CMSec: A Vulnerability Prevention Tool for Supporting Migrations in Cloud Composite Services

Mohamed Oulaaffart, Rémi Badonnel, Olivier Festor

▶ **To cite this version:**

Mohamed Oulaaffart, Rémi Badonnel, Olivier Festor. CMSec: A Vulnerability Prevention Tool for Supporting Migrations in Cloud Composite Services. CloudNet 2022 - IEEE International Conference on Cloud Networking, Nov 2022, Paris, France. hal-03886094v2

HAL Id: hal-03886094
https://hal.inria.fr/hal-03886094v2

Submitted on 9 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

CMSec: A Vulnerability Prevention Tool for Supporting Migrations in Cloud Composite Services

Mohamed Oulaaffart
RESIST Research Team
LORIA / INRIA Nancy Grand Est,
University of Lorraine, CNRS
54600 Villers-les-Nancy, France
mohamed.oulaaffart@loria.fr

Rémi Badonnel
RESIST Research Team
LORIA / INRIA Nancy Grand Est,
University of Lorraine, CNRS
54600 Villers-les-Nancy, France
remi.badonnel@loria.fr

Olivier Festor
RESIST Research Team
LORIA / INRIA Nancy Grand Est,
University of Lorraine, CNRS
54600 Villers-les-Nancy, France
olivier.festor@loria.fr

***Abstract*—The growing maturity of orchestration languages, such as the Topology and Orchestration Specification for Cloud Applications (TOSCA) language, and the continuous improvement of virtualization techniques contribute to the large-scale deployment of cloud composite services based on multiple resources that may be hosted over different cloud service providers. The resources composing these services may be subject to migrations, either live or offline, in order to cope with performance objectives, such as load balancing, latency minimization, and energy savings. However, these migrations may induce changes affecting the configurations of migrated resources. In particular, these changes may lead to vulnerabilities, that may compromise the security of resources, or even the security of the whole cloud service. This demonstration showcases a vulnerability prevention tool, called CMSec, for supporting the resource migrations of TOSCA-based cloud composite services by considering OVAL vulnerability descriptions together with verification techniques. Based on two main scenarios, we will expose how the CMSec tool is able to detect vulnerable configurations prior to resource migrations, and to suggest corrective operations for remediating them.**

I. BACKGROUND

The maturity of orchestration languages contribute to the design and deployment of elaborated cloud services through the composition and configuration of multiple computing resources . These resources are subject to changes over time, that may cause vulnerabilities compromising the resources, or even the whole cloud service. The recent advances on virtualization techniques facilitate cold and hot migrations, that consist in transferring the resource(s) of a cloud composite service from a given provider (or a given infrastructure) to another provider (or infrastructure). While it is often motivated by performance and cost objectives, the migration process may impact the protection of cloud services by involuntarily generating configuration vulnerabilities. It is therefore of major importance to prevent such vulnerabilities in order to minimize the exposure of these services to security attacks.

Supported by the project CONCORDIA that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 830927

The challenge of automating the protection of cloud services and their underlying resources have already been investigated in the literature through several security approaches. In particular, endogenous security mechanisms directly impact the resources, such as generating specific cloud images with a low attack surface [1], modifying the internal parameterization of resources [2], and exploiting certification techniques for guaranteeing the resource behaviours [3]. However, these solutions are often not designed for addressing the properties of cloud environments (e.g. elasticity, heterogeneity), or do not focus on vulnerable configurations. In addition, exogenous security mechanisms enable protecting cloud resources through the exploitation of external security functions. For instance, a security orchestrator based on the NFV MANO[1] architecture has been introduced in [4] for supporting the deployment of network security functions in front of the considered resources. However, such solutions have mainly investigated access control rules that are specified using orchestration language extensions. In order to strengthen the security of cloud resources, several approaches have also focused on trust features for cloud infrastructures, by considering the virtualization systems used in these environments, such as enabling secure migration based on trusted platform modules [5]. These methods are complementary to our solution for cloud composite services, which addresses configuration vulnerabilities that may occur even in fully trusted cloud environments.

In this demonstration paper we will present the implementation architecture of our CMSec vulnerability prevention tool, detail its main components and their interactions, and finally describe the two main scenarios considered for the demonstration.

