C3S-TTP: A Trusted Third Party for Configuration Security in TOSCA-based Cloud Services

Mohamed Oulaaffart, Rémi Badonnel, Olivier Festor

C3S-TTP: A Trusted Third Party for Configuration Security in TOSCA-based Cloud Services

Mohamed Oulaaffart*, Rémi Badonnel and Olivier Festor

RESIST Research Team, LORIA / INRIA Nancy Grand Est,
University of Lorraine, CNRS, 615 Rue du Jardin Botanique,
54600, Villers-les-Nancy, France.

*Corresponding author: mohamed.oulaaffart@loria.fr
Contributing authors: remi.badonnel@loria.fr, olivier.festor@inria.fr

Abstract

The large-scale deployment of cloud composite services distributed over heterogeneous environments poses new challenges in terms of security management. In particular, the migration of their resources is facilitated by recent advances in the area of virtualization techniques. This contributes to increase the dynamics of their configuration, and may induce vulnerabilities that could compromise the security of cloud resources, or even of the whole service. In addition, cloud providers may be reluctant to share precise information regarding the configuration of their infrastructures with cloud tenants that build and deploy cloud composite services. This makes the assessment of vulnerabilities difficult to be performed with only a partial view on the overall configuration. We therefore propose in this article an inter-cloud trusted third-party approach, called C3S-TTP, for supporting secure configurations in cloud composite services, more specifically during the migration of their resources. We describe the considered architecture, its main building blocks and their interactions based on an extended version of the TOSCA orchestration language. The trusted third party 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. After designing and formalizing this third party solution, we perform large series of experiments based on a proof-of-concept prototype in order to quantify its benefits and limits.

Keywords: Security Management, Composite Services, Cloud Security, Orchestration Language, Trusted Third Party, Resources Migration

1 Introduction

The growing maturity of orchestration languages contributes to the large-scale design of cloud composite services. These services are built from elementary cloud resources, that may be subject to offline and live migrations over time, this being catalyzed by recent advances in virtualization techniques. These migrations may typically involve multiple cloud service providers, with the motivation of performance and operational objectives, such as fault management [1], load balancing [2], and energy efficiency [3]. Their 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, the security mechanisms implemented by the cloud provider. The configuration changes induced by the migration of cloud resources may introduce new vulnerabilities that may be exploited by malicious entities to compromise the migrated resource, or even the whole cloud composite service.

In addition, while several initiatives have been taken to increase the interoperability amongst cloud infrastructures and prevent vendor lock-in problems [4, 5], 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 [6]. Our objective is rather 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.

In this article we propose an inter-cloud trusted third-party approach, called C3S-TTP[1], for supporting configuration security in TOSCA[2]-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 the 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

[1] Composite Cloud Configuration Security - Trusted Third Party
[2] Topology and Orchestration Specification for Cloud Applications

side (i.e. application of a security patch). However, cloud stakeholders generally avoid sharing detailed technical configuration information about their resources and infrastructures amongst them, in particular those related to their security or simply impacting their security [7, 8]. 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 several of its cloud resources. First, it collects configuration information related to the cloud resources that are candidates for migration. It then requests cloud service providers to get configuration information regarding potential hosting environments, including the resource versioning and parameterization of these environments, as well as information regarding security mechanisms that may be implemented by the cloud providers.This information exchange is performed by extending TOSCA templates that describe considered composite services. Considering this overall knowledge and using our OVAL[3]-based vulnerability assessment framework detailed in [9], 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 main contributions of this article include:

- The design of an inter-cloud trusted third-party architecture for configuration security in cloud composites services, relying on a dedicated trusted third party to minimize the disclosure of configuration information,
- The extension of the TOSCA orchestration language for supporting the interactions amongst the different stakeholders of the proposed architecture, in particular during the migration of cloud resources,
- The formalization of the configuration assessment process implemented by the inter-cloud trusted third party in order to detect and prevent potential vulnerabilities,
- The proof-of-concept prototyping as well as the evaluation of our solution based on extensive series of experiments using the official OVAL vulnerability datasets.

[3]Open Vulnerability and Assessment Language

The remainder of the article is organized as follows. Section II gives an overview of related work in the area of inter-cloud management and security. Section III describes the considered inter-cloud trusted-third party architecture, by detailing its main building blocks and the interactions amongst them to support and secure resource migrations. Section IV depicts the proof-of-concept prototype developed in Python, implementing this architecture on top of a configuration assessment framework based on a SMT solver engine, as well as provides the performance evaluation of our solution based on extensive experiments. Finally, Section V gives the conclusion and points out future research perspectives.

