Policy and Context Management in Dynamically Provisioned Access Control Service for Virtualised Cloud Infrastructures

Canh Ngo[1], Peter Membrey[2], Yuri Demchenko[1], Cees de Laat[1]
University of Amsterdam[1], Hong Kong Polytechnic University[2]
email: {t.c.ngo, y.demchenko, delaat}@uva.nl[1], peter@membrey.hk[2]

***Abstract*— Cloud computing is developing as a new wave of ICT technologies, offering a common approach to on-demand provisioning of computation, storage and network resources which are generally referred to as infrastructure services. Most of currently available commercial Cloud services are built and organized reflecting simple relations of a single provider to multiple customers with simple security and trust model. New architectural models should allow multi-provider heterogeneous services environment that can be delivered to organizational customers representing multiple user groups. These models should be supported by new security approaches for multi-provider, multi-tenant crossing security domains to create consistent and dynamically configurable security services for virtualised infrastructures. This paper proposes an on-demand provisioned access control infrastructure with dynamic trust establishment for entities in a Cloud IaaS architecture model. It applies XACML-based RBAC model for the flexible authorization policy configuration and management. It uses authorization ticket as a security session management mechanism to solve the security context synchronization and exchange between multiple Cloud providers. The paper describes practical implementation of the proposed Dynamic Access Control Infrastructure as the part of a complex infrastructure services provisioning system.**

Keywords*— *Dynamic Access Control Infrastructure, RBAC, XACML, Policy Generation, Dynamic Trust Establishment, Security Context Management.

I. INTRODUCTION

Cloud computing is emerging as a common approach and a service model for provisioning infrastructure services on-demand that include both computation, storage and advanced network infrastructure. Beside a wide spectrum of currently available Cloud services, there is a number of research and standardization activities focusing on definitions, use-cases, and reference models such as NIST Cloud Architecture collaboration group [1], [2]. OGF Infrastructure Services on Demand Research Group (ISOD-RG) [3], and OASIS Cloud Identity Technical Committee (IDCloud TC) [4].

The current widely accepted Cloud computing definition is based on the NIST definition that identifies five essential Cloud characteristics [2]: (1) on-demand self-service; (2) broad network access and diversity of client devices; (3) resource pooling that allows providers to serve multi-tenant customers by managing resource utilization more efficiently using virtualization, resource partitioning and workload balancing; (4) rapid elasticity that allows scaling resources dynamically; (5) measured service with the pay-per-use business model. Other additional feature is the heterogeneity on both provider and customer sides, and multi-provider services.

Current trend in moving services to cloud facilitates changing approaches to applications, services and utilities provisioning and management. It motivates development of new services provisioning models, and consequently refactoring and re-thinking existing security models to allow consistent security in a dynamic virtualised environment.

Infrastructure Services Provisioning On-Demand (ISOD) is an important part of all cloud service models which can be delivered directly to end-users in the cloud IaaS model or as the underlying infrastructure in supporting for PaaS and SaaS cloud models. In this sense, security is an important component of both approaches, delivering services to end-users or securing underlying infrastructure.

The paper presents the on-going research on developing a security framework for Cloud IaaS architecture. It aims to deliver a security infrastructure to support consistent trust establishment, identity management, access control, as well as security context management. It proposes a solution for provisioning authorisation services with dynamic trust establishment for entities in the Cloud IaaS architecture and adopts the Security Service Lifecycle Management model described in [5]. It supports security context sharing across distributed security domains among multiple providers by using authorization tickets as a security/authorisation session management mechanism. It also uses the proposed XACML policy profile to allow dynamic policy generation based on the templates that support Role-Based Access Control (RBAC) model to manage authorization services for complex hierarchical user groups and infrastructures. The paper provides update on the current implementation results based on the GAAA-TK toolkit library [6]. The presented research is conducted as a part of the two EU projects GEYSERS [7] and GEANT3 [8] which provide a prototype implementation and testbed.

