Complete Dynamic Multi-cloud Application Management

Project no. 644925

Innovation Action

Co-funded by the Horizon 2020 Framework Programme of the European Union

Call identifier: H2020-ICT-2014-1

Topic: ICT-07-2014 – Advanced Cloud Infrastructures and Services

Start date of project: January 1st, 2015 (36 months duration)

Deliverable D4.4

Consolidated CYCLONE secure action and resource models

Due date:	31/12/2016
Submission date:	30/12/2016
Deliverable leader:	TUB
Editors list:	Mathias Slawik (TUB), Ilke Zilci (TUB)

Dissemination Level

☒	PU:	Public
☐	PP:	Restricted to other programme participants (including the Commission Services)
☐	RE:	Restricted to a group specified by the consortium (including the Commission Services)
☐	CO:	Confidential, only for members of the consortium (including the Commission Services)

List of Contributors

Participant	Short Name	Contributor
Interoute S.P.A.	IRT	
Sixsq SARL	SIXSQ	
QSC AG	QSC	
Technische Universitaet Berlin	TUB	Ilke Zilci, Mathias Slawik, Dirk Thatmann
Fundacio Privada I2CAT, Internet I Innovacio Digital A Catalunya	I2CAT	Jose Aznar Baranda, Isart Canyameres
Universiteit Van Amsterdam	UVA	Fatih Turkmen, Yuri Demchenko
Centre National De La Recherche Scientifique	CNRS	Christophe Blanchet, Brian Brancotte

Change history

Version	Date	Partners	Description/Comments
1.0	October 15, 2016	TUB	Initial structure / proposal
2.1	November 02, 2016	TUB	Added initial descriptions for sections 2.1, 2.2, 2.4
3.2	November 09, 2016	TUB	Extended sections 2.2, 2.4
3.9	December 05, 2016	TUB	Proofreading, wording, Revised section 2.1
4.3	December 18, 2016	UvA	Filled section distributed authorization/ trusted bootstrapping
4.4	December 21, 2016	TUB, i2CAT	Version ready for review released.
4.5	December 23, 2016	TUB	Incorporated comments and feedback (internal review), revised introduction
4.6	December 29, 2016	CNRS	Update on section 2.3.2 – Testing Tool
Final	December 30, 2016	I2CAT, IRT	Final version ready to be submitted to the EC

Table of Contents

Figures Summary

Executive Summary

This document provides technical details of the new security components and the added features they provide to the existing security architecture explicated in D4.1 and D4.2.

1. Introduction

This deliverable is an update to D4.3 and complementary to D4.2. It explains CYCLONE developments and their technical details implemented in Y2. The focus of the work is on making the CYCLONE components more inline with the use cases of Y1 and Y2. To achieve this, we extended some of the components and added new compatible ones. As we follow an agile approach to developing our components, the new requirements are based on the feedback of the use case partners on the existing components. The building blocks of CYCLONE components are open source libraries which receive frequent updates. We put the necessary effort to keep our deployments up-to-date.

The most mature outcomes of Y2 are the implementation of the CYCLONE PAM module to enable using Secure Shell Login using federated eduGAIN identities and the improvements of the CYCLONE Federation Provider to improve privacy of eduGAIN users. These components are integrated in the use cases and have finished a first round of feedback from the use case owners. Furthermore, we report substantial conceptual work on possibly more complex authorization scenarios. OpenNaaS CNSMO components which aim to integrate with CYCLONE logging system and monitoring mechanisms of SlipStream are under implementation.

The document is structured as follows. Section 2.1 gives details of the workflow with the PAM module. Section 2.2 walks through the implementations which were part of the CYCLONE Federation Provider privacy enhancements. Section 2.3 provides the concept for more complex authorization scenarios. Section 2.4 explains the current state of CNSMO components.

