ARIMA épisciences

Service Promotion in a Federation of Security Domains

Abdramane BAH[1,2], Pascal ANDRE[1], Christian ATTIOGBE[1], Jacqueline KONATE[2]

[1]LS2N CNRS UMR 6004 - University of Nantes, France
[2]FST-USTTB - University of Sciences, Techniques and Technologies of Bamako, Mali

*E-mail : pascal.andre@ls2n.fr

DOI : 10.46298/arima.6757
Submitted on September 3, 2020 - Published on January 15, 2021
Volume : **34** - Year : **2020**
Special Issue : **Volume 34 - 2020 - Special Issue CARI 2020**
Editors : *Nabil Gmati, Mathieu Roche, Tri Nguyen-Huu, Laurent Debreu*

Abstract
Service Oriented Architecture (SOA) provides standardised solutions to share services between various security domains. But acces sur s control to services is defined for each domain, and therefore the federation of security domains brings some flexibility to users of the services. To facilitate the authentication of users, a solution is a federated access control that relies on the identity federation, which allows an user to authenticate once in one domain and to access the services of others according to her authorisation attributes. Since the access control requirements of services are specified using domain-specific authorisation attributes, the secure sharing of services in the federation becomes a real challenge. On the one hand, domains cannot abandon their access control models in favour of a global one; on the other hand, the redefinition of the access control requirements of services compromises the existing service consumers. This article extends our paper at CARI2020; we propose the promotion of services as a method that consists in publishing the services of domains at the federation level by redefining their access control requirements with the federation's authorisation attributes. Our promotion method relies on mappings between federation's authorisation attributes and those of domains to preserve existing service consumers and to support domain autonomy. We formally describe interaction and access to promoted services using operational semantics. The promotion method has been implemented with web services technologies.

Keywords
Federated SOA, Service Federation, Web service, Access Control

I INTRODUCTION

Service Oriented Architecture (SOA) implemented through web service technologies provides standardised solutions for sharing resources across organisational boundaries as services. The federation of services is defined as the sharing of services of independent organisations of a federation that are accessed on behalf of users. For a secure federation of services, it becomes critical to ensure that the shared services are accessible only to authorised users. However, every organisation in the federation is autonomous and has control over the security and the access to its services according to its own access control (**AC**) policies, such as *Role-Based Access Control* (RBAC), *Mandatory Access Control* (MAC), *Attribute-Based Access Control* (ABAC) Haguouche and Jarir [27]. Although all access control models can be converted to ABAC Hafeez, Rajpoot, and Shibli [21] through its attribute and policy concepts, the attributes may have different or even incompatible semantics from one organisation to another Preuveneers, Joosen, and Ilie-Zudor [33]. The attributes are the characteristics of the users, of the services or the environment conditions Hu, Ferraiolo, Kuhn, Schnitzer, Sandlin, Miller, and Scarfone [26] whose semantics of the authorisation methods vary from one model to another. For example, with RBAC, the authorisations of a user are determined through his *role* while with MAC, the user's *clearance* is used. We call these access control informations (e.g. role, clearance), the *authorisation attributes* or attributes.

Since the access control requirements of services are specified using the authorisation attributes, sharing services in the federation becomes a real challenge due to the AC requirements heterogeneity. On the one hand, the service consumers are not aware of all the AC requirements of services and they are not able to respond to them; on the other hand, the service providers cannot abandon their control models to a global AC model of the federation or map their model with those of all service consumers (peer AC). In Bah, André, Attiogbé, and Konaté [36] we proposed a method to enable the interaction between independent domain services inside a federation, despite the heterogeneity of their access control mechanisms. However, there are two new issues related to the service sharing challenge in the federation. The first one is the need of a common definition for service access control requirements; the service federation requires that these AC requirements be defined with mechanisms that can be easily understood by all federated domains. The second issue is the lack of flexibility for discovering and composing, despite their security mechanisms, the services defined inside the same federation. Indeed, without such flexibility which can ensure transparency of access, the involved services do not benefit from being in the same federation.

This article is an extension of a paper in CARI 2020 that contributes to solve these issues. First, we propose a global mechanism to define shared access control requirements across the federation. Second, we propose a method for promoting services at the level of the federation without modifying the access modalities of the existing consumers of the services. This promotion is formally defined in order to ensure the secure access to services; it consists in transforming the existing service access control requirements with federated mechanisms in a way that the services can be discovered and used directly at the federation level.

The rest of the paper is organised as follows: in Section II we formally define the basic concepts of service contracts, security domains, federation of services and the federated service access control. In Section III, we detail our method for promoting services at the federation level and the semantics of the access to both federated and non-federated services. Section IV describes the implementation of our method with web service protocols and its web service experimentation is presented in Section V. Related works are discussed in Section VI. We end with a prospective conclusion in Section VII.

II SERVICE SHARING IN A FEDERATION: THE CONCEPTS

We formally describe the concepts of *service-oriented architecture (SOA)*, of *security domain* and constituents, and the *federation of domain services*; we present the functioning of the access control of the services in a domain as well as the access control issues related to the federation of services.

Service-Oriented Architecture (SOA)

SOA is an approach to organise distributed resources as autonomous units of functionalities called services Dikmans and Van Luttikhuizen [23]. The services are discoverable and accessible to end-user applications or other services via standard interfaces and message protocols. SOA has three main components: the service provider, the service registry and the service consumer (the client). The service provider hosts and runs the service on the behalf of the service consumer who discovered the service description in the service registry. Consumers of a service may be unknown to the service provider Bertino, Martino, Paci, and Squicciarini [18].

A service is a self-contained, self-describing processing logic unit for remote access to business information and functionality. A service offers capabilities that meet the needs of service consumers. A service has two separate parts Bertino, Martino, Paci, and Squicciarini [18] and Dikmans and Van Luttikhuizen [23]: a contract s_{cont} and an implementation s_{imp}. The service contract describes the functionalities offered by the service as well as its access policies (e.g. its security requirements); the service implementation achieves these functionalities. The service contract facilitates the discovery of the service and hides its implementation details to its consumers. We define a *service contract* as a tuple $s_{cont} = \langle I, P_{R[X]}, Edp \rangle$ where I is the interface of the service, $P_{R[X]}$ its constrained policy and Edp the address on which the service receives message invocations; this address is called *endpoint*. The service interface I describes the set of operations provided by the service and the protocols used to access them. The service policy $P_{R[X]}$ describes the capabilities (e.g. supported encryption algorithms) and the requirements on the invocation message to access the service ; for instance the parts of the invocation message that must be encrypted, the message protection requirements in form of attributes (a_i) or *terms* (combinations of attributes such as $a_i\ OR\ (a_j\ AND\ b_k)$). To simplify the policy, we focus on the abstraction of its protection requirements in the form of authorisation attributes or terms $l_r \in R[X]$ where X is the set of attributes used to express the requirements R ; hence the notation $\langle I, P_{R[X]}, Edp \rangle$ for the service contract.