The structure of paper is organized as follows: section II revisits a Cloud IaaS model which motivates the

development of the Dynamic Access Control Infrastructure (DACI). Section III presents the policy management in the DACI, including policy generation model for RBAC and its approach applying for XACML standard. After that, section IV discusses on trust management and establishment for the infrastructure. Section V describes on-going implementation details of the DACI. Section VI provides an overview of some related works motivating the presented research. Finally, section VII contains the summary and suggestions for further research directions.

II. Access Control for On-demand Provisioned Virtualised Infrastructures Services

A. Virtualised Infrastructure Services Provisioning Model

On-demand Infrastructure Services Provisioning model [9] is illustrated in Fig. 1. It defines the multilayer infrastructure provisioning model that includes Physical Infrastructure Providers (PI Provider), Virtual Infrastructure Providers (VI Provider). Tenants, defined as Virtual Infrastructure Operators (VIO), can subscribe IT-resources (e.g.: storage, computing), network resources (e.g.: network links) to form a completed Virtual Infrastructure (VI) distributed crossing multiple physical providers.

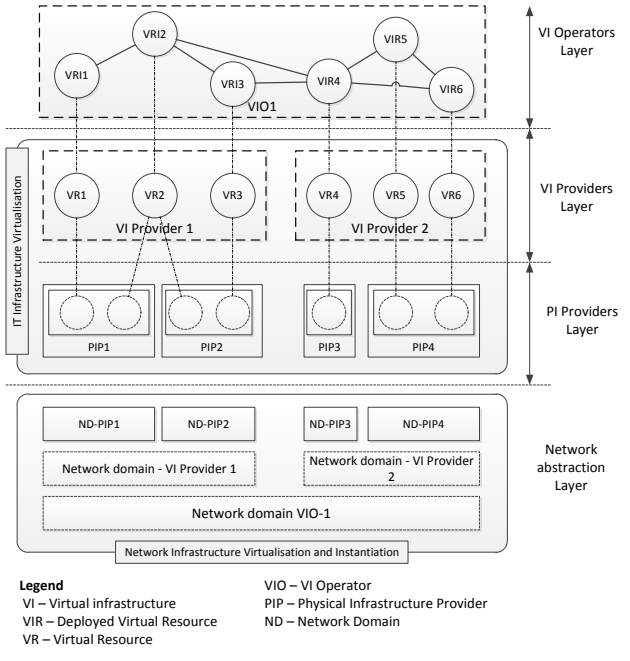

Figure 1. Virtualised Infrastructure Service Provisioning Model

VI Provider runs the Virtual Infrastructure Composition and Management (VICM) layer that consists of three layers components. It includes the Logical Abstraction Layer and the VI/VR Adaptation Layer facing correspondingly lower PI Providers and upper Application layer of VI Operators [9]. The proposed architecture motivates for the development of an access control infrastructure to support multi-tenant, multi-provider, dynamically configurable services that be provisioned and operate across different security and network domains. The Dynamic Access Control Infrastructure (DACI), including core architecture and some scenarios, is described in the following section.

B. Access Control Architecture

The involvement of multi-providers, multi-tenants across multi-domains resources of the Virtualised Infrastructure Service Provisioning model requires an access control infrastructure that should handle the dynamic relationships as well as security services with configurable parameters during the infrastructure provisioning lifecycle. The DACI is proposed to solve the mentioned above issues and challenges of the access control for Virtualised Infrastructure Services Provisioning.

Basic DACI architecture (Fig. 2) is initially introduced in [10] and has been improved based on current implementation in the GEYSERS project [7]. It includes three interfaces to integrate with the VICM layer of the VI Provider. The internal Authentication and Authorization Interface is used to control internal authentication and authorization operations during virtual infrastructure composition and management at the VI Provider. During provisioning phases of a VI, VICM initiates the instantiation process for the access control instance (DACS) of this VI through the DACS Management interface. The DACS plays as the access control as a service for the VI, which its administrations on authentication & authorization are delegated to the VIO, owner of the VI. The APIs for policy management are defined in section III.

