Cloud Orchestration at the Level of Application

Project Acronym: **COLA**

Project Number: **731574**

Programme: **Information and Communication Technologies**

Advanced Computing and Cloud Computing

Topic: **ICT-06-2016 Cloud Computing**

Call Identifier: **H2020-ICT-2016-1**

Funding Scheme: **Innovation Action**

Start date of project: 01/01/2017 Duration: 33 months

Deliverable:

D2.5 Report on Standardization Activities

Due date of deliverable: 31/07/2019 Actual submission date: 28/09/2019

WPL: CloudSME UG

Dissemination Level: PU

Version: Final

Table of Contents

1. List of Figures and Tables

Figures

Tables

2. Status and Change History

Status:	Name:	Date:	Signature:
Draft:	Gabriele Pierantoni	26/07/19	Gabriele Pierantoni
Reviewed:	Anastasia Anagnostou	04/09/19	Anastasia Anagnostou
Approved:	Tamas Kiss	28/09/19	Tamas Kiss

Table 1 - Status Change History

Version	Date	Pages	Author	Modification
0.1	08/07/19	All	UoW	Skeleton
1.0	25/07/19	All	All	First Draft with contributions from all the partners
1.1	26/07/19	7-11, 16-17	J. DesLauriers	Added section 5,6,7
1.2	26/07/19	5-6	G. Pierantoni	Added Sections 4
2.0	14/08/19	All	G. Pierantoni	First complete version
2.1	04/09/19	All	A. Anagnostou	Reviewing the deliverable
2.2	14/09/19	All	G. Pierantoni	Modifying the deliverable considering the reviewer's comments
2.3	23/09/19	All	G Terstyanszky	Extending section 4 and 7
final	28/09/19	All	T. Kiss	Finalizing and submitting the deliverable

Table 2 - Deliverable Change History

3. Glossary

TOSCA	Topology and Orchestration Specification for Cloud Applications
DSL	Domain Specific Language
CSP	Cloud Service Provider
ADT	Application Description Template
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
CSA	Cloud Security Alliance
ISMS	Information Security Management System
TLS	Transport Layer Security
IKE	Internet Key Exchange
IPsec	Internet Protocol Security
PKI	Public Key Infrastructure
RBAC	Role-based access control
TPM	Trusted Platform Module
IETF	Internet Engineering Task Force
INCITS	International Committee for Information Technology Standards
NIST	National Institute of Standards and Technology
RFC	Request for Comments
FIPS	Federal Information Processing Standards

Table 3 - Glossary

4. Introduction

This document describes the overall approach COLA took regarding standards and standardization considering technologies used to describe, deploy and run applications in the Cloud through the MiCADO framework. Further, it gives a short overview of standards used to describe applications (WP5), to deploy and run application in the Cloud (WP4 and WP6) and to provide security to manage applications (WP7). Finally, it outlines the standardisation activities undertaken in COLA.

4.1 Objectives of the overall work package

WP2 was responsible for dissemination, communication, training and standardisation activities in order to share the results achieved during the project with the wider community.

Task 2.5 Contribution to standardisation led COLA activities related to standards and standardisation. This task is described in the DOA as follows: “Standardisation of cloud orchestration at application level will be primarily pursued via the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) Technical Committee (TC). The COLA project will establish connection with and will contribute to the work of this TC. UoW will lead this task, as the leader of the application template definition activities, with contribution from other academic and technology provider partners involved in platform development and standardisation.” T2.5 monitored and coordinated how standards are used in the MiCADO framework and how standardization activities are carried out across different work packages targeting **objective 2.5 “**To drive and significantly contribute to the standardisation of cloud orchestration at application level.”

WP2 compiled **D2.5 Report on standardisation activities** to describe what standards were used by WP4-WP7 to design and implement the MiCADO framework and to summarise the contribution of the COLA project to standardisation of cloud orchestration at applications level.

4.2 Structure of the deliverable

This Deliverable is structured as follows:

- Section 4 introduces the deliverable and describes the WP2 objective (O2.5) that this deliverable addresses.
- Section 5 gives a short overview of how standards were considered and used in the MiCADO framework.
- Section 6 describes how TOSCA, an OASIS standard, is used to describe cloud application (WP5).
- Section 7 details standards used to design and implement the Orchestration Layer of the MiCADO framework (WP6).
- Section 8 explains the standards upon which MiCADO security services have been built (WP7).
- Section 9 outlines the standards to which to cloud interfaces must be compliant (WP4)
- Section 10 describes COLA standardisation activities in application description and security.
- Section 11 concludes the report and lists some future works on standardisation.

5. Standards Used and Standardization Efforts in Cola

5.1 MiCADO framework

MiCADO is generic framework implemented as a set of microservices to support description, deployment and execution of applications in the Cloud. The generic layered architecture of MiCADO, that has been included in several previous deliverables, is illustrated in Figure 1. This deliverable only provides a very brief overview of these layers in order to identify at what layers standardization is desired and implemented in MiCADO.