A ***security domain*** is a single unit of security administration; it can be a physical or logical unit; it consists of a set of elements (e.g. human users, applications, services), security authorities, and a security policy in which the elements are managed in accordance with the security policy *Baseline identity management terms and definitions* [42]. The scope of a domain can range from a simple computer, a business department to an entire organisation.

A domain d_i is a tuple $\langle U_i, S_i, R_i, SP_i, SS_i \rangle$ where U_i is a set of users, S_i is a set of services, R_i is the service registry, SP_i is the security policy and SS_i the set of security services of d_i. R_i contains the subset of the services of S_i that are published. The permissions to access the services are defined and controlled using $SP_i = \langle AT, AR \rangle$ where AT is a set of the authorisation attributes of d_i and AR is a set of authorisation rules based on the attributes of AT (for instance the rules describe which attributes a_k are authorised to access a service s_i: $\{(s_i, a_k)\}$). The security services SS_i include the authentication service named *local token service* (LTS), the authorisation service (ATS) and the interceptor $Interceptor$. These services are described in the next section. An user u_i of U_i is characterised by his/her authorisation attributes; $u_i = \langle uID, uAT \rangle$ where uID is his/her identity attributes (e.g. his/her name) and uAT is the set of authorisation attributes such as his/her role (e.g. teacher, manager, administrator).

A service is a communication mean between applications. It allows an application to access the business information and functions of other applications. Since an application is used by the end users, then the services are accessed for the users: the services perform actions on behalf of an user or application Papazoglou [13]. The access control ensures that only authorised users have access to the services according to the domain security policy SP. Consider, for example, a bank transfer service (*transfertService*) that *debits* a first account of a given amount to *credit* a second account of the same amount. The bank's security policy for its service defines that only employees and customers with an account in the bank have access to the *transfertService*; a customer cannot make a transfer into his/her account from someone else's account.

The access control (AC) relies on two preliminary steps Hu, Ferraiolo, and Kuhn [6]: (1) identification and authorisation of users; (2) authentication of users.

The first step consists of assigning an unique identifier and access permissions to users. The authorisation of users is done using AC models such as RBAC or ABAC. ABAC uses the notion of attribute and policy rules. The attributes are the characteristics of users, of services or the environment conditions Hu, Ferraiolo, Kuhn, Schnitzer, Sandlin, Miller, and Scarfone [26]. The policy rules express the access permissions on the attributes assigned to users and services Preuveneers, Joosen, and Ilie-Zudor [33]. With a RBAC model, the access permissions are associated to the roles and the roles are assigned to users. However, all access control models can be converted to ABAC Hafeez, Rajpoot, and Shibli [21]. Therefore, the access control consists to check whether the user has the required attributes and if these attributes have the appropriate permissions. Authorisation control relies on the authentication of users. Unlike traditional applications, a service cannot authenticate users. The authentication is to confirm the identity of an user or service Jasiul, Sliwa, Piotrowski, Goniacz, and Amanowicz [19]. The authentication is provided by security services such as the security token service in the case of web services. The authentication service of a domain is named the *local token service* (LTS). Users are authenticated to the LTS which delivers a security token as authentication credential.

A domain's service AC is implemented by the three security services according to a XACML architecture Ferraiolo, Chandramouli, Kuhn, and Hu [30] where the $Interceptor$ is the *policy enforcement point* (PEP) and the ATS is the *policy decision point* (PDP). The LTS authenticates the users and delivers a security token as authentication credential used to invoke the service. A *security token* represents a set of *claims* that are declarations made about the user's attributes. The security token is made according to the access control requirements of the wished service. The interceptor receives all call messages to domain services; it has several components including a call queue $CallQueue$. In the following we use the notation $Interceptor \downarrow CallQueue$ to denote the selection of the call queue. The interceptor extracts the user's attributes from the security token embedded in the call message and request an access decision to ATS. The latter checks whether the user's attributes have the permissions to run the service in accordance with the domain's security policies (SP). This access control process must be considered when sharing the services in a federation in order to minimize the dependencies between the domains.

Federation of Domains and Services

The interoperability between domains requires common collaboration agreements and a secure trusted environment. In such a situation, the federation is one recommended solution Tu, Zacharewicz, and Chen [31]. A ***federation of domains*** F is a set of autonomous domains that adhere to common rules and governance policies to control interactions between them Duan [16]. The administrators of the involved domains are committed to set and follow the common security rules and agreements mediation. The federation creates a trusted environment for the secure sharing of services between domains.

A federation contains a service registry R_f, a security policy SP_f, security services SS_f that enforce this policy, a set of services S_f, and a set of users U_f who use these services. A federation can be considered as a domain except that its users and services come from the domains that make it up. A federation has a security policy SP_f that defines the security rules for interactions between its domains. The users of federated domains can access the services shared in the federation. A federation can be considered as a domain except that its users and services come from its constituents domains. To have an uniform definition with a domain, we define a federation F of n domains $d_i = \langle U_i, S_i, R_i, SP_i, SS_i \rangle$ by the tuple $F = \langle U_f, S_f, R_f, SP_f, SS_f \rangle$ where U_f is the union of domain users ($U_f = \bigcup_{i=1}^{n} (U_i)$); SP_f is the security policy and SS_f the security services of the federation. Initially S_f and R_f are empty. In the following section, we show how S_f and R_f are constructed by promoting the services of domains at the federation level.

The federation of services consists of sharing the services of the domains in a federation. A ***federated service*** is a service of a domain published at the federation level, and thus, available for other domains. However, in order to federate the domain services, it is necessary to take into account also the AC of users from outside the definition domain of the services. It is necessary to take into account both the access control in the domains and the access control between the domains. The difference between intra-domain and inter-domain access control is that the services and some of its users are in separate domains. Users of the federation from outside the definition domain of the services must be allowed to access it. Since each domain is autonomous, the user's authorisation attributes of the domains defined separately can be different or have different semantics for each domain of the federation. This heterogeneity constitutes a challenge for the access control. The access control between the domains faces a challenge which is the heterogeneity of the access control models of the domains and thus that of their authorisation attributes. Consequently, the heterogeneity of domain authorisation attributes constitute an obstacle to the secure federation of services.

To overcome this challenge, we have proposed in Bah, André, Attiogbé, and Konaté [36] a method of inter-domain AC for the secure federation of services. This method is based on a federation architecture in which we introduced a new essential component at the federation level, the ***Global Access Control Mediator*** (GACM). The GACM represents the federation. It defines the authorisation attributes of the federation called the *federated attributes* (AF) that are independent of those of the domains. The security policy (SP_f) of the federation is made of the federated attributes and a set of authorisation rules (FR) associated to these attributes ($SP_f = \langle AF, FR \rangle$). Initially, FR is empty because the federation F does not dictate domain service access permissions. Federated domains keep the control on their authorisation rules. The federated attributes are used to make mappings between the authorisation attributes of the domains in order to allow AC between them. To achieve the attribute mappings and to establish trust across domains, we introduced an authentication service called *federated token service* (FTS) in SS_f at the level of the GACM. Thanks to this federation architecture, it is possible to securely share the services in the federation despite the heterogeneity of the domains attributes.