The VI Authorization interface is the DACS public interface to receive authorization requests from end-users to access a deployed virtual resource (VIR) of the VI in which they belong to. Based on the isolation mechanisms using VI Global Reservation Identifier, final decisions are evaluated and combined from the DACS Authorization Service instance of VI and the internal AAI authorization service of VICM.

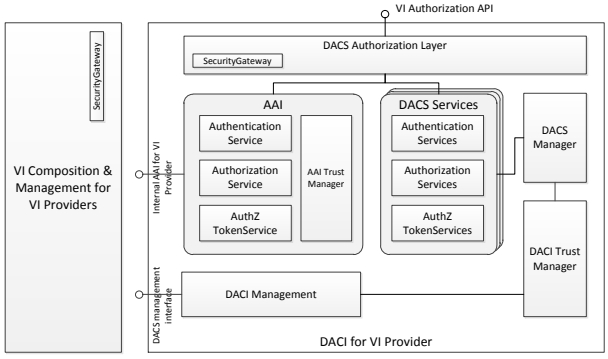

Figure 2. DACI Architecture incorporated with the Virtual Infrastructure Provider architecture

This interface also may issue an authorization ticket that will include the request information and authorization decisions as the security credential proof for end-users to access virtual resources directly at PI Providers in subsequent VI operational stage of a single virtual infrastructure as in Fig. 3, or collaboration between multiple virtual infrastructures as in Fig. 4. Here we assume that the trust relationships between the VI Provider and its PI Providers, as well peer-to-peer trust of VI Providers are established based on the dynamic trust model in [10].

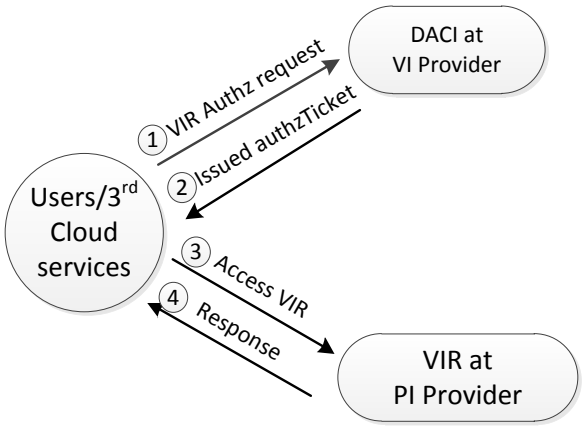

Figure 3. Access virtual resources scenarios from Users/3rd Cloud services.

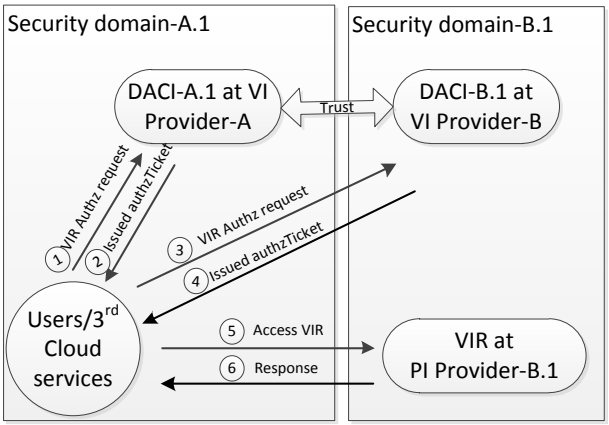

Figure 4. Access virtual resources scenarios from Users/3rd Cloud services in different virtual infrastructures

III. POLICY MANAGEMENT IN DACI

A. Policy generation model

The hierarchy model of RBAC [11] defines $\{U, R, P, PA, UA, RH\}$ in which U is set of users, R is set of roles, P is set of permissions, PA is the permission assignment to role, UA is the role assignment to user and RH is role hierarchy:

- $PA \subseteq P \times R$ is the permission to role assignment. $PA(p,r) = true$ if the permission $p \in P$ is assigned to role $r \in R$.
- $UA \subseteq U \times R$ is the user to role assignment. $UA(u,r) = true$ if the role $r \in R$ is assigned to user $u \in U$.
- $RH \subseteq R \times R$ is a partial order on R called the role hierarchy, also written as $r_1 \geq r$ is that the role r_1 inherits all permissions from the role *r*.