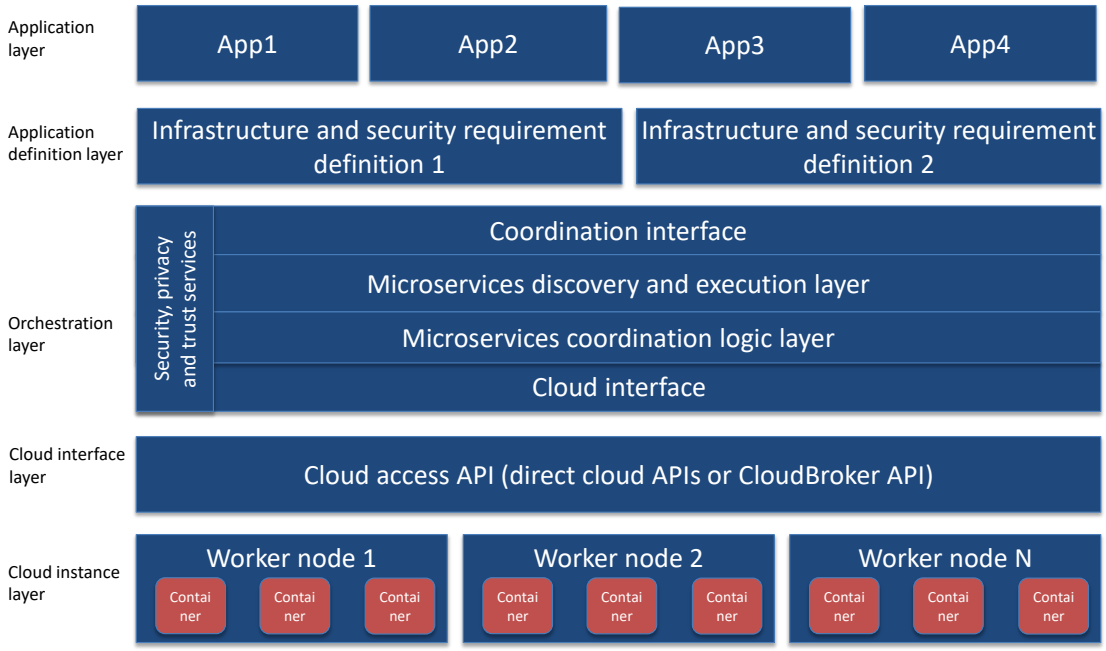

Figure 2 MiCADO framework

MiCADO has a multi-layered architecture with five horizontal layers:

- **Application layer**. Application layer contains the actual application code and data described by application definition (layer 2) to work in such a way that a desired functionality is reached. For example, this layer could populate database with initial data, and configure HTTP server with look and feel and application logic.
- **Application definition layer**. This layer allows definition of the functional architecture of applications using application templates. At this level, software components and their requirements (both infrastructure and security specifications) as well as their interconnectivity are defined using application descriptions uploaded to a public repository. As the infrastructure is agnostic to the actually executed application, the application template can be shared with any application that requires such an environment.
- **Orchestration layer.** This layer is divided into four horizontal and one vertical sub-layers. The horizontal sub-layers are:
  - **Coordination interface API.** This sub-layer provides access to the orchestration layer and decouples it from the application definition layer. This set of APIs enables application developers to utilize the dynamic orchestration capabilities of the underlying layers and supports the convenient development of dynamically and automatically scalable cloud-based applications by embedding these API calls into application code.
  - **Microservices discovery and execution layer.** This sub-layer manages the execution of microservices and keeps track of services running. Execution management combines both start-up, running and shut down of microservices.

Service management gathers information about currently running services, such as service name, IP address and port where the service is reachable and optional service tags to help service coordination.

- **Microservices coordination logic.** To reap the benefits from cloud-based execution, it becomes necessary to understand how the current execution environment is performing. Information needs to be gathered and processed. If bottlenecks are detected or the currently running infrastructure appears underutilized, it may be necessary to either launch or shut down cloud instances, and possibly move microservices from one physical worker node to another.
- **Cloud interface API.** It is responsible for abstracting cloud access from layers above. Cloud access APIs can be complex interfaces, as they typically cater for a large number of services provided by the cloud provider. On the other hand, the microservices discovery and execution and coordination logic layers only need to shut down and start instances. Abstracting this to a cloud interface API simplifies the implementation of the aforementioned layers, and if new Cloud access APIs are implemented, only this layer needs to change.

The vertical sub-layer is:

- **Security, privacy and trust services.** These services span among multiple levels of the orchestration layer, as it is illustrated in Figure 2. The main aim is to save the application developers from detailed security management. To achieve this, the security, privacy and trust services of the orchestration layer take the general security policies defined at the Application definition layer, as well as security credentials for the application domain. These inputs are used by the special purpose security policy enforcement services to enforce the security policies at orchestration level.