Issues on the Federation of Domain Services

Although the GACM provides inter-domain access control for the shared services in the federation, the service contracts still remain in the service registry of their domains. This means that the access control requirements in these service contracts are specified using the authorisation attributes of the domains. In the local service registry of domains, the services shared with the federation are mixed with the private services (not shared) of the domains. On the one hand, the services shared with the federation are not visible at the level of the federation; this makes it difficult to discover and use these shared services in the federation. On the other hand, the services can only be federated with the access control requirements understandable by all domains. Since services are shared between the domain that provides them and the other domains of the federation, redefining their access control requirements will compromise the operation of existing service consumers. According to these shortcomings, the new challenge is to share services of the domains in the federation while allowing the local use of these services. We address this issue in our proposal for the promotion of services in the following section.

III PROMOTION OF DOMAIN SERVICES IN THE FEDERATION

In this section we show how to overcome the challenge of sharing services at the federation level. The services shared by the domains must be visible in a single location at the federation level to facilitate their discovery and composition inside the federation. At this level, the service contract, especially the service access control requirements, must be specified by the authorisation attributes of the federation. Redefining service access control requirements with federated attributes is called the *promotion of service*; it must be transparent to existing consumers of the shared services.

Initialising the Federation

Assume we have n domains $d_i = \langle U_i, S_i, R_i, SP_i, SS_i \rangle$, $(i = 1 \cdots n)$ that wish to collaborate in a federation $F = \langle U_f, S_f, R_f, SP_f, SS_f \rangle$. The promotion consists in:

- *Set up the GACM.* The GACM is the physical representative or the operator of the federation. The GACM includes the security services that enforce the security policies of the federation. Initially, the security services include only the FTS, ($SS_f = \{FTS\}$) and the security policy SP_f includes only the federated attributes defined in common agreement between the domains. The FTS provides authentication and trust management across domains. The GACM hosted and managed by one of the federated domains, provides secure access to the services that will be shared across domains.
- *Federate the users of the domains.* The federation of users or *identity federation* allows users to access services from different domains using an unique identity (for example, their user account in their domain). Because of the heterogeneity, domains communicate with the authorisation attributes of the federation. To federate users, on the one hand the domains negotiate with the GACM to establish the mappings between their authorisation attributes and the federated attributes; on the other hand, they establish locally the mappings between the federated attributes to their authorisation attributes. The federated domain users are initialised with $\bigcup_{i=1}^{n}(U_i)$.

From now, we denote by $d_i \sqsubset F$ that a domain d_i is member of a federation F. The federation does not have services yet, $S_f = \{\}$. In the next section, we create a service registry R_f for the federation that will contain the services shared by the domains.

To facilitate the discovery and the use of shared services between domains, we need a service registry at the federation level: the *federated (service) registry*, or *federated registry* in short, is the registry R_f of the federation F. It is added into the GACM, and is empty at this point. The federated domains will publish in this registry the contracts of the services they wish to share with the federation.

Promoting the Services

To promote a service s in the federation F, we create a new federated service contract s_{fcont} from the existing service contract $s_{cont} = \langle I, P_{R[AT]}, Edp\rangle$. The AC requirements $R[AT]$ of the existing service contract should be redefined using the federated attributes $af_j \in AF$ to create the access control requirements of the federated service contract. Considering that an AC requirement is a *term* t_r built with the domain authorisation attributes $at_i \in AT$, the AC requirement for the federation results in transforming the domain attributes at_i in t_r with the federated attributes af_j. This results in AC requirements $R[AF]$ for the federated service policy $P_{R[X]}$. The mappings m between the domain authorisation attributes AT and the federated ones AF are already defined in the domains as functions: $m : AT \to AF$; they are basically, sets of couples $\{(at_i, af_j)\}$. A federated service authorisation requirement is obtained by transforming each term t_r found in the existing service contract with the mapping m; this results in a set of terms built with AF: $R[AF]$. The federated service contract $s_{fcont} = \langle I, P_{R[AF]}, Edp\rangle$ is then published in the service registry of the federation (see Figure 1) which contains the services s_f promoted by the domains ($\exists d_i \sqsubset F \wedge s_f \in S_i$); they are called the *federated services* instead of *promoted services* in order to be aligned with federated attributes $R_f = \{s_f, ...\}$.

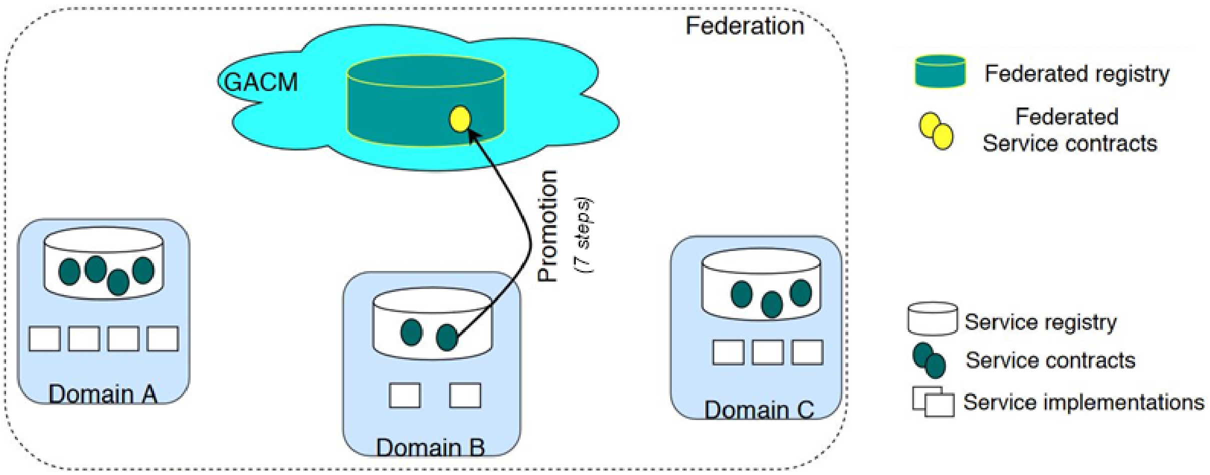

Figure 1: Overview of service promotion

We have a mapping function m between the authorisation attributes AT_i of the domain d_i and the federated attributes AF_j; $m : AT_i \to AF_f$. We generalise the application of the mapping function to a set of elements by using the notation $map(m, s_a)$ where s_a is a set of attributes. The $map(m, s_a)$ results in a set s'_a of attributes. We also use the converse function m^{-1} in the same way. ACR stands for access control requirements.

Promoting a service $s_i = \langle I, P_{R[AT_i]}, Edp\rangle$ of $d_i = \langle U_i, S_i, R_i, SP_i, SS_i\rangle$ where $SP_i = \langle AT_i, AR_i\rangle$ with $AT_i = \{at_u\}\, u \in 1..q \wedge q \in \mathbb{N}$, $AR_i = \{(s_u, at_v)\}\, u, v \in \mathbb{N}$ into the federation $F = \langle U_f, S_f, R_f, SP_f, SS_f\rangle$, consists in performing the following steps.