2. CYCLONE Action and Resource Model Extensions

In Y2, CYCLONE developments focused on improving the existing components which are used by the use case partners and additional features to address Y2 requirements. This section provides details to the federated SSH login, highlights extensions to CYCLONE Federation Provider and reflects a multi-tenant distributed authorization on top of XACML. OpenNaaS CNSMO.

2.1. Secure, federated SSH login (PAM Module)

2.1.1.1. SSH Login with Federated Identities Workflow

To implement the concept described in D4.2, CYCLONE relies on a PAM module written in Python that interacts with the Linux PAM API on the VM. The login process is as follows:

1. An eduGAIN user connects via SSH login to the instance
2. The SSH server invokes PAM that in turn invokes the CYCLONE PAM module
3. The PAM module starts an HTTP server that is configured to let users login to the Federation Provider via OpenID Connect
4. The PAM module invoked by the SSH server shows the link to its web server through SSH keyboard-interactive mode
5. The eduGAIN user opens the link in the browser
6. The OIDC client integrated in the webserver runs the OpenID Connect Authorization Code Flow to log in eduGAIN users with their federated identity
7. After completing authentication, the PAM module checks if the user's email is in the list of allowed user IDs for the account. If so, the PAM module sends a "success AuthN" message to the PAM subsystem on the VM, logging in the user.

2.1.1.2. User Management

The user management is as simple as possible: when an eduGAIN user issues a login request, the CYCLONE-PAM module searches for a `~/edugain` file to check if the user's ID (their email address) was designated to be allowed to login as that specific system user.

The workflow is as follows:

1. EduGAIN users issue an SSH log in request as, for example, alice@example.com
2. The PAM module carries out the process described in the last section
3. After successful authentication, the CYCLONE-PAM module fetches the users' attributes from the JSON Web Token
4. The CYCLONE-PAM module opens `~/.edugain` and checks if the provided JWT `email` attribute matches any entry in the file

5. If the file doesn't exist or the attribute does not match, authentication is rejected

The structure of the `.edugain` file is as follows:

```
{
  "users": [
    "mail1@example.com",
    "mail2@example.com"
  ]
}
```

It is a JSON file and therefore easy to interpret, text based, and easy to edit.

2.2. Extensions to CYCLONE Federation Provider

There were three main extensions to the CYCLONE Federation Provider in Y2 that are explicated in the following. Besides these, we constantly spent effort on keeping the CYCLONE Federation Provider up-to-date, updating its underlying libraries. At last, we also spent effort on “parameterizing” the Federation Provider, modifying the Docker components as well as their configurations to accept more parameters from the environment, e.g., via Nuv.la deployment parameters.

2.2.1. Generating an OpenID Subject ID from eduGAIN attributes

CYCLONE SAML Bridge now enhances JWT Subject IDs by using mapping mechanisms of SimpleSAMLPHP.

As described in D4.3, eduGAIN recommends member identity providers to include certain SAML attributes in the SAML tokens to service providers however does not enforce it. This results in a heterogeneous set of attributes received by service providers, they are named differently or are in different formats. In CYCLONE SAML Bridge, we unify these attributes by mapping them always to the same structure. We also select the most reliable ID which is unique and not changeable among the provided attributes. More details can be seen in the related GitHub commits [1].

2.2.2. Data-privacy aware session data removal

The European Data Protection Directive specifies that computer systems processing personal data should not keep them longer than needed. The CYCLONE Federation Provider processes SAML assertions originating from the eduGAIN Identity Providers. Therefore, these assertions should be removed when they are not needed anymore. We implemented a removal solution that runs periodically and deletes this data from the database [2]. In a later version of the Federation Provider, this feature will be provided out-of-the-box by the underlying Keycloak Identity Server – at least according to the plans of the Keycloak developers.

2.2.3. Consent screen

Many Service Providers will be registered with the CYCLONE Federation Provider, each having different people responsible for their operation as well as different data protection officers. By default, Keycloak presents users a consent screen before their attributes are relayed to the services. We extended this consent screen with comprehensive information about the Service Providers so that people can make a more informed decision, especially regarding the Terms of Use, data protection rules, and the people responsible for the service that users log in to.