- **Cloud interface layer.** This layer provides functions to launch and shut down cloud instances. There can be one or more cloud interfaces to support multiple clouds. Besides directly accessing cloud APIs, generic cloud access services, such as the CloudBroker platform [22] can be also used at this layer to support accessing multiple, heterogeneous and distributed clouds via its uniform access layer.
- **Cloud instance layer.** This layer contains cloud instances provided by Infrastructure-as-a-Service (IaaS) cloud providers. These instances can run various containers that execute actual microservices. The layer typically represents state-of-the-art of cloud technology provided by various public or private cloud providers.

5.2 Standards and the MiCADO framework

As major technology output, the COLA Project implemented the MiCADO framework, as introduced in the previous section. In order to implement the layered MiCADO architecture, COLA deals with multiple interconnected activities many of which have significant relationship with standardization and interoperability efforts. The overall picture of these activities is illustrated in Figure 2 highlighting how the various layers of MiCADO, implemented in different COLA Work Packages relate to standards and standardisation efforts. These standardisation efforts are summarised below by each Work Package.

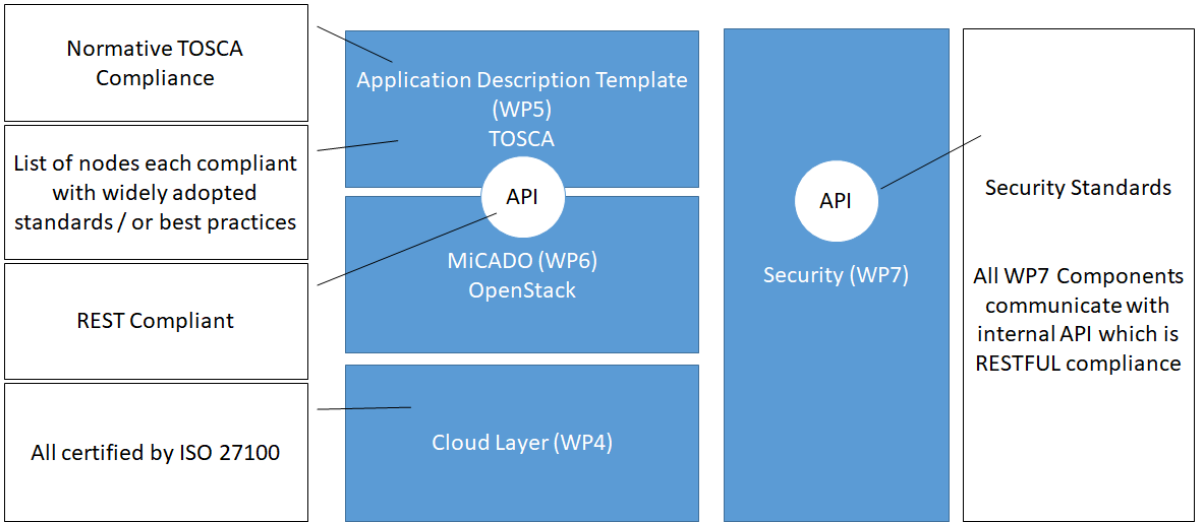

Figure 2 – Relationship of COLA work packages with standards

WP5 Application Description Template. The work package designed and implemented the Application Description Templates (ADTs) in compliance with OASIS TOSCA Language Specification [1]. ADTs are created in the Application Definition layer of the MiCADO framework (See Fig. 1). WP5 created a list of standardized nodes that can be easily reused to deploy applications to widely used cloud middleware (e.g. AWS, OpenStack, etc.) and to follow best practices in cloud deployment and execution. The work package is having an ongoing effort to actively cooperate with the OASIS committee that defines the future standards of the next TOSCA specifications. Although there have been interesting developments (OASIS is well aware of the COLA efforts), a formal cooperation requires further steps and formal commitment, mainly from the University of Westminster. We envisage that such a formal cooperation will bring benefit to the MiCADO framework development community and the research in the University of Westminster and, even after the formal end of the COLA project. Further details on how TOSCA is used in ADTs and standardization efforts of WP5 are described in Sections 6 and 10, respectively.

WP6 - Microservices deployment and execution layer. The MiCADO Orchestration layer (See in Fig 1.) processes ADTs, written in TOSCA, to deploy and run applications in the Cloud. ADTs are forwarded to the MiCADO Submitter through a REST compliant interface that supports the standard REST operations. Details on standards used in the Orchestration layer to manage applications are given in Section 7.

WP7 – **Security, privacy and trust at the level of cloud applications**. This work package deals with privacy, security and trust services of the vertical sub-layer of the Orchestration layer. These services span the entire technological stack of the project. It is composed of various services that interact with each other through a standardized REST interface to improve flexibility and adaptability. The list of standards to which WP7 implementation complies with is detailed in Section 8.

WP4 - **Cloud access layer and testbed infrastructure.** Cloud access layers do not employ a single standardized cloud interface as such interfaces do not exist. However, all the technologies it encompasses are compliant to the ISO 27100 Certification. Further details on standards used in the Cloud interface layer (See Fig.1) are given in Section 9.