S1: Copy the service contract $\langle I, P_{R[AT_i]}, Edp\rangle$ of s_i from the service registry R_i;

S2: Isolate the local ACR $P_s = \{at_u, \ldots\}\, u \in 1..p \wedge p \in \mathbb{N}; P_s \subset AT$;

S3: Transform[1] P_s into terms $R[AT_i] = \{t_1, t_2, ...\}$ from $P_{R[AT_i]}$; each term is an ACR specified with the local authorisation attributes $at_u \in AT_i$;

S4: Recover the mapping function m defined by the domain d_i between its authorisation attributes AT and the federated attributes AF;

S5: Transform $R[AT_i]$ into the federated ACR $R[AF_f]$ by applying m on the set of terms $R[AT_i]$ to change the attributes $at_u \in AT_i$ with the federated attributes $af_j \in AF_f$: $map(m, R[AT_i])$;

S6: Create a new federated service contract $s_{fcont} = \langle I, P_{R[AT_f]}, Edp\rangle$ with the federated ACR $R[AT_f]$;

S7: Publish the service contract s_{fcont} in the service registry R_f of the federation F.

From an empty federated registry, after the promotion steps above, the federated registry contains one federated service s_f visible and accessible by all domains of the federation, $R_f = \{s_f\}$; the process is incremental. The following rule formally defines the promotion of domain services in the federation:

$$\frac{\begin{array}{c} F = \langle U_f, S_f, R_f, SP_f, SS_f\rangle \\ d_i = \langle U_i, S_i, R_i, SP_i, SS_i\rangle \qquad s_i = \langle I, P_{R[AT_i]}, Edp\rangle \\ d_i \sqsubset F \wedge s_i \in S_i \qquad R[AF_f] = map(m, R_i) \\ s_j = \langle I, P_{R[AF_f]}, Edp\rangle \end{array}}{F = \langle U_f, S_f, R_f \cup \{s_j\}, SP_f, SS_f\rangle} (promotion, d_i, s_i)$$

Handling the service calls in the federation

Domain services are federated to facilitate the discovery and use of the services of the federation. The GACM now serves as an interface between service consumers and service providers. The federated services are considered as provided by the federation represented by the GACM. To meet the federated service AC requirements, service consumers call the federated services with security tokens obtained at the GACM level containing federated attributes. GACM calls service implementations with this security token. The AC of domains is made using the mappings defined between federated attributes and their authorisation attributes. Once a service call is authorised, the origin domain return the responses to the GACM which in turn returns them to the service consumers (refer to Bah, André, Attiogbé, and Konaté [36] for the authorisation process). The service composition is greatly facilitated by the fact that all federated services are provided by the GACM.

We now explain how the promoted services are accessed, in a transparent way, inside a federation; it is important to keep the access as simple as possible for the users of the federation. For this purpose the services are gathered into categories, and the service's calls are managed according to these categories. We formally define the access and the interaction between the communicating entities by means of operational semantics rules.

Categories of services. Not all the services of a domain are visible at the federation level. We distinguish two categories of services: (1) the *local services*; (2) the *federated services*.

The local services are published only in the service registry of the domains; they are shared at the domain level. The AC requirements of a local service are specified with the domain authorisation attributes. As a result, accessing to local services outside their definition domains requires a common understanding of the authorisation attributes of all the others domains of

[1]In the case of web services, the AC requirements are claims expressed in XML as URIs.

the federation. The federated services are published in the service registry of the federation. The AC requirements of federated services are specified with the authorisation attributes of the federation that are understandable by all domains. There are two points of view for the federated services. From the domains (service consumer) perspective, a federated service is provided by the federation. From the domains (service provider) perspective, a federated service is a local service that is shared at the federation level. The federation of services facilitates the use and composition of the services of different domains in terms of access control. In addition, the federation of services is transparent to service consumers because it does not change the services calling rules. We formalise, with semantic rules, the processing of the calls of the two categories of services.

In the following semantic rules, the function *localToken*(s_i, u_i, s_j, ss_i) is used to get the local security token st_i delivered by a security service ss_i on behalf of the user u_i to call a service s_j from a service s_i; *domainToken*(s_i, tk_i, s_j, ss_j) is used from the service s_i of a domain d_i to request a security token tk_j required by a service s_j from the security service ss_j of another domain d_j of the federation F on behalf of a user of a domain d_i authenticated by its local security token tk_i. The expression *CallAttempt*(s_i, u, s_j) denotes a call attempt to a service s_j from a service s_i on behalf of a user u, and *SecureCall*(s_i, tk, s_k) denotes the secured call by providing the required token tk. Finally in the semantic rules, the symbols $\rightsquigarrow$ has the meaning *results in*.

Semantics of intra-domain services calls.. When a service s_1 of a domain d_i calls another service s_2 of the same domain d_i on behalf of an authorised user u of d_i, then the security token associated to u by the service security ss of d_i is used to call s_2.

$$\frac{\begin{array}{c} d_i = \langle U_i, S_i, R_i, SS_i, SP_i \rangle \\ u \in U_i \quad s_1 \in S_i \quad s_2 \in S_i \\ s_2 \in R_i \quad s_2 = \langle I, P_{R[AT_i]}, Edp \rangle \\ ss \in SS_i \quad tk = \mathit{localToken}(s_1, u, s_2, ss) \end{array}}{\mathit{CallAttempt}(s_1, u, s_2) \ \rightsquigarrow \ \mathit{SecureCall}(s_1, tk, Edp)} (intradomainLocalServCall)$$

Semantics of inter-domain services calls.. Local services can still be called by authorised domains; that is an inter-domain service call. When a service s_i of a domain d_i calls on behalf of a user u_i of d_i a service s_j of another domain d_j of the federation F, then the security token tk_j obtained from the security service ss_j of d_j with the local security token tk_i associated with the user u_i by the security service ss_i of d_i, is used to call the service s_j of d_j.

$$\frac{\begin{array}{c} F = \langle U_f, S_f, R_f, SS_f, SP_f \rangle \\ d_i = \langle U_i, S_i, R_i, SS_i, SP_i \rangle \quad d_i \sqsubset F \\ d_j = \langle U_j, S_j, R_j, SS_j, SP_j \rangle \quad d_j \sqsubset F \\ u \in U_i \quad s_i \in S_i \quad s_j \in S_j \quad ss_i \in SS_i \\ s_j \notin R_f \quad s_j = \langle I_j, P_{R[AT_j]}, Edp_j \rangle \\ tk_i = \mathit{localToken}(s_i, u, s_j, ss_i) \\ ss_j \in SS_j \quad tk_j = \mathit{domainToken}(s_i, tk_i, s_j, ss_j) \end{array}}{\mathit{CallAttempt}(s_i, u, s_j) \ \rightsquigarrow \ \mathit{SecureCall}(s_i, tk_j, Edp_j)} (interdomainLocalServCall)$$