2 Related Work

While they are facilitated by recent advances on virtualization techniques, inter-cloud migrations are facing several 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 [10] 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 [11], contribute to the development of these properties by encouraging the collaboration amongst different cloud vendors. In particular, the authors propose an inter-cloud framework that supports the interoperability amongst heterogeneous cloud environments with the goal of dispatching and optimizing the workload 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 [12] 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 management, as well as authorization and access control. For instance, the authors of [13] introduce an inter-cloud identity management infrastructure for cloud environments. When a cloud user performs the request for a resource that cannot be directly handled by its home

cloud, or 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 both a person or a software entity, has been authenticated and determines his rights. Once the identity has been verified, a specific request for external resource is forwarded to the so-called foreign cloud. In the same manner, the authors of [14, 15] describe basic models and architecture patterns for supporting federated access control in heterogeneous inter-cloud environments. They first focus on an inter-cloud operation framework for enabling the integration and interoperability of resources in multi-domain heterogeneous cloud environments. They then extend it with a security layer to provide access control features with respect to these resources. More specifically, Single Sign On (SSO) approaches, such as [15], have been designed and implemented for supporting more specifically cloud federation. In particular, the authors of [16] 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 are transferred amongst different entities in the cloud federation during the SSO mechanism, and highlighted the benefits in large-scale environments. Moreover, ontology-based frameworks, such as the Security Ontology For the Inter-Cloud (SOFIC) solution [17], 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 improve 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. For instance, the authors of [18] propose an endogenous approach based on the generation of protected unikernel images, corresponding to lightweight virtual machines containing only the strict necessary packages and libraries in order to minimize the attack surface of cloud resources. We also started to work on securing composite services based on vulnerability descriptions expressed with the OVAL language [19], but without considering information disclosure issues [9]. Certification techniques discussed in [20, 21], 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, at the basis of certification activities against real and synthetic execution traces. The objective is to increase trust and transparency

Table 1 Synthesis and comparison of existing work on inter-cloud security management

Approaches	Main Contributions	Third Party	Distributed Context	Orchestr. Language Extension	Shortages
[11]	Interoperability	No	Yes	No	Does not focus on security aspects of cloud resources
[12]	Interoperability and Portability	No	Yes	No	Covers interoperability and portability exclusively, with less emphasis on security concerns
[13]	Identity Management	Yes	Yes	No	Does not address security aspects related to vulnerabilities that may exist in cloud infrastructures
[14]	Access Control	Yes	Yes	No	Is not totally adapted to the dynamic aspect of cloud resources
[15]	Authentication	Yes	Yes	No	Considers only the authentication aspects related to cloud resources
[17]	Security Management	No	Yes	No	Does not consider composite cloud resources
[18]	Security Hardening	No	No	Yes	Focuses on specific individual cloud resources
[19]	Vulnerability Assessment	No	No	Yes	Does not consider the multi-tenant cloud environments
[20, 21]	Certification Aspects	No	Yes	No	Focuses on specific aspects rather than encompassing cloud services in their entirety
[22, 23]	Audit of Infrastructures	No	Yes	No	Relies only on auditing infrastructures
[24]	Vulnerability Management	No	Yes	No	Does not focus on cloud composite services
[25, 26]	Hardware Security	No	No	No	Proposes security solutions relying on the Trusted Platform Module, yet these solutions are not entirely suited for migration of resources
[27]	Risk Management	No	Yes	No	Does not take into consideration cloud composite services

of a given cloud composite service during its whole life cycle, and ensure a continuous certification process during the service operation. This is based on the evidence collected about the behavior of the composite service, which is used to verify whether it maintains the support of the desired certified properties. While the authors do not refer to any specific (unitary) certification techniques, the authors propose heuristic algorithms to ensure the certification of composite services deployed at the cloud application layer (BPaaS) by

considering the ISO 15408 common criteria certification framework [28] for monolithic services. They use Business Process Model and Notation (BPMN), a generic and standard notation for modeling business processes and service compositions, in order to dynamically generate BPMN-compliant compositions that hold a set of security properties. Some other efforts are more endogenous to the cloud resources. Solutions, such as [22, 23], aim at auditing cloud environments before the migration of considered resources. For this, they rely on the evidence collected from the cloud service providers, and 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 [24] also evaluate the risk associated to each cloud environment based on a quantification and classification of vulnerabilities inherent to the environment. Approaches, such as [29], 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 [25, 26] 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 [27] how risk management methods can be applied to cloud infrastructures, in order to automatically determine the counter-measures to be applied during resource migrations.

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) [30]. It also provides security groups to specify network firewall rules related to EC2 virtual machine instances [31]. Microsoft Azure also offers several security services, such as Azure Active Directory (AD) for identity management [32], and supports several compliance certifications, including the one related to the Health Insurance Portability and Accountability Act (HIPAA). Similarily, 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 [33].

Table 1 synthetizes and compares existing work related to our approach. In fact, most of existing techniques for inter-cloud security management are centered on identity management and access control, as in [13, 14], and do not exploit the knowledge provided by the specification of cloud composite services.

Moreover, approaches such as [22, 23] aim to perform some audit activities and check the compliance of cloud environments before migrating cloud resources. However, considered languages and techniques are not specified, and even the entity or the person that is in charge of doing such an audit is not clearly identified. Other approaches, such as [20, 21], provide frameworks to improve the security of cloud services, but rely on a peer-to-peer scenario between the cloud tenant and the cloud provider, requiring the potential sharing of critical information regarding resource configurations.

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.

