e.g. a specific version for this server). Components found in the system matching the OVAL object description are called OVAL items. These items are compared against the specified OVAL state in order to build the OVAL test result. The overall result for the criterion specified in the OVAL definition will be built using the results of each referenced OVAL test and will permit determining whether the configuration resulting from a migration corresponds to a vulnerable state. More details about the OVAL-based vulnerability assessment framework can be found in [133]. When a known vulnerability is detected, this framework is capable of determining remediation actions (such as activating a given firewall rule) and if any corrections are associated with the OVAL definitions. These actions are typically specified using the Extensible Configuration Checklist Description Format (XCDDF) language, which supports the mapping of OVAL vulnerability descriptions to remediation actions. These actions may then be enforced by the cloud provider and/or the cloud tenant to protect the considered migrated resources.

5.3.2 Interactions amongst the main building blocks supported by an extended version of the TOSCA language

Listing 1 Extract of an extended TOSCA topology sent by the cloud tenant to the trusted third party prior to the migration

```
topology_template:
...
  node_templates:
    web_server:
        type: tosca.nodes.WebServer.Apache
        properties:
            version: 2.4
        requirements:
            host:
             properties:
               num_cpus: 1
               disk_size: 10 GB
               mem_size: 4 MB
            os:
              properties:
               ...
    webapp_struts:
        type: tosca.nodes.WebApp.Struts
        properties:
            version: 2.3.12
        requirements:
            host: web_server
            database_endpoint:
                 node: mysql_db
            migration:
                 status: true
                 type: online
    mysql_db:
        type: tosca.nodes.Database.MySql
        properties:
            version: 8.0.29
        requirements:
            host: db_server
            migration:
                 status: false
            ...
```

The interactions amongst the trusted third party, the cloud tenant, and the cloud provider are supported by an extended version of the TOSCA orchestration language in order to facilitate the sharing of configuration information required to perform vulnerability assessments prior to the migration. Let us consider the simple scenario of a Struts web micro-service migrating to

another cloud provider at the request of a cloud tenant. This latter may initiate the migration process for a candidate cloud resource that is currently hosted on its on-premise infrastructure or that is already hosted on the infrastructure of another cloud provider.

This process is typically triggered by performance objectives, such as improving the quality of service or minimizing operational costs. Before performing the migration, the cloud tenant will first send a request to the inter-cloud trusted third party in order to assess the impact of the resource migration from a security configuration perspective, specifically to determine whether the migration may introduce new configuration vulnerabilities on the cloud composite service. In that context, it is important to take into account the dependencies that may exist amongst the migrated resource and the other resources of the cloud composite service. The request sent by the cloud tenant (represented by Step 1 on Figure 5.1) therefore integrates an extended TOSCA specification of the cloud composite service that indicates the structure of the cloud service (nodes, dependencies), but also the resource(s) that are candidate(s) for migration. An extract of such extended specification is given on Listing 1. We can observe on this extract that the Struts web micro-service, corresponding to the node of type tosca.nodes.WebApp.Struts, is dependent on two other main nodes, corresponding to an Apache webserver instance (host property on line 22) and a backend MySQL database instance (database_endpoint property on line 24). The migration may involve one or several resources of the cloud composite service. In this example, only one resource is a candidate for migration, as shown by the migration status property being set to true (on line 26) for the Struts web micro-service. The request enables the trusted third party to know about the dependencies that may exist both vertically (e.g. hosting of the Struts micro-service over a given web server) and horizontally (e.g. interconnection of the Struts micro-service to a given database) for the migrated resource.

Once the inter-cloud trusted third party receives the request from the cloud tenant, it activates its TOSCA interpreter in order to analyze the cloud composite service and determines the dependencies of the cloud resource candidate for migration. It then interacts with the cloud provider in order to get the contextual configuration provided by its hosting infrastructure (as shown by Step 2 on Figure 5.1). This includes, in particular, the nature of the hosting nodes, and the availability of security mechanisms to protect the infrastructure. This response from the cloud provider corresponds to Step 3 on the figure and again integrates an extended specification of the TOSCA topology. In our scenario, the inter-cloud trusted third party gets back a response, including the extract presenting available security mechanisms (Listing 2). The response indicates that the cloud provider supports three different versions of the Apache web server (not detailed on this extract), namely Apache versions 2.4.54 and 2.4.46, which support the Struts framework version 2.3.12, and Apache version 2.4.31, which supports the Struts framework version 2.3.14, in order to host the Struts web micro-service, and that it provides at least two security mechanisms, namely the open-source web application ModSecurity firewall, as shown on line 4, which is designed as a module for the Apache server, and the open-source Suricata intrusion detection system, as shown on line 10 on the listing.

The inter-cloud trusted third party then exploits its vulnerability assessment framework to determine the projection of the cloud resource configuration after migration. In our scenario, the only configuration parameter that may change due to the migration is the hosting environment of the Struts web micro-service, meaning the Apache web server. The version of the back-end database is kept unchanged. The vulnerability assessment framework then compares

Listing 2 Extract of available security mechanisms sent by the cloud provider to the trusted third party

```
topology_template:
...
  node_templates:
    apache_modsecurity:
        type: tosca.nodes.WAF.ModSecurity
        properties:
            version: 3.0.7
        requirements:
        ...
    oisf_suricata:
        type: tosca.nodes.IDS.Suricata
        properties:
            version: 6.0.6
        requirements:
        ...
```

the projection of the resource configuration to the OVAL vulnerability description database in order to check if this configuration matches any of the known vulnerability descriptions. If no vulnerability is identified by the framework, the inter-cloud trusted third party can directly reply back to the cloud tenant and authorize the migration (Step 6 on Figure 5.1), which is enforced by the cloud tenant by interacting with the cloud provider. If one or several vulnerabilities are detected, the framework may consult a XCCDF [150] database to determine if countermeasures are associated with the considered vulnerabilitie(s). It turns out that in our scenario, the migration generates a vulnerability corresponding to CVE-2017-5638 that can be exploited to perform remote code execution by allowing a remote attacker to inject operating system commands into the web application through the content-type header. Two counter-measures associated with this OVAL vulnerability description are provided by the XCDDF database, namely a ModSecurity firewall rule, and the upgrade of the Struts web micro-service to a more recent version of the Struts framework, meaning to a version 2.3.x equal to or higher than 2.3.32, or to a version 2.5.x equal to or higher than 2.5.10.1. Let us consider that the upgrade is not currently possible due to interoperability issues; the only remediation action is to enforce the new web application firewall rules over the ModSecurity firewall. The inter-cloud trusted third party requests the cloud provider to activate this security mechanism (Step 4 on Figure 5.1), and receives a response from the cloud provider that acknowledges that the additional firewall rules have been properly activated over the cloud infrastructure (Step 5 on Figure 5.1). The third party then confirms to the cloud tenant that the migration of the considered resource is authorized over the cloud provider's hosting environment(Step 6 on Figure 5.1). Finally, the cloud tenant directly interacts with the cloud provider to enforce the migration of the cloud resource, which corresponds to Steps 7 and 8 on the figure.

Complementarily, Figure 2 synthesizes the different interactions external and internal to the trusted third party during the migration assessment corresponding to the considered scenario.

First, the cloud tenant sends the migration assessment request, including the TOSCA specification of the service. The trusted third party then requests contextual information from the cloud provider and exploits the vulnerability assessment framework (in dotted lines) to identify the potential vulnerabilities. Finally, it decides to activate the web application firewall rules in order to protect the Struts web micro-service on the new cloud infrastructure before letting the cloud tenant perform the effective migration. This strategy permits restricting the configuration information that is shared between the cloud tenant and the cloud provider. In particular, the cloud tenant does not have to disclose the whole TOSCA specification of the cloud composite service to the cloud provider. In the meantime, the cloud provider does not have to disclose the whole configuration of its hosting infrastructure, including security mechanisms, to the cloud tenant.

5.3.3 Migration assessment formalization

The migration assessment aims at determining the potential configuration vulnerabilities that may occur due to the migration of the cloud resource(s) over the hosting infrastructure offered by the cloud provider. We formalize the migration assessment supported by the inter-cloud trusted third party, depending on the observability jointly provided by the cloud tenant and cloud provider.

Let us consider that the cloud tenant A owns a cloud service composed of a set of resources denoted $C = \{c_1, c_2, \dots\}$. Each resource c_i, such as the Struts micro-service, is in turn characterized by a set of properties $P = \{p_1, p_2, \dots\}$ that can be seen as unary predicates $p_i(c)$ that characterize the current state $s_i(c)$ of the component, with $s \in S$ and $S = \{s_1, s_2, \dots\}$ representing the set of component states. For instance, a given predicate may relate to the version of a given Apache web server. The migration of a cloud resource c_i of the cloud tenant A to the cloud provider B may introduce new configuration vulnerabilities, with regard to a given set of vulnerabilities noted $V = \{v_1, v_2, \dots\}$. Each vulnerability description can be specified as a logical formula corresponding to a state $s \in S$, the vulnerability dataset can be seen as a disjunction of logical formulas given by $\phi = v_1 \vee v_2 \dots \vee v_n = \bigvee(v_i)$ with $v_i \in V$. Concretely speaking, this dataset corresponds in our case to the official OVAL repository of vulnerability descriptions. We also consider the function $state : C \rightarrow S$ taking a cloud component $c \in C$ as input and returning its current state $s \in S$. For instance, this state can provide the different properties related to the Struts micro-service, such as the versions of the Struts framework and the Apache web server. $M = \{m_1, m_2, \dots\}$ denotes the set of migrations that can be applied over a cloud composite service during its operation, they may concern one or several components c_i of the given composite service, and are performed at the request of the cloud tenant A.

In that context, the cloud tenant sends to the trusted third party TTP a migration assessment request, that contains a specification of the cloud composite service, that may be only partial. Let us consider $P_{TTP}(A)$ the set of properties that are disclosed by the cloud tenant A to the trusted third party TTP during this initiation process. For instance, the $P_{TTP}(A)$ set may typically include the property characterizing the version of the MySQL database server currently used by the cloud tenant A. The third party then requests to the cloud provider contextual configuration information related to the requested migration. We denote $P_{TTP}(B)$ the set of properties that are disclosed by the cloud provider B to the same trusted third party TTP

during this second phase. For instance, the $P_{TTP}(B)$ set may include the versions of the Apache web server supported by the cloud provider B. From this knowledge, the trusted third party exploits the vulnerability assessment framework to establish a projection of the cloud resource configuration that will result from the considered migration m. For that purpose, we consider two different functions already defined in [133], that are respectively:

- $impact : M \rightarrow S \equiv$ function that takes a considered migration $m \in M$ as input and returns a state $s \in S$ that projects the affected characteristics corresponding to a migration m, independently from the considered component, by considering the set of properties P,
- $\Pi : S \times S \rightarrow S \equiv$ function that takes a projected state $s_1 \in S$ together with a component state $s_2 \in S$ as inputs and returns s_2 updated with the properties of $s1$.

Based on these definitions, we then specify a migration assessment function $\Phi_{TTP} : P \times S \rightarrow Boolean$ that determines if the cloud component $c \in C$ is vulnerable under V, meaning that the assessment of ϕ over the state of c is true, considering the set of observable properties P_{obs}. By observable properties, we mean properties that the trusted third party TTP is capable to evaluate, because they have been disclosed by the cloud tenant A or the cloud provider B.

$$\begin{gathered}\Phi_{TTP}(P_{obs},\Pi(impact(m_i), state(c))) \\ with\ P_{obs} \in P, m_i \in M, c \in C\end{gathered} \tag{5.1}$$

This migration assessment function Φ_{TTP} detailed by Equation 6.3 enables assessing potential configuration vulnerabilities related to the projection of the cloud resource c under V, considering the set of observable properties P_{obs}. This projection of the migrated resource is itself obtained from the Π function applied to the $impact(m_i)$ and $state(c)$ functions. The knowledge of the trusted third party directly depends on the properties $P_{TTP}(A)$ disclosed by the cloud tenant A, and on the properties $P_{TTP}(B)$ disclosed by the cloud provider B.

$$\begin{gathered}\Phi_{TTP}(P_{TTP}(A) \cup P_{TTP}(B), \Pi(impact(m_i), state(c))) \\ with\ (P_{TTP}(A), P_{TTP}(B)) \in P^2, m_i \in M, c \in C\end{gathered} \tag{5.2}$$

The migration assessment performed by the trusted third party TTP corresponds to a satisfiability issue, where we make sure that the new configuration of the cloud resource (e.g. the one after the migration) does not match any vulnerable configuration, and is solved by the vulnerability assessment framework using a back-end SMT solver. The completeness of the assessment Φ_{TTP} under the vulnerability dataset V is directly dependent on the observability of the stakeholder performing this activity, namely the trusted third party in our solution. In the regular case, without a trusted third party, this observability may be restricted in order to prevent the disclosure of critical properties between the cloud tenant and the cloud provider. However, the lack of such knowledge impacts on the vulnerability descriptions that can be covered by the vulnerability assessment. For instance, not knowing the version of the Apache web server prevents to properly assess configuration vulnerabilities involving this property, such as a vulnerability affecting a specific version of this web server. In the meantime, the introduction of the trusted third party encourages the sharing of this configuration knowledge, in order to improve the migration assessment activity and the detection of configuration vulnerabilities that are part of the vulnerability dataset V.

5.4 Performance Evaluation

We have evaluated the performance of our trusted third-party approach for cloud composite services through an extensive series of experiments. For that, we have developed a proof-of-concept prototype of the proposed C3S-TTP architecture using the Python v3.10 language, implemented on top of our OVAL-based vulnerability assessment framework [133], and considering the open-source CVC4 tool as a back-end SMT solver. The assessment framework is compatible with other solvers, such as Z3 and VeriT, as it internally generates a SMT-LIB (Satisfiability Modulo Theories LIBrary) specification that can be interpreted by a large variety of SMT solvers. The different experiments have been conducted on a regular laptop equipped with a 2 GHz Intel Core i5 processor and 8 GB of RAM memory. We have considered a simple TOSCA specification, and used vulnerability descriptions coming from the official OVAL repository, focusing on the largest available datasets, namely those related to CentOS Linux, Red Hat Enterprise Linux, and Microsoft Windows. The methodology can, however, be easily extended to any other resources for which OVAL vulnerability descriptions are available.

The objective of these experiments was in particular to quantify the benefits and limits of using our trusted third party, in comparison to a baseline approach with direct interactions between the cloud tenant and cloud provider, where the configuration is partially observable. This baseline corresponds to a peer-to-peer scenario, where only a certain percentage of observability is enabled. The extreme case of having 0% of observability means that no information is shared at all. Therefore, the TTP does not intervene and the assessment cannot be performed. While the linearity of the management overhead in number of messages is trivial, we wanted to quantify how the observability enabled by such a third party contributes to perform vulnerability assessment activities in a more efficient manner. We can define this observability as a ratio between the knowledge provided to the third party TTP by the cloud tenant A and the cloud provider B, corresponding to $P_{TTP}(A) \cap P_{TTP}(B)$, and the overall knowledge required for the vulnerability assessment, and corresponding to the overall set of properties P, as given by Equation 5.3.

$$observability = \frac{P_{TTP}(A) \cap P_{TTP}(B)}{P} \quad with\ (P_{TTP}(A), P_{TTP}(B)) \in P^2 \tag{5.3}$$

These properties are required to assess whether the resource configuration matches a given vulnerability description. A property (or predicate) may appear several times in the vulnerability dataset V, and contribute to the assessment of different vulnerability descriptions. A higher observability, meaning a higher ratio, is expected to enable higher vulnerability assessment performance but requires a higher disclosure of configuration information that may not be acceptable in direct interactions between the cloud tenant and the cloud provider.

We conducted four major series of experiments in that context. The first series have focused on the impact of the observability ratio on the migration assessment time. The second series has investigated to what extent this observability impacts on the vulnerability assessment performance, while the third series has looked at how it contributes to reducing the attack surface and minimizing risks, considering the common vulnerability scoring system (CVSS) [151]. The fourth experimental series has focused on comparing our solution to a proactive security approach in order to counter low configuration observability.

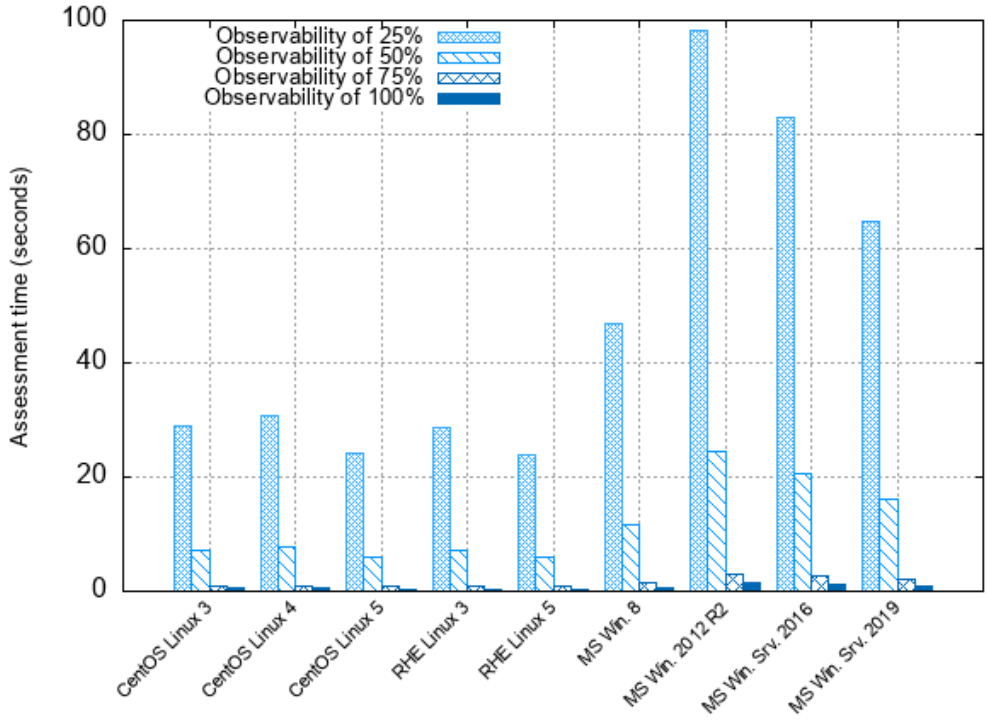
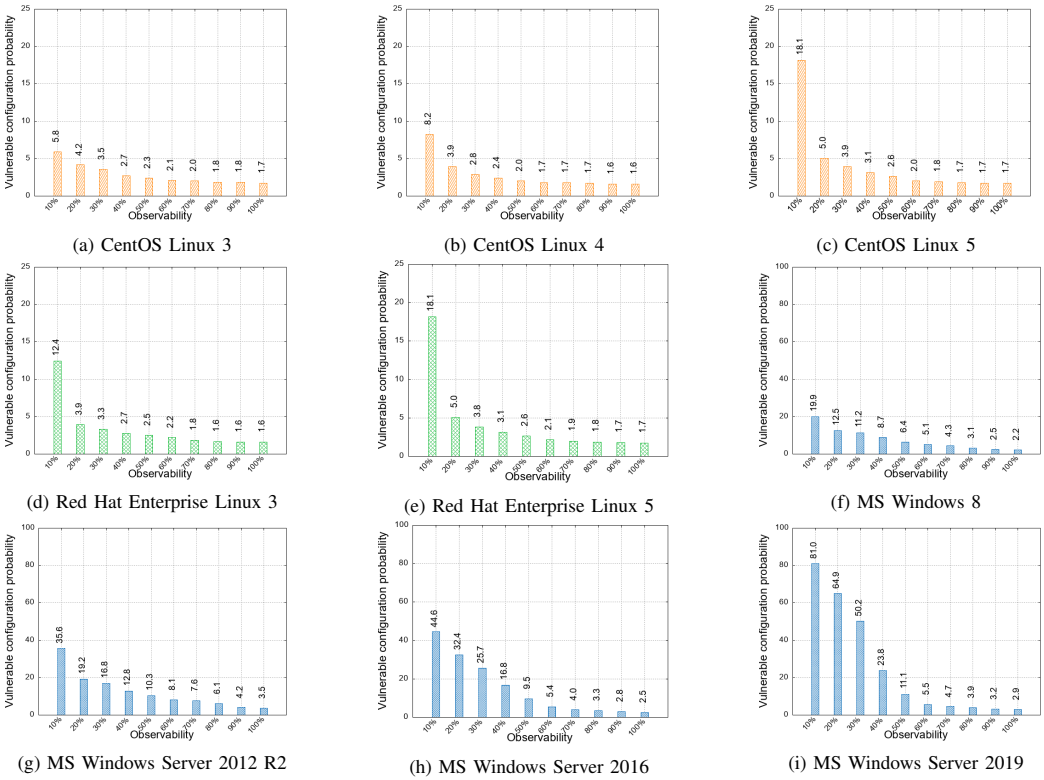


Figure 5.3: Migration assessment time depending on the configuration observability (expressed in terms of percentage of known properties) with different operating systems

5.4.1 Impact of observability on the migration assessment time

In a first series of experiments, we were interested in quantifying the impact of the observability enabled by our trusted third party on the migration assessment time. For that, we considered the migration of a set of cloud resources to a cloud provider infrastructure, with a projected configuration corresponding to an average of 12 properties, and this with different operating systems. We then measured the time required to perform the migration assessment from the official OVAL dataset for the considered operating systems (corresponding to more than 22100 vulnerability descriptions in total) while varying the observability ratio from 0% to 100%. The OVAL properties considered in the experiments are selected in a random manner. These experiments have been repeated until convergence in order to establish average values, and we have observed a variability of up to 5% in our experiments. These experimental results are synthesized on Figure 5.3, where the x axis indicates the operating system and its version and the y axis provides the migration assessment time for a given observability ratio. An observability ratio of 0% means that no property is observable, and therefore the assessment simply cannot be performed.