We have set of resource identifiers, called set of objects *O*. Equivalent to each object $o \in O$, there is a set of actions *A*. We define a permission $p \in P$ as:

$$P \subseteq O \times A$$

Based on the administrative permissions of RBAC model, policy composition and management have following operations:

- Create a permission p from object o and action a, $addPermission(o,a)$:
$$P' = P \cup \{p\} | p \notin P; p = (o,a); o \in O; a \in A;$$
- Assign a permission to a role, $assignPermission(p,r)$:
$$PA' = PA \cup \{(p,r)\} | (p,r) \notin PA; p \in P; r \in R$$
- Assign a user to a role, $assignRole(r,u)$:
$$UA' = UA \cup \{(r,u)\} | r \in R; u \in U; (r,u) \notin UA$$
- Create a role that inherits from an existing role $inheritRole$(r):
$$R' = R \cup \{r_1\} | r_1 \notin R; r_1 \geq r; r \in R$$

Equivalent to *create* or *assign* operations, there are also *remove* or *unassign* operations:

- Remove a permission from permission set $removePermission(p)$
$$P' = P \setminus \{p\} | p \in P$$
- Unassign a permission from a role, $unassignPermission(p,r)$:
$$PA' = PA \setminus \{(p,r)\} | (p,r) \in PA; p \in P; r \in R$$
- Unassign a user from a role, $unassignRole(r,u)$:
$$UA' = UA \setminus \{(r,u)\} | (r,u) \in UA; r \in R; u \in U$$
- Remove a role, $removeRole(r)$:
$$R' = R \setminus \{r\} | r \in R$$

Each operation can be implemented by a generated XACML policy following RBAC profile of XACML in [12]. We will illustrate policy generation in the implementation section with example.

B. XACML policy generation

The DACS uses XACML policy language [13] for authorization request evaluation. To adapt typical hierarchical organizations' structures, we select the

XACML RBAC profile [12] as described in the implementation section. With parameters for RBAC configuration such as resource identifiers of VIRs in the instantiated VI along with equivalent actions, permissions assignment to roles and roles assignment to end-users, we use XACML policy templates to generate XACML polices for the DACS authorization service. In this manner, VIOs are provided with tools to administer both identity management and authorization policies composition.

IV. DACS TRUST MANAGEMENT

The trust-path from the end-users to the PI Providers is built up based on X.509 certificates exchange between entities in the path. Upon establishing trust links during initial deployment stage, each entity has a trusted certificates list of neighbour partners' in the trust-path. As described in Fig. 4, because of the VI Provider's certificates is stored in the PIP's trust list, the PIP can verify requests from users which contain the authorization ticket issued and signed by this VIP. The verification is based on SAML assertion as the subject confirmation credential, along with the authorization ticket in previous section.

The DACI relies on a number of key trust anchors which can be secured further by anchoring them to a trusted machine in a known state [10]. As the Direct Anonymous Attestation (DAA) component of the TPM specification is not currently well supported an alternative is required. A machine with a Trusted Platform Module (TPM) can boot into a known and trusted state. The TPM can create a non-migratable key-pair that is sealed and only accessible in this known state. If any part of the boot process is changed or tampered with, it will not be possible to access the private key. As the key is non-migratable, it is not possible to backup or extract the private key from the TPM. Only the specific machine in a specified and trusted state will be able to decrypt or sign messages using the key-pair.