2.2.4. Self-service registration

When a new application is deployed that should use the CYCLONE Federation Provider for authentication, it needs to have a new OpenID Connect Client created. Currently, this is a manual process, requiring the Federation Provider Operator (TUB) to create a new client in the back end. We extended the Keycloak server with a self-service registration API that allows application DevOps to create new OpenID Connect clients by themselves.

While Keycloak already had internal Java classes to create clients provided by a REST API, this API did not feature user access control and was not multi-tenant capable. Our extension adds this multi-tenant capability for the client registration API, binding created clients to eduGAIN users. While this works for the rest interface, as a next step we'll implement a Keycloak user UI to manage those clients.

Details about this component can be found on our GitHub repositories [3].

2.3. Components for distributed authorization

CYCLONE currently supports the use of various authorization mechanisms ("*Federated Authorization Mechanisms*" use case in deliverable D4.1 [7]), these mechanisms mainly aim to manage access to VMs. From security point of view, there are several limitations of this approach:

- The authorization mechanisms do not provide sufficient level of granularity to specify policies. For instance, an access control rule cannot refer to a specific sensitive resource (application, networking services or a data item in the object store) inside/outside the VM.
- Analysing the existing permissions (and possibly revoking them) in these mechanisms is very difficult when the number of users is large. This is due to the fact that policies are tightly coupled to VMs. A security administrator should be given the means to control and manage authorizations with the relative ease.
- The security mechanisms lack of context when evaluating authorization requests. The context varies from trust/delegation relations between users to environmental conditions such as time of day.

In fact, most of the authorization mechanisms available in CYCLONE fall into a form of authentication according to the authorization definition provided in REST profile of XACML [6]. In order to provide an extended support for fine-grained access control policies, we employ a multi-tenant attribute-based access control mechanism built on top of XACML.

In the model, there are the following actors that can specify their security constraints in the form of XACML policies:

1. Provider: The original owner of the physical resources that are leased to a tenant such as LAL cloud in CYCLONE cloud. A provider may specify **delegation policies** that identify which tenants have (contractual/temporal) what permissions over its resources.
2. Tenant/s: Tenant is often considered as the temporary owner of a resource leased to her and may specify a **user access control policy**. It is also possible that a tenant shares its resources (with the users of other tenants) through **sharing policies** that enables trusted tenants (trustee) to use the resources leased to them. Sharing policies are relevant in particular to CYCLONE use cases (e.g. UC13 in use cases portal) where researchers share their experimental settings with other researchers. The use of XACML based authorizations enables more granular authorizations (e.g. allowing only one application execution rather than all in a particular VM instance) in these use cases.

3. Users (of a tenant): Tenants or trustees have users in their domains of specialization (e.g. bio-engineer in a bioinformatics lab).

The extended CYCLONE authorization model considers the deployment of a dedicated authorization service for modelling/managing the distributed authorizations (details are available in deliverable D4.2). Figure 1 and Figure 2 shows the development of the access management system from the time of application/service development/deployment (by the application developer) to execution of the application (by the user). First, the developer prepares the application deployment scripts (e.g. Slipstream recipes) together with the security configurations and the associated application components (i.e. C1, C2, C3 in Figure 1). Then the application regulates access to sensitive resources (e.g. a data item in the object store) through the components contained in its pipeline. More specifically, it builds an authorization service that lies on top of the existing infrastructure services (possibly containing basic security services) by using the attributes obtained from an identity provider. Note here that the security configurations refer to a spectrum of parameters from authorization policies to attributes used for permission management. For the latter, CYCLONE use case owners can employ a set of attributes that are obtained from EDUGAIN IdPs. For instance the following attributes: name, surname, organization type, affiliation, e-mail, username can be used to specify authorization policies. This way, the developer can put in place (on top of the available services from the provider) an authorization service to manage user permissions in a more granular way.