On the figure, we can observe for each plotted observability ratio that the assessment time is higher for Microsoft Windows Server in comparison to CentOS Linux and Red Hat Enterprise Linux. This can be explained by the fact that the OVAL datasets related to Microsoft Windows Server contain more vulnerability descriptions than the two others. For instance, the Microsoft



(a) CentOS Linux 3 (b) CentOS Linux 4 (c) CentOS Linux 5

(d) Red Hat Enterprise Linux 3 (e) Red Hat Enterprise Linux 5 (f) MS Windows 8

(g) MS Windows Server 2012 R2 (h) MS Windows Server 2016 (i) MS Windows Server 2019

Figure 5.4: Probability of having a vulnerable configuration after the cloud resource migration depending on the configuration observability (expressed in terms of percentage of known properties) with different operating systems

Windows Server 2012 R2 dataset corresponds to 5344 vulnerability descriptions, and the migration assessment for an observability of 25% reaches 98.7s, while the CentOS Linux 5 dataset contains only 1171 vulnerability descriptions and requires a migration assessment time of 28.2s for the same observability ratio. We expected that a low observability would contribute to reducing the migration assessment time while deteriorating the vulnerability detection performance. Actually, it appears that a higher observability reduces the migration assessment time for all the considered operating systems. Considering low observability means that the cloud destination environment may potentially contain a large number of vulnerabilities. This increases the time assessment and the probability of matching a vulnerability containing the shared properties. In this case, looking for the first vulnerability that appears requires less time as the probability of matching a vulnerability is higher. However, when searching for overall vulnerabilities, the assessment time becomes more important. For instance, a subset of properties may match a vulnerability that the framework finds at the beginning of the assessment process (considering the given dataset), while another subset may match another vulnerability that is found at the end of this process. Let consider the case of Microsoft Window Server 2019, which contains more than 1400 vulnerability descriptions. The migration assessment time is of 69.2 s for an observability ratio of 25%, while it decreases respectively to 2.7 s and 1.2 s for observability ratios reaching

75% and 100%. The same phenomenon is observed with the different versions of CentOS Linux and Red Hat Enterprise Linux. The migration assessment with respect to a given vulnerability dataset is formalized as a satisfiability issue that is solved by the vulnerability assessment framework integrated with the trusted third party using a CVC4 back-end solver. Increasing the observability ratio means knowing more properties and therefore having more constraints regarding the satisfiability issue to be solved, which explains the reduction in the assessment time when the observability is increasing. Therefore, the higher observability enabled by the trusted third party contributes to reducing the migration assessment time.

5.4.2 Impact of observability on the probability of having a vulnerable configuration after the migration

In a second series of experiments, we wanted to evaluate the impact of observability on the probability of having a vulnerable configuration after the migration, considering the vulnerability descriptions provided by the official OVAL repository. A low observability means that the migration assessment process will not be able to properly evaluate all the vulnerability descriptions related to a given operating system because some configuration properties are not observable. We have quantified the probability of having a vulnerable configuration while varying the observability ratio from 0% to 100%. Concretely, we have evaluated the percentage of vulnerability descriptions that cannot be assessed due to unknown properties for each operating system. These unknown properties correspond to those that the cloud tenant refuses to disclose to the cloud provider. The experimental results are given on Figure 5.4, where we have plotted for each considered system this probability (represented on the y axis), with different observability ratios (represented on the x axis). Let consider the case of the CentOS Linux 3 operating system. We can observe that the probability of having a vulnerable configuration is of 5.8% with an observability ratio of 10%, while this probability decreases to 1.7% with an observability ratio of 100%. We can notice that with the maximal observability (ratio reaching 100%) this probability of having a vulnerability is close but not equal to zero due to residual vulnerable configurations inherent to the cloud environment intended for migration. For each considered operating system, we can observe that the probabilities of having a vulnerable configuration are decreasing when the observability is increasing. The difference between an observability of 10% and an observability of 100% is on average of 9.03% with CentOS Linux, of 13.6% with Red Hat Enterprise Linux, and of 42.5% with Microsoft Windows Server. This phenomenon is related to the intrinsic nature of the vulnerability description datasets, in particular the redundancy of properties appearing in several descriptions, and is also partially correlated to the size of the vulnerability description datasets. For instance, the average size of the Microsoft Windows Server dataset is around 3800 descriptions, while it reaches around 1400 descriptions for the two other operating systems. We can also observe from these sub-figures that the benefits of increasing the observability is overally decreasing over time. For instance with the Microsoft Windows Server 2019 system, the benefits is of 16.1% between an observability of 10% and 20%, while it is only of 0.3% between an observability of 90% and 100%.

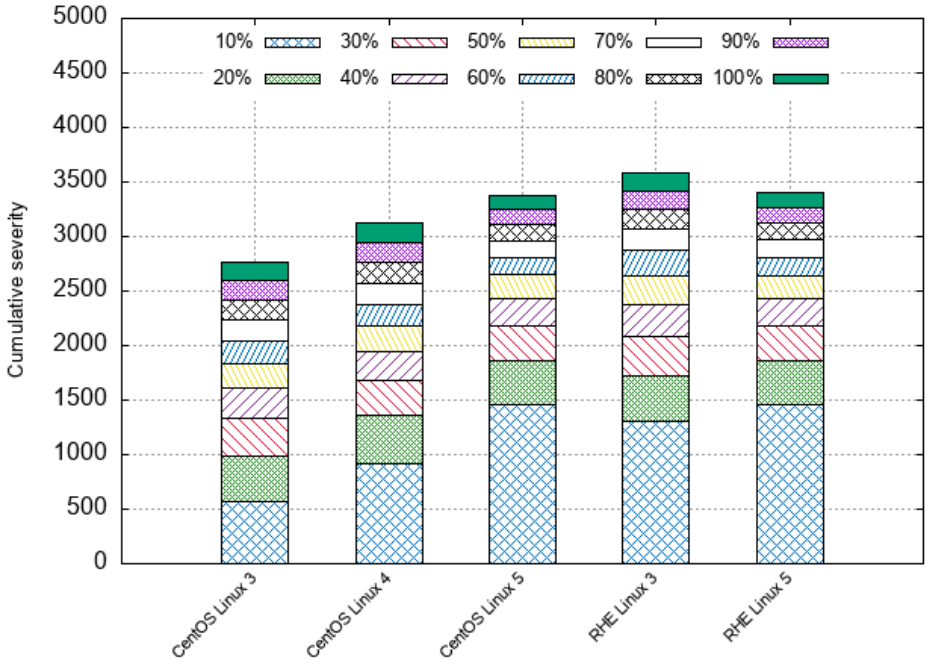
5.4.3 Cumulative severity related to potential configuration vulnerabilities after the migration of cloud resources

The severity of vulnerabilities may vary depending on their exploitability and their impact on the cloud resource and the whole cloud composite service. In order to quantify the risk associated with potential vulnerable configurations, we have performed a third series of experiments where we have quantified the cumulative severity of potential vulnerabilities while varying the observability ratio. For that, we have relied on the CVSS scoring system, which is also part of the SCAP[2] security automation languages and provides a normalized framework to characterize and quantify severity scores for configuration vulnerabilities. In these experiments, we have analyzed the cumulative severity of potential vulnerabilities, by adding up the severity scores provided by the CVSS system for these vulnerabilities. The experimental results presenting this severity for each considered operating system are described on Figure 5.5, while varying the observability ratio. The highest severity is observed again with Microsoft Windows Server 2012 R2, which should be mitigated by the fact that this also corresponds to the OVAL dataset with the highest number of vulnerability descriptions. The benefits of increasing the observability are also decreasing over time with regard to the cumulative severity, as we have already observed this phenomenon in the previous series of experiments. In most of the cases, an observability ratio of 30% enables to cover at least 50% of the cumulative severity associated to potential vulnerable configurations. However, a high observability ratio is required to cover all the potential vulnerable configurations characterized by the highest severity, considering the distribution of these vulnerabilities in the different analyzed datasets. As a result, our trusted third party benefits from increased configuration observability during cloud resource migration between a cloud tenant and a cloud provider.

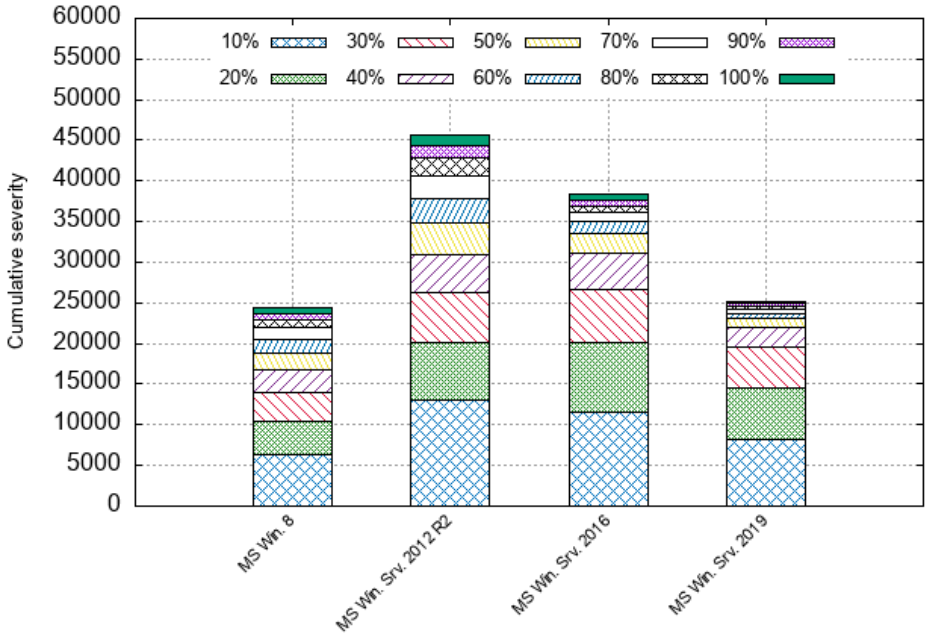
5.4.4 Comparison to a proactive security strategy supporting the protection of migrated cloud resources

The lack of observability of the cloud tenant with respect to the cloud provider might be compensated by implementing a proactive security strategy. This one consists of enforcing countermeasures (such as patches) to cover any potential vulnerable configurations prior to the migration. This, however, represents an additional cost to the cloud tenant in order to cover these vulnerabilities that may not be effectively present. We have performed a fourth series of experiments to quantify this cost while varying the observability ratio. The experimental results are summarized on Figure 5.6, where we have plotted the total cost of such a proactive security approach, quantified in terms of the minimal number of countermeasures to be implemented by the cloud tenant, considering the vulnerability description datasets of different operating systems. The cost of such a proactive strategy could be refined by evaluating experimentally the operational costs (i.e. processing time, memory usage) due to the enforcement of countermeasures, but this work goes beyond the scope of this article. We can observe on the figure that the cost is decreasing, when the observability becomes higher. The lowest cost has been observed with Red Hat Enterprise Linux 5, while the dataset does not correspond to the lowest number of vulnerability descriptions. This can be explained by the fact that some countermeasures may cover a

[2]Security Configuration Automation Protocol



(a) Linux operating systems



(b) Microsoft Windows operating systems

Figure 5.5: Cumulative severity induced by potential vulnerable configurations depending on the configuration observability

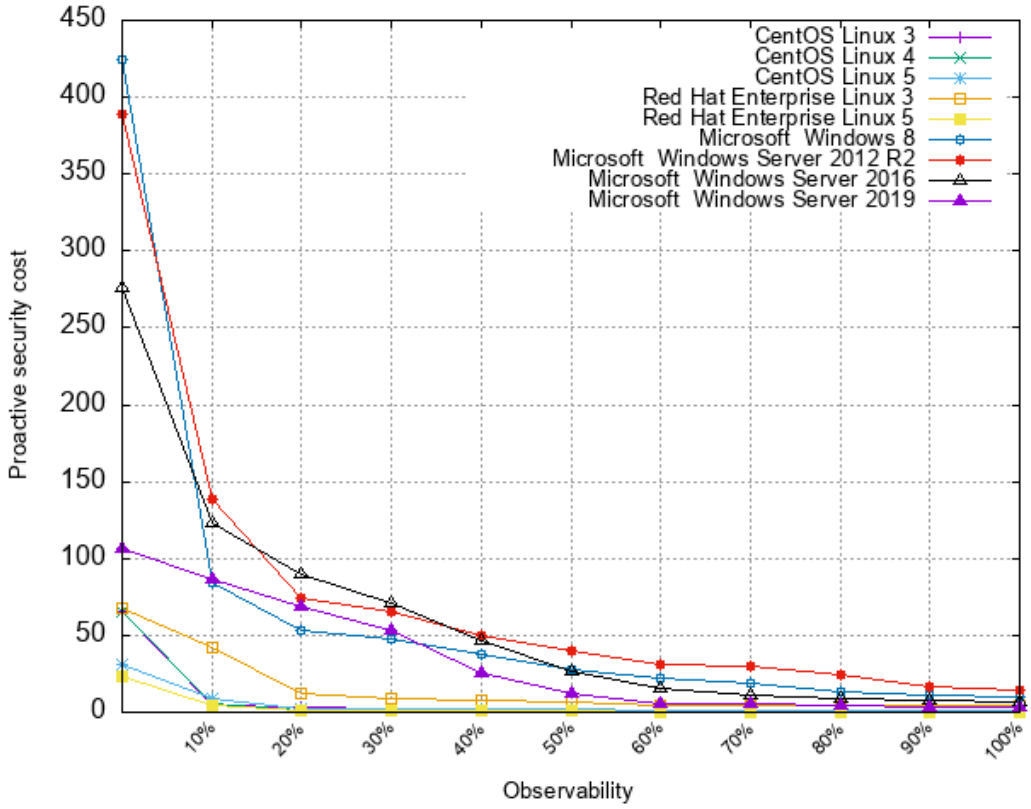


Figure 5.6: Cost of a proactive security strategy to protect cloud resources while varying the configuration observability

large number of potential vulnerability descriptions, while others are only capable of addressing a specific one. Overall, the cost induced by this proactive security strategy is not negligible for any of the considered operating systems and may be balanced by better configuration knowledge of the cloud provider infrastructures, as supported by our trusted third party solution for cloud composite services.

Our C3S-TTP approach therefore provides interesting and relevant experimental results. It is essential to consider potential limitations to gain a complete perspective on its applicability. The knowledge given by the OVAL vulnerability descriptions serves as the foundation for the assessment process. For this, consistent and continuous updates are required for maintaining vulnerability description datasets, and keep a performant vulnerability assessment. It is also important to note that our solution is focused on the detection of known vulnerabilities. Another aspect would be to extend the approach to the case of unknown vulnerabilities, and therefore covering vulnerability discovery. For instance, threat intelligence methods and techniques might be integrated to our third-party architecture to cover such vulnerabilities, whose descriptions are not yet available. Finally, achieving the balance between security assessment and quality of service is essential. Critical migrations may require immediate movements, which does not allow the necessary time to perform the desired security assessment. This goes in favor of an assessment that is performed as much as possible in anticipation of the resource migration.

5.5 Synthesis

We have proposed an inter-cloud trusted third-party approach, called C3S-TTP, for supporting configuration security in cloud composite services. This third-party entity serves as an intermediary between the cloud tenant and the cloud provider, and is responsible for collecting configuration information and preventing configuration vulnerabilities before performing the migration of cloud resources. We have first described the considered architecture and its main building blocks. We have specified an extension of the TOSCA orchestration language to support the interactions amongst the trusted third party, the cloud tenant, and the cloud provider. We have formalized the assessment process performed by the trusted third party by considering vulnerability descriptions extracted from the official OVAL repository. We have also developed a proof-of-concept prototype of this architecture, implemented on top of our SMT-based assessment framework. Finally, we have performed an extensive series of experiments in order to quantify the benefits and limits of our approach in comparison to a baseline solution without a trusted third party. We have shown to what extent the observability enabled by the trusted third party contributes to increase the performance of vulnerability prevention and to minimize risks related to vulnerable configurations. The experimental results have shown that such observability reduces the security assessment time for all the considered systems (CentOS Linux, Red Hat Enterprise Linux, Microsoft Windows). In addition, the probability of having a vulnerable configuration decreases as the number of known configuration properties about the cloud environment increases. The difference between an observability of 10% and an observability of 100% is on average of 9.03% with CentOS Linux, of 13.6% with Red Hat Enterprise Linux, and of 42.5% with Microsoft Windows Server. In the same manner, increasing the observability reduces significantly the cost of activating security functions, when following a proactive security strategy. In the next chapter, we extend our work to the case of moving target defense strategies in cloud composite services.

Chapter 6

Verified AI-based Moving Target Defense Strategy for Securing Cloud Composite Services

Contents

6.1 Introduction

Prior performing an attack against cloud composite services, the attackers typically conduct reconnaissance activities to gather information about the cloud infrastructure, including its architecture, configurations, and potential weaknesses. By collecting such knowledge through reconnaissance activities, they can identify vulnerabilities and exploit them to gain unauthorized access, execute malicious actions, or extract sensitive data from the cloud environments [152].

In addition, the adaptability, the unpredictability, and the irrationality of attackers lead to an important asymmetry between them and the defenders. The static aspects of most of current defense mechanisms result in a large attack surface for cloud composite services. Indeed, the static nature of these mechanisms, such as antivirus and firewalls, often provides attackers with an important advantage, while defenders are reacting retroactively when a suspicious event is detected or an attack is initiated [153].

In order to overcome these issues, new approaches that introduce dynamic defense strategies, are used to protect these services. This includes moving target defense (MTD) strategies. Such techniques aim at modifying, controlling, and changing, continuously and dynamically, the attack surface of a given system. They propose to shift configurations in order to prevent attackers opportunities. The objective is to reduce the likelihood to identify and exploit targets and increase the overall attack costs. Without having a consolidated view on valuable assets and critical elements on the targeted infrastructures, attackers will waste more time and efforts to perform reconnaissance activities. MTD defense approaches are typically characterized by three major pillars. The first one concerns the configuration parameters that are subjects to change on the system. For instance, the strategies may modify the attack surface of the system through the modification of the address space layout, of the software programs, of the used interfaces, or of the considered virtual machines. The second pillar consists in determining the appropriate time space to perform these changes. These latter may be done reactively, with changes triggered by specific events observed on the system and its environment [154]. This may also be done proactively. For instance, the changes can be performed according to a given frequency [155]. The strategies may also combine reactive and proactive features in a hybrid manner, such as in [156, 157]. The third pillar refers to the nature of the changes in themselves. This corresponds to another aspect contributing to the uncertainty with respect to the view of the attackers. For instance, the strategies may implement different shuffling techniques, and rely on a large variety of implementations. While these moving target defense strategies contribute to discourage attackers by increasing the costs and performance of reconnaissance activities, they may also involuntarily generate new vulnerabilities on the system, by trying to further explore configuration states.