Federated Services Calls. When the service s_j of d_j is a federated service, then the security token used to call s_j is obtained from the security service ss_f of the federation.

$$\frac{\begin{array}{c} F = \langle U_f, S_f, R_f, SS_f, SP_f \rangle \\ d_i = \langle U_i, S_i, R_i, SS_i, SP_i \rangle \quad d_i \sqsubset F \\ d_j = \langle U_j, S_j, R_j, SS_j, SP_j \rangle \quad d_j \sqsubset F \\ u \in U_i \quad s_i \in S_i \quad s_j \in S_j \quad ss_i \in SS_i \\ s_j \in R_f \quad s_j = \langle I_j, P_{R[AT_j]}, Edp_j \rangle \\ tk_i = \mathit{localToken}(s_i, u, s_j, ss_i) \\ ss_f \in SS_f \quad tk_f = \mathit{domainToken}(s_i, tk_i, s_j, ss_f) \end{array}}{\mathit{CallAttempt}(s_i, u, s_j) \rightsquigarrow \mathit{SecureCall}(s_i, tk_f, Edp_j)} (interdomainFederatedServCall)$$

These rules are used to implement the interactions between the services of the domains. A service of any domain can still call another service of the same domain by providing its security requirement. But a service can now call directly a service of another domain promoted at the federation level; in the last case, the security token of the initial caller is used to get via the federation, the right security token for calling the promoted service. Therefore we ensure the simplicity of service composition with respect to heterogeneous security policies, in the context of the promoted service.

Let a domain $d_j = \langle U_j, S_j, R_j, SP_j, SS_j \rangle$ of a federation $F = \langle U_f, S_f, R_f, SP_f, SS_f \rangle$ with $SP_f = \langle AF_f, FR_f \rangle$ and $AF_f = \{af_u\}\ u \in 1..p \wedge p \in \mathbb{N}$, the calls to the services of d_j received in the call queue of the interceptor of d_j ($Interceptor_j \in SS_j$) are not directly enabled. The calls are first checked according to the access control requirements of the called services, then they are authorised in accordance with the domain's security policy $SP_j = \langle AT_j, AR_j \rangle$ where the attributes $AT_j = \{at_u\}\ u \in 1..m \wedge m \in \mathbb{N}$ and the rules $AR_j = \{(s_u, at_u)\}\ u \in 1..r \wedge r \in \mathbb{N}$. Because local services and federated services are both provided by a domain, the calls to services are handled by the domains.

Conditions to enable calls to local service s_j from s_i with a token tk_j.. A call by a service s_i to a service s_j in a domain d_j of F is enabled under the following conditions either s_i is in the domain d_j or s_i is a service of a domain of the federation and s_i satisfies the security requirements of s_j; that is the conformance with the required access rules (denoted $conformance(tk_j, AT_j, P_{R[AT_j]})$ which checks that the token is built with attributes in AT_j and satisfies the requirements in $P_{R[AT_j]}$). Moreover the attributes in tk_j should satisfy the rules in AR_j.

$$\boxed{\begin{array}{l} \mathit{SecureCall}(s_i, tk_j, Edp_j) \in Interceptor_j{\downarrow}CallQueue \\ s_i \in S_j \vee (\exists d_i \sqsubset F \wedge s_i \in S_i) \qquad SP_j = \langle AT_j, AR_j \rangle \\ \exists s \in S_j \wedge s = \langle I, P_{R[AT_j]}, Edp \rangle \qquad Edp_j = Edp \\ conformance(tk_j, AT_j, P_{R[AT_j]}) \qquad \forall at_u \in tk_j.(s, at_u) \in AR_j \end{array}}$$

Conditions to enable the calls to s_j from s_i via the federation level with a token tk_f.. In this case the reverse mapping from the federated token (denoted by $mapToken(m, tk_f)$) should be in conformance with the local requirements of s_j.

$$\boxed{\begin{array}{l} \mathit{SecureCall}(s_i, tk_f, Edp_j) \in Interceptor_j{\downarrow}CallQueue \\ \exists d_i \sqsubset F \wedge s_i \in S_i \qquad SP_j = \langle AT_j, AR_j \rangle \\ \exists s \in S_j \wedge s = \langle I, P_{R[AT_j]}, Edp \rangle \qquad Edp_j = Edp \\ m = \{(at_j, af_f)\} \wedge tk_j = mapToken(m, tk_f) \\ conformance(tk_j, AT_j, P_{R[AT_j]}) \qquad \forall at_u \in tk_j.(s, at_u) \in AR_j \end{array}}$$

IV IMPLEMENTATION OF THE PROMOTION OF SERVICES

Web services technologies such as SOAP, WSDL, UDDI provide a SOA implementation through standard internet protocols (e.g. HTTP). A web service contract is described using the WSDL, WS-Policy, and WS-SecurityPolicy standards Aruna S and VIT Vellore [29] providing a framework to specify the service policy Bertino, Martino, Paci, and Squicciarini [18]. The interest of SOA is noticeable when reusable services can be composed to create new services or applications Dikmans and Van Luttikhuizen [23]. However, the service-oriented environment is open and decentralised; as a result, the services may belong or be scattered in several security domains.

The services of each domain are implemented with the SOAP, WSDL and UDDI web service technologies. Service contracts are described with WSDL and the service's security policies; the access control requirements are specified with the WS-SecurityPolicy standard. The promotion of services which involves a local domain and the federation is implemented with three software modules (Figure 2) and involves the following steps:

- the module *extract-mod* extracts the WSDL contract (*localContract*) of the service to be promoted from the service registry of the local domain;
- the *localContract* contract is passed to the module *promotion-mod* to obtain the promoted service contract (*promotedContract*);
- the promoted contract (*promotedContract*) is published by the module *publish-mod* as a federated service in the service registry of the federation. The federated service has additional information about the domain that is not in the promoted service contract.

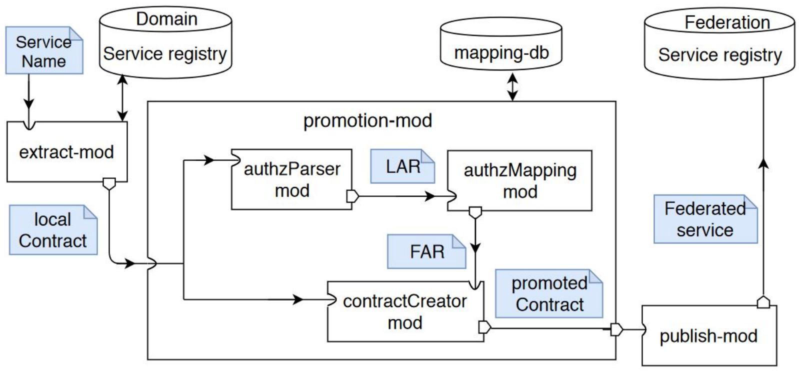

Figure 2: Overview of the implemented software modules to achieve service promotion

We implemented the *promotion-mod* module in Java using the Java API for XML Processing (JAXP). Its three sub-modules are: (1) the access control requirements parser *authzParser-mod*; (2) the access control requirements mapping module *authzMapping-mod* and; (3) the promoted service contract construction module *contractCreator-mod*.