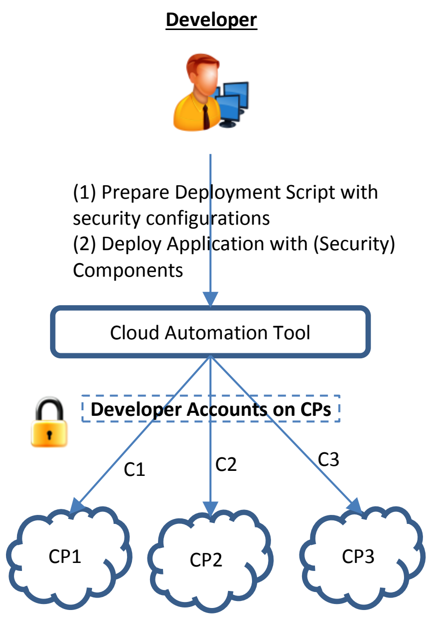
Developer
(1) Prepare Deployment Script with security configurations
(2) Deploy Application with (Security) Components
Cloud Automation Tool
Developer Accounts on CPs
C1
C2
C3
CP1
CP2
CP3

Figure 1 Modeling Authorization System

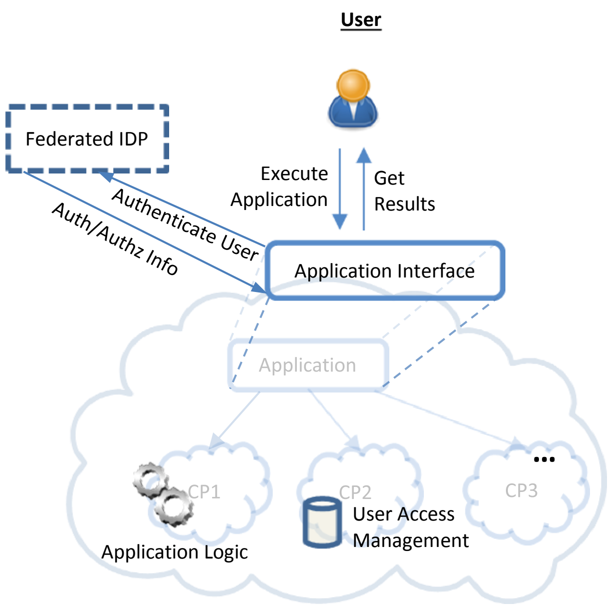

Figure 2 Managing Authorizations for Users

The described authorization service decouples the management of permissions from the applications and the objects so that access control policies become easier to manage when assigning and revoking permissions.

2.3.1. Trust Bootstrapping

There are several approaches for bootstrapping the cloud nodes with relevant security keys [4,5,6]. Among the available proposals CYCLONE employs keylime [4] due to its high security guarantees, scalability and suitability to cloud settings considered in CYCLONE. The requirements of keylime and how it is employed inn CYCLONE is discussed deliverable D4.2 [8].

2.3.2. Federation Provider Testing Tool

In order to validate the use of the federation provider to authenticate user with the federation provider, we propose a first tool [9] allowing to start an apache web server in a docker container, and let user getting authenticated through the federation provider. A second tool [10] shows the integration of the authentication with the federation provider within an inter-operable Web Application. For this purpose we provide a Django project (Django is a high-level Python Web framework), which is hosted by apache, itself being hosted in a docker container. This provides a web application using the federation provider easily deployable on any system supporting Docker. These tools were developed as prototypes for both the CYCLONE use case portal, and the bioinformatics cloud web portal. These developments, including the PAM module to log in with SSH and the federation identity, are especially important for the life science community where people are not familiar with the use of electronic certificates nor SSH public keys.