We therefore propose in this chapter a moving target defense strategy for protecting cloud composite services, which relies on the coupling of artificial intelligence methods together with verification techniques. The objective is dual by both preventin attacker from taking advantage of information gathered through reconnaissance activities, but also by making sure that the system does not reach vulnerable configurations that may be critical for the considered system. The core of the strategy combines reinforcement learning algorithms to support the exploration of configuration states, combined with verification techniques to minimize the occurrence of vulnerabilities. In that context, the remainder of this chapter is organized as follows. First, Section 6.2 gives a background on moving target defense strategies and their implementation. Section 6.4 then describes the considered moving target defense framework, and illustrates its operation. It also provides a mathematical formalization considering a given reinforcement learning algorithm coupled with verification techniques. Section 6.5 depicts the proof-of-concept prototype implemented using the python language on top of open-source SMT solvers. It also describes the performance evaluation based on extensive series of experiments. Finally, Section 6.6 provides a synthesis of the chapter.

6.2 Background on moving target defense

As for the on-premise infrastructures, attackers follow different steps, also called the cyber kill-chain, in order to perform an attack against cloud infrastructures and their services. An important step consists in reconnaissance activities, in order to collect information about the targeted system. These reconnaissance activities typically include port scanning, network mapping, social engineering, and open-source intelligence gathering [158]. Port scanning consists to scan a given system ports to identify potential security weaknesses or services that can be exploited. Network mapping entails identifying network topology and mapping out the system structure. Social engineering activities involve manipulating individuals to reveal sensitive information or gain unauthorized access. Attackers also use open-source intelligence gathering that consists of collecting information from publicly available sources. The gathered knowledge may include the operating system version, open ports, libraries, and running software. This information is then used to identify vulnerabilities and construct schemes and strategies in order to perform attacks by exploiting these vulnerabilities. While regular static defense mechanisms, such as firewalls, access control mechanisms, and intrusion detection systems, are exploited to reduce the attack surface. The attackers may adjust their strategies to corrupt the defense system, while the corresponding countermeasures are costly, time-consuming, and only effective for a short time. This contributes to an asymmetry in favor of attackers, making static defense incapable of countering malicious attackers with a promising long-term success rate.

In order to break the asymmetry between attackers and defense systems, moving target defense strategies have been introduced for countering reconnaissance activities [159]. Through a continuous modification of the configuration of cloud services, these strategies aim to deceive attackers by making the knowledge collected by reconnaissance activities quickly outdated, so that they cannot properly know the attack surface of the system. For instance, the IP address of a given cloud resource may periodically be changed over time. The objective is to increase the efforts and time required by attackers for making the inventoring of the system and determining potential vulnerabilities to be exploited. Moving target defense strategies exploit different shuffling techniques to dynamically shift the configuration of the system, and therefore modify the attack surface. These techniques may apply to different configuration parameters, such as IP addresses [160], MAC addresses [161], open ports [162], or even operating systems [163], and network topologies[164]. The techniques may also rely on the use of multiple instances of the same resource, such as software components [165] or network sessions [166]. The diversity of these resources may also by increased by modifying the implementations of the resources themselves. For instance, this diversity may be strengthen by using different libraries, interfaces and development platforms [167, 168], or by considering various programming languages [169] supporting the same functionalities. Also, several moving target defense strategies may be combined, such as [170, 171], in order to increase the overall performance and exploration of possible states for the system to be protected.

More specifically in the area of cloud computing, several solutions leverage moving target defense techniques to address cloud security[172]. For instance, the authors of [173] introduce a defense strategy based on the migration of resources over several cloud infrastructures. The purpose is to determine the most adequate hosting environment of a given resource over time. The approach is built in order to counter several attack schemes, by considering both static

and adaptive attackers. The experimental results show to what extend the diversity of hosting environments impact on the overall performance. Some efforts also investigate the exploitation of software-defined networking (SDN) as a lever to increase the dynamics of cloud services, and and evaluate its efficiency against large-scale attacks[174]. In the same manner, the authors of [175] design proactive and reactive migration scheme for virtual machines, in order to protect cloud-based applications. In particular, they target a balance between performance and security. The objective is to optimize the cost and frequency of migrations, in order to provide the most efficient defense strategy. In addition, some efforts, such as [176], provide moving target defense solution to specifically counter co-resident attacks, such as prime and probe attacks [177] or flush and reload attacks [178]. Such attacks are mostly performed against virtual machines co-located in the same cloud infrastructure. They propose to model the problem using game theory, and identify optimal strategies to detect and defend against virtual machines owned by attackers, while maintaining a limited impact on the service performance.

As security attacks are gaining in sophistication, moving target defense strategies are also taking benefits from automation and artificial intelligence techniques. For instance, approaches such as [179] targets predicting attacks in order to improve the selection of movements that have to be performed by the considered system and its resources. They leverage the co-evolutionary relationship between attackers and defenders, in order to reduce the knowledge of attackers about the systems to be defended. The authors of [176] exploit reinforcement learning algorithms to automate a moving target defense strategy that enables the defender to adopt the appropriate security policy depending on the knowledge an attacker is expected to get from the system during its reconnaissance activities. In the same manner, the approach defined in [180] proposes a moving target defense framework, based on reinforcement algorithms, capable to adapt the movements performed on the system, depending on the results provided by an intrusion detection mechanisms. In the approach developed in [181], the authors use a sequential decision-making process and a multi-objective reinforcement learning algorithm to drive the moving target defense. The solution generates a multi-objective markov decision process, considering the components of the system and their behaviors, in order to reduce the attack surface. In addition, the work presented in [182] specifies a strategy for placing attack detection mechanisms in cloud infrastructures. This strategy is formulated as a zero-sum Markov game that uses the knowledge coming from attack graphs together the severity score on vulnerabilities given by the Common Vulnerability Scoring System (CVSS). The objective is to maximize the detection of vulnerabilities, while ensuring the required level of security in the considered cloud infrastructure. Complementarily, some efforts such as [183] rather focuses on cloud data, with a game theoretic-based strategy that relies on different machine learning techniques to counter data tampering attacks.

Moving target defense approaches play an important role to counter reconnaissance activities performed by attackers. Significant efforts have been done to automate these approaches, in particular using artificial intelligence methods that may serve to drive the movements performed on the system and its resources to be protected. Solutions have been proposed in the area of cloud computing, such as techniques based on the migration of cloud resources. In the meaning, these existing approaches tend to increase the exploration of the states of the system. While they reduce or confuse the visibility of the attackers on the system, they may also expose it to vulnerable configurations.

6.3 Exploitation of artificial intelligence

As our moving target defense strategy is driven by artificial intelligence algorithms, we remind here three main categories of learning techniques: supervised learning, unsupervised learning, and reinforcement learning. Each approach is characterized by strengths and weaknesses, and may be used to solve different types of AI problems.

Supervised learning is an approach of artificial intelligence that involves training a model using labeled data to make accurate predictions. Labeled data is data that has already been classified or categorized by humans, providing the model with examples to learn from. During training, the model is presented with labeled data, and it tries to identify patterns and relationships in the data. The model then adjusts its parameters to minimize the difference between its predicted output and the true output, allowing it to make accurate predictions on new, unseen data. Supervised learning can be used to solve a wide range of problems, including image and speech recognition, natural language processing, and even predicting future outcomes [184, 185]. One of the primary advantages of supervised learning is its ability to provide accurate predictions on new data, making it useful for a wide range of applications such as image and speech recognition, natural language processing, and predicting future outcomes. One of the main drawbacks of supervised learning is its reliance on large amounts of labeled data, which can be difficult, challenging and time-consuming to obtain.

Unsupervised learning remains an approach of artificial intelligence that aims to train a model using unlabeled data. In contrast to supervised learning, where the model is trained on labeled data, in unsupervised learning, the model is presented with data that has no labels or categories. The goal of unsupervised learning is to identify patterns and relationships in the data, allowing the model to group similar data points together and identify outliers or anomalies. This can be useful for applications such as clustering, anomaly detection, and dimensionality reduction. The advantages of unsupervised learning remain in its ability to identify patterns and relationships in data without the need for labeled data [185, 186]. This can be useful in situations where labeled data is not available or when the cost of labeling data is too high. However, one of the challenges of unsupervised learning is that it can be difficult to evaluate the performance of the model, as there is no clear metric for determining the correctness of the model output. Unsupervised learning has a broad range of fields of application, from identifying anomalies in financial transactions to clustering similar customer groups in marketing. Clustering helps to group similar data points together based on their features, enabling insights that may not be immediately obvious from the dataset. Anomaly detection can help identify outliers in datasets, such as fraudulent transactions, which may be missed by traditional supervised learning approaches. Dimensionality reduction is another application of unsupervised learning that involves reducing the number of features in a dataset, making it easier to visualize and analyze. Association rule learning helps discover patterns and relationships between variables in a dataset, which can be valuable for understanding customer behavior and identifying cross-selling opportunities. Finally, generative models can be used to create new data points that are similar to the original data. Unsupervised learning is a powerful tool for analyzing large datasets and can be used across many industries to uncover insights and patterns that can drive better decision-making.

Reinforcement learning is an approach to artificial intelligence that aims to train an agent to

make decisions based on the outcomes of its actions in a dynamic environment. The agent receives feedback in the form of rewards or penalties for each action it takes, and its goal is to maximize the total reward it receives over time. Reinforcement learning is particularly useful in situations where there is no clear optimal solution, and the agent must learn through trial and error. Reinforcement learning and Markovian models are closely related, as Markov Decision Processes (MDPs) are often used as a mathematical framework for reinforcement learning. In an MDP, an agent makes a decision based on the current state of the environment, and the environment responds by transitioning to a new state and providing a reward signal to the agent. The goal of the agent is to maximize the expected cumulative reward over time. This process is formalized using the Markov property, which states that the future state and reward only depend on the current state and action, not the history of states and actions [187]. Reinforcement learning is applied in several fields including game playing, robotics, and autonomous vehicles [188]. Reinforcement learning (RL) encapsulates several algorithms that involve learning from trial and error through interaction with an environment, including Q-learning which is a popular model-free algorithms that can handle large state spaces [189], and has been considered to implement our solution.

6.4 Verified AI-based defense strategy

In this section, we will describe our moving target strategy based on the coupling of artificial intelligence with verification techniques. We will first overview the considered approach. We will then detail the underlying framework, by describing its main building blocks and their interactions. We will finally detail the mathematical formalization of our solution.

6.4.1 Coupling of artificial intelligence with verification techniques

The objective of our moving target defense strategy is to enhance the security of composite cloud services. This strategy aims to thwart potential attackers by preventing them from exploiting any knowledge they may gather about these resources. While existing moving target defense strategies typically involve shifting the configuration of a system to mislead and confuse attackers, thereby thwarting their efforts to leverage collected information, our approach takes a step further by also reducing vulnerable configurations using verification techniques. Our strategy is based on the selection of movements that are operated on cloud composite services. These movements correspond to the migration of cloud resources. They may involve one or several resources that compose such a service. These migrations are not carried out haphazardly or randomly; they are executed with the specific purpose of countering reconnaissance activities, so exploring a large spectrum of the configuration states of the considered composite services using artificial intelligence. At the same time, the solution should safeguard the considered resources and the overall service from potential vulnerabilities using verification techniques. The strategy relies on three main pillars, including the automation provided by artificial intelligence algorithms, the vulnerability assessment supported by verification techniques to control the exploration performed by artificial intelligence algorithms, and the migration of cloud resources [190, 191].

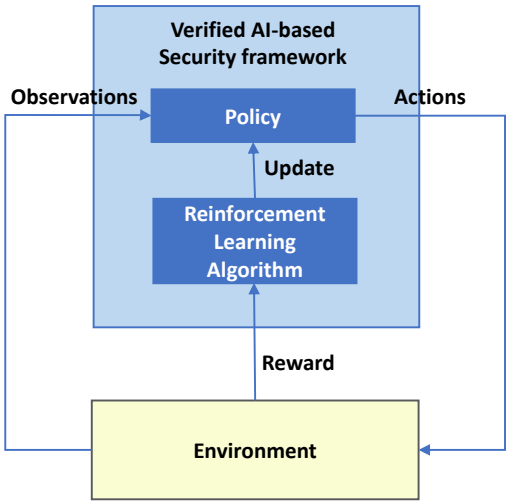
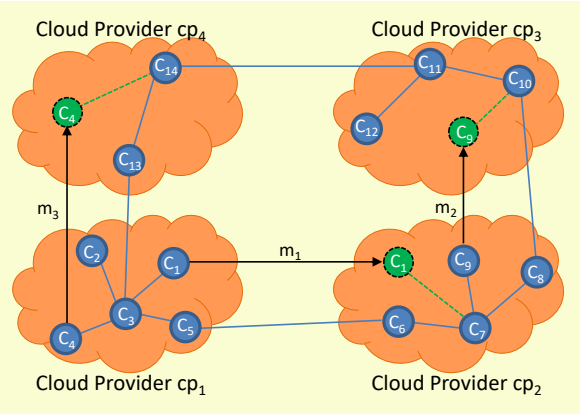
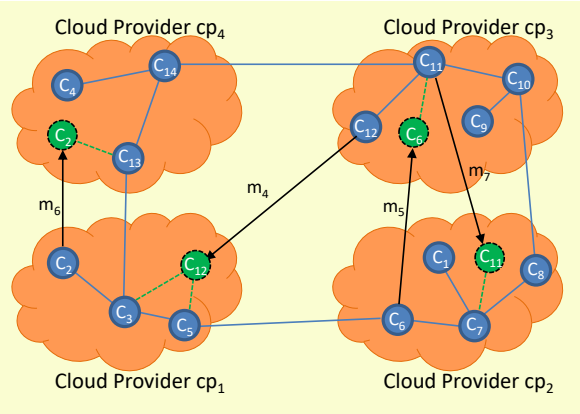


Figure 6.1: Overview of our moving target defense strategy

As depicted on Figure 6.1, our moving target defense strategy is centered on a reinforcement learning cycle, more precisely the Q-learning algorithm [189]. Reinforcement learning enables us to determine the defense actions to be performed on the cloud composite services, while the rewards are calculated based on the results of verification techniques. The Q-learning algorithm has been preferred for its performance in comparison to other reinforcement learning algorithm, such as SARSA [192]. However, it could be easily extended to other reinforcement learning algorithms. The verification techniques are used to assess the state of a cloud resource after performing a migration to a destination cloud environment. The assessment does not only concern the host environment, but take into account both the configuration of the resource together with the cloud environment. We have considered our SMT-based framework previously presented, in order to support such vulnerability assessment [133]. The severity of vulnerabilities is also taken into account by using the scores provided by the common vulnerability score system (CVSS) [151]. The choice of reinforcement learning, which is one of the three main learning paradigms, together with supervised learning and unsupervised learning [193], is based on the fact that it does not require labeled input/output pairs to be presented, nor sub-optimal actions to be explicitly corrected, contrary to alternative techniques. Instead, the objective is to find a balance between the exploration of configuration states, and the prevention of vulnerable states. During the learning process depicted on the figure, the agent interacts with an environment, which corresponds in our case to the cloud composite service, whose resources may be hosted over several cloud providers, offering heterogeneous configurations. The objective of this agent is to determine at each iteration the movements to be performed by the cloud composite service, concretely the migration of resources to be considered. The reward is then calculated by the results provided by the vulnerability assessment framework. This enables to update the policy driving the defense strategy, in order to prevent that the exploration driven by the artificial intelligence brings the system into configurations that are characterized by high severity scores.



(a) Phase 1: cloud composite service before performing migrations m_1, m_2 and m_3



(b) Phase 2: cloud composite service after performing migrations m_1, m_2 and m_3

Figure 6.2: A sample scenario illustrating the migrations of resources on a cloud composite service, driven by our moving target defense strategy

In order to illustrate how our moving target defense strategy is executed, let us consider a simple scenario, as given on Figure 6.2. It represents a cloud composite service that consists of twelve components deployed across four heterogeneous cloud providers with different infrastructures. These resources are orchestrated using a dedicated language, namely the Topology and Orchestration Specification for Cloud Applications (TOSCA) language. The individual resources may have inter-dependencies, and may be elastically scaled to address client requests. As an example for such services, we can consider a three-tier cloud application, composed of resources including components handling presentation interfaces, stateless components handling business logic tiers, and storage components that allow for the storage of data. These resources can be either deployed over several infrastructures of the same cloud provider with similar or different configurations, or amongst multiple cloud providers. The first phase of the considered scenario (left part of Figure 6.2) assumes that the moving target strategy triggers the migration of resources c_1, c_4 and c_9. The component c_1, initially deployed over the cloud provider cp_1, migrates to the cloud provider cp_4. Similarly, the components c_4 and c_9 migrate respectively from cloud provider cp_1 and cp_2 to cloud providers cp_4 and cp_3. Such migrations may be performed in a periodic manner, or due to specific events, including the detection of reconnaissance activities by attackers. These events are coming in general from security equipments, such as network intrusion detection systems, or threat intelligence sharing platforms. In the second phase of the considered scenario (right part of Figure 6.2), we can observe that the components c_1, c_4 and c_9 have moved, and that new dependencies, that are shown in green dotted lines, have appeared, and others have disappeared. In the remainder of our scenario, further migrations are performed by the moving target defense strategy, corresponding to the migrations m_4, m_5 m_6 and m_7. The number of migrations/movements may depends on the trigger (time-based strategy, event-based strategy, or hybrid strategy). While the exploration of states reduces the visibility of attackers, verification techniques minimize the occurrence of vulnerabilities due to these movements.

6.4.2 Underlying framework and its main building blocks

The underlying framework supporting our moving target defense strategy is depicted on Figure 6.3. We can observe the reinforcement learning cycle in the top part of the figure (blue color), while the environment corresponds to the orchestrated resources of the cloud composite service represented in the bottom part of the figure (yellow color). It takes benefits from our SMT-based framework (corresponding to the configuration verifier sub-block) to assess configuration vulnerabilities based on OVAL vulnerability descriptions datasets together with CVSS datasets to evaluate the severity of a given identified vulnerability. The framework supports event-based

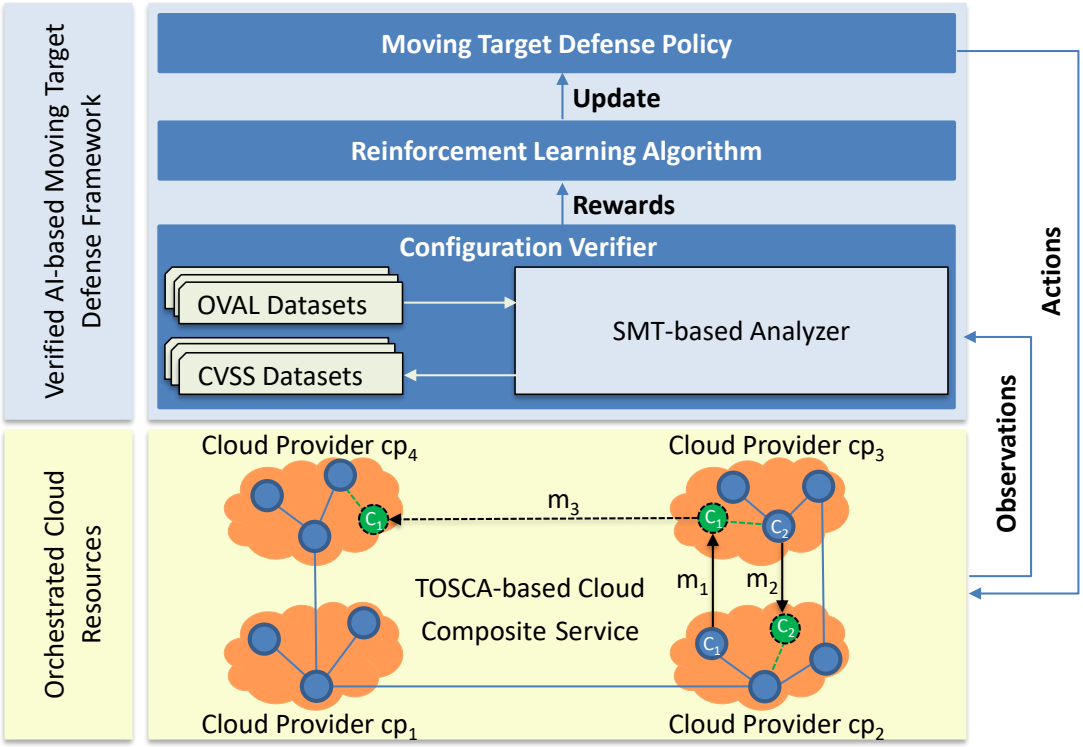


Figure 6.3: Main building blocks of the underlying framework

and time-based strategies for selecting the movements of resources composing a considered cloud composite service. These movements being implemented by migration operations. The event-based technique permits to start a migration after receiving a specific event, such a new threat or an attack attempt. The time-based approach triggers the migration of resources in a periodic manner depending on the time elapsed since the last migration. In order to illustrate the operation of the framework, let us consider the scenario of a single resource migration. First, the moving target defense framework determines the movement to be operated on the cloud resource depending on the current moving target defense policy (first sub-block of the blue part). It therefore implements such an action on the cloud composite service through the resource migration. The configuration verifier sub-block then exploits the SMT-based analyzer to perform

the assessment of the migrated resource over the new hosting environment. As previously mentioned, it relies on OVAL vulnerability descriptions [125] to assess the new configuration. During this phase, the current configuration is compared to a set of configurations that are known to be vulnerable. The configuration verifier may identify one or several vulnerabilities that match the current configuration. It then determines the score associated to these vulnerabilities based on the score provided by the CVSS dataset. These scores are then exploited by the framework to calculate the reward sent to the reinforcement learning algorithm. This one corresponds to the Q-learning algorithm, which then updates the moving target defense policy. This policy initially aims at performing the largest exploration of possible configurations in order to counter reconnaissance activities, by making observations performed by attackers outdated on a short time interval. The rewards provided by the configuration verifier will restrict this exploration to prevent vulnerable configurations with high severity. The process is performed iteratively. In each iteration, the agent is taking decisions regarding the next movements to be applied on the cloud composite services. New assessments are performed by the configuration verifier, and rewards are transmitted to the reinforcement learning algorithm accordingly. The moving target defense policy is updated each time that changes are brought to the configuration of resources composing the cloud services, to the configuration of infrastructures hosting cloud resources, or in case of contextual changes, such as the publication of a new vulnerability description.