The sub-module *authzParser-mod* receives as input the *localContract* and it outputs the list of access control requirements named *LAR* contained in this file. The *LAR* list contains the access control requirements specified with the domain-specific authorisation attributes and possibly the attributes of the domain's LTS implemented with WS-Trust. Then, the *LAR* list is passed as input to the sub-module *authzMapping-mod* which computes another list of access control requirements (named *FAR*) specified with the authorisation attributes of the federation. The mapping between the authorisation attributes of the domain and those of the federation is already defined in the mapping database *mapping-db* of the domain. The *FAR* list is transmitted as input to the *contractCreator-mod* sub-module which creates the *promotedContract* from the *localContract* by replacing the access control requirements in the *LAR* by the access control requirements given in the *FAR*.

V EXPERIMENTATION

We illustrate the service promotion with a simple web service named *HelloService* provided by a domain *IUG* identified by the URI `http://iug.net`. The contract of *HelloService* extracted from *IUG*'s service registry[2]. *HelloService* requires a security token issued by the security token service (STS) of *IUG* named *iugSTS*. This token must contain some authorisation attributes of the user on whose behalf the service is called. The access control requirements are claim requirements as defined in the WS-Trust specification.

The claims are expressed using a dialect that indicates the used syntax and semantics. However, WS-Trust does not define any dialect for the expression of claims. *IUG* has its own dialect identified by the URI `http://schemas.iug.net/authorizations/attributes`. Each authorisation attribute of *IUG* is identified by an URI. For example, the URI of the user's role is `http://schemas.iug.net/authorizations/attributes/role`.

The service *HelloService* must be shared in the federation *ICV* composed of different domains. Each domain expresses its authorisation attributes using its own dialect. To gain a common understanding of authorisation attributes, *ICV* has a dialect for its federated attributes that are shared by all domains. This dialect is identified by the URI `http://federation-icv.org/ac/ws/authorizations/attributes`. *IUG* defines mappings between the URIs of its authorisation attributes and those of the federation. For example, the user role of IUG `http://schemas.iug.net/authorizations/attributes/role` corresponds to the URI `http://federation-icv.org/ac/ws/authorizations/attributes/subject-function` of the federation.

To promote *HelloService* in the federation *ICV*, its contract is passed to the *promotion-mod* module which replaces each access control requirement expressed with the dialect of *IUG* by the corresponding one expressed with the dialect of the federation, using the mappings defined by *IUG*. The attributes of the *iugSTS* issuing the security token in *IUG* are also replaced by those of the STS of the federation. The result of this promotion is published in the service registry of *ICV*[2]. Thus, other domains can discover the *HelloService* and understand its access control requirements.

We implemented the service promotion by the means of a portable jar file. The mapping policy database is here a collection of Java HashMap instances included in the archive file. During the execution, the WSDL file of the service to promote is given in the parameters of the application as illustrated by the top of Figure 3. The result is a new WSDL file for the federated service `federatedservice.xml` as illustrated by the bottom of Figure 3.

[2]Available at `https://uncloud.univ-nantes.fr/index.php/s/N6xBRyJRsddTk5g`

Listing 1: Authorization requirement of HelloService

```
<sp:AsymmetricBinding> <wsp:Policy> <sp:InitiatorToken> <wsp:Policy>
  <sp:IssuedToken sp:IncludeToken="http://docs.oasis−open.org/ws−sx/ws−securitypolicy
        /200702/IncludeToken/AlwaysToRecipient">
    <sp:RequestSecurityTokenTemplate>
      <t:TokenType>http://docs.oasis−open.org/wss/oasis−wss−saml−token−profile−1.1#
            SAMLV1.1</t:TokenType>
      <t:KeyType>http://docs.oasis−open.org/ws−sx/ws−trust/200512/PublicKey</t:KeyType>
           <t:Claims Dialect="http://schemas.iug.net/authorizations/attributes" xmlns:authz="
                 http://schemas.iug.net/authorizations/attributes">
        <authz:ClaimType Uri="http://schemas.iug.net/authorizations/attributes/country"/>
        <authz:ClaimType Uri="http://schemas.iug.net/authorizations/attributes/role"/>
        <authz:ClaimType Uri="http://schemas.iug.net/authorizations/attributes/status"/>
      </t:Claims>
    </sp:RequestSecurityTokenTemplate>
  <wsp:Policy> <sp:RequireInternalReference/> </wsp:Policy> <sp:Issuer>
     <wsaw:Address>http://iug.net/ss−services/sts/iugSTS</wsaw:Address> </sp:Issuer> </
          sp:IssuedToken>
  </wsp:Policy> </sp:InitiatorToken> <sp:RecipientToken> ... </sp:RecipientToken>
   ... </wsp:Policy>
</sp:AsymmetricBinding>
```