6. Standards Used in Application Description (WP5)

WP5 elaborated the Application Description Template (ADT) to describe applications in order to support their deployment and execution in the Cloud. ADT describes the application in full, including the cloud infrastructure which supports it, the containers inside the applications run, and the policies which regulate aspects of the application's orchestration and lifecycle such as scalability and security. ADT is compliant to TOSCA. (Topology and Orchestration Specification for Cloud Applications) [1]. TOSCA is an OASIS standard used to describe applications in the Cloud. TOSCA aims to avoid vendor lock-in and ensure simplified portability and management of applications and infrastructure across different public, private and hybrid clouds. TOSCA Simple Profile in YAML replaces the original XML-based TOSCA DSL (Domain Specific Language) [2] with a more readable data serialisation standard, YAML (YAML Ain't Markup Language) [3]. When the Application Description Template in MiCADO was conceptualised, TOSCA Simple Profile in YAML was at v1.0. Since then, OASIS has accepted v1.1 and v1.2 as standards, and the ADT has taken strides to keep up to date with new advancements in these versions.

The Application Description Template extends normative TOSCA with a set of rich types specific to applications in MiCADO. These types follow the TOSCA model and remain compliant with TOSCA in YAML v1.0. MiCADO uses the open-source OpenStack TOSCA Parser, which automatically validates all templates before parsing. Some of the concepts and structures of the TOSCA standard which we follow and benefit from in ADTs are:

- **TOSCA Types.** TOSCA works strongly with types, which provide variable levels of abstraction and inheritance. Normative TOSCA defines basic *types* for all TOSCA structures (nodes, relationships, interfaces, etc.) and wherever possible, MiCADO ADTs reuse these types. Where TOSCA normative types are not sufficient, we define custom types, always ensuring that these derive from a TOSCA normative type. Custom parent types can specify defaults, constraints and requirements for types which derive from them. The restrictions or rules defined by the parent type are then enforced in all children types when the ADT is processed.
- **TOSCA Nodes.** TOSCA uses the concept of nodes. A node is any component of an application in the cloud topology including virtual machines, containers, volumes, software and networks. MiCADO supports virtual machines and pre-built application containers, and uses TOSCA nodes in an ADT to describe their desired state. Auxiliary components such as volumes and networks can also be described as nodes when needed. Some basic node types which are present in MiCADO ADTs are:
  - tosca.nodes.MiCADO.Compute.EC2 (derived from *tosca.nodes.Compute*)
  - tosca.nodes.MiCADO.Container.Volume (derived from *tosca.nodes.BlockStorage*)
- **TOSCA Interfaces.** Interfaces in TOSCA are used to manage the lifecycle of cloud applications. The TOSCA Standard Interface is used to set input parameters or environment variables for custom scripts or code snippets which then manage specific lifecycle operations of a given node. In the case of cloud orchestration, for example, these operations might be *create virtual machine* or *stop virtual machine*. For MiCADO ADTs we define custom interface types for managing lifecycle operations using MiCADO components. These custom types derive from the normative *tosca.interfaces.node.lifecycle.Standard*.
- **TOSCA Relationships.** TOSCA relationships define the interoperability between nodes. Basic examples of relationships might include hosting one node on another, or attaching nodes together. In MiCADO ADTs the normative *tosca.relationships.HostedOn* is used to restrict a given application container to running on a specified virtual machine. For volume management in ADTs, the normative type *tosca.relationships.AttachesTo* is used when attaching a volume to an application container.

7. Standards Used in Application Orchestration (WP6)

Users initiate application deployment and execution submitting an Application Description Template (ADT) to the MiCADO Orchestration layer. An API was designed to support this submission. The API follows Representational State Transfer (REST) concept, a widely used architectural style and set of best practices, which is built on top of several well-known standards, seen in Table 4 and explained below:

Requirements	**Standard** *SDO*
Data interchange	RFC8259: JavaScript Object Notation; *IETF*
Messaging protocol	RFC6585: Hypertext Transfer Protocol; *IETF*
Resource Pointer	RFC3986: Uniform Resource Identifier (URI); *IETF*

Table 4 – Standards used in application orchestration

- **JavaScript Object Notation** (JSON) [4] is a lightweight, text-based, language-independent data interchange format. It was derived from the ECMAScript Programming Language Standard. JSON defines a small set of formatting rules for the portable representation of structured data.
- **HTTP**. (HyperText Transfer Protocol) [5] The communications protocol used to connect to Web servers on the Internet or on a local network (intranet). Its primary function is to establish a connection with the server and send HTML pages back to the user's browser.
- **URI** (Uniform Resource Identifier) [6] is a compact sequence of characters that identifies an abstract or physical resource. This specification defines the generic URI syntax and a process for resolving URI references that might be in relative form, along with guidelines and security considerations for the use of URIs on the Internet. The URI syntax defines a grammar that is a superset of all valid URIs, allowing an implementation to parse the common components of a URI reference without knowing the scheme-specific requirements of every possible identifier. This specification does not define a generative grammar for URIs; that task is performed by the individual pecifications of each URI scheme.

