An Authorization Framework for Sharing Data in Web Service Federations

Martin Wimmer and Alfons Kemper

Technische Universität München,
{martin.wimmer | alfons.kemper}@in.tum.de,
D-85748 Garching bei München, Germany

Abstract. In this paper we present our authorization framework that supports the dynamic set-up of Web service federations for sharing data within virtual federations. Building on previous work, where we showed how the access control of Web services can be consolidated with the access control of the underlying database systems, we focus on the delegation of trust across administrative boundaries, thus enabling inter-organizational collaboration. In order to restrict the flow of (possibly sensitive) access control information, authorization proceeds as an interplay of local and distributed policy enforcement. Scalability and performance of distributed policy enforcement are provided through caching techniques, which have to ensure strong cache consistency.

1 Introduction

In areas like e-science, e-business, and e-health, inter-organizational collaborations are becoming more and more prevalent. Thereby, tightly and loosely coupled systems are differentiated. Considering e-health, an example for a tightly coupled system is a collaboration network built of hospitals for providing shared access on medical records. Collaboration might also be established and again be terminated dynamically, i.e., in a loosely coupled manner. As an example consider the exchange of therapy results for testing new medications for a research project. These scenarios have in common that data has to be shared among users of different organizations participating in a collaboration network. As each organization is considered to act autonomously regarding the management of its data, the dynamic setup of database federations with a consolidated schema is inflexible and can hardly be realized. Service oriented architectures (SOAs) provide a remedy for this integration task. As SOAs are based upon widely adopted Web service standards, they are well suited for the integration of heterogeneous applications supplied by different providers.

Providing access on data within a Web service federation poses two challenges on access control. On the one hand, access control of the Web service interfaces has to be consolidated with the security policies of the underlying database systems. On the other hand, inter-organizational privilege delegation and policy enforcement have to be realized.

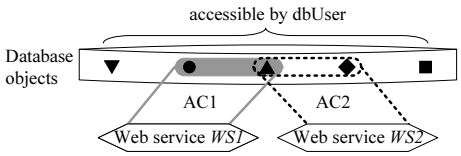

Fig. 1. Illustration of access corridors (AC1 and AC2)

Foregoing Work When Web services are used as interfaces for database systems, access control proceeds in two stages. First, the authorization of requests is checked on the side of the Web services, e.g., by employing security standards and protocols like SAML, XACML and WS-Security. Second, when services execute queries, access control is performed by the underlying database systems. In the general case, these authorization steps are done independently. With regard to this, we presented engineering techniques addressing the consolidation of the access control frameworks for Web services and databases in [1]. We further introduced a partial order on policies, allowing the verification of access control dependencies, i.e., ensuring that database policies embrace the access rights required by the depending Web services. Mainly due to restrictions of SQL, access control provided by database systems is not as fine-grained as required by today's (Web) applications. By proceeding the way as described in [1], effective security gatekeepers for database systems can be realized on the application layer.

A central aspect of the access control consolidation is the adjustment of database profiles. Today it is common practice to access databases via powerful (and potentially dangerous) super-user accounts. This might be reasonable for large software installations like SAP R/3 running within closed trust domains. But at the latest if data is made accessible via lightweight software components like Web services, the same does not hold. Figure 1 illustrates an abstract example of two services that access the same database. Each service requires only access to an extract of the database content, expressed via the access corridors `AC1` and `AC2`. As both corridors overlap, it seems preferable to use the account `dbUser` for both applications. In case of services or the service platform being attacked, information not provided through the service interfaces will also be accessible, e.g., ▼ and ■ in the figure. We avoid this risk by automatically generating adequate database profiles from the service specifications and policies. Thus, we follow the least privilege principle according to which accounts are only granted those privileges needed to provide the service functionality.

Focus of this contribution Based on our previous work for the reliable design of Web service interfaces for database systems, we concentrate on our authorization framework that enables the set-up of tightly as well as loosely coupled collaboration networks. Our approach relies on an interplay of local and distributed policy enforcement. While local authorization enforces policies within one closed trust domain, distributed authorization is required for verifying privilege and role assignments that span several organizations. In order to optimize

distributed policy enforcement, i.e., reduce communication costs and achieve low execution times, we devise a caching strategy for the goal-oriented validation of assignments. As authorizations must not succeed based on outdated cache entries, we employ caching techniques that provide the required strong cache consistency and analyze them with regard to their applicability in the authorization context. The tight integration of these techniques in our Web service platform ServiceGlobe [2] supports the secure sharing of data in Web service federations.

Document Structure The remainder of this paper is structured as follows: In Section 2 the employed notation and policy representation is introduced. In Section 3 the algorithms for local and distributed policy evaluation are described. The optimization of distributed policy evaluation through caching is shown in Section 4. Related work is presented in Section 5 and Section 6 provides a summary of the paper.