6.4.3 Formalization of the defense strategy

We will now formalize our strategy which corresponds to a moving target defense approach that combines a Q-learning method with verification techniques to protect cloud composite services. For that purpose, we will consider a formalism similar to the one used to specify the verification process in the previous chapters. We consider a cloud composite service e, which is composed of a set of resources (also called components), noted $C = \{c_1, c_2, \dots\}$. Each individual component is in turn characterized by a set of properties, noted $P = \{p_1, p_2, \dots\}$, that can be seen as unary predicates $p_i(c)$ defined for the considered component c and its execution environment. We remind that these predicates permit to define the properties possessed by the component, as well as to specify the properties to be observed for the configuration assessment. The cloud composite service can be considered as a component itself. We also consider $S = \{s_1, s_2, \dots\}$, the set of states describing in a compact manner a set of properties required to be observed over a cloud resource or a composite service. The cloud resources may be hosted over different cloud service providers, noted $CP = \{cp_1, cp_2, \dots\}$. For simplification, we consider that each of these latter only offer a single cloud infrastructure. However, the formalization is extensible to the case of cloud providers offering multiple infrastructures. These infrastructures are expected to provide different hosting configurations, in order to contribute to the diversity required by the moving target defense approach.

The reinforcement learning is performed in an iterative manner. Each time a new movement is performed, corresponding to a migration m_i applied to one or several components of a cloud service e, the framework assesses the resulting configuration/state, noted s, using the configuration verifier. This assessment is performed with respect to a set of vulnerability descriptions, noted $V = \{v_1, v_2, \dots\}$, where each vulnerability description can be specified as a logical formula combining a set of properties p_i. The vulnerability datasets correspond in our case again

to the official OVAL repository of vulnerability descriptions. This assessment allows to identify a subset of vulnerabilities noted $V'_{m_i} \in V$ related to the migration m_i applied to the cloud composite service e. For this, we consider the function $\Phi' : S \rightarrow V'$ which determines this subset of vulnerabilities under which the cloud composite service e is vulnerable for a given migration m_i to a cloud environment, as shown on Equation 6.3.

$$V'_{m_i} \leftarrow \Phi'(\Pi(impact'(m_i), state'(e))), m_i \in M \tag{6.1}$$

In this equation, we consider that the function $state' : E \rightarrow S$ takes a cloud composite service e as input and returns its current state s, while $impact : M \rightarrow S$ corresponds to a function taking a projected migration $m \in M$ as input and returning a state s that projects the affected characteristics corresponding to the migration. In the meantime, the function $\Pi : S \times S \rightarrow S$ takes a projected state $s_1 \in S$ together with a state $s_2 \in S$ as inputs and returns the state s_2 updated with the properties of s_1.

The moving target defense framework then calculates the reward r_{m_i} based on the subset of vulnerabilities V'_{m_i} related to the migration m_i applied to a component of the cloud composite service e. This reward takes into account the score of configuration vulnerabilities provided by the Common Vulnerability Score System (CVSS). It is given by Equation 6.2, where the *score* function provides the CVSS score for a given vulnerability v.

$$r_{m_i} = (\sum_{v \in V'_{m_i}} score(v))^{-1} \tag{6.2}$$

The reinforcement learning algorithm, by using this reward r_{m_i} and the current state as inputs, determines the action(s) that corresponds to the movement to be applied on the cloud composite service. Choosing a given movement means selecting an adequate migration m_i from the set of possible migrations $M = \{m_1, m_2, \dots\}$. The learning algorithm consists in calculating and updating a function, called value function (or Q function), given by $Q : S * M \rightarrow R$. This function consists to learn the value of an action, this later corresponding to a migration m_i in our case. At each migration m_i, the algorithm determines the reward r_{m_i} associated to the action together with the new state s' of the system, as given by Equations 6.3 and 6.4.

$$Q(s, m_i) \leftarrow Q(s, m_i) + \alpha[r_{m_i} + \gamma * max_{m'_i} Q(s', m'_i)) - Q(s, m_i)] \tag{6.3}$$

$$Q(s, m_i) \leftarrow Q(s, m_i) + \alpha[(\sum_{v \in V'_{m_i}} score(v))^{-1} + \gamma * max_{m'_i} Q(s', m'_i)) - Q(s, m_i)] \tag{6.4}$$

Considering that s' is the new configuration state and s is the previous one for the cloud composite service e, m_i is the chosen migration, and r_{m_i} is the reward received by the algorithm, α is a number between 0 and 1, called learning rate, and γ is the discount factor. The learning rate α determines how much the new calculated information will exceed the previous one. If $\alpha = 0$ the agent learns nothing. However, if $\alpha = 1$, the agent still ignores everything he has learned until now and will only take into account the latest learned information. The discount factor γ determines the importance of future rewards. A factor 0 only considers current rewards, while a factor close to 1 would also involve more long-term high rewards.

6.5 Performance evaluation

In this section, we will now describe the prototyping of our moving target defense strategy, as well as detail experimental results performed to quantify the performance and limits of our approach.

6.5.1 Prototyping and experimental setup

For our experimental evaluation, we implemented a dedicated prototype using the Python. We use vulnerability descriptions coming from the official OVAL repository, and we exploit the assessment framework developed in [133] on top of the SMT solver CVC4. During the experiments, we consider a TOSCA-based composite cloud service that consists of 3 web micro-services that interact with 2 containerized MySQL database instances as the back end. We then perform the assessment of a containerized MySQL database instance that migrates across a hundred of heterogeneous virtualized cloud infrastructures. The latter present diversified and heterogeneous configurations, including some configurations that may generate vulnerabilities (with different CVSS severity scores) after a resource migration. We have performed the experimental series on a regular laptop equipped with a 2 GHz Intel Core i5 processor and 8 GB of RAM memory.

6.5.2 Time convergence and diversity generated by the considered strategy

In a first series of experiments, we wanted to analyze the convergence time of our moving target defense strategy, as well as the diversity provided by this our when varying the percentage of the overall considered cloud environments. We quantified the time required to provide a policy that puts the system in its safest state. In the meantime, we have investigated the progression of diversity, when we involve additional cloud configurations. Figure 6.4 illustrates the results of these experiments, where the horizontal axis of the figure indicates the percentage of the considered cloud environments and the vertical axis of the figure represents respectively the average diversity and the convergence time. This figure therefore showcases the obtained average diversity, while varying the percentage of cloud environments allowed for performing the migrations related to the defense strategy. In addition, we consider the convergence time of the proposed strategy in seconds for the same percentage of cloud hosting environments.

In these experiments, we assume that all the migrations are possible amongst the considered cloud environments. The analysis of Figure 6.4 shows that the convergence time increases with the number of considered cloud environments. This number of environments determines the number of migrations enabled for a given cloud resource. This corresponds in this case to $n*(n-1)$ migrations assuming n is the number of cloud environments being taken into account. For instance, 10 environments allow 90 migrations, and the convergence of the strategy reinforcement learning algorithm takes a time period of 32.1 seconds. In the same figure we can notice that the diversity (on average) increases until the 50% of cloud providers, then it decreases as we introduce other cloud environments into the migration process. Intuitively, one might expect a curve that monotonically grows with the number of considered cloud environments, however, from the 50% we observe a continuous decrease in the diversity average, despite the increase of the number of providers involved in the migration. This can be explained in particular by the fact that our strategy favors migrations to the most heterogeneous environments at the beginning of the moving target defense process.

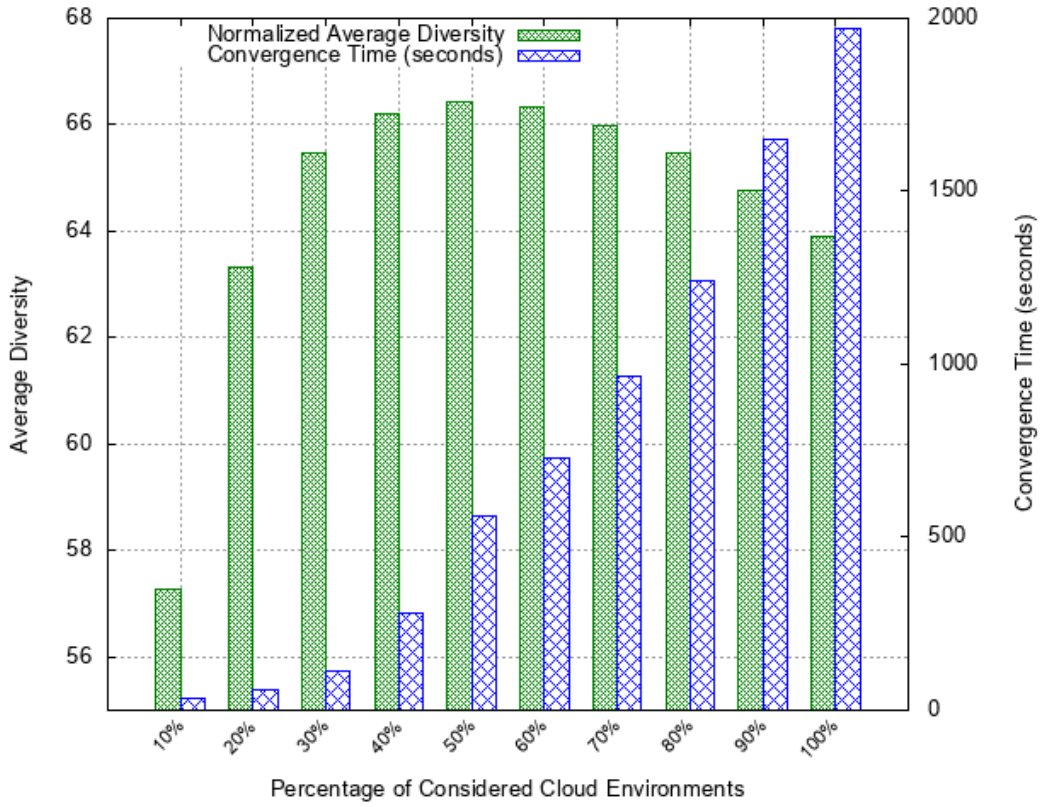


Figure 6.4: Convergence time of our moving target strategy and the average diversity with different percentages of considered cloud environments

6.5.3 Predictability of our moving target defense strategy

The objective of our moving target defense strategy is to confuse attackers by reducing the exploitability of the information retrieved through reconnaissance activities. In this way, the unpredictability of the movements, and then the one of the configuration of the cloud composite service contributes to reduce the probability of attack success. In our case, we wanted to introduce a metric that quantifies to what extend the prediction of the next configuration for a given cloud composite service is easy for potential attackers. In that context, we consider the experiment illustrated in Figure 6.5 that shows the predictability of the considered system according to the number of involved cloud configurations and the authorized migrations amongst cloud providers. We calculate the predictability using the following formula $Pr = 1/\prod((\sum_{i=1}^{n} cp_i)(\sum_{j=1}^{k} m_j))$. Where n is the number of the involved cloud providers $CP = \{cp_1, cp_2, \dots\}$ and k is the number of allowed migrations. In order to highlight the predictability, we have examined two scenarios. In the first scenario, we have looked at the predictability (represented in the vertical axis of Figure 6.5) considering the number of cloud configurations (represented in the horizontal axis of the first sub-figure of Figure 6.5) that are involved in the migration process. In the second scenario, we consider a risk threshold based on the cumulative CVSS score (represented in the

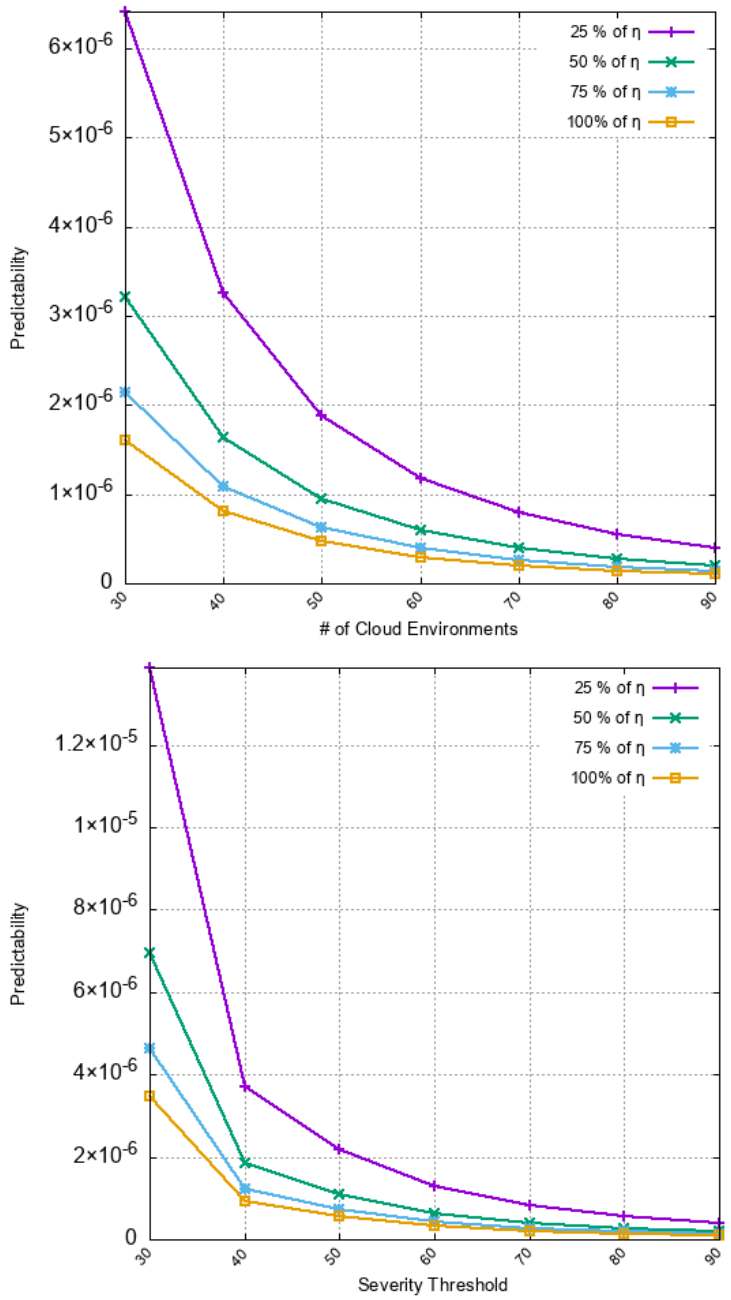


Figure 6.5: Predictability of the moving target defense strategy regarding the number of cloud environments and the authorized migrations

horizontal axis of the second sub-figure of Figure 6.5). In this case, we believe that the user accepts this threshold, which allows us to assume that all cloud configurations with a cumulative CVSS less than or equal to this threshold are allowed for migrations. Of course, the number of possible migrations amongst cloud providers may vary depending on portability, interoperability, cost and operation issues. In that context, we consider for the two scenarios mentioned above a given number of authorized migrations that we represent by η. For instance, the curve in blue assumes that only 25% migrations can be performed.

The analysis of the two sub-figures shows a significant decrease of the predictability when the number of considered cloud configurations increases, or as we enable a higher risk threshold for the moving target defense strategy. Similarly, the percentage of possible migrations is characterized by an important decrease in term of predictability. For instance, the case where we allow 100% of the migrations gives much lower predictability values, than when we tolerate only 25% of the migrations. When we look at the differences between these different percentages, we have also noticed that the differences are much more interesting, and they are becoming less and less when increasing both the number of cloud environments and the risk threshold. In addition, we can observe that the predictability decreases drastically at the beginning of each sub-figures, taking into account the risk threshold as the varying metric, and that afterward the shape of the curve becomes less and less important. In the meatime, considering the number of cloud configurations, the decrease in term of predictability appears to be progressive in our experiments.

6.5.4 Comparison between our strategy and a stochastic strategy

In order to compare our strategy to a baseline, we have proposed new series of experiments, shown in Figure 6.6, where we have compared the performance of our solution to a stochastic strategy, by looking at the average severity (represented in the vertical axis of the figure). The latter depends on the cumulative severity score related to the configurations generated by the migrations. In this figure, we have representd the average severity for each strategy and quantified the gain in term of risk represented by the difference between the average severities of the two strategies. By varying the number of cloud environments taken into account, we have observed that our moving target defense strategy generates a lower severity average compared to the baseline strategy. The latter performs migrations without necessarily considering existing vulnerabilities and their associated severity score. In the same manner we have observed that the more we increase the number of cloud environments involved in the experiments, the more we gain in terms of average severity, and this goes up to 20 environments. From this value, the risk gain starts to decrease until it reaches its lowest value, considering the total number of cloud environments for the experiments. The results come from the fact that our strategy favors, from the beginning of our experiments, cloud environments with fewer vulnerabilities or cloud configurations with vulnerabilities of the lowest severity. However, the baseline strategy does not take into account this aspect. On the other hand, by introducing a greater number of environments, our strategy is forced to migrate to cloud environments with significant severity in order to maintain unpredictability. It therefore adopts a behavior similar to the baseline strategy, which results in a progressive reduction of the risk gain, as we introduce further cloud configurations. As a conclusion, our strategy reduces the overall risk compared to the baseline strategy, in particular when we consider a relatively limited number of cloud configurations in the experiments.