The API for ADT submission defines resources as the applications currently deployed to the MiCADO framework. When an ADT is submitted to the MiCADO framework, an HTTP POST call generates the resource for the desired deployment. This resource can be updated with PUT, which updates the deployment, or deleted with DELETE, which removes the deployment from the MiCADO framework. While not a strict standard, use of a RESTful API makes submission to the MiCADO framework a more familiar process for developers and operators.

8. Security Standards in the MiCADO Framework (WP7)

This section summarises cryptographic techniques and network protocols used in the MiCADO security architecture designed and implemented by WP7. It also lists the standards behind these cryptographic techniques and network protocols.

8.1 Cryptographic and security standards used in the MiCADO framework

- **Transport Layer Security (TLS) Protocol – RFC5246** [7]. The TLS protocol aims at providing data privacy and integrity for secure communication between two entities. It prevents attacks such as eavesdropping, tampering, and message forgery. The protocol was first defined as TLS 1.0 in RFC 2246 in 1999 as an Internet Engineering Task Force (IETF) standard. The next versions include TLS 1.1 defined in RFC 4346 in 2006, TLS 1.2 in RFC 5246 in 2008, and TLS 1.3 in RFC 8446 in 2018. In the project, we utilize TLS 1.2 to protect communication between users and the MiCADO framework.
- **Asymmetric Key Packages - RFC5958** [8]. It defines the syntax for private-key information and a content type for it. Private-key information includes a private key for a specified public-key algorithm and a set of attributes. The Cryptographic Message Syntax (CMS), as defined in RFC 5652, can be used to digitally sign, digest, authenticate, or encrypt the asymmetric key format content type. In the project, we extensively rely on public key cryptography to protection network communication as well as integrity attestation of platforms.
- **Internet Key Exchange (IKE) – RFC2409** [9]. The IKE aims at establishing the shared security attributes between two entities to support their secure communication. It was first defined as IKEv1 in RFC 2409 by Internet Engineering Task Force (IETF). The next version IKEv2 was proposed in RFC 4306 in 2005, then improved in RFC 5996 in 2010, and finally upgraded to an Internet Standard and published in RFC 7296 in 2014. In the project, we configure to use IKEv2 in IPsec settings in order to protect communication among MiCADO master node and worker nodes.
- **Internet Protocol Security (IPsec) – RFC6071** [10]. IPsec protocol suite aims at authenticating and encrypting the data packets sent over the Internet Protocol network. It protects data flows between two entities. It was first standardized as an open standard by the Internet Engineering Task Force (IETF) IP Security Working Group, and published in RFC-1825 through RFC-1827 in 1995. In the project, we utilize IPsec provided by the open source strongSwan (https://www.strongswan.org) to protect communication among MiCADO master node and worker nodes. The strongSwan is an IPsec-based VPN solution which complies with multiple standards such as IKEv2 (RFC 7296), NAT-Traversal via UDP encapsulation and port floating (RFC 3947) and Dead Peer Detection (DPD, RFC 3706).
- **Internet X.509 Public Key Infrastructure (PKI) - RFC2510** [11]. X.509 standard aims at defining the format for public key certificates. It was first issued in 1988, and specified in RFC 5280 in 2008. In the project, we utilize X.509 certificates for TLS protocol and master node - worker node secure communication.
- **Secure Hash Algorithm (SHA1) – RC3174** [12]**.** It describes the use of secure hashing algorithms. NIST-approved digital signature algorithms require the use of an approved cryptographic hash function in the generation and verification of signatures. Approved cryptographic hash functions and digital signature algorithms can be found in FIPS 180-3, Secure Hash Standard (SHS), and FIPS 186-3, Digital Signature Standard (DSS), respectively. The security provided by the cryptographic hash function is vital to the security of a digital signature application. This Recommendation specifies a method to enhance the security of the cryptographic hash functions used in digital signature applications by randomizing the messages that are signed. Within project COLA, secure hashing algorithms are used in the Image Integrity Verifier Component.

- **Trusted Platform Module (TPM) - ISO/IEC 11889** [13]**.** The TPM Library specification v. 2.0 describes the library access format to Trusted Platform Modules (TPMs). TPMs are a basic building block used in most other specifications, for providing an anchor of trust. They can be used for validating basic boot properties before allowing network access (TNC), or for storing platform measurements (PC Client), or for providing self-measurement to provide anchors of trust to hypervisors (Virtualization). Within project COLA, the Trusted Platform Module library specification is used in the Image Integrity Verifier Component.

Table 5 summarizes the relevant cryptographic and security standards and techniques used in the MiCADO security enablers.