A Vanguard application is a program sent to a remote machine to verify whether it is in a trustworthy state [10]. For machines with an active TPM, Vanguard and any subsequent files can be encrypted using the machine's non-migratable key-pair. This ensures that only the destination machine can decrypt the application. The payload contains a shared secret that Vanguard can use to authenticate itself to the machine that sent it. Vanguard can verify the status of the TPM and can in addition verify the setup and configuration details of the machine. It can then exchange the initial infrastructure files needed to bootstrap the infrastructure. Once the infrastructure has been transferred and verified, Vanguard executes and begins the bootstrapping process. At this stage the infrastructure being deployed takes control of the process.

V. IMPLEMENTATION DETAIL

We implement DACI as Java OSGi bundles corresponding to components shown in Fig. 2, based on the GAAA-TK library using Sun's XACML and OpenSAML libraries [14], [15].

These bundles can be deployed in OSGi platforms such as Apache Felix or ServiceMix [16]. To make compatibility with other components in Geysers project [7], we currently supports two packages types: OSGi bundles in Apache ServiceMix and web services in Apache Tomcat, in which OSGi bundles are wrapped by SOAP web service layers.

A. SecurityGateway

The SecurityGateway provides the unified interface to integrate authentication, authorization functionalities with authorization token supports, along with attributes collecting and validation prior authorization evaluation.

To unify authentication, authorization and attribute validation, we define the *SecurityContext* container as the placeholder for authentication credentials and authorization requests' attributes, along with related security context data such as trusted entity identifiers.

SecurityContext = {AuthenticationData, AuthorizationData, SessionData, SecurityData}

Credential types are then forwarded to corresponding authentication adapters during the validation process. Current implementation supports basic username/password credential with token-based validation using SAML assertion. Other schemes such as LDAP, Kerberos, and HTTP authentication will be supported in the future.

B. XACML policy composition

The Authorization service follows the XACML standard recommendation for Policy Decision Point implementation [13]. It implements the Role-Based Access Control model (ANSI-RBAC [17]) using XACML RBAC policy profile [12] running on SunXACML library [14]. We define roles and permissions as loose coupling XACML Policy and PolicySet entities so that it is easy to add or remove permissions to multiple roles.

In the following examples, Fig. 5 defines a role policy that matches if the subject in the authorization request has the role attribute 'VIO'. The PolicySetIdReference indicates that PDP must look up a PolicySet with reference 'PPS:VIO:role' for the evaluation process. The Fig. 6 specifies which permissions are assigned to the role VIO by PolicyIdReference values. The PDP then keep referencing those policies and one of them is the policy in Fig. 7, to allow the request for a new virtual infrastructure. With this approach, it is easy to implement policy management functions described in section III: to create permission, we use the policy template in Fig. 7 with two parameters that are action-id and resource-id; to assign a permission to a role, we use the template in Fig. 6 by inserting appropriate created permission policy identifier in the PolicyIdReference element. We can create a role by using policy template in Fig. 5 and can assign a user to a role by updating user's role attributes in the authentication service.

```
01 <?xml version="1.0"?>
02 <PolicySet PolicySetId="RPS:VIO:role"
03   PolicyCombiningAlgId="permit-overrides">
04   <Target>
05     <Subjects>
06       <Subject>
07         <SubjectMatch MatchId="string-equal">
08           <AttributeValue DataType="string">VIO
09              </AttributeValue>
10           <SubjectAttributeDesignator DataType="string"
11              AttributeId="http://authz-
interop.org/AAA/xacml/subject/subject-role"/>
12         </SubjectMatch>
13       </Subject>
14     </Subjects>
15   </Target>
16   <PolicySetIdReference>PPS:VIO:role
17   </PolicySetIdReference>
18 </PolicySet>
```