II. CLOUD MIGRATION SECURITY TOOL

The proposed tool, called CMSec (Cloud Migration Security), aims at preventing known vulnerabilities that may affect cloud composite services during the migration of their

[1]Management And Network Orchestration

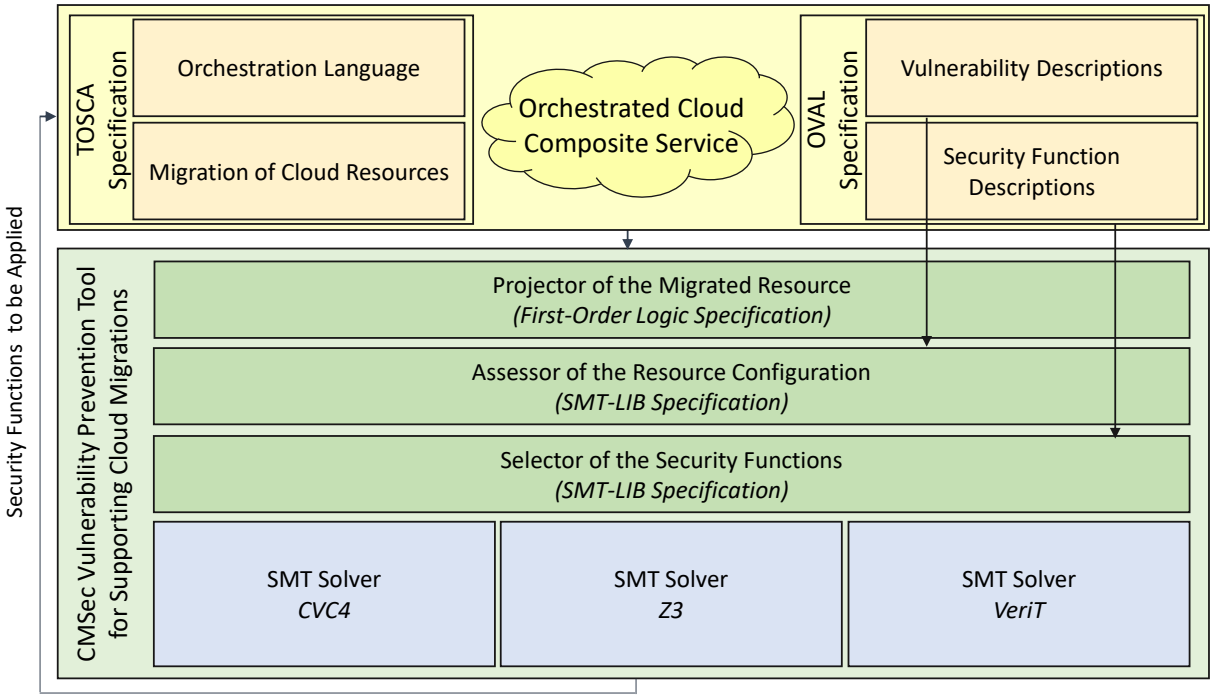

Fig. 1: Main building blocks of the CMSec vulnerability prevention tool for supporting cloud migrations

resources, through the assessment of their configurations and the selection of adequate countermeasures when vulnerabilities are identified. It corresponds to the implementation of the strategy that we have formalized in [6]. The CMSec architecture, detailed on Figure 1, has been prototyped using the Python language version 3.7, and relies on SMT solvers compatible with the SMT-LIB (Satisfiability Modulo Theories LIBrary) format [7] as back-end services. As shown on the figure, the three main building blocks (represented in green) of the CMSec tool include: (1) the migrated resource projector, (2) the resource configuration assessor, and (3) the security function selector, that intervene successively in the process of preventing configuration vulnerabilities during the migration of resources in cloud composite services.

The CMSec tool considers cloud composite services orchestrated using the TOSCA[2] orchestration language [8]. This language is both open-source and characterized by a high expressivity for supporting flexible, interoperable and distributed applications regardless of the underlying cloud service providers. It provides an important knowledge source about cloud services to our tool, by describing the elementary resources of these services, as well as their relationships. In particular, it specifies a cloud service as a TOSCA topology which defines a set of TOSCA nodes (e.g. web micro-service nodes) that are interconnected by a set of TOSCA relationships (e.g. the interconnection to MySQL database nodes). In addition, our CMSec tool relies on a set of vulnerability descriptions coming from the official OVAL[3] repository to perform vulnerability prevention activities. OVAL is an open standardized language supported by MITRE [9] for describing vulnerability descriptions 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 is 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. The OVAL language provides also a support for specifying countermeasures (also called security functions) that may be used to remediate identified configuration vulnerabilities (e.g. applying a given security patch to the cloud resource, or activating a specific firewall rule).