No.	Cryptographic techniques/ Network protocol	Relevant standard	Example Security Enablers
1	TLS 1.2	RFC 5246	Zorp
2	Assymetric Key Package	RFC 5958	Zorp, Integrity Image Verifier
3	IKEv2	RFC 7296	Zorp
4	IPsec	RFC-1825 through RFC-1827	Zorp
5	X.509 Public Key Infrastructure	RFC 5280	Security Policy Manager, Master node - worker node communication, CryptoEngine, Zorp
6	SHA1, SHA256	NIST Special Publication 800-106	Image Integrity Verifier
7	strongSwan	RFC 7296, RFC 3947, RFC 3706	Master node - worker node communication
8	Role-based access control (RBAC)	INCITS 359-2004, INCITS 359-2012	Zorp, Credential Manager
9	Bcrypt	Based on NIST approved Blowfish algorithm	Credential Manager
10	Trusted Platform Module	TPM 2.0 Library Specification	Image Integrity Verifier

Table 5 – Security standards and techniques used in the MiCADO framework

8.2 Cryptographic and security techniques used in the MiCADO framework

- **Role-based access control (RBAC)** [14]. RBAC is a popular access control mechanism which is widely adopted by industry. It allows assigning access to resources based on the individual's need, which is represented by their roles. It was first standardized by NIST in 2000, but adopted, copyrighted and distributed by the International Committee for Information Technology Standards (INCITS) as INCITS 359-2004, and then INCITS 359-2012. In the project, we apply the role-based access control to authorize the users to deploy/ un-deploy applications in MiCADO.
- **Password hashing function (Bcrypt)** [15]**.** It's a hash algorithm, proposed by Niels Provos and David Mazieres. It is constructed based on the NIST approved Blowfish

algorithm. It enforces adding salt in hashing every password which improves the password security. Furthermore, it slows down the speed of hashing compared to other hash functions such as SHA that helps to avoid dictionary attacks. In the project, we use the hash password function of the open source Flask-User v1.0, which provides customizable user authentication and user management. Such hash password function is configured to use Bcrypt for hashing as default, but this can be configured to utilize other hash algorithms such as SHA256.

8.2 Security Evaluation Standards

WP7 evaluated the MiCADO framework's security based on the well-established industrial standards NIST Special Publication 800-53. The NIST SP 800-53 is the special publication for the Security and Privacy Controls for Federal Information Systems and Organizations published by the National Institute of Standards and Technology. Its purpose is to deliver a set of standards and guidelines to protect sensitive and personal information from cyber-attacks. To do so, it introduces a list of security control families, spanning across operational, technical, and management safeguards. In accordance to the defined security control families, we assess the relevant security features of MiCADO framework, and identify their implementation status into one of the following types:

- **Supported**: This reflects that the MiCADO framework support features that comply with the requirement.
- **Partial:** This reflects that either the MiCADO framework support features that partially comply with the requirement, or MiCADO supports features that fully comply with the requirement; however, these features are not enabled in the current release.
- **Not available**: The feature is currently not in place. However, it is considered as an addition in the future versions of MiCADO.
- **Not applicable**: The underlying feature is not in the direct control of the MiCADO framework and the application owners should take care to comply with the any necessary requirements.
- **None**: The underlying feature is not supported by MiCADO. The key reasons include:
  - The underlying requirement is mainly related to information systems in general. However, MiCADO is not considered as such a framework, or
  - MiCADO does not support relevant features as they are not required.

Results of the detailed security evaluation using the above described standard are reported in COLA Deliverable D7.6 MiCADO Security Evaluation Report.

9. Standards in the Cloud Layer (WP4)

9.1 Certified Cloud Service Providers (CSP)

The MiCADO framework aims to be cloud agnostic and vendor free, and so operates across a wide range of cloud service providers (CSP). This means a mix of standards coming from a potentially unrestricted number of supported clouds. The current public clouds with which MiCADO is compatible - CloudSigma AG, Amazon Web Services and Microsoft Azure - are all certified in the ISO 27000 family of security standards. The open-source OpenStack cloud can be deployed privately, but also supports a growing number of public clouds. The standards to which these cloud deployments conform are dependent on the respective OpenStack operator. In the MiCADO platform clouds are not only accessed directly but also available through the commercial CloudBroker Platform. CloudBroker, strives to be compliant with a number of standards produced by ISO, IEC and CSA, and where it is not fully compliant it still follows many of their best practices and regulations.