Figure 5. XACML PolicySet template for role VIO

```
01 <?xml version="1.0"?>
02 <PolicySet PolicySetId="PPS:VIO:role"
03           PolicyCombiningAlgId="permit-overrides">
04   <Target/>
05   <PolicyIdReference>permission:vi:request-action
06   </PolicyIdReference>
07   <PolicyIdReference>permission:vi:instantiate-action
08   </PolicyIdReference>
09   <PolicyIdReference>permission:vi:decommission-action
10   </PolicyIdReference>
11 </PolicySet>
```

Figure 6. XACML PolicySet template for permissions of role VIO

```
01 <?xml version="1.0"?>
02 <Policy PolicyId="permission:vi:request-action"
03   RuleCombiningAlgId="permit-overrides">
04   <Target/>
05   <Rule RuleId="vi:request-action" Effect="Permit">
06     <Target>
07       <Resources>...</Resources>
08       <Actions>
09         <Action>
10           <ActionMatch MatchId="string-equal">
11             <AttributeValue DataType="string">
12                   MLI:Request-VI</AttributeValue>
13             <ActionAttributeDesignator DataType="string"
14                   AttributeId="action-id"/>
15           </ActionMatch>
16         </Action>
17       </Actions>
18     </Target>
19   </Rule>
20 </Policy>
```

Figure 7. XACML Policy template for a permission.

It should be noted that here XACML policies are organised in hierarchical order, in which at the root level there are roles assignment policies (Fig. 5), subsequently permission role policies (Fig. 6) are placed at the lower layer and finally the permission policies (Fig. 7). This structure is also applied to and deployed in the implementation using Sun's XACML library [14].

C. *Security context sharing and Authorization ticket format*

To support authorization session context sharing among multiple providers and multiple domains as illustrated in Fig. 3 and Fig. 4, we use authorization tickets (AuthzTicket) to convey security context from the issuers to verifiers as described in Fig. 8. It contains authorization request contexts and authorization decisions from the VI Providers as the DACI issuers with their digital signatures. Upon receiving at PI Providers, based on trust credentials established in the trust establishment process of the DACI deployment phase [5], these tickets are verified and then evaluated against authorization policies for VIR resources managed at PI Providers. If all conditions are satisfied, PI Providers allow users to access the deployed virtual resources.

The authorization ticket schema in this paper is based on the proposed AuthzTicket format for the GAAA-NRP profile [18] that is extended for the specific needs of the Virtualised Infrastructure Services Provisioning Model.

In authorization ticket schema, Subject, Resource and Action elements are defined the same as equivalent element definitions from the XACML context schema [13]. It not only simplifies generation of both AuthzTicket and XACML authorisation request but also ensures their compatibility across the VI components and between multiple providers. Decision element contains VI Providers' authorization results along with the target VIR resource identifier at PI Providers. The validity time of the authorization ticket is specified in the Condition element, including NotBefore and NotOnOrAfter attributes. The proposed schema also supports delegation with Delegation element, to describe whether defined capabilities in AuthzTicket is delegated and restricted for delegation.

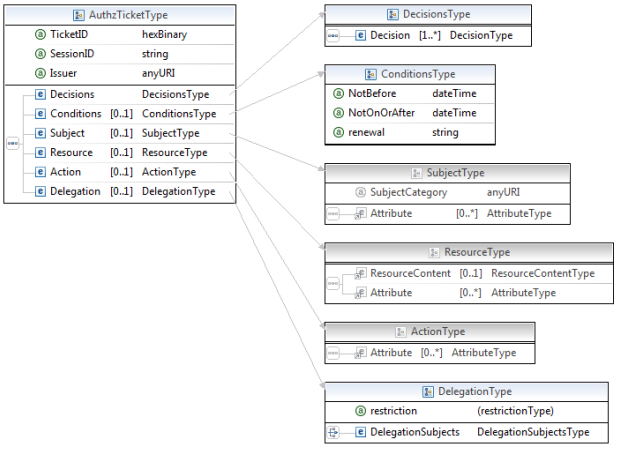

Figure 8. Authorization Ticket format

VI. RELATED WORK

Sandhu et al. [11] and Ferraiolo et al. [19] proposed Role-Based Access Control (RBAC) models, from the basic RBAC to hierarchical RBAC. NIST then standardized it as the ANSI-RBAC model [17] in 2004. After that, Kuhn et al. [20] analysed RBAC combining