[2]Topology and Orchestration Specification for Cloud Applications

[3]Open Vulnerability and Assessment Language

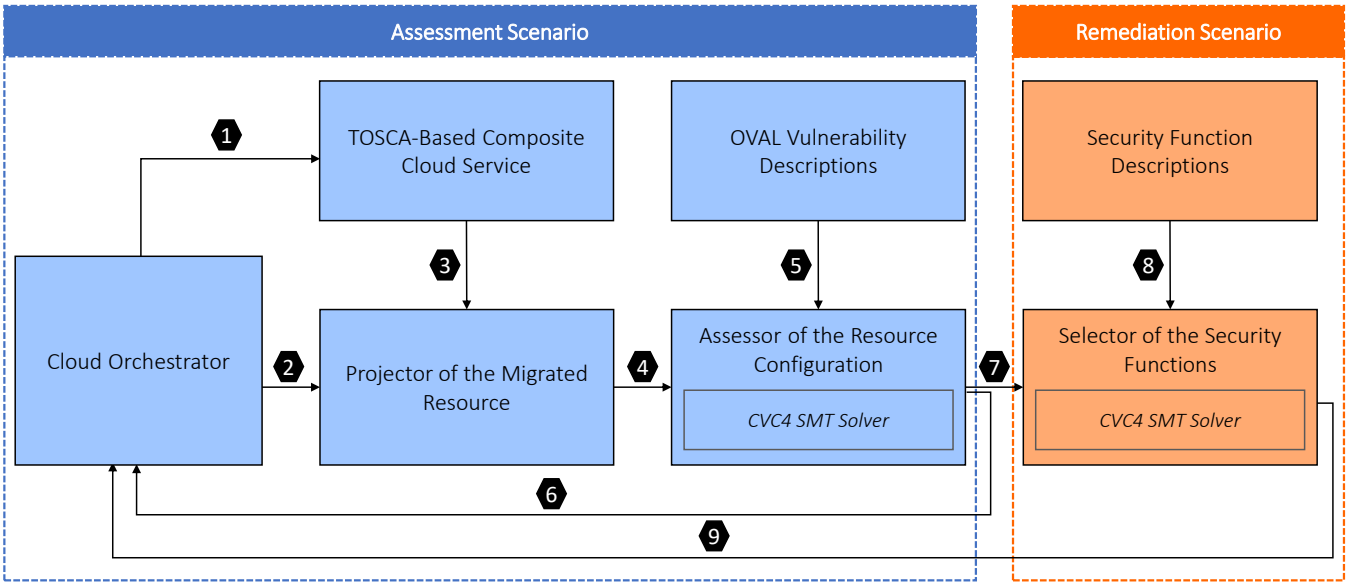

Fig. 2: Two main demonstration scenarios corresponding respectively to the vulnerability assessment of a migrated cloud resource (in blue color) and the selection of countermeasures (in orange color) by the CMSec vulnerability prevention tool

A. Migrated Resource Projector

The first building block, called 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 permits also to identify the other resources that may be impacted by the considered migration.

B. Resource Configuration Assessor

The second building 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, 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 CVSS[4] scoring associated to 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. We mainly consider the CVC4 SMT solver [10], which provides better overall performance than other SMT solvers according to recent benchmarking on open-source solvers [11]. However, the CMSec tool also supports other solvers compatible with the SMT-LIB format, such as Z3 [12] and VeriT [13]. When configuration vulnerabilities are detected by this configuration assessor, the tool activates the third building block to determine countermeasures.

[4]Common Vulnerability Scoring System

C. Security Function Selector

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

III. The Demonstration

In this demonstration we will show the operation of the CMSec vulnerability prevention tool for supporting migrations

in cloud composite services orchestrated with the TOSCA language, by considering two main scenarios (shown on Figure 2) illustrating respectively the assessment of a migrated resource and the selection of countermeasures. In that context, we consider an e-commerce cloud composite service involving a set of micro-services together with back-end databases. The micro-services are implemented in the form of a web application built with the Struts open-source framework, that 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.