Cloud Service Providers comply with the following standards:

- **ISO/IEC 27000 standard** (Information Security Management Systems (ISMS) standards). [16] ISO/IEC 27000 provides an overview of information security management systems, and defines related terms. It also contains an ISMS family of standards intended to assist in implementation and operation of information security international standards. CSPs pay special attention to some standards which are part of ISMS family and listed in ISO/IEC 27000.
- **ISO/IEC 27001 standard** (Information Security Management standard) [17]. CSPs, in particular CloudBroker, aim to meet requirements of ISO/IEC 27001. It formally specifies an Information Security Management System (ISMS), a suite of activities concerning the management of information security risks. ISO/IEC 27001 does not formally mandate specific information security controls since the controls that are required vary markedly across the wide range of organizations adopting the standard. Following the recommendations provided in ISO/IEC 27001 CSPs tend to do its best to comply with them:
  - Information security requirements identified for the organization:
    - Information assets and their value.
    - Business needs for information processing, storage and communication.
    - Legal, regulatory and contractual requirements.
  - Information security risks are assessed
- **ISO/IEC 27002 standard (**Information technology — Security techniques — Code of practice for information security controls standard) [18]. CSPs tend to develop controls for risk reduction as recommended in ISO/IEC 27002 that contains details about information security controls addressing information security control objectives arising from risks to the confidentiality, integrity and availability of information. For example, CSPs implement the following recommendations:
  - User access management. The allocation of access rights to users are controlled from initial user registration through to removal of access rights when no longer required, including special restrictions for privileged access rights and the management of passwords plus regular reviews and updates of access rights.
  - User responsibilities. Users are made aware of their responsibilities towards maintaining effective access controls.
  - System and application access control. Information access should be restricted in accordance with the access control policy e.g. through secure log-on, password management, control over privileged utilities and restricted access to program source code.
  - Cryptographic controls. There should be a policy on the use of encryption, plus cryptographic authentication and integrity controls.

- Backup. Appropriate backups should be taken and retained in accordance with a backup policy.
- Network security management. CloudBroker keeps its Networks and network services secure.

9.2 Cloud Service Provider Interfaces (Cloud APIs)

Each of the potential clouds which MiCADO supports defines its own unique interface with which an orchestrator must communicate. The most well-known and widely adopted open-source and open-standard cloud is OpenStack [7]. By supporting the OpenStack interface in MiCADO, the COLA project ensures compatibility with any private OpenStack cloud, as well as any public cloud that is powered by OpenStack. Proprietary commercial clouds have their own unique APIs with which an external source must interact. In order to facilitate interoperability between MiCADO and each independent cloud, a unique set of parameters are defined for each CSP and expressed using the OASIS Standard TOSCA. These parameters are used inside the ADT to describe the desired cloud topology for a given application deployed by MiCADO. At orchestration time, the ADT and its parameters are translated from TOSCA into the interface that is required by the desired cloud provider API.

- **OpenStack** [19]. OpenStack is an open-source and open-standard project for managing large pools of cloud resources such as compute, networking & storage. MiCADO supports interfacing with Nova, the OpenStack component responsible for managing compute nodes. At present, MiCADO expresses the following Nova properties inside the TOSCA based Application Description Templates:
  - Project ID
  - Image ID
  - Network ID
  - Flavor Name
  - Key Name
  - Security Groups

  These properties adopt standard OpenStack nomenclature and so they will be familiar to any OpenStack user or operator. Interfacing with OpenStack Nova ensures that MiCADO will be compatible with any private or public cloud that is powered by an OpenStack which adheres to the same open-standard.

- **CloudSigma** [20]. CloudSigma is a European commercial cloud offering Infrastructure-as-a-Service. MiCADO interfaces with CloudSigma to build a cloud topology. Currently MiCADO supports the following CloudSigma properties to provision compute nodes:

  - Number CPUs
  - Memory Size
  - VNC Password
  - Library Drive ID
  - Public Key ID
  - NIC/Firewall/IP Configuration

  These parameters are common across the CloudSigma IaaS offering and so support reuse and portability for operators moving to and from MiCADO.

- **AWS** [21]. AWS offers one of the largest IaaS clouds through its Elastic Compute Service (EC2). MiCADO supports the following basic EC2 parameters which can be further extended by the user as needed:

  - Region
  - Amazon Machine Image (AMI) ID
  - Instance Type
  - Security Groups
  - Tags

The benefits of adopting a set of parameters from AWS means that MiCADO directly supports other platforms which interface with EC2 such as OpenNebula - a platform for building private clouds. MiCADO supports OpenNebula on EC2 with no additional configuration.

- **CloudBroker** [22]. CloudBroker offers a brokerage service for a number of public clouds, including AWS, Microsoft Azure and CloudSigma, and can connect to and broker private clouds such as OpenStack and OpenNebula. CloudBroker offers its own standard interface on top of all of these clouds, for which MiCADO supports the following parameters:
  - Deployment ID
  - Instance Type
  - Key Pair ID
  - Ports

  These basic parameters do not change as CloudBroker adds additional cloud providers, so MiCADO automatically supports new CloudBroker offerings as soon as they are added.

10. COLA Standardization Efforts

The standardisation efforts in COLA targeted two areas:

- application description and
- security.