with attributed-based access control approaches to take advantage of both their strengths, by defined set of RBAC-A approaches, including Dynamic roles, Attribute-centric and Role-centric.

OASIS eXtensible Access Control Markup Language (XACML) [13] was defined as the standard for authorization policies composition and management. It operates based on attribute-based access control model (ABAC) by using set of attributes of Subject, Resource, Action and Obligation elements to evaluate decisions against authorization policies. To incorporate with RBAC model, OASIS defined RBAC profile for XACML in [12]. Current XACML 3.0 focuses on administration and delegation [21] features. Because XACML standard only defines authorization context and policies schemes without specific protocol, Security Assertion Markup Language (SAML) standard is often used as the transportation for XACML requests and responses between Policy Enforcement Point (PEP) and Policy Decision Point (PDP) in the authorization model [22].

The problem of trust management in distributed, decentralized environment was initially investigated by Blaze et al. [23]. Subsequent work represented Datalog trust policy languages by Li and Michell [24] and then Role-based trust management language [25], in which trust policies map subjects to roles based on attributes in their credentials, then decisions were given from roles. Because of distributed properties of attributes in decentralized environment, they developed a credential chain discovery algorithm to retrieve and collect credentials. Such these algorithms belonged to trust negotiation process aware of privacy of sensitive attribute information such as automated trust negotiation of Li et al. [26] or the Privacy-aware role-based access control framework by Ni et al. [27].

Several previous works have proposed authorization models and implemented distributed authentication and authorization systems relating to cloud computing. Shibboleth [14] is a federated identity-based authentication system based on SAML. However it requires manual trust establishment between security domains through agreements and does not include authorization management supporting RBAC models.

Calero et al. [28] described an authorization model supporting hierarchical role-based access control, path-based object hierarchies and federation for multi-tenancy Cloud environment. Their work did not mention on multi-provider property and security context sharing crossing security domains among hierarchy providers in the Cloud IaaS model as well as provisioning configurable services.

VII. CONCLUSION

We have proposed an approach and solutions to build access control infrastructure that can be provisioned as a part of the multi-tenant, multi-provider and multi-domain Cloud IaaS infrastructure that complies with the general service lifecycle management model. It supports dynamic trust establishment with configurable security services. Furthermore, the paper presents the RBAC-based policy management approach which is practical for implementation using XACML policy generation based on the pre-defined templates. It solves the security context sharing problem between distributed entities in the Cloud IaaS architecture model by using authorization tickets.

The proposed DACI is currently being implemented in the framework of the GEYSERS project [6] on dynamic infrastructure service provisioning.

We are currently working with trust management model using policy language to support dynamic trust establishment and interconnect trust policies with authorization policies in the dynamically provisioned/configured access control services. We also extend the policy generation and management to support flexible RBAC properties assignment/configuration such as separation of duties (SoD) and delegations. We also plan to extend DACS to be able to interoperate and integrate with existing identity and management systems.

REFERENCES

[1] NIST SP 800-145, "A NIST definition of Cloud computing," Sep. 2011.

[2] NIST SP 500-291, "NIST Cloud Computing Standards Roadmap," NIST, NIST SP 500-291, Jul. 2011.

[3] "OGF ISOD-RG," *Infrastructure Services On-Demand Provisioning Research Group*. [Online]. Available: http://www.ogf.org/gf/group_info/view.php?group=ISOD-RG.

[4] OASIS IDCloud TC, "OASIS Identity in the Cloud TC." [Online]. Available: http://wiki.oasis-open.org/id-cloud/.