A. Assessment Scenario

The first scenario, depicted in blue color on the figure, 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, 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 CMSec vulnerability prevention tool (*Arrow 2*) before performing the effective migration. The CMSec tool 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 (*Arrow 5*). The vulnerability assessment process is formalized as a satisfiability issue [6], by considering the vulnerability descriptions together with the resource projection, and generates 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. When no vulnerability is identified with regard to known vulnerable configuration, the CMSec tool notifies the cloud orchestrator that the Struts web micro-service can be migrated to the destination cloud environment (*Arrow 6*).

B. Remediation Scenario

The second main scenario, depicted in orange color on the figure, 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 [14], 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 when the migration is requested. First, the CMSec vulnerability prevention tool 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 are identified by the CMSec tool: the first one corresponds to the execution of an internal patch over the cloud resource, while 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 may be conditioned by several factors, such as the portability, the interoperability and the operation of other resources using the same cloud platform.

C. Performance Evaluation

In addition to these two main scenarios, the demonstration will also present some performance evaluation results, in particular regarding the vulnerability assessment process with different vulnerability description dataset from the official OVAL repository. The experiments are performed over a regular laptop equipped with a 2Ghz Intel Core i5 processor and 8GB of RAM memory. The experimental results are synthesized on Table I, considering a projected configuration corresponding to an average of 12 properties. We can observe that the vulnerability assessment time takes less than 2 seconds with all the considered OVAL datasets. 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. We will also quantify during the demonstration the overall execution time required by the CMSec vulnerability prevention tool to

Operating Systems	Nb. of Vulnerabilities	Assessment Time
CentOS Linux 3	1426	0.44s
CentOS Linux 4	1623	0.48s
CentOS Linux 5	1171	0.39s
Red Hat Enterprise Linux 3	1539	0.44s
Red Hat Enterprise Linux 5	1172	0.38s
MS Windows 8	4669	0.72s
MS Windows Svr 2012 R2	5344	1.53s
MS Windows Svr 2016	3790	1.29s
MS Windows Svr 2019	1459	1.01s

TABLE I: Vulnerability assessment time considering different operating system datasets from the official OVAL repository

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, during our experiments.

This demonstration showcases the architecture of the CM-Sec vulnerability prevention tool, as well as the benefits and limits of this solution through concrete usage scenarios. As future work, we are interested in investigating to what extent this tool could be integrated into the architecture of a trusted third-party to support inter-cloud configuration security.

REFERENCES

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

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

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

[4] 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*, Dec. 2018.

[5] B. Danev, R. J. Masti, G. O. Karame, and S. Capkun, "Enabling Secure VM-vTPM Migration in Private Clouds," in *Proc. of the Annual Computer Security Applications Conference (ACSAC)*. Association for Computing Machinery, Dec. 2011, pp. 187–196.

[6] 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/IFIP Network Operations and Management Symposium (NOMS)*, 2022, pp. 1–9.

[7] C. W. Barrett, A. Stump, and C. Tinelli, "The SMT-LIB Standard Version 2.0," 2010.

[8] C. L. Paul Lipton. TOSCA Simple Profile in YAML Version 1.3. [Online]. Available: https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/os/TOSCA-Simple-Profile-YAML-v1.3-os.html

[9] Mitre. OVAL Vulnerability Description Repository. [Online]. Available: https://oval.cisecurity.org/repository

[10] C. W. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, and C. Tinelli, "CVC4," in *Proc. of the International Conference on Computer Aided Verification (CAV)*, ser. Lecture Notes in Computer Science, vol. 6806. Springer, 2011, pp. 171–177.

[11] H. Barbosa, C. Barrett, F. Bobot, and M. Brain. (2020) About CVC4. [Online]. Available: https://cvc4.github.io/

[12] L. De Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Proc. of the International Conference on Theory and Practice of Software*. Berlin, Heidelberg: Springer-Verlag, 2008, p. 337–340.

[13] T. Bouton, D. Caminha B. de Oliveira, D. Déharbe, and P. Fontaine, "VeriT: An Open, Trustable and Efficient SMT Solver," in *Automated Deduction*, R. A. Schmidt, Ed. Springer Berlin Heidelberg, 2009.

[14] R. D. Booth, H. and G. Witte. The National Vulnerability Database (NVD): Overview, ITL Bulletin, National Institute of Standards and Technology. [Online]. Available: https://tsapps.nist.gov/publication