2.4. OpenNaaS CNSMO

Besides the previously mentioned services that CYCLONE provides through the integration with SlipStream, the OpenNaaS CNSMO component also integrates with the security architecture by providing logs to the logging system. The goal is to collect logs of all CYCLONE components and allowing DevOps to have a far easier access to the logging information. In terms of OpenNaaS CNSMO, we aim to keep track of the deployment and configuration of the services and enable the possibility to react in case something went wrong. At this point, this tasks are under implementation. Further details and outcomes will be provided in D4.5 at the end of Year 3.

3. Conclusions

This deliverable describes our recent components. With the PAM-based federated SSH login we support a Shell-based access to remote VMs by using a Web-based authentication. A simple user management provides a convenient way to use it at runtime. With our CYCLONE SAML Bridge we unify certain attributes provided in a non-standardized notation by different IdPs. By this we enable a seamless further processing of SAML assertions. Furthermore, an assertion removal extension for the CYCLONE Federation Provider (Keycloak) has been invented. This increases compliance to the upcoming European Data Protection Directive. Another extension is a self-service registration API, which allows application DevOps to create new OpenID Connect clients by themselves. The Distributed Authorization components complement already existing authorization flows, and by using Keylime we can increase the level of security, since the use of a TPM in a VM becomes accessible. Further plans addressed by WP4 are given in D4.2. All further developments, integrations and efforts Y3 efforts will be reported in the upcoming D4.5.

References

[1] https://github.com/cyclone-project/cyclone-federation-provider/blob/bb94f6cc639ea6754fe6eee71fc1b00351850ccf/components/samlbridge/config/config/config.php

[2] https://github.com/cyclone-project/cyclone-federation-provider/tree/master/components/cache-clean

[3] https://github.com/cyclone-project/cyclone-client-registration/

[4] Nabil Schear, Patrick T. Cable II, Thomas M. Moyer, Bryan Richard, Robert Rudd, "Bootstrapping and Maintaining Trust in the Cloud", Annual Computer Security Applications Conference (ACSAC), 2016

[5] Peter Membrey, Keith C. C. Chan, Canh Ngo, Yuri Demchenko, Cees de Laat, Trusted Virtual Infrastructure Bootstrapping for On Demand Services, Seventh International Conference on Availability, Reliability and Security (ARES), 2012

[6] Bryan Parno , Jonathan M. McCune, Adrian Perrig, "Bootstrapping Trust in Modern Computers", IEEE Symposium on Security and Privacy 2010.

[7] CYCLONE deliverable D4.1, Security infrastructure specification and initial implementation, http://www.cyclone-project.eu/documents.html#deliverables, Dec 2015

[8] CYCLONE deliverable D4.2, Multi-cloud security, http://www.cyclone-project.eu/documents.html#deliverables, Dec 2016

[9] https://github.com/cyclone-project/cyclone-federation-provider-apache-oidc-demo

[10] https://github.com/cyclone-project/cyclone-federation-provider-apache-oidc-django-demo

Abbreviations

B2B	Business to Business
CNSMO	CYCLONE Network Service Manager/Orchestrator
DC	Data Center
E2E	End to End
IaaS	Infrastructure-as-a-Service
IPR	Intellectual Property Rights
IT	Information Technology
MaaS	Metal as a Service
NaaS	Network-as-a-Service
Net-HAL	Network Hardware Abstraction Layer
NFV	Network Function Virtualization
PaaS	Platform-as-a-Service
PC	Project Coordinator
PAM	Pluggable Authentication Modules
PMB	Project Management Board
PoP	Point of Presence
REST	REpresentational State Transfer
SaaS	Software-as-a-Service
SAML	Security Assertion Markup Language
SCI	Smart Core Interworks
SDN	Software Defined Networks
SP	Service Provider
TC	Technical Coordinator
TCTP	Trusted Cloud Transfer Protocol
TMB	Technical Management Board
TPM	Trusted Platform Module
VM	Virtual Machine
WP	Work Package
WPL	Work Package Leader

<END OF DOCUMENT>