3.1 Background

Before detailing the proposed approach, we will provide some background about cloud composite services and the considered use case. Figure 1 describes the considered use case together with the different stakeholders. The cloud services (represented in the middle of the figure) are owned by a cloud tenant (represented on the left of the figure). The resources related to these composite services are hosted over one or several cloud providers (represented on the right of the figure). The cloud tenant may request the migration of resources related to these services, in order to target several performance objectives, such as scalability and cost effectiveness. In the meantime, the cloud services and the related resources may be subject to vulnerabilities that can be exploited by attackers to perform security attacks. There are two major categories of attackers. The first category stands for outsider attackers, which are external entities that attempt to gain unauthorized access to cloud resources and underlying infrastructures from outside the considered cloud environment [34]. Using remote software and hardware and taking advantage of the human weaknesses of cloud users, external attackers can launch numerous attacks on cloud infrastructures throughout the network. The second category stands for insider attackers, which are internal entities 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 provider infrastructures. Insider attackers have a significant

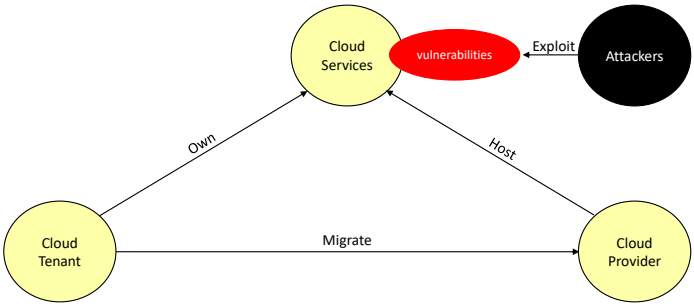


Fig. 1 Overview of the considered use case and the different stakeholders

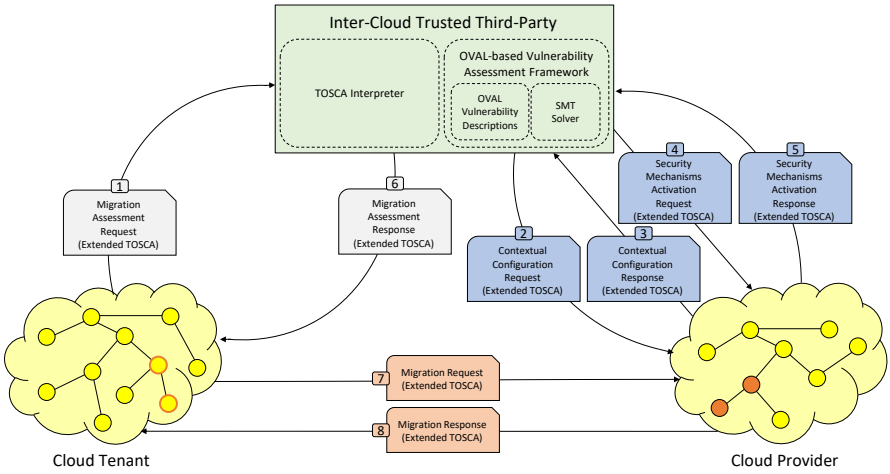


Fig. 2 Main building blocks of our inter-cloud trusted third-party architecture

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 security attacks on targeted cloud infrastructures [35]. Preventing vulnerabilities is a major activity to minimize the attack exposure of cloud composite services, in particular during resource migration, but this may imply sharing critical configuration information.

3.2 Architecture and Main Building-Blocks

The architecture of our C3S-TTP third-party approach is described in Figure 2, with the different involved building blocks and their interactions. It mainly relies on an inter-cloud trusted third party (top part of the figure) supporting 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, 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 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 system. Minimizing configuration vulnerabilities on cloud composite services is therefore important to prevent security attacks. 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 assessment, typically prior to the migration operations. The cloud tenant requires to know detailed configuration information regarding the hosting environment given by the cloud providers, this including information about the security mechanisms that may be provided or implemented by them. In the same manner, if this assessment is rather done 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 to properly control the attack surface and let configuration vulnerabilities that may be exploited by attackers over cloud composite services. The second consequence is to have a high security overload by provisioning in a preventive manner 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 therefore 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 [36], and is characterized by a more advanced expressivity than OpenStack HOT [37], 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 topology is given in yellow on the left part of Figure 2, with a topology composed of more than 10 nodes. It is then possible to specify orchestration procedures over this topology, such as starting,

shutting down a node or changing a relationship. Each element (node or relationship) takes benefits from inheritance and template mechanisms offered by the language. Such specification owned by the cloud tenant provides detailed configuration knowledge regarding the resources of 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 on the figure, it serves as a trusted intermediate 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 as a logical combination of conditions that if observed on a target system, the security problem described by such vulnerability is 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 permits to determine 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 [9]. When a known vulnerability is detected, this framework is capable to determine remediation actions (such as activating a given firewall rule), if any corrections are associated to the considered OVAL definitions. These actions are typically specified using the XCDDF[4] 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.