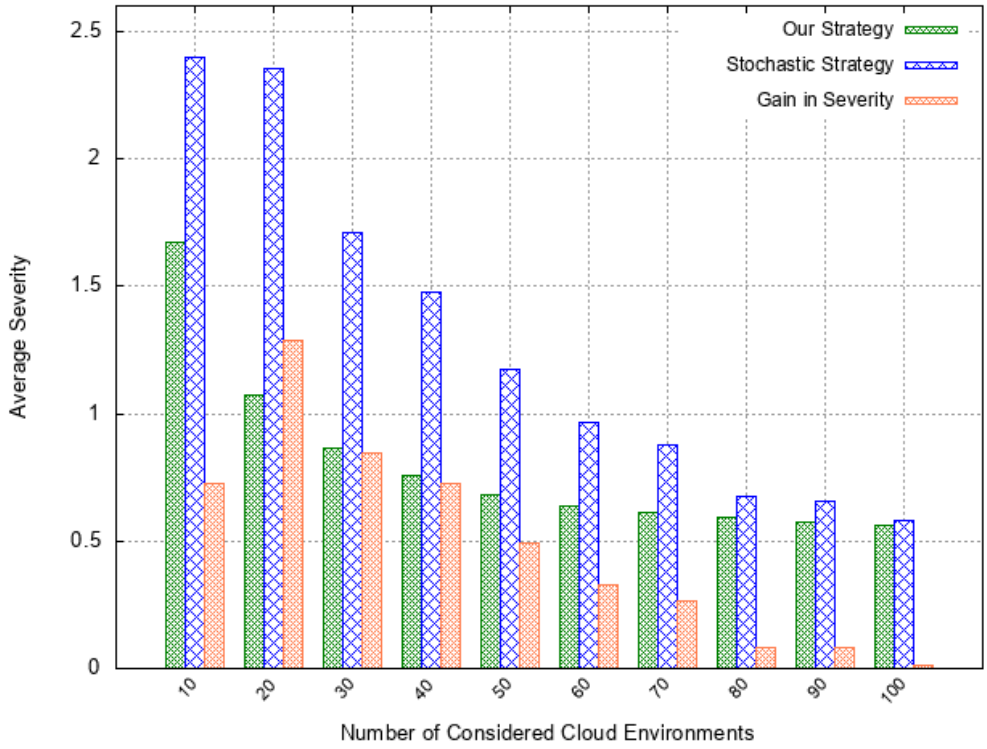


Figure 6.6: Average severity regarding our moving target defense strategy and a stochastic strategy

6.5.5 Cost of the considered strategy regarding the service unavailability

The movements that are performed by our moving target defense strategy may be performed periodically during specific time slots, according to a so-called time-based strategy, or based on some specific events, according to a so-called event-based strategy. This later may typically trigger movements based on events sent by security equipments, such as intrusion detection systems. Such equipements monitor the network and report any suspicious activities in the form of alerts. In that case, the number of migrations will depend on the number of alerts that occur over time. The costs of a time-based strategy are more predictable, as changes are performed in a periodic or pseudo-periodic manner. These migrations may generate service unavailability in order to perform them. In the experimental series shown on Figure 6.7, we wanted to quantify the cost of several time-based or event-based strategies with respect to the unavailability time (represented in the vertical axis of the figure), considering several bandwidth links. For that purpose, we have relied on the results given by [194], in order to quantify such unavailability for each moving target defense strategy. In Figure 6.7, α represents the number of events taken into account for each strategy, while β refers to the time slot between two successive migrations. By analyzing the results of this figure, we can notice that strategies that are characterized by short time slots before each migration require more bandwidth, and that they generate more unavailability time. However, such strategy contribute to make the information collected by

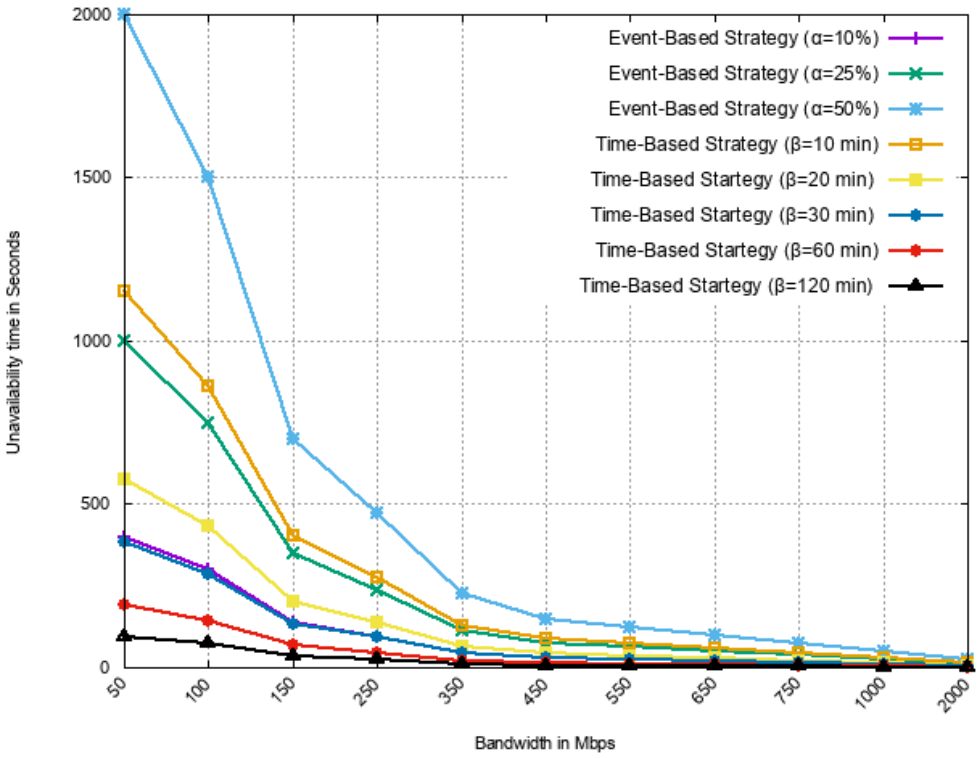


Figure 6.7: Unavailability time during the migration process considering multiple based-event and time-event moving target defense strategies

adversaries during reconnaissance activities to have a short validity time period. In this way, the system attack surface changes frequently over time, and is more difficult to be analyzed by attackers. Similarly, increasing the number of alerts taken into account by the event-based strategy increases the overall unavailability time due to migrations that are more numerous for the cloud composite service. In these experiments, we have considered such unavailability time as the cost related to a given moving target defense strategy, but other elements could be taken into account such as the service latency. Overall, these experiments show that the choice and parameterization of a given moving target defense strategy implies a trade-off between the frequency of resource migrations required for the defense and the available bandwidth for the operation of the service.

6.6 Synthesis

We have designed a moving target defense strategy that couples artificial intelligence with verification techniques. We have specified the considered framework which consists in two main blocks. A learning block based on reinforcement learning algorithms automates the selection of movements according to the rewards. These rewards are calculated based on the results of a verifier block, which exploits configuration verification techniques to assess the movements chosen

by the artificial intelligence. We have also mathematically formalized our solution by considering our previous modelling. The objective of our moving target defense strategy is to have an approach which makes the movements hard to predict for the attackers, while minimizing the risk of reaching vulnerable configurations in a cloud composite services. We have conducted several series of experiments based on a proof-of-concept prototype, in order to evaluate the performance of our solution. In that context, we have compared our approach to a baseline strategy, and taken into account the severity scores of vulnerabilities provided by a CVSS repository. The experimental results show that our strategy reduces the exposure to severe security attacks, while the overhead due to the assessment time is very limited. We have also evaluated the impact on the predictability of movements by attackers, which impacts on reconnaissance activities, and quantified the costs of the movements with respect to service unavailability.

Chapter 7

Conclusions and Future Work

Contents

7.1 General conclusions

The advent of the Internet and cloud computing has brought significant transformation to the manner in which data and resources are stored, processed, and accessed. Central to the concept of cloud computing are virtualization techniques, which facilitate the optimal allocation of resources and the delivery of scalable and adaptable services. Through virtualization, the abstraction of underlying hardware enables the creation of virtual machines or containers capable of supporting multiple operating systems and applications concurrently on a single physical server. This allows for more efficient utilization of hardware resources, which is essential for delivering the cost savings and scalability that cloud computing promises. As a result, cloud computing offers a wide range of services, such as storage, applications, and infrastructure, that can be accessed from anywhere through the internet. Additionally, cloud computing is highly scalable, allowing users to easily adjust their computing resources based on their needs. Moreover, the maturity of orchestration languages and the benefits of resource migrations have enabled further improvements in the composition of elaborated composite cloud services and enhancements in the performance of cloud infrastructures. These languages enable the automation of complex

processes that would otherwise be time-consuming and error-prone. This automation can help reduce the risk of human error and improve the reliability and consistency of cloud infrastructure. They also provide a high level of abstraction that can help simplify the process of managing cloud infrastructures and ease the management and coordination of the different components of cloud environments. Furthermore, orchestration languages improve the performance of cloud applications by enabling efficient resource allocation and load balancing. By automating the management of resources and optimizing the allocation of workloads, orchestration languages can help ensure that cloud applications are running at peak performance. Similarly, migration techniques are of particular importance for improving cloud performance in several ways. Firstly, they enable better resource utilization by allowing resource relocation to underutilized physical servers. Furthermore, migrations help to improve service availability by allowing resources to be moved across cloud infrastructures. This capability enables the continuity of services without experiencing downtime or service interruptions. Despite these advantages of technical migrations to cloud infrastructures, it is important to emphasize that these activities can generate new security threats. This may lead to vulnerabilities and attacks that compromise the security of cloud resources involved in the migration. They complement those that exist in traditional environments by exploiting features of cloud environments, such as auto-scaling.

The composition of cloud services, as well as orchestrated migrations combined with cloud characteristics, bring several benefits, particularly in terms of resource performance. However, these services come with their own set of threats and risks that must be carefully managed. The leverage of these threats can be particularly severe, as they can affect several security properties as well as the organizations reputation, their financial stability, and even their legal compliance. These threats are heightened by the presence of new vulnerabilities that may emerge as a result of resource migrations across heterogeneous cloud infrastructures and the underlying dynamic processes. These vulnerabilities can then lead to security attacks that may compromise the security of cloud resources. Such attacks may even be spread to other components or services, amplifying the impact of the security breaches. These migrations can, in certain cases, generate new vulnerabilities and thus lead the migrated resources to end up in unsafe states. Therefore, the security of cloud composite services may be significantly impacted by these threats and risks. Since cloud composite services are built by integrating multiple cloud services from different providers, they can be particularly vulnerable to attacks that exploit weaknesses in any of the services involved. This can lead to a range of security issues, including data breaches, service interruptions, and the unauthorized access or the alteration of sensitive data. To mitigate the security threats, several approaches propose solutions based on exogenous and endogenous security mechanisms. Exogenous security mechanisms refer to external security measures that are implemented by cloud service providers, while endogenous security mechanisms are internal security mechanisms that are implemented by the organization itself. Endogenous security mechanisms include the generation of resources with the necessary packages, the analysis of resource configuration, and ensuring resource verification. However, exogenous mechanisms cover several features, such as the virtualization of network functions, the building of security chains, hardware-based encryption systems, and audits of cloud platforms prior to migrations.

This thesis presents a proposal for automating the security of cloud composite services focusing on resource migrations. It is structured around three fundamental axes. The first axis entails the introduction of an automated SMT-based framework for evaluating cloud resources during

migrations. The second axis involves the design of a trusted third party to ensure the security of these resources within a distributed context. Lastly, the third axis leverages the moving target defense paradigm by integrating verification techniques with artificial intelligence algorithms, to elaborate a strategy for enhancing the security of such resources.

7.1.1 SMT-based security for migrations in cloud composite services

The first axis of the thesis corresponds to an SMT-based security framework for supporting migrations in cloud composite services, such as those orchestrated with the TOSCA language. The goal is to exploit verification techniques for automatically assessing the configuration changes that affect the resources of cloud services during their migration, and determining adequate countermeasures. The proposed security framework operates into three main phases. The first phase establishes a projection of the resource configuration induced by the migration, considering the changes that affect the migrated resource and its context. It then assesses this configuration by exploiting the knowledge provided by vulnerability descriptions. The third phase consists of determining potential countermeasures to be executed on the resource itself or activated in its environment to prevent the observed vulnerabilities and maintain the overall cloud composite service security. In order to demonstrate the effectiveness and the feasibility of our approach, we have developed a prototype using Python on the top of SMT solvers. We then performed extensive series of experiments using different assessment and remediation scenarios using existing vulnerability descriptions coming from the official OVAL repository, and corresponding to different operating systems. These experiments have shown that the highest assessment time (1.53 seconds) is observed with the Microsoft Windows Server 2012 R2 dataset, which also corresponds to the dataset with the highest number of vulnerability descriptions, meaning a total of 5344 OVAL vulnerability descriptions. The overall execution time required for the framework to both assess and select countermeasures, corresponding to an average time of around 3.20 seconds with the identification of a single countermeasure, and to an average time of 7.10 seconds for identifying all potential countermeasures. Finally, the obtained results compared to live migration baseline scenarios show the feasibility of our proposed solution.

7.1.2 Inter-cloud trusted third party for configuration security in orchestrated cloud services

The second axis leverages the previous framework to propose an architecture of an inter-cloud trusted third-party approach, called C3S-TTP (Composite Cloud Configuration Security-Trusted Third Party), in order to enhance the security of cloud composite services in a distributed context. The objective is to support the security of TOSCA-based cloud services, whose elementary resources may be subject to migration over time. In fact, this approach is motivated by the difficult and challenging aspects of analyzing configuration changes that are a result of partial sharing of the information regarding configurations amongst stakeholders, namely cloud tenants and cloud providers. Maintaining the security of cloud services during the migration of their resources requires increased transparency with respect to configurations before performing these migrations. This knowledge sharing contributes to better security management, with the preparation of an adequate destination environment from the cloud provider side, and the

optimal hardening of the migrated resource from the cloud tenant side. Cloud stakeholders, on the other hand, generally avoid sharing detailed technical configuration information about their resources and infrastructures, especially those related to security, or affecting security. The proposed solution therefore aims at preventing or restricting such configuration information disclosure amongst cloud tenants and cloud providers, while managing vulnerabilities that may occur during the migration of cloud resources. It relies on an inter-cloud trusted third-party architecture that exploits a third-party stakeholder for sharing configuration information amongst cloud tenants and cloud providers and assessing potential vulnerabilities. This trusted third party is triggered as soon as a cloud tenant requires the migration of one or more of its cloud resources. First, it collects configuration information related to the cloud resources that are candidates for migration. It then requests that cloud service providers get configuration information regarding potential hosting environments, including the resource versioning and parameterization of these cloud environments, as well as information regarding security mechanisms that may be implemented by the cloud providers. Considering this overall knowledge and using our Open Vulnerability and Assessment Language OVAL-based vulnerability assessment framework, the third party determines configuration vulnerabilities that may appear during the migration of resources over potential hosting environments. These assessment results enable the trusted third party to recommend or decline the migration of a given resource back to the cloud tenant, while reducing the disclosure of configuration information amongst involved cloud tenants and cloud providers. We have evaluated our approach through several experiments using a proof-of-concept prototype of the architecture, implemented on top of our security assessment framework, which exploits the open-source CVC4 SMT solver to analyze configurations and detect potential vulnerabilities based on OVAL datasets. We have shown to what extent the observability enabled by the trusted third party contributes to increase the performance of vulnerability prevention and to minimize risks related to unsafe configurations. The experimental results have shown that such observability reduces the security assessment time for all the considered systems (CentOS Linux, Red Hat Enterprise Linux, Microsoft Windows). In addition, the probability of having a vulnerable configuration decreases as the number of known configuration properties about the cloud environment increases. The difference between an observability of 10% and an observability of 100% is on average of 9.03% with CentOS Linux, of 13.6% with Red Hat Enterprise Linux, and of 42.5% with Microsoft Windows Server. In the same manner, increasing the observability reduces significantly the cost of activating security functions, when following a proactive security strategy.

7.1.3 Verified AI-based moving target defense strategy for securing cloud composite services

The last axis focuses on the design of a moving target defense strategy which couples reinforcement learning algorithms with verification techniques. It consists in continuously changing the attack surface by performing movements on the target. These movements are typically concretized by the migration of resources that are part of a cloud composite service. They are selected by artificial intelligence to minimize their predictability by attackers. The objective is to deceive these attackers into conducting and performing reconnaissance activities. However, the movements in themselves may introduce new configuration vulnerabilities that may be exploited

by attackers. Verification techniques are therefore used to minimize the attack surface, when movements are decided. The core of the moving target defense strategy relies on reinforcement learning algorithms, in particular Q-learning, where rewards are calculated based on the results of verification techniques. We have also mathematically formalized our solution by considering our previous modelling. The selection of a movement is triggered by the detection of suspicious events, or is performed periodically after a specific time interval. In order to quantify the effectiveness of our solution, we have conducted several series of experiments with a comparison to a baseline strategy. The obtained results show the reduction of the exposure to severe security attacks, with a vulnerability assessment introducing a limited overhead. They also illustrate a low impact on the predictability of movements, which contributes to counter reconnaissance activities performed by attackers, while verification techniques contribute to minimize the occurrence of vulnerabilities.

7.2 Research perspectives

As future work, we propose three main directions to explore (described on Figure 7.1), in order to improve and strengthen the security of cloud composite services, namely the coupling with threat intelligence, the contribution to green security, and the extension to edge computing.

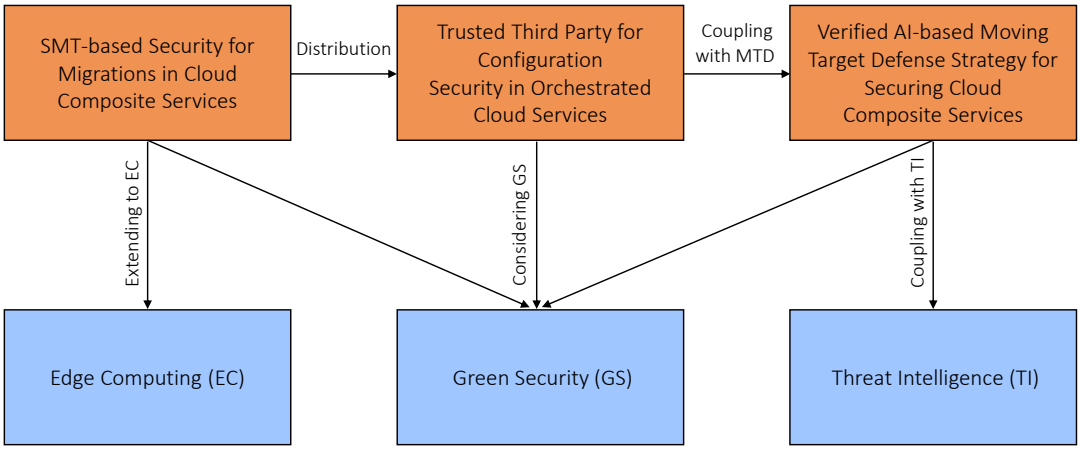


Figure 7.1: The overview of the research perspectives

7.2.1 Coupling with threat intelligence

The first axis aims to leverage threat intelligence to enhance our security approach. The attackers continuously improve their tactics, techniques, and procedures to be less predictable, more persistent, more resourceful, and better funded, which makes their activities difficult to detect and investigate. For this, threat intelligence aims to gather and aggregate knowledge and information and share it amongst the stakeholders in order to anticipate, prevent, detect,

and respond to security incidents. The main goal is to determine and identify specific threats and cyber criminals, analyze their methods, and therefore put in place adequate reactive and proactive defense mechanisms and strategies. In this regard, we plan to investigate the coupling of our solution with threat intelligence, in order to increase the performance of our framework by using new knowledge sources in the context of cloud composite services. For instance, the knowledge on attempted attacks can be exploited to prioritize the corrective operations with respect to vulnerabilities. In particular, the inter-cloud trusted third party could take benefits from this knowledge to improve its vulnerability assessment based on configuration information collected from the cloud tenant and the cloud provider. We may also adjust the moving target defense policy depending on such threat intelligence knowledge, in order to adapt the frequency or nature of movements.

7.2.2 Contribution to green security

The second axis is centered on green security in cloud environments. Along with its benefits, cloud computing has side effects that have a negative impact on the environment, such as carbon emissions. Even though there are efforts to make cloud infrastructures less harmful to the environment, they still need to be improved. Cloud providers try to reduce the environmental impact of their infrastructures by making them more energy-efficient and using less hardware. They can also get economies of scale by combining data centers and using advanced cooling and power management technologies to cut down on energy use. However, the growth of cloud computing has also led to an increase in the number of data centers, which can consume a significant amount of energy. This is exacerbated by migrations within these infrastructures and the activation and deactivation of security mechanisms to protect these resources. Such operations can result in activities that consume important resources, especially when resources are migrated across several cloud infrastructures. An important challenge is to reduce the resource consumption due to security mechanisms, including those used to protect cloud composite services. In particular, the choice of countermeasures should take into account this parameter, in order to factorize as much as possible the costs induced by their enforcement. For instance in some scenarios, the implementation of an endogenous mechanism, implemented into the cloud resource, might be preferred to an exogenous mechanism supported by the cloud infrastructure. In other cases, the exploitation of an exogenous mechanism, already activated to support other resources in the cloud infrastructure, might be favored to mutualize its usage.