2 Policy Representation

2.1 Notation

Privilege assignments define relations between principals (the subjects) and resources (the objects). In our authorization model any access is denied unless a privilege granting a certain type of access can be inferred. This so-called closed world assumption in combination with positive authorization is reasonable for modeling access control of dynamic collaboration networks with trust relationships being existent only temporarily.

The assignment of an access right to a subject is represented by use of the following notation:

$$[\texttt{Subject} \rightarrow_P \texttt{Privilege}]_{\texttt{Condition}} \quad \texttt{Issuer} \tag{2.1}$$

A `Privilege` summarizes information about a particular resource and the way it can be accessed. An Issuer is an entity that is granted the administrative rights to declare an assignment. The validity of an assignment can further depend on a `Condition` like a temporal constraint. But it is also possible to define constraints on the `Subject`'s context. For example, the access to the medical record of a certain patient can be restricted to his/her attending physicians. In addition to this common discretionary access control model, we integrated role based access control (RBAC [3]) concepts in our authorization framework. Privileges are associated with roles through privileges-to-roles assignments in the meaning of assignment (2.1) with the subject being a `Role`. Subjects are granted privileges through the indirection of assigning roles to them. As roles constitute subjects themselves, hierarchical RBAC is realized, too. Similar to the previous assignment, role assignments can be constrained as well:

$$[\texttt{Subject} \rightarrow_R \texttt{Role}]_{\texttt{Condition}} \quad \texttt{Issuer} \tag{2.2}$$

Assignments (2.1) and (2.2) support a static rights management. The following assertions enable the delegation and revocation of privilege and role assignments, which provide a dynamic rights management.

$$[\texttt{Subject} \rightarrow_P^{(\texttt{assign}\,|\texttt{revoke})} \texttt{Privilege}]_{\texttt{Condition}}\ \texttt{Issuer} \tag{2.3}$$

$$[\texttt{Subject} \rightarrow_R^{(\texttt{assign}\,|\texttt{revoke})} \texttt{Role}]_{\texttt{Conditon}}\ \texttt{Issuer} \tag{2.4}$$

Prefix notation is used to denominate organization affiliations of subjects, roles and privileges. In the case of the subject's affiliation varying from the granted privilege's one, we talk about an inter-organizational privilege assignment or delegation. Analogous considerations apply to the assignment of roles. The right to assign a privilege, respectively role, to a subject does not presuppose the issuer to possess the privilege (role) himself/herself. A self-assignment can be prohibited via appropriate conditions.

Though entities in the subsequent examples are represented by names (e.g., Kerry Weaver as a subject or physician as a role), the introduced access control model is not restricted to identities. In general, an entity, i.e., a subject or an object, is described via a set of attributes. To give an example, physicians are characterized by their names, field of activity, social security number (SSN), age and so on. Thus, if Kerry Weaver is a chief physician at the Cook County General Hospital (CCG), the following role assignment may be used:

$$[\{\texttt{name} = \texttt{K.Weaver} \wedge \texttt{SSN = 1234}\} \rightarrow_R \{\texttt{role = CCG.ChiefPhysician}\}]\ \texttt{CCG}$$

2.2 Implementation Details

The above notation constitutes a representation of the rights management we realized in our authorization framework. We chose XACML [4, 5] as policy language for several reasons. One is its usability in the Web services context as both, XACML and the Web services technology, are based upon XML. Thus, no new terminology or processing technology is required. Moreover, the introduced formal notation can seamlessly be realized in XACML. We distinguish three types of policies.

- *Permission policies* specify access rights, i.e., objects and the way they can be accessed. They are not restricted to certain subjects. Subcategories of this type of policy exist for the assignment, respectively revocation of privileges and roles. In these cases, other permission policies (i.e., privileges) or roles constitute the objects and actions are in $\{\texttt{assign}, \texttt{revoke}\}$.
- By means of *base policies*, privileges (defined as permission policies) are assigned to subjects. Thus, this type of policy is used to express assignments (2.1), (2.3) and (2.4).
- *Role assignment policies* are used to assign roles to subjects in the meaning of assertion (2.2).

The strict separation of privilege definitions and their assignment to users, respectively roles, enable the system's scalability: The administrative effort is kept at a low level and the rights management remains concise and flexible. We integrated the described policy management into our research Web service platform ServiceGlobe. ServiceGlobe [2] is a lightweight, distributed and extensible Service Oriented Architecture. It is completely implemented in Java and based on standards like XML, SOAP, WSDL and UDDI. Services in ServiceGlobe are mobile code that can be executed on arbitrary service hosts participating in the ServiceGlobe federation. Access control functionality is supplied by separate authorization components provided by the service platform. So-called Delegation Services supervise the policy repositories of organizations, provide the functionality for privilege and role delegation, respectively revocation, and can be used to evaluate authorization requests.