[4]Extensible Configuration Checklist Description Format

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
            # Beginning of the TOSCA extension
            migration:
                 status: true
                 type: online
            # End of the TOSCA extension
    mysql_db:
        type: tosca.nodes.Database.MySql
        properties:
            version: 8.0.29
        requirements:
            host: db_server
            # Beginning of the TOSCA extension
            migration:
                 status: false
            ...
```

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

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 assessment prior to the migration. Let 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 process of migration for a candidate cloud resource that is currently hosted on its on-premise infrastructure, or that is already hosted over another cloud provider infrastructure.

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, concretely 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 2) 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 of two main other nodes corresponding to an Apache webserver instance (host property on line 22) and to 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 candidate for migration, as shown by the migration status property 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) to the migrated resource.

Once the inter-cloud trusted third party receives the request from the cloud tenant, it exploits 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 2). 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 integrates again 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

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

```
1  topology_template:
2  ...
3    node_templates:
4      apache_modsecurity:
5      # Beginning of the TOSCA extension
6          type: tosca.nodes.WAF.ModSecurity
7          properties:
8              version: 3.0.7
9          requirements:
10         ...
11     oisf_suricata
12         type: tosca.nodes.IDS.Suricata
13         properties:
14             version: 6.0.6
15     # End of the TOSCA extension
16         requirements:
17         ...
```

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, supporting the Struts framework version 2.3.12, and Apache version 2.4.31 supporting 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 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 2), 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 [38] database to determine if counter-measures are associated to the considered vulnerabilitie(s). It turns out that in our scenario, the migration generates a vulnerability corresponding to CVE-2017-5638 [39], which can be exploited to

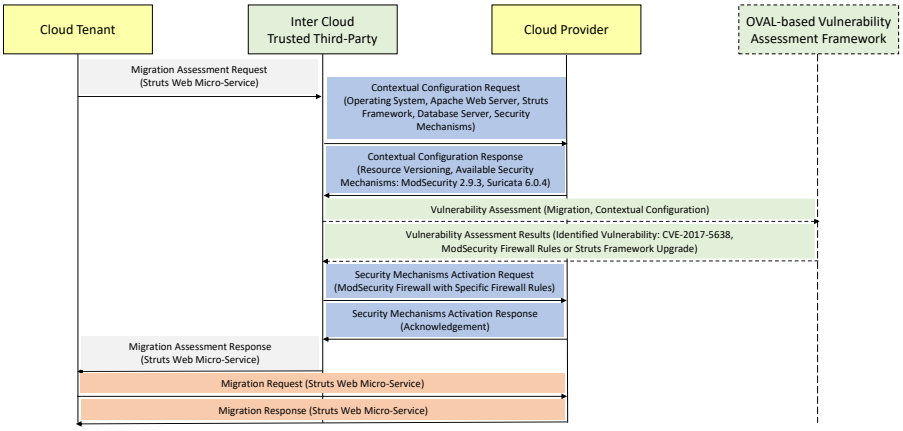


Fig. 3 Sequence diagram of the TOSCA-based interactions amongst the architectural building blocks

perform remote code execution, by allowing a remote attacker to inject operating system commands into the web application through the content-type header. Two alternative counter-measures associated to 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 equals 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 2), and receives a response from the cloud provider which acknowledges that the additional firewall rules have been properly activated over the cloud infrastructure (Step 5 on Figure 2). The third party then confirms to the cloud tenant that the migration of the considered resource is authorized over the hosting environment of the cloud provider (Step 6 on Figure 2). Finally, the cloud tenant directly interacts with the cloud provider to enforce the migration of the cloud resource, corresponding 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 performing the effective migration. This strategy permits to restrict the configuration information that are shared amongst the cloud tenant and the cloud provider. In

particular, the cloud tenant does not require to disclose the whole TOSCA specification of the cloud composite service to the cloud provider. In the meantime, the cloud provider does not require to disclose the whole configuration of its hosting infrastructure, including security mechanisms, to the cloud tenant.

3.4 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 [9], 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}. We mean by observable properties, 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.

$$\begin{aligned} &\Phi_{TTP}(P_{obs},\Pi(impact(m_i), state(c))) \\ &\qquad with\ P_{obs} \in P, m_i \in M, c \in C \end{aligned} \tag{1}$$