7.2.3 Extension to edge computing

The last axis will focus on the placement of security mechanisms for protecting composite services by extending our solution to the case of edge computing. The purpose is to make the right decision about whether putting a given security mechanism at the edge or in a cloud data center, in order to efficiently secure these services. This may both concern the vulnerability assessment and the selection of countermeasures to remediate them. On the edge, the security mechanisms may be activated closer to the source of the data, rather than being sent to more a centralized location, such as a cloud data center. This enables faster and more efficient processing, and reduces the latency associated with transmitting data to data centers. In the meantime, the edge devices and their applications may also be more exposed to security attacks, due to their distributed nature,

making important to implement strong authentication and access controls, data encryption, and secure communication protocols to protect them. In that context, our work will consider security issues related to composite services in a edge-cloud continuum, and will investigate automated offloading strategies for securing them.

Bibliography

[1] P. M. Mell and T. Grance, "SP 800-145. The NIST Definition of Cloud Computing," Gaithersburg, MD, USA, Tech. Rep., 2011.

[2] T. S. Derek Palma. (2013) Topology and Orchestration Specification for Cloud Applications Version 1.0. Visited on 2023-03-20. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html

[3] X. Xu, X. Zhang, M. Khan, W. Dou, S. Xue, and S. Yu, "A Balanced Virtual Machine Scheduling Method for Energy-Performance Trade-offs in Cyber-Physical Cloud Systems," *Future Generation Computer Systems*, vol. 105, pp. 789–799, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17318927

[4] S. Basu, A. Bardhan, K. Gupta, P. Saha, M. Pal, M. Bose, K. Basu, S. Chaudhury, and P. Sarkar, "Cloud Computing Security Challenges Solutions - A survey," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, 2018, pp. 347–356.

[5] R. Ranjan, B. Benatallah, S. Dustdar, and M. P. Papazoglou, "Cloud Resource Orchestration Programming: Overview, Issues, and Directions," *IEEE Internet Computing*, vol. 19, no. 5, pp. 46–56, Sep. 2015.

[6] R. Ranjan, B. Benatallah, S. Dustdar, and M. P. Papazoglou, "Cloud Resource Orchestration Programming: Overview, Issues, and Directions," *IEEE Internet Computing*, vol. 19, no. 5, pp. 46–56, 2015.

[7] O. Tomarchio, D. Calcaterra, and G. Di Modica, "Cloud Resource Orchestration in the Multi-Cloud Landscape: a Systematic Review of Existing Frameworks," *Journal of Cloud Computing*, vol. 9, p. 49, 09 2020.

[8] R. Dukaric and M. B. Juric, "BPMN Extensions for Automating Cloud Environments Using a Two-Layer Orchestration Approach," *Journal of Visual Languages Computing*, vol. 47, pp. 31–43, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1045926X18300429

[9] C. Liu, B. T. Loo, and Y. Mao, "Declarative Automated Cloud Resource Orchestration," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2038916.2038942

[10] D. Weerasiri, M. Barukh, B. Benatallah, Q. Sheng, and R. Ranjan, "A Taxonomy and Survey of Cloud Resource Orchestration Techniques," *ACM Computing Surveys*, vol. 50, pp. 1–41, 05 2017.

[11] G. Katsaros, M. Menzel, A. Lenk, J. R. Revelant, R. Skipp, and J. Eberhardt, "Cloud Application Portability with TOSCA, Chef and Openstack," in *Proceedings of IEEE International Conference on Cloud Engineering*, March 2014, pp. 295–302.

[12] P. Lipton, D. Palma, M. Rutkowski, and D. A. Tamburri, "The Efficacy of Live Virtual Machine Migrations over the Internet," *IEEE Cloud Computing*, vol. 5, no. 02, pp. 37–47, March 2018.

[13] "A Comparison Between TOSCA and OpenStack HOT through Cloud Patterns Composition," vol. 8, no. 4, p. 299–311, Jan. 2017. [Online]. Available: https://doi.org/10.1504/IJGUC.2017.088259

[14] (2020) Welcome to the OpenStack wiki. Visited on 2023-04-11. [Online]. Available: https://docs.openstack.org/ussuri/

[15] "AWS CloudFormation - Infrastructure as Code AWS Resource Provisioning," library Catalog: aws.amazon.com. [Online]. Available: https://aws.amazon.com/cloudformation/

[16] J. Carrasco, J. Cubo, and E. Pimentel, "Towards a Flexible Deployment of Multi-Cloud Applications Based on TOSCA and CAMP," vol. 508, 02 2015, pp. 278–286.

[17] A. Atrey, H. Moens, G. Van Seghbroeck, B. Volckaert, and F. De Turck, "An Overview of the OASIS TOSCA Standard: Topology and Orchestration Specification for Cloud Applications," 09 2015.

[18] A. Brogi, J. Soldani, and P. Wang, "TOSCA in a Nutshell: Promises and Perspectives," in *Proceedings of the European Conference on Service-Oriented and Cloud Computing*, vol. 8745, 09 2014, pp. 171–186.

[19] P. Hirmer, U. Breitenbücher, T. Binz, and F. Leymann, "Automatic Topology Completion of TOSCA-based Cloud Applications," in *Proceedings of the Informatik 2014*, E. Plödereder, L. Grunske, E. Schneider, and D. Ull, Eds. Bonn: Gesellschaft für Informatik e.V., 2014, pp. 247–258.

[20] D. Calcaterra, V. Cartelli, G. Di Modica, and O. Tomarchio, "Combining TOSCA and BPMN to Enable Automated Cloud Service Provisioning," in *Proceedings of the 7th International Conference on Cloud Computing and Services Science*. Setubal, PRT: SCITEPRESS - Science and Technology Publications, Lda, 2017, p. 187–196. [Online]. Available: https://doi.org/10.5220/0006304701870196

[21] S. Liyanage, S. Khaddaj, and J. Francik, "Virtual Machine Migration Strategy in Cloud Computing," 08 2015, pp. 147–150.

[22] R. Boutaba, Q. Zhang, and M. F. Zhani, *Virtual Machine Migration in Cloud Computing Environments: Benefits, Challenges, and Approaches*, 01 2013, pp. 383–408.

[23] M. R. Desai and H. B. Patel, "Efficient Virtual Machine Migration in Cloud Computing," in *Proceedings of the fifth International Conference on Communication Systems and Network Technologies*, 2015, pp. 1015–1019.

[24] E. Harney, S. Goasguen, J. Martin, M. Murphy, and M. Westall, "The Efficacy of Live Virtual Machine Migrations over the Internet." New York, NY, USA: Association for Computing Machinery, 2007. [Online]. Available: https://doi.org/10.1145/1408654.1408662

[25] M. Mesbahi and A. Rahmani, "Load Balancing in Cloud Computing: A State of the Art Survey," *International Journal of Modern Education and Computer Science*, vol. 8, pp. 64–78, 03 2016.

[26] J. Hu, J. Gu, G. Sun, and T. Zhao, "A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment," *Parallel Architectures, Algorithms and Programming, International Symposium on*, pp. 89–96, 12 2010.

[27] L. Youseff, M. Butrico, and D. Da Silva, "Toward a Unified Ontology of Cloud Computing," in *Grid Computing Environments Workshop*, 2008, pp. 1–10.

[28] R. Hu, G. Liu, J. Jiang, and L. Wang, "A New Resources Provisioning Method Based on QoS Differentiation and VM Resizing in IaaS," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–9, 10 2015.

[29] Z. Zhang, Z. Li, K. Wu, D. Li, H. Li, Y. Peng, and X. Lu, "VMThunder: Fast Provisioning of Large-Scale Virtual Machine Clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3328–3338, Dec 2014.

[30] K. Takahashi, K. Sasada, and T. Hirofuchi, "A Fast Virtual Machine Storage Migration Technique Using Data Deduplication," in *Proceedings of the Third International Conference on Cloud Computing, GRIDs, and Virtualization*, 2012.

[31] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Improving Virtual Machine Migration in Federated Cloud Environments," in *Proceedings of the 2nd International Conference on Evolving Internet*, Sep. 2010, pp. 61–67.

[32] W. Cerroni, "Multiple Virtual Machine Live Migration in Federated Cloud Systems," in *Proceedings of the IEEE Conference on Computer Communications Workshops*, April 2014, pp. 25–30.

[33] Y. Lu, X. Xu, and J. Xu, "Development of a Hybrid Manufacturing Cloud," *Journal of Manufacturing Systems*, vol. 33, pp. 551–566, 05 2014.

[34] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," *Internet Computing, IEEE*, vol. 13, pp. 14 – 22, 11 2009.

[35] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live Migration of Virtual Machines," in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design amp; Implementation - Volume 2*. USA: USENIX Association, 2005, p. 273–286.

[36] T.-Y. Wu, N. Guizani, and J.-S. Huang, "Live Migration Improvements by Related Dirty Memory Prediction in Cloud Computing," *Journal of Network and Computer Applications*, vol. 90, pp. 83 – 89, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517301133

[37] A. Shribman and B. Hudzia, "Pre-Copy and Post-Copy VM Live Migration for Memory Intensive Applications," in *Proceedings of the Euro-Par 2012: Parallel Processing Workshops*, I. Caragiannis, M. Alexander, R. M. Badia, M. Cannataro, A. Costan, M. Danelutto, F. Desprez, B. Krammer, J. Sahuquillo, S. L. Scott, and J. Weidendorfer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 539–547.

[38] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast Transparent Migration for Virtual Machines," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*. USA: USENIX Association, 2005, p. 25.

[39] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi, "Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension," in *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 73–83.

[40] J. Kim, D. Chae, J. Kim, and J. Kim, "Guide-copy: Fast and Silent Migration of Virtual Machine for Datacenters," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Nov 2013, pp. 1–12.

[41] M. Noshy, A. Ibrahim, and H. Ali, "Optimization of Live Virtual machine Migration in Cloud Computing: A Survey and Future Directions," *Journal of Network and Computer Applications*, vol. 110, pp. 1–10, 03 2018.

[42] P. Svärd, B. Hudzia, J. Tordsson, and E. Elmroth, "Evaluation of Delta Compression Techniques for Efficient Live Migration of Large Virtual Machines," in *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ser. VEE '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 111–120. [Online]. Available: https://doi.org/10.1145/1952682.1952698

[43] Y. Ma, H. Wang, J. Dong, Y. Li, and S. Cheng, "ME2: Efficient Live Migration of Virtual Machine with Memory Exploration and Encoding," in *2012 IEEE International Conference on Cluster Computing*, 2012, pp. 610–613.

[44] H. Liu, H. Jin, X. Liao, C. Yu, and C. Xu, "Live Virtual Machine Migration via Asynchronous Replication and State Synchronization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 12, pp. 1986–1999, 2011.

[45] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live Migration of Virtual Machine Based on Full System Trace and Replay." New York, NY, USA: Association for Computing Machinery, 2009. [Online]. Available: https://doi.org/10.1145/1551609.1551630

[46] P. Riteau, C. Morin, and T. Priol, "Shrinker: Efficient Wide-Area Live Virtual Machine Migration using Distributed Content-Based Addressing," INRIA, Research Report RR-7198, Feb. 2010. [Online]. Available: https://hal.inria.fr/inria-00454727

[47] M. Li, M. Zheng, and X. Hu, "Template-based Memory Deduplication Method for Inter-Data Center Live Migration of Virtual Machines," in *Proceedings of the International Conference on Cloud Engineering*, March 2014, pp. 127–134.

[48] M. Zheng and X. Hu, "Template-Based Migration Between Data Centers Using Distributed Hash Tables," in *Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery*, Aug 2015, pp. 2443–2447.

[49] X. Zhang, Z. Huo, J. Ma, and D. Meng, "Exploiting Data Deduplication to Accelerate Live Virtual Machine Migration," in *Proceedings of the IEEE International Conference on Cluster Computing*, Sep. 2010, pp. 88–96.

[50] P. Svard, J. Tordsson, B. Hudzia, and E. Elmroth, "High Performance Live Migration through Dynamic Page Transfer Reordering and Compression," in *Proceedings of the IEEE Third International Conference on Cloud Computing Technology and Science*, Nov 2011, pp. 542–548.

[51] M. Hines and K. Gopalan, "Post-copy Based Live Virtual Machine Migration Using Pre-Paging and Dynamic Self-Ballooning," in *Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE'09*, 01 2009, pp. 51–60.

[52] S. Sahni and V. Varma, "A Hybrid Approach to Live Migration of Virtual Machines," in *Proceedings of the IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, 2012, pp. 1–5.

[53] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi, "Reactive Consolidation of Virtual Machines Enabled by Postcopy Live Migration," in *Proceedings of the 5th International Workshop on Virtualization Technologies in Distributed Computing.* New York, NY, USA: Association for Computing Machinery, 2011, p. 11–18. [Online]. Available: https://doi.org/10.1145/1996121.1996125

[54] Bhagyalakshmi and D. Malhotra, "A Critical Survey of Virtual Machine Migration Techniques in Cloud Computing," in *Proceedings of the first International Conference on Secure Cyber Computing and Communication (ICSCCC)*, Dec 2018, pp. 328–332.

[55] U. Deshpande and K. Keahey, "Traffic-Sensitive Live Migration of Virtual Machines," in *Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2015, pp. 51–60.

[56] U. Deshpande, D. Chan, S. Chan, K. Gopalan, and N. Bila, "Scatter-Gather Live Migration of Virtual Machines," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 196–208, 2018.

[57] U. Deshpande, D. Chan, T. Guh, J. Edouard, K. Gopalan, and N. Bila, "Agile Live Migration of Virtual Machines," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016, pp. 1061–1070.

[58] A. Mashtizadeh, E. Celebi, T. Garfinkel, and M. Cai, "The Design and Evolution of Live Storage Migration in VMware ESX," in *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*. USA: USENIX Association, 2011, p. 14.

[59] A. J. Mashtizadeh, M. Cai, G. Tarasuk-Levin, R. Koller, T. Garfinkel, and S. Setty, "XvMotion: Unified Virtual Machine Migration over Long Distance," in *Proceedings of the USENIX Conference on USENIX Annual Technical Conference*. USA: USENIX Association, 2014, p. 97–108.

[60] K. K. Ramakrishnan, P. Shenoy, and J. Van der Merwe, "Live Data Center Migration across WANs: A Robust Cooperative Context Aware Approach," in *Proceedings of the 2007 SIGCOMM Workshop on Internet Network Management*. New York, NY, USA: Association for Computing Machinery, 2007, p. 262–267. [Online]. Available: https://doi.org/10.1145/1321753.1321762

[61] D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G. Varghese, G. M. Voelker, and A. Vahdat, "Difference Engine: Harnessing Memory Redundancy in Virtual Machines," *Communications of the ACM*, vol. 53, no. 10, p. 85–93, oct 2010. [Online]. Available: https://doi.org/10.1145/1831407.1831429

[62] K. R. Jayaram, C. Peng, Z. Zhang, M. Kim, H. Chen, and H. Lei, "An Empirical Analysis of Similarity in Virtual Machine Images," in *Proceedings of the Middleware 2011 Industry Track Workshop*. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2090181.2090187

[63] P. Kokkinos, D. Kalogeras, A. Levin, and M. Varvarigos, "Survey: Live Migration and Disaster Recovery over Long-Distance Networks," *ACM Computing Surveys*, vol. 49, pp. 1–36, 07 2016.

[64] S. Alshathri, B. Ghita, and N. Clarke, "Sharing with Live Migration Energy Optimization Scheduler for Cloud Computing Data Centers," *Future Internet*, vol. 10, p. 86, 09 2018.

[65] M. Guazzone, C. Anglano, and M. Canonico, "Exploiting VM Migration for the Automated Power and Performance Management of Green Cloud Computing Systems," in *Proceedings of the Energy Efficient Data Centers*. Springer-Verlag, 05 2012, pp. 81–92.

[66] W. Cerroni and F. Esposito, "Optimizing Live Migration of Multiple Virtual Machines," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 1096–1109, 2018.

[67] J. Narantuya, H. Zang, and H. Lim, "Service-Aware Cloud-to-Cloud Migration of Multiple Virtual Machines," *IEEE Access*, vol. 6, pp. 76 663–76 672, 2018.

[68] W. Zhang, Y. Chen, X. Gao, Z. Mo, Q. Zheng, and Z. Lu, "Cluster-Aware Virtual Machine Collaborative Migration in Media Cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2808–2822, Oct 2017.

[69] S. Al-Kiswany, D. Subhraveti, P. Sarkar, and M. Ripeanu, "VMFlock: Virtual Machine Co-Migration for the Cloud," in *Proceedings of the 20th International Symposium on High Performance Distributed Computing*. New York, NY, USA: Association for Computing Machinery, 2011, p. 159–170. [Online]. Available: https://doi.org/10.1145/1996130.1996153

[70] S. Hacking and B. Hudzia, "Improving the Live Migration Process of Large Enterprise Applications," in *Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing*. New York, NY, USA: Association for Computing Machinery, 2009, p. 51–58. [Online]. Available: https://doi.org/10.1145/1555336.1555346

[71] J. Roemer, M. Groman, Z. Yang, Y. Wang, C. C. Tan, and N. Mi, "Improving Virtual Machine Migration via Deduplication," in *Proceedings of the 11th International Conference on Mobile Ad Hoc and Sensor Systems*, 2014, pp. 702–707.

[72] M. Hines, U. Deshpande, and K. Gopalan, "Post-copy Live Migration of Virtual Machines," *Operating Systems Review*, vol. 43, pp. 14–26, 07 2009.

[73] G. Verma and R. Sushil, "Performance Analysis of Live and Offline VM Migration Using KVM," *International Journal of Modern Education and Computer Science*, vol. 11, pp. 50–57, 11 2016.

[74] B. Ivanovska, S. Ristov, M. Kostoska, and M. Gusev, "Using the P-TOSCA Model for Energy Efficient Cloud," in *Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2015, pp. 245–249.

[75] E. Aguiar, Y. Zhang, and M. Blanton, *An Overview of Issues and Recent Developments in Cloud Computing and Storage Security*. New York, NY: Springer New York, 2014, pp. 3–33. [Online]. Available: https://doi.org/10.1007/978-1-4614-3296-8_1

[76] Z. Tari, X. Yi, U. Premarathne, P. Bertok, and I. Khalil, "Security and Privacy in Cloud Computing: Vision, Trends, and Challenges," *IEEE Cloud Computing*, vol. 2, pp. 30–38, 03 2015.

[77] Y. Al-Issa, M. A. Ottom, and A. Tamrawi, "eHealth Cloud Security Challenges: A Survey," *Journal of Healthcare Engineering*, vol. 2019, 2019.

[78] M. Pearce, S. Zeadally, and R. Hunt, "Virtualization: Issues, Security Threats, and Solutions," *ACM Comput. Surv.*, vol. 45, no. 2, mar 2013. [Online]. Available: https://doi.org/10.1145/2431211.2431216

[79] H. Takabi, J. B. Joshi, and G.-J. Ahn, "Security and Privacy Challenges in Cloud Computing Environments," *IEEE Security Privacy*, vol. 8, no. 6, pp. 24–31, 2010.

[80] B. Asvija, R. Eswari, and M. Bijoy, "Security in Hardware Assisted Virtualization for Cloud Computing—State of the Art Issues and Challenges," *Computer Networks*, vol. 151, pp. 68–92, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618302998

[81] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, "Security and Privacy in Cloud Computing: A Survey," in *Proceedings of the Sixth International Conference on Semantics, Knowledge and Grids*, 2010, pp. 105–112.

[82] P. J. Sun, "Privacy Protection and Data Security in Cloud Computing: A Survey, Challenges, and Solutions," *IEEE Access*, vol. 7, pp. 147 420–147 452, 2019.

[83] S. Kumari, K. Solanki, S. Dalal, and A. Dhankhar, "Analysis Of Cloud Computing Security Threats and Countermeasures," in *Proceedings of the 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2022, pp. 1–6.

[84] O. Nagesh, T. Kumar, and V. Venkateswararao, "A Survey on Security Aspects of Server Virtualization in Cloud Computing," *International Journal of Electrical and Computer Engineering*, vol. 7, pp. 1326–1336, 06 2017.

[85] M. S. Dildar, N. Khan, J. B. Abdullah, and A. S. Khan, "Effective Way to Defend the Hypervisor Attacks in Cloud Computing," in *In the proceedings of the 2nd International Conference on Anti-Cyber Crimes (ICACC)*, 2017, pp. 154–159.

[86] O. Yevsieieva and S. M. Helalat, "Analysis of the Impact of the Slow HTTP DOS and DDOS Attacks on the Cloud Environment," in *In the proceedings of the 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC ST)*, 2017, pp. 519–523.