3 Policy Evaluation

We distinguish between local and distributed authorization. In case subject s of organization D_s, denoted D_s.s, invokes a Web service of organization D, first of all the policies of D are evaluated. This is what we refer to as local policy evaluation. If the requested access cannot be granted solely based on the access control information available at D, it is checked whether D and D_s are part of a collaboration network which grants D_s.s the required privileges. As we do not rely on central authorities, i.e., each organization administers and enforces access rules locally, this step refers to distributed policy evaluation.

3.1 Local Policy Evaluation

Local policy evaluation is employed to check whether a subject D_s.s is granted a privilege (to execute a respective Web service) within a closed trust domain D, based only on D-local access rules. As introduced in the previous section, the policies of D are separated into three categories: permission policies, base policies and role assignment policies. Thus, local policy evaluation proceeds in three steps:

1. Determine the set of local roles $\mathcal{R}^{(l)}$ that are granted to D_s.s, defined as

$$\mathcal{R}^{(l)} \stackrel{def}{=} \{\text{D.r} \mid \text{D.r is a role} : \exists \left[\text{D}_s\text{.s} \rightarrow_R \text{D.r}'\right]_c \text{X, with D.r}' \succeq \text{D.r}\}$$

   In the above equation, X is a placeholder for an issuer and c for a condition. Through $\succeq$ a partial order on roles is defined. $\text{D.r}' \succeq \text{D.r}$, iff every privilege that is granted to D.r is also granted to D.r′. On the other hand, every subject that possesses the role D.r′ is implicitly granted the role D.r, too. In the terminology of role hierarchies [3], D.r′ is said to be a senior role of D.r. In return, D.r is called a junior role of D.r′.
2. Determine the set $\mathcal{P}$ of privileges that are granted to D_s.s directly or to any role $r \in \mathcal{R}^{(l)}$.

3. If any privilege $D.p \in \mathcal{P}$ applies, the request is permitted. Thereby, D.p applies to the request *req*, iff D.p addresses a resource that includes or is equal to the resource addressed by *req* and the action allowed by D.p is equal to or more comprehensive than the action expressed by *req*.

3.2 Distributed Policy Evaluation

Collaboration networks are established by assigning roles and/or privileges to entities of foreign trust domains. For example, if two organizations D and D′ intend to collaborate, roles and/or access rights of organization D are assigned to principals (i.e., subjects like users or roles) of D′ and vice versa. As mentioned before, the respective assignment policies are administered and enforced at the organization that is the owner of the granted privileges and roles. Consequently, access control for collaboration networks relies on distributed policy evaluation that proceeds as follows:

1. First of all the set $\mathcal{P}$ of privileges that grant the requested service execution is determined.
2. Authorization will succeed, if $D_s.s$ possesses a role that is granted at least one of the privileges in $\mathcal{P}$ and which is defined in a trust domain that cooperates with D. Let $\mathcal{R}$ be the set of roles that fulfill these requirements, defined as

$$\mathcal{R} \stackrel{def}{=} \{D_i.r_i \mid D_i.r_i \text{ is a role}, D_i \neq D, \exists\, D.p \in \mathcal{P} : [D_i.r_i \rightarrow_P D.p]_{c_i} X \vee (\exists\, D.r \text{ is a role} : [D_i.r_i \rightarrow_R D.r]_{c_i} X \wedge [D.r \twoheadrightarrow_P D.p]_{c'} X')\}$$

   X and X′ represent arbitrary issuers and c_i, c' conditions of assignments. $D.r \twoheadrightarrow_P D.p$ stands for the assignment of the privilege D.p to D.r, whereby D.p is granted to D.r either directly (referring to assertion (2.1) in Section 2.1) or via role inheritance according to the D-local role hierarchy. For each $D_i.r_i \in \mathcal{R}$ it is checked whether the role is assigned to $D_s.s$.
3. In case this assumption holds for any $D_i.r_i$, authorization succeeds. In order to verify this, the Delegation Service of organization D_i is queried, which evaluates the policies of D_i. This proceeds analogously to a D_i-local policy evaluation that either succeeds or requires further distributed evaluation.
   (a) If successful, the Delegation Service of D_i returns a positive response and the initial request for executing D's Web service can be granted.
   (b) Otherwise, the set $\mathcal{R}_i^{(f)}$ of roles that are granted $D_i.r_i$ or any senior role of $D_i.r_i$ is determined. $\mathcal{R}_i^{(f)}$ is defined as

$$\mathcal{R}_i^{(f)} \stackrel{\text{def}}{=} \{D_f.r_f \mid D_f.r_f \text{ is a role}, D_f \neq D_i \wedge [D_f.r_f \rightarrow_R D_i.r_i']_c X \text{ with } D_i.r_i' \succeq D_i.r_i\}$$

   Distributed evaluation branches by invoking the Delegation Services of the trust domains D_f with $D_f.r_f \in \mathcal{R}_i^{(f)}$. The services are called, querying whether $D_s.s$ is granted $D_f.r_f$. Each of these invocations can lead to further distributed policy evaluation calls, i.e., step 3 can be executed repeatedly.