This migration assessment function Φ_{TTP} detailed by Equation 1 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.

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

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.

4 Performance Evaluation

We have evaluated the performance of our trusted third-party approach for cloud composite services, through extensive series of experiments. For that, we have developed a proof-of-concept prototyping of the proposed C3S-TTP architecture using the Python version 3.10 language, implemented on top of our OVAL-based vulnerability assessment framework [9], and considering the open-source CVC4 tool as a back-end SMT solver. The assessment framework is compatible with other solvers, such as Z3 [5] and VeriT [6], as it internally generates a SMT-LIB [7] (Satisfiability Modulo Theories LIBrary) specification that can be interpreted by a large variety of SMT solvers. The different experiments have been conducted over a regular laptop equipped with a 2Ghz Intel Core i5 processor and 8GB of RAM memory.

Table 2 Description of the different elements of the Proof of Concept

Hardware resources	2Ghz Intel Core i5 Processor and 8GB of RAM Memory
Programming Language	Python Version 3.10
Vulnerability Datasets	OVAL Datasets Version 5.11.1
SMT Solver	CVC4 Version 1.8
Orchestration Language	TOSCA Language Version 1.3
Compatibility	Compatible with other SMT Solvers such as Z3
Extension of the PoC	Extensible to other OVAL Vulnerability Descriptions

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 used in cloud infrastructures. The methodology can however easily be 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

[5] A theorem prover from Microsoft Research. It is licensed under the MIT license.

[6] Open-source SMT solver. It is proof-producing and complete for quantifier-free formulas with uninterpreted functions and linear arithmetic on real number and integers.

[7] An international initiative aimed at facilitating research and development in Satisfiability Modulo Theories (SMT).

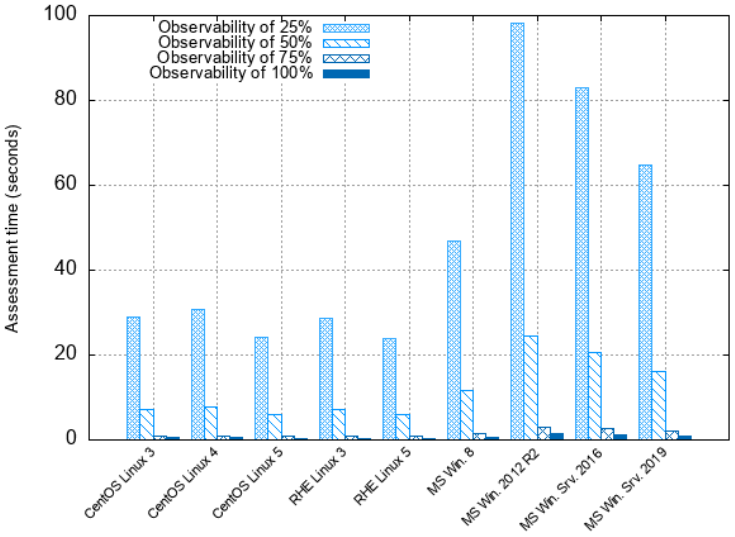
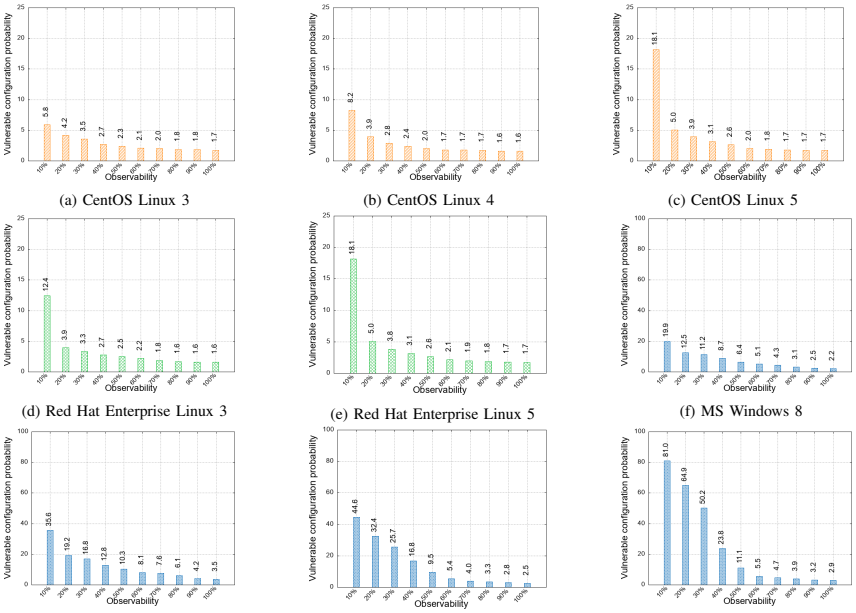


Fig. 4 Migration assessment time depending on the configuration observability (expressed in terms of percentage of known properties) with different operating systems

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 $P_{TTP}(A) \cup P_{TTP}(B)$, and the overall knowledge required for the vulnerability assessment, and corresponding to the overall set of properties P, as given by Equation 3.

$$observability = \frac{P_{TTP}(A) \cup P_{TTP}(B)}{P} \quad with\ (P_{TTP}(A), P_{TTP}(B)) \in P^2 \tag{3}$$

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.



(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

Fig. 5 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

In that context, we performed four main series of experiments. The first series has 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) [40]. The fourth experimental series has focused on comparing our solution to a proactive security approach in order to counter a low configuration observability.

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

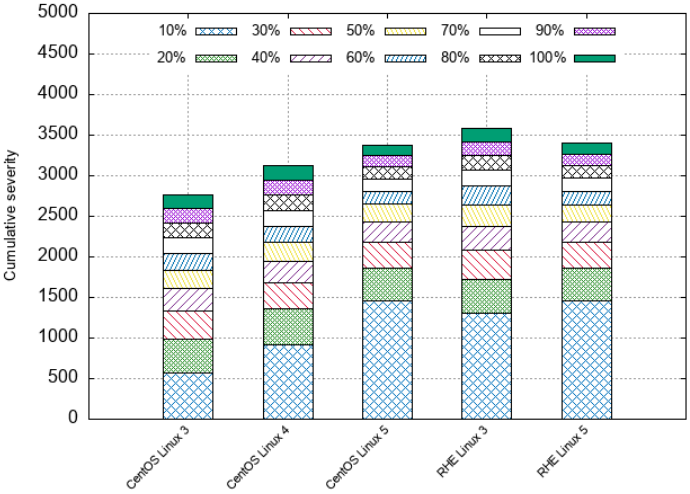
experiments are selected in a random manner. These experiments have been repeated until convergence in order to establish average values, and we have observed a variability of up to 5% in our experiments. These experimental results are synthesized on Figure 4, where the x axis indicates the operating 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 the 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 Entreprise Linux. This can be explained by the fact that the OVAL datasets related to Microsoft Windows Server contains more vulnerability descriptions than the two others. For instance, the Microsoft 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 reduce the migration assessment time, while deteriorating the vulnerability detection performance. Actually, it appears that a higher observability reduces the migration assessment time, and this 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 to the trusted third party using a CVC4 back-end solver. Increasing the observability ratio means knowing more properties, and therefore more constraints regarding the satisfiability issue to be solved, which explains this reduction of the assessment time when the observability is increasing. Therefore, the higher observability enabled by the trusted third party contributes to reduce the migration assessment time.