[87] S. Dong, K. Abbas, and R. Jain, "A Survey on Distributed Denial of Service (DDoS) Attacks in SDN and Cloud Computing Environments," *IEEE Access*, vol. 7, pp. 80 813–80 828, 2019.

[88] R. Doshi and V. Kute, "A Review Paper on Security Concerns in Cloud Computing and Proposed Security Models," in *In the proceedings of the International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 2020, pp. 1–4.

[89] L. Coppolino, S. D'Antonio, G. Mazzeo, and L. Romano, "Cloud Security: Emerging Threats and Current Solutions," *Computers Electrical Engineering*, vol. 59, pp. 126–140, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790616300544

[90] G. Zhu, Y. Yin, R. Cai, and K. Li, "Detecting Virtualization Specific Vulnerabilities in Cloud Computing Environment," in *In the proceedings of the 10th International Conference on Cloud Computing (CLOUD)*, 2017, pp. 743–748.

[91] N. Rakotondravony, B. Taubmann, W. Mandarawi, E. Weishäupl, P. Xu, B. Kolosnjaji, M. Protsenko, H. Meer, and H. Reiser, "Classifying Malware Attacks in IaaS Cloud Environments," *Journal of Cloud Computing*, vol. 6, 12 2017.

[92] N. Almutairy, K. Al-Shqeerat, and H. AlHamad, "A Taxonomy of Virtualization Security Issues in Cloud Computing Environments," *Indian Journal of Science and Technology*, vol. 12, p. 19, 01 2019.

[93] J.-R. Yeh, H.-C. Hsiao, and A.-C. Pang, "Migrant Attack: A Multi-resource DoS Attack on Cloud Virtual Machine Migration Schemes," in *In the procedding of the 11th Asia Joint Conference on Information Security (AsiaJCIS)*, 2016, pp. 92–99.

[94] X. Gao, J. Xiao, H. Wang, and A. Stavrou, "Understanding the Security Implication of Aborting Virtual Machine Live Migration," *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, pp. 1275–1286, 2022.

[95] L. Coppolino, S. D'Antonio, G. Mazzeo, and L. Romano, "Cloud Security: Emerging Threats and Current Solutions," *Computers Electrical Engineering*, vol. 59, pp. 126–140, 2017.

[96] M. Barrère, R. Badonnel, and O. Festor, "Supporting Vulnerability Awareness in Autonomic Networks and Systems with OVAL," in *Proceedings of the 7th International Conference on Network and Service Management*, 2011, pp. 1–8.

[97] M. Barrére, R. Badonnel, and O. Festor, "A SAT-based Autonomous Strategy for Security Vulnerability Management," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–9.

[98] K. A. Torkura, F. Cheng, and C. Meinel, "A Proposed Framework for Proactive Vulnerability Assessments in Cloud Deployments," in *Proceedings of the 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2015, pp. 51–57.

[99] M. Compastié, R. Badonnel, O. Festor, and R. He, "A TOSCA-Oriented Software-Defined Security Approach for Unikernel-Based Protected Clouds," in *Proceedings of the IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 151–159.

[100] M. De Benedictis and A. Lioy, "Integrity Verification of Docker Containers for a Lightweight Cloud Environment," *Future Generation Computer Systems*, vol. 97, pp. 236–246, 2019.

[101] S. Ghavamnia, T. Palit, A. Benameur, and M. Polychronakis, "Confine: Automated System Call Policy Generation for Container Attack Surface Reduction," in *International Symposium on Recent Advances in Intrusion Detection*, 2020.

[102] M. Anisetti, C. A. Ardagna, and E. Damiani, "Security Certification of Composite Services: A Test-Based Approach," in *Proceedings of the IEEE 20th International Conference on Web Services*, 2013, pp. 475–482.

[103] F. Kerschbaum and P. Robinson, "Security Architecture for Virtual Organizations of Business Web Services," *J. Syst. Archit.*, vol. 55, no. 4, p. 224–232, Apr. 2009. [Online]. Available: https://doi.org/10.1016/j.sysarc.2008.10.001

[104] M. Anisetti, C. A. Ardagna, E. Damiani, and F. Gaudenzi, "A Semi-Automatic and Trustworthy Scheme for Continuous Cloud Service Certification," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 30–43, 2020.

[105] N. Xi, C. Sun, J. Ma, and Y. Shen, "Secure Service Composition with Information Flow Control in Service Clouds," *Future Generation Computer Systems*, vol. 49, pp. 142–148, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X14002696

[106] G. Hurel, R. Badonnel, A. Lahmadi, and O. Festor, "Towards Cloud-Based Compositions of Security Functions for Mobile Devices," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 578–584.

[107] S. Seeber and G. D. Rodosek, "Improving Network Security Through SDN in Cloud Scenarios," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM) and Workshop*, 2014, pp. 376–381.

[108] M. Pattaranantakul, R. He, Z. Zhang, A. Meddahi, and P. Wang, "Leveraging Network Functions Virtualization Orchestrators to Achieve Software-Defined Access Control in the Clouds," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1.

[109] J.-L. Luo, S.-Z. Yu, and S.-J. Peng, "SDN/NFV-Based Security Service Function Tree for Cloud," *IEEE Access*, vol. 8, pp. 38 538–38 545, 2020.

[110] N. Schnepf, R. Badonnel, A. Lahmadi, and S. Merz, "Automated Verification of Security Chains in Software-Defined Detworks with Synaptic," in *Proceedings of the IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–9.

[111] N. Schnepf, R. Badonnel, A. Lahmadi, and S. Merz, "Synaptic: A Formal Checker for SDN-based Security Policies, year=2018," in *In the Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, pp. 1–2.

[112] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan, "Conflict Classification and Analysis of Distributed Firewall Policies," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 10, pp. 2069–2084, 2005.

[113] X. Wang, X. Chen, Y. Wang, and L. Ge, "An Efficient Scheme for SDN State Consistency Verification in Cloud Computing Environment," *Concurrency and Computation: Practice and Experience*, vol. 32, 07 2019.

[114] A. Celesti, A. Salici, M. Villari, and A. Puliafito, "A Remote Attestation Approach for a Secure Virtual Machine Migration in Federated Cloud Environments," in *Proceedings of the First International Symposium on Network Cloud Computing and Applications*, 2011, pp. 99–106.

[115] M. Aslam, C. Gehrmann, and M. Björkman, "Security and Trust Preserving VM Migrations in Public Clouds," in *Proceedings of the IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012, pp. 869–876.

[116] B. Danev, R. J. Masti, G. O. Karame, and S. Capkun, "Enabling Secure VM-vTPM Migration in Private Clouds," in *Proceedings of the 27th Annual Computer Security Applications Conference*. New York, NY, USA: Association for Computing Machinery, Dec. 2011, pp. 187–196. [Online]. Available: https://doi.org/10.1145/2076732.2076759

[117] K. W. Ullah, A. S. Ahmed, and J. Ylitalo, "Towards Building an Automated Security Compliance Tool for the Cloud," in *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2013, pp. 1587–1593.

[118] U. M. Ismail, S. Islam, and H. Mouratidis, "Cloud Security Audit for Migration and Continuous Monitoring," in *Proceedings of the IEEE Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 1081–1087.

[119] U. M. Ismail and S. Islam, "A Unified Framework for Cloud Security Transparency and Audit," *Journal of Information Security and Applications*, vol. 54, p. 102594, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212620307626

[120] J. Sighom, P. Zhang, and L. You, "Security Enhancement for Data Migration in the Cloud," *Future Internet*, vol. 9, 06 2017.

[121] R. Asif, "Securing Cloud Hypervisors: A Survey of the Threats, Vulnerabilities, and Countermeasures," *Security and Communication Networks*, vol. 2018, 06 2018.

[122] M. C. S. Filho, C. de Castro Monteiro, P. R. M. Inácio, and M. M. Freire, "Approaches for Optimizing Virtual Machine Placement and Migration in Cloud Environments: A Survey," *Journal on Parallel Distributed Computing*, vol. 111, pp. 222–250, 2018.

[123] N. S. Dey and T. Gunasekhar, "A Comprehensive Survey of Load Balancing Strategies Using Hadoop Queue Scheduling and Virtual Machine Migration," *IEEE Access*, vol. 7, pp. 92 259–92 284, 2019.

[124] D. Fernandes, L. Soares, J. Gomes, M. Freire, and P. Inácio, "Security Issues in Cloud Environments - A Survey," *Int. J. Inf. Secur.: Security in Cloud Computing*, p. 113–170, 04 2013.

[125] J. Baker, M. Hansbury, and D. Haynes, "The OVAL Language Specification, Version 5.11.2," *MITRE Corporation*, 2016.

[126] D. Waltermire, S. Quinn, K. Scarfone, and A. Halbardier, "The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP version 1.3," *NIST Special Publication*, vol. 800, p. 126, 2018.

[127] H. Barbosa, C. Barrett, F. Bobot, and M. Brain. (2020) About CVC4. Visited on 2023-04-06. [Online]. Available: https://cvc4.github.io/

[128] D. Booth H., Rike and G. Witte, "The National Vulnerability Database (NVD): Overview, ITL Bulletin, National Institute of Standards and Technology," 2020, visited on 2023-06-09. [Online]. Available: https://tsapps.nist.gov/publication

[129] S. Akoush, R. Sohan, A. Rice, A. W. Moore, and A. Hopper, "Predicting the Performance of Virtual Machine Migration," in *Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (IEEE MASCOTS)*. IEEE Computer Society, 2010, p. 37–46.

[130] R. Pellegrini, P. Rottmann, and G. Strieder, "Preventing Vendor Lock-ins via an Interoperable Multi-cloud Deployment Approach," in *Proc. of the 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, IEEE, Ed., 2017, pp. 382–387.

[131] J. Opara-Martins, R. Sahandi, and F. Tian, "Critical Analysis of Vendor lock-in and its Impact on Cloud Computing Migration: a Business Perspective," *Journal of Cloud Computing*, vol. 5, 04 2016.

[132] R. Kumar and R. Goyal, "On Cloud Security Requirements, Threats, Vulnerabilities and Countermeasures: A Survey," *Computer Science Review*, vol. 33, pp. 1–48, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013718302065

[133] M. Oulaaffart, R. Badonnel, and C. Bianco, "An Automated SMT-based Security Framework for Supporting Migrations in Cloud Composite Services," in *Proc. of the IEEE Network Operations and Management Symposium (NOMS)*, IEEE, Ed., 2022.

[134] J. O. Martins, R. Sahandi, and F. Tian, "Critical Analysis of Vendor Lock in and its Impact on Cloud Computing Migration: a Business Perspective," *Journal of Cloud Computing*, vol. 5, pp. 1–18, 2016.

[135] T. Nodehi, R. Jardim-Goncalves, A. Zutshi, and A. Grilo, "ICIF: an Inter-cloud Interoperability Framework for Computing Resource Cloud Providers in Factories of the Future," *International Journal of Computer Integrated Manufacturing*, vol. 30, no. 1, pp. 147–157, 2017.

[136] C. Ramalingam and P. Mohan, "Addressing Semantics Standards for Cloud Portability and Interoperability in Multi Cloud Environment," *Symmetry*, vol. 13, no. 2, 2021.

[137] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Security and Cloud Computing: InterCloud Identity Management Infrastructure," in *Proc. of the 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, IEEE, Ed., 2010, pp. 263–265.

[138] Y. Demchenko, C. Ngo, C. de Laat, and C. Lee, "Federated Access Control in Heterogeneous Intercloud Environment: Basic Models and Architecture Patterns," in *Proc. of the IEEE International Conference on Cloud Engineering*, IEEE, Ed., 2014, pp. 439–445.

[139] Y. Demchenko, F. Turkmen, M. Slawik, and C. d. Laat, "Defining Intercloud Security Framework and Architecture Components for Multi-cloud Data Intensive Applications,"

in *Proc. of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, IEEE, Ed., 2017, pp. 945–952.

[140] M. V Thomas, A. Dhole, and K. Chandrasekaran, "Single Sign-On in Cloud Federation Using CloudSim," *International Journal of Computer Network and Information Security*, vol. 7, pp. 50–58, 05 2015.

[141] J. Bernal Bernabe, G. Martinez Perez, and A. Skarmeta, "Intercloud Trust and Security Decision Support System: an Ontology-based Approach," *Journal of Grid Computing*, vol. 13, 09 2015.

[142] M. Walkowski, J. Oko, and S. Sujecki, "Vulnerability Management Models Using a Common Vulnerability Scoring System," *Applied Sciences*, vol. 11, no. 18, 2021.

[143] M. Oulaaffart, R. Badonnel, and O. Festor, "Towards Automating Security Enhancement for Cloud Services," in *Proc. of the International Symposium on Integrated Network Management (IM)*, IEEE, Ed., 2021.

[144] B. Gupta, P. Mittal, and T. Mufti, "A Review on Amazon Web Service (AWS), Microsoft Azure and Google Cloud Platform (GCP) Services," in *Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2020, 27-28 February 2020, Jamia Hamdard, New Delhi, India*, IEEE, Ed. EAI, 3 2021.

[145] M. Z. Neto, G. A. A. Santana, F. Sapata, M. Munoz, A. M. S. P. Moraes, T. Morais, and D. L. Goldfarb, *Security Troubleshooting on AWS*. IEEE, 2021, pp. 339–362.

[146] V. Jalili, E. Afgan, J. Taylor, and J. Goecks, "Cloud bursting galaxy: federated identity and access management," *Bioinformatics*, vol. 36, no. 1, pp. 1–9, 06 2019.

[147] S. Potti, "Supercharging security with generative AI," 2023, visited on 2023-04-18. [Online]. Available: https://cloud.google.com/blog/products/identity-security/rsa-google-cloud-security-ai-workbench-generative-ai?hl=en

[148] A. CloudFormation, "AWS CloudFormation API Reference," p. 283, 2020, visited on 2023-02-23. [Online]. Available: https://docs.aws.amazon.com/AWSCloudFormation/latest/APIReference/Welcome.html

[149] A. Esposito, B. Di Martino, and G. Cretella, "Defining Cloud Services Workflow: a Comparison between TOSCA and OpenStack Hot," IEEE, Ed., 2015.

[150] NIST, "XCCDF - The Extensible Configuration Checklist Description Format," 2020, visited on 2022-12-14. [Online]. Available: https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/xccdf

[151] K. Scarfone and P. Mell, "An Analysis of CVSS version 2 Vulnerability Scoring," in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, IEEE, Ed., 2009, pp. 516–525.

[152] F. Amato, N. Mazzocca, and F. Moscato, "Model Driven design and Evaluation of Security Level in Orchestrated Cloud Services," *Journal of Network and Computer Applications*, vol. 106, pp. 78–89, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804517304022

[153] J. Zheng and A. Siami Namin, "A Survey on the Moving Target Defense Strategies: An Architectural Perspective," *Journal of Computer Science and Technology*, vol. 34, pp. 207–233, 01 2019.

[154] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating Crossfire Attacks Using SDN-Based Moving Target Defense," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, 2016, pp. 627–630.

[155] D. C. MacFarland and C. A. Shue, "The SDN Shuffle: Creating a Moving-Target Defense Using Host-Based Software-Defined Networking," in *Proceedings of the Second ACM Workshop on Moving Target Defense*. New York, NY, USA: Association for Computing Machinery, 2015, p. 37–41. [Online]. Available: https://doi.org/10.1145/2808475.2808485

[156] R. Zhuang, S. Zhang, A. Bardas, S. A. DeLoach, X. Ou, and A. Singhal, "Investigating the application of moving target defenses to network security," in *Proceedings of the 6th International Symposium on Resilient Control Systems (ISRCS)*, 2013, pp. 162–169.

[157] H. Alavizadeh, D. Kim, J. Hong, and J. Jang-Jaccard, *Effective Security Analysis for Combinations of MTD Techniques on Cloud Computing* , 12 2017, pp. 539–548.

[158] S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, "A Survey of Moving Target Defenses for Network Security," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1909–1941, 2020.

[159] C. Lei, H.-Q. Zhang, T. Jinglei, Y.-C. Zhang, and X.-H. Liu, "Moving Target Defense Techniques: A Survey," *Security and Communication Networks*, vol. 2018, pp. 1–25, 07 2018.

[160] D. P. Sharma, D. S. Kim, S. Yoon, H. Lim, J.-H. Cho, and T. J. Moore, "FRVM: Flexible Random Virtual IP Multiplexing in Software-Defined Networks," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 579–587.

[161] S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "Attack Graph-Based Moving Target Defense in Software-Defined Networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1653–1668, 2020.

[162] Y.-B. Luo, B.-S. Wang, and G.-L. Cai, "Effectiveness of Port Hopping as a Moving Target Defense," in *2014 7th International Conference on Security Technology*, 2014, pp. 7–10.

[163] M. Thompson, N. Evans, and V. Kisekka, "Multiple OS rotational environment an implemented Moving Target Defense," in *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, 2014, pp. 1–6.

[164] J. B. Hong, S. Yoon, H. Lim, and D. S. Kim, "Optimal Network Reconfiguration for Software Defined Networks Using Shuffle-Based Online MTD," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, 2017, pp. 234–243.

[165] E. Yuan, S. Malek, B. Schmerl, D. Garlan, and J. Gennari, "Architecture-Based Self-Protecting Software Systems," in *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures*. New York, NY, USA: Association for Computing Machinery, 2013, p. 33–42. [Online]. Available: https://doi.org/10.1145/2465478.2465479

[166] Y. Li, R. Dai, and J. Zhang, "Morphing Communications of Cyber-Physical Systems Towards Moving-Target Defense," in *the proceedings of IEEE International Conference on Communications (ICC)*, 2014, pp. 592–598.

[167] Y. Huang and A. Ghosh, *Introducing Diversity and Uncertainty to Create Moving Attack Surfaces for Web Services*, 08 2011, pp. 131–151.

[168] Y. Huang, A. K. Ghosh, T. Bracewell, and B. Mastropietro, "A Security Evaluation of a Novel Resilient Web Serving Architecture: Lessons Learned Through Industry/Academia Collaboration," in *2010 International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2010, pp. 188–193.

[169] M. Taguinod, A. Doupé, Z. Zhao, and G.-J. Ahn, "Toward a Moving Target Defense for Web Applications," in *2015 IEEE International Conference on Information Reuse and Integration*, 2015, pp. 510–517.

[170] A. Gorbenko, V. Kharchenko, and A. Romanovsky, *Using Inherent Service Redundancy and Diversity to Ensure Web Services Dependability*. Berlin, Heidelberg: Springer-Verlag, 2009, p. 324–341.

[171] H. Alavizadeh, J. Jang-Jaccard, and D. S. Kim, "Evaluation for Combination of Shuffle and Diversity on Moving Target Defense Strategy for Cloud Computing," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 573–578.

[172] K. M. Carter, H. Okhravi, and J. Riordan, "Quantitative Analysis of Active Cyber Defenses Based on Temporal Platform Diversity," *ArXiv*, vol. abs/1401.8255, 2014.

[173] K. M. Carter, J. F. Riordan, and H. Okhravi, "A Game Theoretic Approach to Strategy Determination for Dynamic Platform Defenses," in *Proceedings of the First ACM Workshop on Moving Target Defense*. New York, NY, USA: Association for Computing Machinery, 2014, p. 21–30.

[174] J. Steinberger, B. Kuhnert, C. Dietz, L. Ball, A. Sperotto, H. Baier, A. Pras, and G. Dreo, "DDoS Defense Using MTD and SDN," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–9.

[175] S. Debroy, P. Calyam, M. Nguyen, R. L. Neupane, B. Mukherjee, A. K. Eeralla, and K. Salah, "Frequency-Minimal Utility-Maximal Moving Target Defense Against DDoS in SDN-Based Systems," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 890–903, 2020.

[176] M. Zhu, Z. Hu, and P. Liu, "Reinforcement Learning Algorithms for Adaptive Cyber Defense against Heartbleed," in *Proceedings of the First ACM Workshop on Moving Target Defense*, 2014, p. 51–58.

[177] D. A. Osvik, A. Shamir, and E. Tromer, "Cache Attacks and Countermeasures: The Case of AES," in *Topics in Cryptology – CT-RSA 2006*, D. Pointcheval, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–20.

[178] E. Tromer, D. A. Osvik, and A. Shamir, "Efficient Cache Attacks on AES, and Countermeasures," *J. Cryptol.*, vol. 23, no. 1, p. 37–71, jan 2010.

[179] R. Colbaugh and K. Glass, "Predictability-Oriented Defense Against Adaptive Adversaries," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2012, pp. 2721–2727.