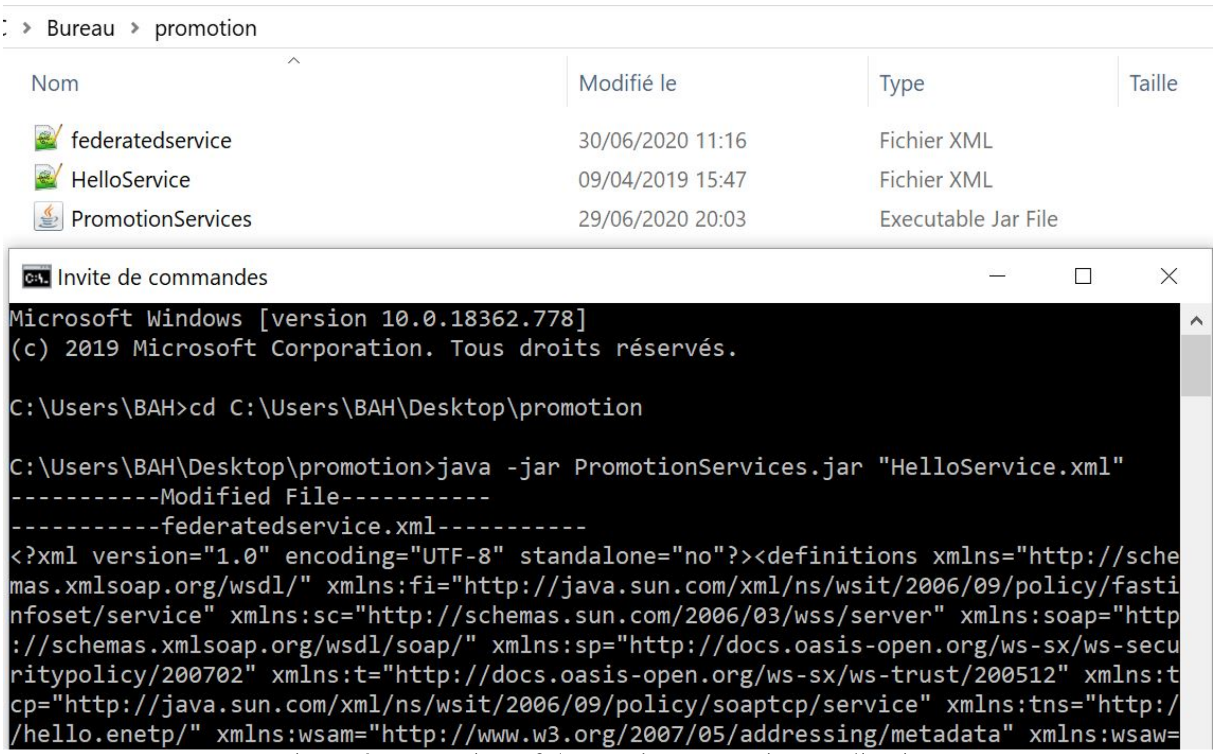

Figure 3: Execution of the service promotion application

VI RELATED WORK

The first concern about service sharing in a federation is how to make domain services available to the service consumers in other domains. The authors in Silva-Lepe, Rouvellou, Akolkar, and Iyengar [20] argue that a federation is not supposed to define a centralised service registry and it is not desirable to publish or search in the service registry of each domain; this can be cumbersome with a large number of registry and published services. They propose the communication between the service registries of the domains through notification messages of interest or availability of services satisfying a given description. Sellami et al. Sellami, Bouchaala, Gaaloul, and Tata [25] propose to organise service registries as communities according to the functionalities of the services they advertise in order to reduce the search space of service consumers. A service registry can belong to different communities at the same time with different degrees of membership. Compared to these techniques, we gather the services into the service registries (central and local) according to the description of their security properties. The service discovery is thus well targeted and effective in the federation. In Yuanmin Chen, Wei Mao, and Xiaodong Li [15] an approach similar to ours is given for the service discovery in ubiquitous computing. In their agent-based system, when a service search fails in a (domain) Directory Agent (DA), that registers Service Agents records, the request is sent to one of the Federation Guide (our GACM) which returns a list of (DA) likely to respond. The service consumers no longer need to know the details of all domains providing the services, they only need those of the federation.

WS-SecurityPolicy, WS-Trust and WS-Federation provide standard mechanisms for expressing the security requirements of the services. The heterogeneity comes from AC models and the authorisation attributes of domains. Preuveneers et al. Preuveneers, Joosen, and Ilie-Zudor [33] propose to align the authorisation attributes of domains by declaratively defining equivalence relations between their names and their values. The authors in Haguouche and Jarir [27] proposed to make the service AC at the consumer side based on collaboration contracts as proposed in Menzel, Wolter, and Meinel [9]. While this approach preserves domain autonomy in terms of security, it is difficult to adapt to the authorisations changes at service providers side. The mapping of attributes is also proposed in Rubio-Medrano, Zhao, Doupe, and Ahn [28]. It consists to transform the local attributes using derivation rules to federated attributes, which are attributes defined by the domains but recognised by the federation. The federated attributes used in our approach are defined by the federation and are independent from the authorisation attribute of the domains. Using federated attributes, the same service can be shared locally, and in different federations, using heterogeneous security informations; we ensure the autonomy of domains in terms of securing services and to minimise dependencies with the federation.

VII CONCLUSION

To meet the challenges of sharing and composing securely the services inside a federation of heterogeneous domains, we have proposed the promotion of services outside their definition domains. The services promoted by the domains become federated services. With our promotion technique, the usage of promoted services remains simple and transparent within the federation. To master the secure access to services and their composition, we have formally defined the interaction and the access to services (federated or not), using operational semantics. We have implemented, as a proof of concept, the proposed promotion of services in JAVA with the JAXP API. The services are implemented with SOAP, WSDL and UDDI web service technologies. The service access control policies are specified with WS-SecurityPolicy. We applied it to a WSDL contract of a secure service whose authorisation requirements are specified in accordance with the WS-Trust specification. A primary benefit of the promotion of services is to easily create applications and new services by composing federated services with their security requirements. Our experimentations confirm that service promotion breaks barriers to service interoperability through the expression of service access control requirements with a common claims dialect of the federation.

As for perspectives, it would be more convenient to use or extend the dialect used in one federation; for this purpose, a standard dialect like that of the WS-Federation specification could be adapted. To gain more accuracy, we plan to experiment with the calling rules of federated services on the basis of the authorisation conditions that we have defined. This will then be reused for studying the performance of the secure composition of federated services and evaluate its relevance for large distributed applications across federations.

REFERENCES

Publications

[1] M. H. Kang, J. S. Park, and J. N. Froscher. "Access control mechanisms for inter-organizational workflow". In: *Proceedings of the sixth ACM symposium on Access control models and technologies*. ACM, 2001, pages 66–74.

[2] M. Coetzee and J. H. P. Eloff. "Towards Web Service Access Control". In: *Comput. Secur.* 23.7 (Oct. 2004), pages 559–570. ISSN: 0167-4048.

[3] D. Chen and N. Daclin. "Framework for enterprise interoperability". In: *Proc. of IFAC Workshop EI2N*. 2006, pages 77–88.

[4] F. Cuppens, N. Cuppens-Boulahia, and C. Coma. "O2O: Virtual Private Organizations to Manage Security Policy Interoperability". en. In: *Information Systems Security*. Volume 4332. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pages 101–115. ISBN: 978-3-540-68962-1 978-3-540-68963-8.

[5] U. Fragoso-Rodriguez, M. Laurent-Maknavicius, and J. Incera-Dieguez. "Federated Identity Architectures". In: (Jan. 2006), page 8.

[6] V. C. Hu, D. Ferraiolo, and D. R. Kuhn. *Assessment of access control systems*. Technical report NISTIR 7316. National Institute of Standards and Technology, Sept. 2006.

[7] A. A. El Kalam, Y. Deswarte, A. Baina, and M. Kaâniche. "Access control for collaborative systems: A web services based approach". In: *Web Services, 2007. ICWS 2007. IEEE International Conference on*. IEEE, 2007, pages 1064–1071.

[8] J. Li and A. H. Karp. "Access control for the services oriented architecture". en. In: *Proceedings of the 2007 ACM workshop on Secure web services - SWS '07*. Fairfax, Virginia, USA: ACM Press, 2007, page 9. ISBN: 978-1-59593-892-3.

[9] M. Menzel, C. Wolter, and C. Meinel. "Access control for cross-organisational web service composition". In: *Journal of Information Assurance and Security* 2.3 (2007), pages 155–160.

[10] A. Singhal, T. Winograd, and K. A. Scarfone. *SP 800-95. Guide to Secure Web Services*. Technical report. Gaithersburg, MD, United States: National Institute of Standards & Technology, 2007.

[11] E. Dalsgaard, K. Kjelstrøm, and J. Riis. "A Federation of Web Services for Danish Health Care". In: *Proceedings of the 7th Symposium on Identity and Trust on the Internet*. IDtrust '08. Gaithersburg, Maryland, USA: ACM, 2008, pages 112–121. ISBN: 978-1-60558-066-1.