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

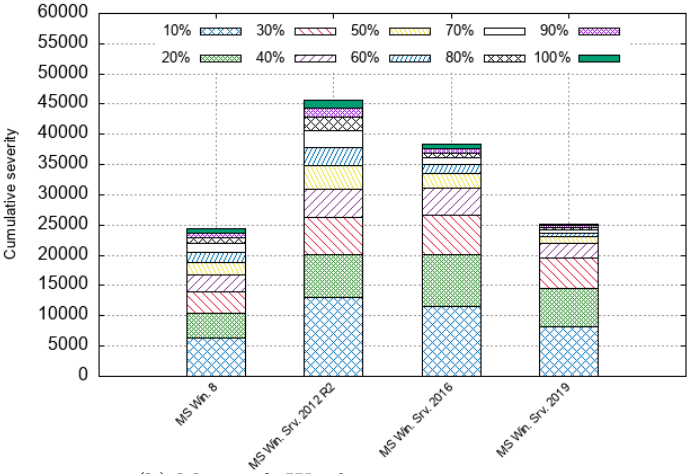
In a second series of experiments we wanted to evaluate the impact of observability on the probability of having a vulnerable configuration after the migration, considering the vulnerability descriptions provided by the official OVAL repository. A low observability means that the migration assessment process will not be capable 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 systems. These unknown properties correspond to properties that the cloud tenant refuses to disclose to the cloud provider. The experimental results are given on Figure 5, 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 also partially correlated to the size of the vulnerability description datasets. For instance, the average size of Microsoft Windows Server dataset is of 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%.

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 to potential vulnerable configurations, we



(a) Linux operating systems



(b) Microsoft Windows operating systems

Fig. 6 Cumulative severity induced by potential vulnerable configurations depending on the configuration observability

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[8] security automation languages family, that 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

[8]Security Configuration Automation Protocol

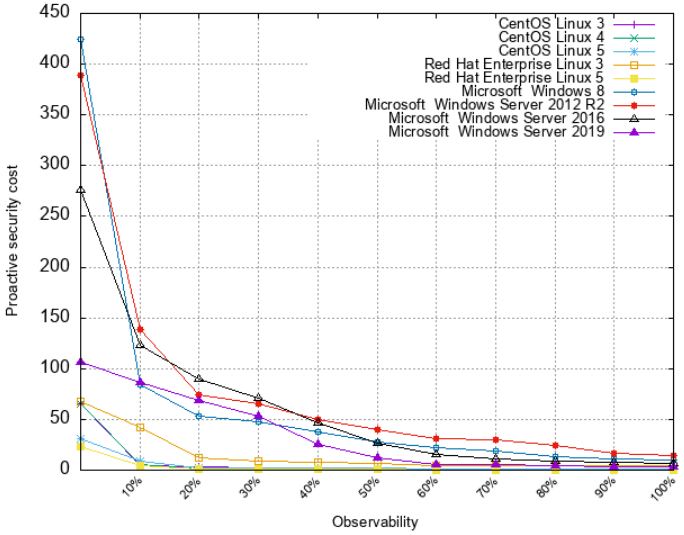


Fig. 7 Cost of a proactive security strategy to protect cloud resources while varying the configuration observability

this severity for each considered operating system are described on Figure 6, 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 is 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. This therefore goes in favor of our trusted third party enabling a higher configuration observability during the migration of cloud resources between a cloud tenant and a cloud provider.

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 in enforcing counter-measures (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 7, where we have plotted the

total cost of such a proactive security approach, quantified in terms of minimal number of countermeasures to be implemented by the cloud tenant, considering the vulnerability description datasets of different operating systems. The cost of such proactive strategy could be refined by evaluating experimentally the operational costs (i.e. processing time, memory usage) due to the enforcement of counter-measures, 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 high number of potential vulnerability descriptions, while others are only capable to address 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 a 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 Conclusions and Future Work

The growing maturity of orchestration languages facilitates the building and large-scale deployment of cloud composite services, whose elementary resources may be hosted over several cloud providers and are subject to migrations over time. These migrations lead to configuration changes that may introduce new vulnerabilities. It is important to prevent these vulnerabilities to minimize the attack surface of cloud composite services. However, vulnerability prevention is challenged by the reluctance of stakeholders, namely the cloud tenant and the cloud provider, to share precise configuration information regarding respectively their cloud composite services and their cloud infrastructures, this information disclosure posing also security issues. In that context, 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 intermediate between the cloud tenant and the cloud provider, and is responsible for collecting configuration information and preventing configuration vulnerabilities before the migration of cloud resources. We have first described the considered architecture and its main building blocks, the trusted third party relying on a TOSCA interpreter and an OVAL-based vulnerability assessment framework. 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 OVAL-based vulnerability assessment framework, which exploits the open-source CVC4 SMT solver to analyze configurations and detect potential vulnerabilities. Finally, we have performed extensive series of experiments,in order to quantify the benefits and limits of our approach, in comparison to a baseline solution without trusted third party. In particular, we have shown to what extent the observability 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.