[180] S. Kim, S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "DIVERGENCE: Deep Reinforcement Learning-based Adaptive Traffic Inspection and Moving Target Defense Countermeasure Framework," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2021.

[181] B. Tozer, T. Mazzuchi, and S. Sarkani, "Optimizing Attack Surface and Configuration Diversity Using Multi-objective Reinforcement Learning," in *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 144–149.

[182] A. Chowdhary, S. Sengupta, D. Huang, and S. Kambhampati, "Markov Game Modeling of Moving Target Defense for Strategic Detection of Threats in Cloud Networks," *ArXiv*, vol. abs/1812.09660, 2018.

[183] E. Farchi, O. Shehory, and G. Barash, "Defending via Strategic ML Selection," *ArXiv*, vol. abs/1904.00737, 2019.

[184] M. Mohammadi, M. Dawodi, W. Tomohisa, and N. Ahmadi, "Comparative Study of Supervised Learning Algorithms for Student Performance Prediction," in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2019, pp. 124–127.

[185] M. Alloghani, D. Al-Jumeily Obe, J. Mustafina, A. Hussain, and A. Aljaaf, *A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science*, 01 2020, pp. 3–21.

[186] S. Fahle, C. Prinz, and B. Kuhlenkötter, "Systematic Review on Machine Learning (ML) Methods for Manufacturing Processes – Identifying Artificial Intelligence

(AI) Methods for Field Application," *Procedia CIRP*, vol. 93, pp. 413–418, 2020, 53rd CIRP Conference on Manufacturing Systems 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827120307435

[187] W. Du and S. Ding, "A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications," *Artificial Intelligence Review*, vol. 54, pp. 1–24, 06 2021.

[188] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A Gentle Introduction to Reinforcement Learning and its Application in Different Fields," *IEEE Access*, vol. 8, pp. 209 320–209 344, 2020.

[189] F. Zhao, Y. Liu, N. Zhu, T. Xu, and Jonrinaldi, "A Selection Hyper-heuristic Algorithm with Q-learning Mechanism," *Applied Soft Computing*, p. 110815, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494623008335

[190] "Optimization of Live Virtual Machine Migration in Cloud Computing: A Survey and Future Directions," *Journal of Network and Computer Applications*, vol. 110, pp. 1–10, 2018.

[191] M. Esam, H. Abbas, and C. Meinel, "Virtual Machines Pre-copy Live Migration Cost Modeling and Prediction: a Survey," *Distributed and Parallel Databases*, pp. 1–34, 12 2021.

[192] Y.-H. Wang, T.-H. S. Li, and C.-J. Lin, "Backward Q-learning: The combination of Sarsa algorithm and Q-learning," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 9, pp. 2184–2193, 2013.

[193] M. Heidari, S. Zad, and S. Rafatirad, "Ensemble of Supervised and Unsupervised Learning Models to Predict a Profitable Business Decision," in *Proceedings of the IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2021, pp. 1–6.

[194] U. Mandal, M. F. Habib, S. Zhang, M. Tornatore, and B. Mukherjee, "Bandwidth and Routing Assignment for Virtual Machine Migration in Photonic Cloud Networks," in *Proceedings of the 39th European Conference and Exhibition on Optical Communication (ECOC 2013)*, 2013, pp. 1–3.

List of Figures

List of Tables

Appendix A

Résumé de la thèse en français

L'avènement de l'Internet et de du cloud informatique a apporté une évolution significative à la manière dont les données et les ressources sont stockées, traitées et accessibles. Au cœur du concept du cloud informatique se trouvent des techniques de virtualisation, qui facilitent l'allocation optimale des ressources et la mise à disposition de services évolutifs et adaptables. Grâce à la virtualisation, l'abstraction du matériel sous-jacent permet la création de machines virtuelles ou de conteneurs capables de prendre en charge plusieurs systèmes d'exploitation et applications simultanément sur une seule infrastructure. Cela permet une utilisation plus efficace des ressources matérielles, essentielle pour fournir les économies de coûts et la scalabilité promises par l'informatique en cloud. En conséquence, le cloud propose une large gamme de services tels que le stockage, les applications et l'infrastructure, qui peuvent être accessibles de n'importe où via Internet. De plus, l'informatique en cloud est scalable, permettant aux utilisateurs d'ajuster facilement leurs ressources informatiques en fonction de leurs besoins. De plus, la maturité des langages d'orchestration et les avantages des migrations de ressources ont permis des améliorations supplémentaires dans la composition de services. Ces langages permettent l'automatisation de tâches complexes et des processus processus qui seraient autrement longs et sujets aux erreurs peuvent être automatisés pour réduire le risque d'erreurs humaines et améliorer la fiabilité et la cohérence des infrastructures du cloud informatique. Cette automatisation offre un niveau élevé d'abstraction, simplifiant ainsi la gestion des infrastructures et facilitant la gestion et la coordination des différents composants des environnements cloud. De plus, les langages d'orchestration améliorent les performances des applications cloud en permettant une allocation efficace des ressources et un équilibrage de charge. En automatisant la gestion des ressources et en optimisant l'allocation de la charge de travail. De même, les techniques de migration sont particulièrement importantes pour améliorer les performances en cloud de plusieurs manières. Tout d'abord, elles permettent une meilleure utilisation des ressources en autorisant le déplacement des ressources vers des serveurs physiques sous-utilisés. De plus, les migrations contribuent à améliorer la disponibilité des services en permettant le déplacement des ressources entre différentes infrastructures cloud. Cette capacité assure la continuité des services sans interruption ni perturbation. Malgré ces avantages des migrations techniques vers les infrastructures en cloud, il est important de souligner que ces activités peuvent générer de nouvelles menaces en matière de sécurité. Cela peut entraîner des vulnérabilités et des attaques pouvant compromettre la sécurité des ressources cloud impliquées dans la migration, exploitant les caractéristiques des environ-

nements cloud. La composition de services, ainsi que les migrations orchestrées combinées aux caractéristiques du cloud, apportent plusieurs avantages, notamment en termes de performances des ressources. Cependant, ces services présentent également leur propre ensemble de menaces et de risques qui doivent être gérés avec soin. Ces menaces peuvent avoir des conséquences graves, affectant plusieurs propriétés de sécurité, la réputation des organisations, leur stabilité financière, voire leur conformité légale. Les vulnérabilités émergentes lors des migrations de ressources entre des infrastructures cloud hétérogènes et les processus dynamiques sous-jacents peuvent accroître ces risques, pouvant conduire à des attaques compromettant la sécurité des ressources cloud. Puisque les services composites en cloud sont construits en intégrant plusieurs services cloud de différents fournisseurs, ils peuvent être particulièrement vulnérables aux attaques exploitant les faiblesses de l'un des services impliqués. Cela peut entraîner divers problèmes de sécurité, tels que des violations de données, des interruptions de service et un accès ou une altération non autorisés de données sensibles. Pour atténuer ces menaces de sécurité, plusieurs approches proposent des solutions basées sur des mécanismes de sécurité exogènes et endogènes. Les mécanismes de sécurité exogènes se réfèrent à des mesures de sécurité externes mises en place par les fournisseurs de services cloud, tandis que les mécanismes de sécurité endogènes sont des mécanismes de sécurité internes mis en œuvre par l'organisation elle-même. Ces mécanismes incluent la génération de ressources avec les paquets nécessaires, l'analyse de la configuration des ressources et la vérification des ressources. Cependant, les mécanismes exogènes couvrent plusieurs fonctionnalités, telles que la virtualisation des fonctions réseau, la création de chaînes de sécurité, les systèmes de chiffrement basés sur le matériel et les audits des plateformes cloud avant les migrations. Cette thèse présente une proposition d'automatisation de la sécurité des services composites en cloud en se concentrant sur les migrations de ressources. Elle est structurée autour de trois axes fondamentaux. Le premier axe implique l'introduction d'un cadre automatisé basé sur SMT (satisfiabilité modulo des théories) pour l'évaluation des ressources cloud pendant les opérations de migrations des ressources. Le deuxième axe concerne la conception d'une tierce partie de confiance visant à assurer la sécurité de ces ressources dans un contexte distribué. Enfin, le troisième axe exploite le paradigme de la défense à cible mouvante en intégrant des techniques de vérification avec des algorithmes d'intelligence artificielle, afin d'élaborer une stratégie pour renforcer la sécurité de ces ressources.

Le premier axe de la thèse correspond à un cadre de sécurité basé sur SMT pour soutenir les migrations dans les services composites en cloud, tels que ceux orchestrés avec le langage TOSCA (Topology and Orchestration Specification for Cloud Applications). L'objectif est d'exploiter des techniques de vérification pour évaluer automatiquement les changements de configuration qui affectent les ressources des services cloud pendant leur migration, et de déterminer des contre-mesures adéquates. Le framework proposé fonctionne en trois phases principales. La première phase établit une projection de la configuration des ressources induite par la migration, en tenant compte des changements qui affectent la ressource migrée. Elle évalue ensuite cette configuration en exploitant les connaissances fournies par les descriptions de vulnérabilités. La troisième phase consiste à déterminer des contre-mesures potentielles à mettre en place sur la ressource elle-même ou activées dans son environnement pour prévenir les vulnérabilités observées et maintenir la sécurité globale du service composite en cloud. Afin de démontrer l'efficacité et la faisabilité de notre approche, nous avons développé un prototype en utilisant Python sur le dessus de solveurs SMT. Nous avons ensuite réalisé des séries d'expériences approfondies en utilisant différents

scénarios d'évaluation et de remédiation à l'aide de descriptions de vulnérabilités existantes provenant du référentiel OVAL officiel, et correspondant à différents systèmes d'exploitation. Ces expériences ont montré que le temps d'évaluation le plus élevé (1,53 seconde) est observé avec l'ensemble de données de Microsoft Windows Server 2012 R2, qui correspond également à l'ensemble de données avec le plus grand nombre de descriptions de vulnérabilités, soit un total de 5344 descriptions de vulnérabilités OVAL. Le temps d'exécution global requis pour le cadre d'évaluer et de sélectionner des contre-mesures, correspondant à un temps moyen d'environ 3,20 secondes avec l'identification d'une seule contre-mesure, et à un temps moyen de 7,10 secondes pour l'identification de toutes les contre-mesures potentielles. Enfin, les résultats obtenus par rapport aux scénarios de migration en direct montrent la faisabilité de notre solution proposée.

Le deuxième axe exploite le cadre précédent pour proposer une architecture d'une approche de tierce partie de confiance inter-cloud, appelée C3S-TTP (Composite Cloud Configuration Security-Trusted Third Party), afin de renforcer la sécurité des services composites en cloud dans un contexte distribué. L'objectif est de soutenir la sécurité des services cloud basés sur TOSCA, dont les ressources élémentaires peuvent être soumises à une migration au fil du temps. En fait, cette approche est motivée par les aspects difficiles et complexes de l'analyse des changements de configuration résultant du partage partiel des informations sur les configurations entre les parties prenantes, à savoir les locataires de cloud et les fournisseurs de cloud. Maintenir la sécurité des services cloud lors de la migration de leurs ressources nécessite une transparence accrue en ce qui concerne les configurations avant d'effectuer ces migrations. Cet échange de connaissances contribue à une meilleure gestion de la sécurité, avec la préparation d'un environnement de destination adéquat du côté du fournisseur de cloud, et l'endurcissement optimal de la ressource migrée du côté du locataire du cloud. D'autre part, les parties prenantes du cloud évitent généralement de partager des informations détaillées sur la configuration technique de leurs ressources et infrastructures, en particulier celles liées à la sécurité, ou affectant la sécurité. La solution proposée vise donc à prévenir ou à restreindre la divulgation de telles informations de configuration entre les locataires de cloud et les fournisseurs de cloud, tout en gérant les vulnérabilités qui peuvent survenir pendant la migration des ressources cloud. Elle repose sur une architecture de tierce partie de confiance inter-cloud qui exploite une tierce partie prenante pour partager des informations de configuration entre les locataires de cloud et les fournisseurs de cloud et évaluer les vulnérabilités potentielles. Cette tierce partie de confiance est activée dès qu'un locataire de cloud demande la migration d'une ou plusieurs de ses ressources cloud. Tout d'abord, elle collecte des informations de configuration relatives aux ressources cloud candidates à la migration. Elle demande ensuite aux fournisseurs de services cloud des informations de configuration concernant les environnements d'hébergement potentiels, y compris la version des ressources et la paramétrisation de ces environnements cloud, ainsi que des informations sur les mécanismes de sécurité qui peuvent être mis en œuvre par les fournisseurs de cloud. En tenant compte de ces connaissances globales et en utilisant notre cadre d'évaluation des vulnérabilités basé sur le langage d'évaluation et d'évaluation ouverte OVAL, la tierce partie détermine les vulnérabilités de configuration qui peuvent apparaître lors de la migration des ressources sur les environnements d'hébergement potentiels. Ces résultats d'évaluation permettent à la tierce partie de confiance de recommander ou de refuser la migration d'une ressource donnée au locataire du cloud, tout en réduisant la divulgation d'informations de configuration entre les locataires de cloud et les fournisseurs de cloud. Nous avons évalué notre approche à travers plusieurs expéri-

ences en utilisant un prototype de la conception, mis en œuvre sur notre cadre d'évaluation de la sécurité, qui exploite le solveur SMT open-source CVC4 pour analyser les configurations et détecter les vulnérabilités potentielles basées sur les ensembles de données OVAL. Nous avons montré dans quelle mesure l'observabilité activée par la tierce partie de confiance contribue à accroître la performance de la prévention des vulnérabilités et à minimiser les risques liés aux configurations non sécurisées. Les résultats expérimentaux ont montré que cette observabilité réduit le temps d'évaluation de la sécurité pour tous les systèmes considérés (CentOS Linux, Red Hat Enterprise Linux, Microsoft Windows). De plus, la probabilité d'avoir une configuration vulnérable diminue à mesure que le nombre de propriétés de configuration connues sur l'environnement cloud augmente. La différence entre une observabilité de 10

Le dernier axe se concentre sur la conception d'une stratégie de défense à cible mouvante qui couple des algorithmes d'apprentissage par renforcement avec des techniques de vérification. Elle consiste à changer continuellement la surface d'attaque en effectuant des mouvements sur la cible. Ces mouvements sont généralement concrétisés par la migration de ressources faisant partie d'un service composite en cloud. Ils sont sélectionnés par l'intelligence artificielle pour minimiser leur prévisibilité par les attaquants. L'objectif est de tromper ces attaquants en les incitant à mener des activités de reconnaissance. Cependant, les mouvements eux-mêmes peuvent introduire de nouvelles vulnérabilités de configuration qui peuvent être exploitées par des attaquants. Les techniques de vérification sont donc utilisées pour minimiser la surface d'attaque lorsque les mouvements sont décidés. Le cœur de la stratégie de défense à cible mouvante repose sur des algorithmes d'apprentissage par renforcement, en particulier le Q-learning, où les récompenses sont calculées en fonction des résultats des techniques de vérification. Nous avons également formalisé mathématiquement notre solution en considérant notre modélisation précédente. La sélection d'un mouvement est déclenchée par la détection d'événements suspects ou est effectuée périodiquement après un intervalle de temps spécifique. Afin de quantifier l'efficacité de notre solution, nous avons mené plusieurs séries d'expériences avec une comparaison à une stratégie de référence. Les résultats obtenus montrent la réduction de l'exposition aux attaques de sécurité graves, avec une évaluation des vulnérabilités introduisant un surcoût limité. Ils illustrent également un faible impact sur la prévisibilité des mouvements, ce qui contribue à contrer les activités de reconnaissance effectuées par les attaquants, tandis que les techniques de vérification contribuent à minimiser l'occurrence de vulnérabilités.

S'agissant des perspectives de recherches, nous proposons trois principales directions à explorer, afin d'améliorer et de renforcer la sécurité des services composites en cloud, à savoir le couplage avec le renseignement sur les menaces, la contribution à la sécurité verte et l'extension à l'informatique en périphérie.

Le premier axe vise à exploiter le renseignement sur les menaces pour renforcer notre approche en matière de sécurité. Les attaquants améliorent continuellement leurs tactiques, techniques et procédures pour être moins prévisibles, plus persistants, plus ingénieux et mieux financés, ce qui rend leurs activités difficiles à détecter et à enquêter. Pour cela, le renseignement sur les menaces vise à recueillir et agréger des connaissances et des informations et à les partager parmi les parties prenantes afin d'anticiper, de prévenir, de détecter, et répondre aux incidents de sécurité. L'objectif principal est de déterminer et d'identifier des menaces spécifiques et des cybercriminels, d'analyser leurs méthodes, et donc de mettre en place des mécanismes de défense et des stratégies réactives et proactives adéquats. À cet égard, nous prévoyons d'étudier le couplage de notre

solution avec le renseignement sur les menaces, afin d'améliorer les performances de notre cadre en utilisant de nouvelles sources de connaissances dans le contexte des services composites en cloud. Par exemple, la connaissance des tentatives d'attaques peut être exploitée pour prioriser les opérations correctives par rapport aux vulnérabilités. En particulier, la tierce partie de confiance inter-cloud pourrait tirer parti de ces connaissances pour améliorer son évaluation des vulnérabilités basée sur les informations de configuration collectées auprès du locataire du cloud et du fournisseur de cloud. Nous pouvons également ajuster la politique de défense à cible mouvante en fonction de ces connaissances en renseignement sur les menaces, afin d'adapter la fréquence ou la nature des mouvements.

Le deuxième axe est centré sur la sécurité verte dans les environnements cloud. Malgré ses avantages, l'informatique en cloud a des effets secondaires qui ont un impact négatif sur l'environnement, tels que les émissions de carbone. Même s'il existe des efforts pour rendre les infrastructures cloud moins nuisibles à l'environnement, elles doivent encore être améliorées. Les fournisseurs de cloud tentent de réduire l'impact environnemental de leurs infrastructures en les rendant plus écoénergétiques et en utilisant moins de matériel. Ils peuvent également réaliser des économies d'échelle en combinant des centres de données et en utilisant des technologies de refroidissement et de gestion de l'alimentation avancées pour réduire la consommation d'énergie. Cependant, la croissance de l'informatique en cloud a également entraîné une augmentation du nombre de centres de données, qui peuvent consommer une quantité importante d'énergie. Cela est exacerbé par les migrations au sein de ces infrastructures et l'activation et la désactivation des mécanismes de sécurité pour protéger ces ressources. De telles opérations peuvent entraîner des activités qui consomment des ressources importantes, en particulier lorsque des ressources sont migrées entre plusieurs infrastructures cloud. Un défi important est de réduire la consommation de ressources due aux mécanismes de sécurité, y compris ceux utilisés pour protéger les services composites en cloud. En particulier, le choix des contre-mesures devrait prendre en compte ce paramètre, afin de factoriser autant que possible les coûts induits par leur application. Par exemple, dans certains scénarios, la mise en œuvre d'un mécanisme endogène, intégré à la ressource cloud, pourrait être préférée à un mécanisme exogène pris en charge par l'infrastructure cloud. Dans d'autres cas, l'exploitation d'un mécanisme exogène, déjà activé pour prendre en charge d'autres ressources dans l'infrastructure cloud, pourrait être privilégiée pour mutualiser son utilisation.

Le dernier axe se concentrera sur le placement des mécanismes de sécurité pour protéger les services composites en étendant notre solution au cas de l'informatique en périphérie. L'objectif est de prendre la bonne décision quant à la mise en place d'un mécanisme de sécurité donné au niveau du périphérique ou dans un centre de données cloud, afin de sécuriser efficacement ces services. Cela peut concerner à la fois l'évaluation des vulnérabilités et la sélection de contre-mesures pour les remédier. Sur le périphérique, les mécanismes de sécurité peuvent être activés plus près de la source des données, plutôt que d'être envoyés vers un emplacement plus centralisé, tel qu'un centre de données cloud. Cela permet un traitement plus rapide et plus efficace, et réduit la latence associée à la transmission des données vers les centres de données. Dans le même temps, les périphériques et leurs applications peuvent également être plus exposés aux attaques de sécurité, en raison de leur nature distribuée, rendant important la mise en place d'une authentification forte et de contrôles d'accès, le chiffrement des données et des protocoles de communication sécurisés pour les protéger. Dans ce contexte, notre travail prendra en compte

les problèmes de sécurité liés aux services composites dans un continuum edge-cloud, et étudiera des stratégies d'externalisation automatisées pour les sécuriser.