[12] J. Kallela. "Federated Identity Management Solutions". In: 2008.

[13] M. P. Papazoglou. *Web services: principles and technology*. und. OCLC: 255863191. Harlow: Pearson/Prentice Hall, 2008. ISBN: 978-0-321-15555-9.

[14] M. Rosen, B. Lublinsky, K. T. Smith, and M. J. Balcer. *Applied SOA: Service-Oriented Architecture and Design Strategies*. Wiley Publishing, 2008. ISBN: 0470223650, 9780470223659.

[15] Yuanmin Chen, Wei Mao, and Xiaodong Li. "Federation framework for service discovery in ubiquitous computing". en. In: *2008 11th IEEE International Conference on Communication Technology*. Hangzhou, China: IEEE, Nov. 2008, pages 600–602. ISBN: 978-1-4244-2250-0.

[16] N. Duan. "Design Principles of a Federated Service-oriented Architecture Model for Net-centric Data Sharing". en. In: *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* 6.4 (Oct. 2009), pages 165–176. ISSN: 1548-5129, 1557-380X.

[17] *The Java API for XML-Based Web Services (JAX-WS) 2.2*. 2.2. Sun Microsystems, Inc. Dec. 2009.

[18] E. Bertino, L. Martino, F. Paci, and A. Squicciarini. *Security for Web Services and Service-Oriented Architectures*. en. Springer, 2010. ISBN: 978-3-540-87741-7 978-3-540-87742-4.

[19] B. Jasiul, J. Sliwa, R. Piotrowski, R. Goniacz, and M. Amanowicz. "Authentication and Authorization of Users and Services in Federated SOA Environments - Challenges and Opportunities". en. In: *Computers & Security* (2010), page 13.

[20] I. Silva-Lepe, I. Rouvellou, R. Akolkar, and A. Iyengar. "Cross-domain service management for enabling domain autonomy in a federated SOA". en. In: *2010 International Conference on Network and Service Management*. IEEE, Oct. 2010, pages 475–480. ISBN: 978-1-4244-8910-7.

[21] K. Hafeez, Q. Rajpoot, and A. Shibli. "Interoperability among access control models". en. In: *2012 15th International Multitopic Conference (INMIC)*. Islamabad, Punjab, Pakistan: IEEE, Dec. 2012, pages 111–118. ISBN: 978-1-4673-2252-2 978-1-4673-2249-2 978-1-4673-2251-5.

[22] D. Baier, V. Bertocci, K. Brown, S. Densmore, E. Pace, and M. Woloski. *A Guide to Claims-Based Identity and Access Control: Authentication and Authorization for Services and the Web*. 2nd. Microsoft patterns & practices, 2013. ISBN: 1621140024, 9781621140023.

[23] L. Dikmans and R. Van Luttikhuizen. *SOA made simple discover the true meaning behind the buzzword that is "service oriented architecture"*. English. Birmingham, UK: Packt Pub, 2013. ISBN: 978-1-84968-417-0 978-1-84968-416-3.

[24] B. Fabian, S. Kunz, S. MüLler, and O. GüNther. "Secure Federation of Semantic Information Services". In: *Decis. Support Syst.* 55.1 (Apr. 2013), pages 385–398. ISSN: 0167-9236.

[25] M. Sellami, O. Bouchaala, W. Gaaloul, and S. Tata. "Communities of Web service registries: Construction and management". en. In: *Journal of Systems and Software* 86.3 (Mar. 2013), pages 835–853. ISSN: 01641212.

[26] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. Technical report NIST SP 800-162. NIST, Jan. 2014.

[27] S. Haguouche and Z. Jarir. "Managing Heterogeneous Access Control Models Cross-Organization". en. In: *Risks and Security of Internet and Systems*. Edited by J. Lopez, I. Ray, and B. Crispo. Volume 8924. Cham: Springer, 2015, pages 222–229. ISBN: 978-3-319-17126-5 978-3-319-17127-2.

[28] C. E. Rubio-Medrano, Z. Zhao, A. Doupe, and G.-J. Ahn. "Federated Access Management for Collaborative Network Environments: Framework and Case Study". en. In: *Proc. of the 20th ACM Symposium on Access Control Models and Technologies - SACMAT '15*. ACM Press, 2015, pages 125–134. ISBN: 978-1-4503-3556-0.

[29] Aruna S and VIT Vellore. “Security in Web Services- Issues and Challenges”. en. In: *International Journal of Engineering Research and* V5.09 (Sept. 2016), IJERTV5IS090245. ISSN: 2278-0181.
[30] D. Ferraiolo, R. Chandramouli, R. Kuhn, and V. Hu. “Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC)”. en. In: *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control - ABAC ’16*. ACM Press, 2016, pages 13–24. ISBN: 978-1-4503-4079-3.
[31] Z. Tu, G. Zacharewicz, and D. Chen. “A federated approach to develop enterprise interoperability”. en. In: *Journal of Intelligent Manufacturing* 27.1 (Feb. 2016), pages 11–31. ISSN: 0956-5515, 1572-8145.
[32] H. Nacer, N. Djebari, H. Slimani, and D. Aissani. “A distributed authentication model for composite Web services”. en. In: *Computers & Security* 70 (Sept. 2017), pages 144–178. ISSN: 01674048.
[33] D. Preuveneers, W. Joosen, and E. Ilie-Zudor. “Policy reconciliation for access control in dynamic cross-enterprise collaborations”. en. In: *Enterprise Information Systems* 12.3 (2018), pages 279–299. ISSN: 1751-7575, 1751-7583.
[34] A. Bah, P. Andre, C. Attiogbe, and J. Konate. “Federated Access Control in Service-Oriented Architecture”. In: *arXiv*. 2019.
[35] A. Bah, P. André, C. Attiogbé, and J. Konaté. *Federated Access Control in Service Oriented Architecture*. Research Report. LS2N, Université de Nantes, Apr. 2019.
[36] A. Bah, P. André, C. Attiogbé, and J. Konaté. “Federation of Services from Autonomous Domains with Heterogeneous Access Control Models”. In: *18th International Information Security for South Africa Conference*. Johanesbourg, Aug. 2019.
[37] *Apache Axis2*. `http://axis.apache.org/axis2/java/core/`. Accessed: 2019-01-24.
[38] *Apache CXF*. `http://cxf.apache.org/`. Accessed: 2019-01-24.
[39] *Metro web service stack*. `https://javaee.github.io/metro/`. Accessed: 2019-01-24.
[40] *Windows Communication Foundation*. `https://dotnet.microsoft.com/`. Accessed: 2019-01-24.
[41] *WSO2 Application Server*. `https://wso2.com/products/application-server/`. Accessed: 2019-01-24.
[42] *Baseline identity management terms and definitions*. X.1252. International Telecommunication Union. Apr. Avril 2010.
[43] *Liberty ID-FF Architecture Overview Version 1.2.* 1.2. Liberty Alliance project. May 5May 2009.
[44] *Web Services Federation Language (WS-Federation) Version 1.2. OASIS Standard*. 1.2. May 22 May 2009.