As future work, we are interested in considering optimization techniques to proactively collect configuration information from cloud providers, and exploiting complementary knowledge that may be provided by threat intelligence platforms, such as the MISP sharing environment [41], in order to drive our vulnerability prevention. We are also planning to work on guaranteeing the security of the trusted third party itself, including the interactions amongst the different stakeholders that should be supported by cryptographic security protocols. Finally, we will investigate the integration of our configuration security approach for cloud composite services to other security solutions proposed in the area of federated clouds, in particular with respect to authentication and control access.

Declarations

- Funding: supported by the CONCORDIA project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 83092.
- Competing interests: no conflicts of interest.
- Ethics approval: not applicable.
- Availability of data and code: OVAL and CVSS descriptions coming from the official open source repositories, code under deposit at INRIA and will be provided on a case-by-case manner.
- Authors' contributions: authors contributed equally to this work.

References

[1] Ray, B., Saha, A., Khatua, S. & Roy, S. Proactive Fault-Tolerance Technique to Enhance Reliability of Cloud Service in Cloud Federation Environment. *IEEE Transactions on Cloud Computing* 1–1 (2020). https://doi.org/10.1109/TCC.2020.2968522 .

[2] Ala'Anzy, M. & Othman, M. Load Balancing and Server Consolidation in Cloud Computing Environments: A Meta-Study. *IEEE Access* **7**, 141868–141887 (2019). https://doi.org/10.1109/ACCESS.2019.2944420 .

[3] Zhou, Z., Yu, J., Li, F. & Yang, F. Virtual Machine Migration Algorithm for Energy Efficiency Optimization in Cloud Computing. *Concurrency and Computation* **30** (2018). https://doi.org/10.1002/cpe.4942 .

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

[5] Opara-Martins, J., Sahandi, R. & Tian, F. Critical Analysis of Vendor lock-in and its Impact on Cloud Computing Migration: a Business Perspective. *Journal of Cloud Computing* **5** (2016). https://doi.org/10.1186/s13677-016-0054-z .

[6] Kumar, R. & Goyal, R. On Cloud Security Requirements, Threats, Vulnerabilities and Countermeasures: A Survey. *Computer Science Review* **33**, 1–48 (2019). URL https://www.sciencedirect.com/science/article/pii/S1574013718302065. https://doi.org/https://doi.org/10.1016/j.cosrev.2019.05.002 .

[7] Rajasree, S. & Elizabeth, B. Trust Based Cloud Service Provider Selection. *International Journal Of Engineering And Computer Science* (2016). https://doi.org/10.18535/ijecs/v5i5.63 .

[8] Gao, X., Gu, Z., Kayaalp, M., Pendarakis, D. & Wang, H. IEEE (ed.) *ContainerLeaks: Emerging Security Threats of Information Leakages in Container Clouds.* (ed.IEEE) *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 237–248 (2017).

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

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

[11] Nodehi, T., Jardim-Goncalves, R., Zutshi, A. & Grilo, A. ICIF: an Inter-cloud Interoperability Framework for Computing Resource Cloud Providers in Factories of the Future. *International Journal of Computer Integrated Manufacturing* **30** (1), 147–157 (2017). https://doi.org/10.1080/0951192X.2015.1067921 .

[12] Ramalingam, C. & Mohan, P. Addressing Semantics Standards for Cloud Portability and Interoperability in Multi Cloud Environment. *Symmetry* **13** (2) (2021) .

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

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

[15] Demchenko, Y., Turkmen, F., Slawik, M. & Laat, C. d. IEEE (ed.) *Defining Intercloud Security Framework and Architecture Components for Multi-cloud Data Intensive Applications.* (ed.IEEE) *Proc. of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 945–952 (2017).

[16] V Thomas, M., Dhole, A. & Chandrasekaran, K. Single Sign-On in Cloud Federation Using CloudSim. *International Journal of Computer Network and Information Security* **7**, 50–58 (2015). https://doi.org/10.5815/ijcnis.2015.06.06 .

[17] Bernal Bernabe, J., Martinez Perez, G. & Skarmeta, A. Inter-cloud Trust and Security Decision Support System: an Ontology-based Approach. *Journal of Grid Computing* **13** (2015). https://doi.org/10.1007/s10723-015-9346-7 .