The COLA Project embraced TOSCA for development of the primary user interface of MiCADO elaborating the concept of the Application Description Template (ADT). Playing such a prominent role in MiCADO, TOSCA has often earned mention at, and has several times been the subject of talks where MiCADO was discussed and/or presented. MiCADO ADTs and TOSCA were the main subject of the following talks:

- *Towards Cloud Application Description Templates Supporting Quality of Service* IWSG 2017: 9th International Workshop on Science Gateways, Poznań, Poland
- *Flexible Deployment of Social Media Analysis Tools,* IWSG 2018: 10th International Workshop on Science Gateways, Edinburgh, Scotland
- *Enabling Modular Design of an Application-Level Auto-Scaling and Orchestration Framework using TOSCA-based Application Description Templates,* IWSG 2019: 11th International Workshop on Science Gateways, Ljubljana, Slovenia

ADTs and TOSCA also earned at least a brief overview at the following events:

- UKRI Cloud Workshop February 2019, Francis Crick Institute. London, UK
- EOSC-hub Week 2019, Prague, Czech Republic
- EGI Conference 2019, Amsterdam, The Netherlands,
- deRSE19: Conference for German Research Software Engineers, Potsdam, Germany

TOSCA in MiCADO ADTs was also covered as a topic of a webinar, which is available on the project website (https://project-cola.eu/media/videos) and more recently, was covered in a series of tutorial videos which will also feature on the website in the near future.

Our understanding and use of different TOSCA structures and language elements continues to grow as the MiCADO framework approaches maturity. The approach to TOSCA in MiCADO ADTs has evolved over the course of the project, and is now at a stable point where working examples of TOSCA in ADTs can be beneficial to the OASIS TOSCA Working Group. With this aim in mind, we are currently registering as a member of OASIS so we may join the TOSCA Technical Committee and participate in the growth and development of TOSCA into the future. COLA partners have already participated in regular weekly TOSCA Simple Profile in YAML working group calls as observers and the University of Westminster looks forward to participating as active members once our OASIS registration is complete.

In the scope of the development of the security enablers, WP7 has contributed to the standardization process within the IETF Trusted Execution Environment Provisioning (TEEP) working group (https://datatracker.ietf.org/wg/teep/about/). In particular, WP7 contributed through several comments on the TEEP architecture draft.

11. Conclusions and Future Work

This report has provided an overview on how COLA used standards and contributed to standardisation when designing and implementing the MiCADO framework. The deliverable reviewed the relevant standards and related standardisation activities.

Overall, the main goals that have been achieved by COLA using standards and standardization activities can be summarized as follows:

The MiCADO framework

- offers the Application Description Template as a TOSCA Normative Compliant language to describe the application and the policies that define their lifecycle;
- provides a REST-Compliant interface for the submission, execution and deletion of applications to the Cloud,
- adopts all relevant standards that norm security aspects and connected the different components of the security framework through a standardized REST interface to foster flexibility and adaptability to changes;
- offers a generic interface to users whereby they can find TOSCA node types that cover all the most used cloud technologies;
- adopts the OpenStack specifications for the submitter and orchestrator components.

Although COLA representatives have already participated in the work of the TOSCA Working Group as observers, effort is still ongoing in formalizing the cooperation between COLA and OASIS. The University of Westminster is currently in the process of joining OASIS as a Contributor Member facilitating future long term collaboration and contribution,.

12. References

[1] TOSCA Simple Profile in YAML Version 1.0. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csd03/TOSCA-Simple-Profile-YAML-v1.0-csd03.html.

[2] M Mernik, J Heering and A. M. Sloane: When and How to Develop Domain-Specific Languages, ACM Computing Surveys 37(4):316—344,·December 2005

[3] http://yaml.org

[4] https://tools.ietf.org/html/rfc8259

[5] https://tools.ietf.org/html/rfc6585

[6] https://datatracker.ietf.org/doc/rfc3986/

[7] https://tools.ietf.org/html/rfc5246

[8] https://tools.ietf.org/html/rfc5958

[9] https://tools.ietf.org/html/rfc2409

[10] https://tools.ietf.org/html/rfc6071

[11] https://tools.ietf.org/html/rfc2510

[12] https://tools.ietf.org/html/rfc3174

[13] https://www.iso.org/standard/66510.html

[14] https://tools.ietf.org/html/draft-cridlig-netconf-rbac-00

[15] K. Malvoni, S. Designer and J. Knezovic: Are Your Passwords Safe: Energy-Efficient Bcrypt Cracking with Low-Cost Parallel Hardware, WOOT'14 8th USENIX Workshop on Offensive Technologies, San Diego, CA, 2014, DOI: 10.13140/2.1.3267.0081

[16] https://www.iso.org/news/ref2266.html

[17] https://www.iso.org/isoiec-27001-information-security.html

[18] https://www.iso.org/standard/54533.html

[19] OpenStack - Built the Future of Open Infrastructure, [Online], available www.openstack.org

[20] Cloud Hosting Pricing | CloudSigma, [Online]. Available: https://www.cloudsigma.com/pricing

[21] Amazon EC2, Amazon, [Online], available https://aws.amazon.com/ec2

[22] CloudBroker GmbH | Compute-intensive applications in the cloud [Online]. Available: http://cloudbroker.com/.