[5] Y. Demchenko, C. de Laat, D. R. Lopez, and J. A. Garcia-Espin, "Security Services Lifecycle Management in On-Demand Infrastructure Services Provisioning," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, Indianapolis, IN, USA, 2010, pp. 644–650.

[6] Phosphorus Project, Deliverable D4.3.1, "GAAA Toolkit pluggable components and XACML policy profile for ONRP," Deliverable D4.3.1, Sep. 2008.

[7] GEYSERS Project, "GEYSERS - Generalised Architecture for Dynamic Infrastructure Services," 2010. [Online]. Available: http://www.geysers.eu/.

[8] GEANT Project, "GEANT3 - JRA3 Composable Services." [Online]. Available: http://www.geant.net/pages/home.aspx.

[9] Y. Demchenko, C. Ngo, and C. de Laat, "Access control infrastructure for on-demand provisioned virtualised infrastructure services," in *2011 International Conference on Collaboration Technologies and Systems (CTS)*, Philadelphia, PA, USA, 2011, pp. 466–475.

[10] C. Ngo, P. Membrey, Y. Demchenko, and C. de Laat, "Security Framework for Virtualised Infrastructure Services Provisioned On-demand," 2011, pp. 698–704.

[11] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, Feb. 1996.

[12] OASIS, "Core and hierarchical role based access control (RBAC) profile of XACML v2.0." 01-Feb-2005.

[13] OASIS, "XACML 2.0 Core: eXtensible Access Control Markup Language (XACML) Version 2.0." 01-Feb-2005.

[14] Internet2 Middleware Initiative, "Shibboleth v2.0," *Shibboleth*. [Online]. Available: http://shibboleth.internet2.edu/.
[15] "OpenSAML library 2.x." [Online]. Available: http://www.opensaml.org/.
[16] The Apache Software Foundation, "Apache ServiceMix," *FUSESource SOA*, 2012. [Online]. Available: http://servicemix.apache.org/.
[17] NIST, "Role Based Access Control," 2004. [Online]. Available: http://csrc.nist.gov/rbac/.
[18] Y. Demchenko, O. Mulmo, L. Gommans, C. de Laat, and A. Wan, "Dynamic security context management in Grid-based applications," *Future Generation Computer Systems*, vol. 24, no. 5, pp. 434–441, May 2008.
[19] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, Aug. 2001.
[20] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding Attributes to Role-Based Access Control," *Computer*, vol. 43, no. 6, pp. 79–81, Jun. 2010.
[21] OASIS, "XACML v3.0 Administration and Delegation Profile," OASIS, Aug. 2010.
[22] "SAML 2.0 Profile of XACML, Version 2," OASIS, v2.0, Aug. 2010.
[23] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," pp. 164–173.
[24] N. Li and J. C. Mitchell, "Datalog with constraints: a foundation for trust management languages," in *Proceedings of the fifth international symposium on practical aspects of declarative languages*, New Orleans, 2003.
[25] Ninghui Li, J. C. Mitchell, and W. H. Winsborough, "Design of a role-based trust-management framework," pp. 114–130.
[26] J. Li, N. Li, and W. H. Winsborough, "Automated trust negotiation using cryptographic credentials," *ACM Transactions on Information and System Security*, vol. 13, no. 1, pp. 1–35, Oct. 2009.
[27] Q. Ni, E. Bertino, J. Lobo, C. Brodie, C.-M. Karat, J. Karat, and A. Trombeta, "Privacy-aware role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 3, pp. 1–31, Jul. 2010.
[28] J. M. A. Calero, N. Edwards, J. Kirschnick, L. Wilcock, and M. Wray, "Toward a Multi-Tenancy Authorization System for Cloud Services," *IEEE Secur. Privacy Mag.*, vol. 8, no. 6, pp. 48–55, Nov. 2010.