[18] Compastié, M., Badonnel, R., Festor, O. & He, R. IEEE (ed.) *A TOSCA-Oriented Software-Defined Security Approach for Unikernel-Based Protected Clouds.* (ed.IEEE) *Proc. of the IEEE Conference on Network Softwarization (NetSoft)*, 151–159 (2019).

[19] Barrere, M., Badonnel, R. & Festor, O. IEEE (ed.) *A SAT-based Autonomous Strategy for Security Vulnerability Management.* (ed.IEEE) *Proc. of the IEEE Network Operations and Management Symposium (NOMS)* (2014).

[20] Anisetti, M., Ardagna, C. A. & Damiani, E. IEEE (ed.) *Security Certification of Composite Services: A Test-Based Approach.* (ed.IEEE) *Proc. of the IEEE International Conference on Web Services (ICWS)* (2013).

[21] Anisetti, M., Ardagna, C., Damiani, E. & Gaudenzi, F. A Semi-Automatic and Trustworthy Scheme for Continuous Cloud Service Certification. *IEEE Transactions on Services Computing* (2017) .

[22] Ismail, U. M., Islam, S. & Mouratidis, H. IEEE (ed.) *Cloud Security Audit for Migration and Continuous Monitoring.* (ed.IEEE) *Proc. of the the IEEE Trustcom Conference*, Vol. 1 (2015).

[23] Ullah, K. W., Ahmed, A. S. & Ylitalo, J. IEEE (ed.) *Towards Building an Automated Security Compliance Tool for the Cloud.* (ed.IEEE) *Proc. of the IEEE TrustCom Conference*, 1587–1593 (2013).

[24] Walkowski, M., Oko, J. & Sujecki, S. Vulnerability Management Models Using a Common Vulnerability Scoring System. *Applied Sciences* **11** (18) (2021). https://doi.org/10.3390/app11188735 .

[25] Celesti, A., Salici, A., Villari, M. & Puliafito, A. IEEE (ed.) *A remote attestation approach for a secure virtual machine migration in federated cloud environments.* (ed.IEEE) *Proc. of the First International Symposium on Network Cloud Computing and Applications*, 99–106 (2011).

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

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

[28] Herrmann, D. S. *Using the Common Criteria for It Security Evaluation* (CRC Press, Inc., USA, 2002).

[29] Schnepf, N., Badonnel, R., Lahmadi, A. & Merz, S. IEEE (ed.) *Automated Verification of Security Chains in SDN Networks with Synaptic.* (ed.IEEE) *Proc. of the Conference on Network Softwarization (NetSoft)* (2017).

[30] Gupta, B., Mittal, P. & Mufti, T. IEEE (ed.) *A Review on Amazon Web Service (AWS), Microsoft Azure and Google Cloud Platform (GCP) Services.* (ed.IEEE) (EAI, 2021).

[31] Neto, M. Z. *et al. Security Troubleshooting on AWS*, 339–362 (IEEE, 2021).

[32] Jalili, V., Afgan, E., Taylor, J. & Goecks, J. Cloud bursting galaxy: federated identity and access management. *Bioinformatics* **36** (1), 1–9 (2019). https://doi.org/10.1093/bioinformatics/btz472 .

[33] Potti, S. Supercharging security with generative AI (2023). URL https://cloud.google.com/blog/products/identity-security/rsa-google-cloud-security-ai-workbench-generative-ai?hl=en.

[34] Coppolino, L., D'Antonio, S., Mazzeo, G. & Romano, L. Cloud Security: Emerging Threats and Current Solutions. *Computers and Electrical Engineering* **59**, 126–140 (2017). URL https://www.sciencedirect.com/science/article/pii/S0045790616300544. https://doi.org/https://doi.org/10.1016/j.compeleceng.2016.03.004 .

[35] Ramachandra, G., Iftikhar, M. & Khan, F. A. A Comprehensive Survey on Security in Cloud Computing. *Procedia Computer Science* **110**, 465–472 (2017). URL https://www.sciencedirect.com/science/article/pii/S1877050917313030. https://doi.org/https://doi.org/10.1016/j.procs.2017.06.124, 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017) / 12th International Conference on Future Networks and Communications (FNC 2017) / Affiliated Workshops .

[36] CloudFormation, A. AWS CloudFormation API Reference (2020).

[37] Esposito, A., Di Martino, B. & Cretella, G. IEEE (ed.) *Defining Cloud Services Workflow: a Comparison between TOSCA and OpenStack Hot.*

(ed.IEEE) (2015).

[38] NIST. XCCDF - The Extensible Configuration Checklist Description Format (2020). URL https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/xccdf.

[39] Booth H., D., Rike & Witte, G. The National Vulnerability Database (NVD): Overview, ITL Bulletin, National Institute of Standards and Technology (2020). URL https://tsapps.nist.gov/publication.

[40] Scarfone, K. & Mell, P. IEEE (ed.) *An Analysis of CVSS version 2 Vulnerability Scoring*. (ed.IEEE) *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 516–525 (2009).

[41] Wagner, C., Dulaunoy, A., Wagener, G. & Iklody, A. IEEE (ed.) *MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform*. (ed.IEEE) *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, 49–56 (ACM, 2016).