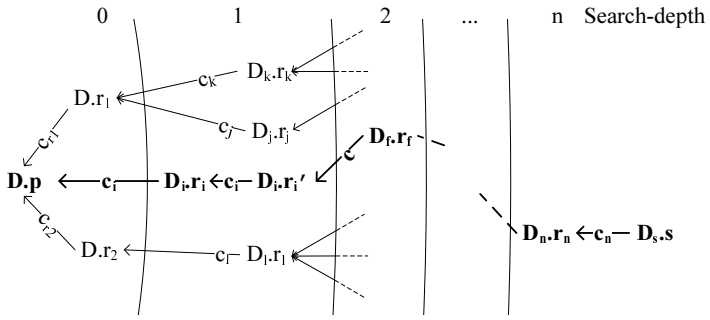

Fig. 2. Graphical representation of distributed evaluation

The evaluation terminates at organization D_i, in case of $D_i.r_i$ being assigned to $D_s.s$ or the set $\mathcal{R}_i^{(f)} = \varnothing$. A chain of distributed role assignments that applies to the subject $D_s.s$ is called *authorization path*. If at least one authorization path exists, the execution of the Web service of D requested by $D_s.s$ is permitted.

Distributed policy evaluation thus corresponds to a browsing of a distributed role hierarchy. As illustrated in Figure 2, this is equivalent to a backwards oriented depth first search (DFS) in the distributed role assignments graph, with roles constituting the nodes of the graph and the edges being annotated with conditions. In other words, there is a directed edge from $D_f.r_f$ to $D_i.r_i'$ annotated with c, if and only if there exists a role assignment policy $[D_f.r_f \rightarrow_R D_i.r_i']_c$ X. An increase of the search-depth proceeds by invocations of collaborating Delegation Services. It can be assumed that real-world collaborations tend to have a quite short delegation depth, denoting that collaboration networks are likely to span only few trust domains. At the same time, organizations tend to directly cooperate with other organizations. Consequently, a breadth first search (BFS) algorithm is supposed to be particularly suitable by providing lower response times than DFS. Unfortunately, a BFS-variant can hardly be realized due to the distributed characteristics of the graph. For BFS, the role assignment graph either has to be traversed with an initial depth bound that is increased successively – which would lead to higher network traffic –, or the complete access control information has to be available at a central authority, which would contradict security considerations of loosely coupled systems. Techniques for enhancing the evaluation process are discussed in more detail in Section 4.

Sensitivity of access control information is also the reason for the pull characteristic of distributed policy evaluation. According to the pull model, the requestor provides only his/her identifying attributes (and/or context information), i.e., $D_s.s$, while his/her privileges are determined by the framework. On the other hand, following a push model, $D_s.s$ would have to provide all security credentials that are required to execute the particular Web service in advance. These security credentials for example consist of the public keys of the roles $D_s.s$ possesses. On the one hand $D_s.s$ has to know which set of keys is required for executing the service method. This information has to be provided by the

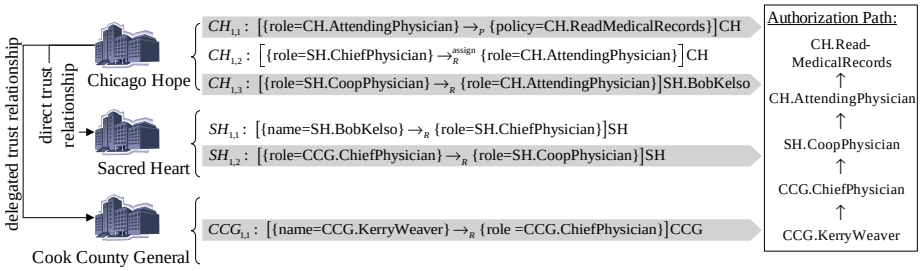

Fig. 3. Example of a hospital collaboration network

Web service but can only be given to D_s.s under the precondition of D_s.s being trusted. D_s.s (or the application calling the service acting in D_s.s' place) on the other hand has to provide the required security credentials. In general, these have to be collected in a distributed manner, as storing all role information with D_s.s would contradict RBAC principles. In our approach, the service platform is responsible for verifying the authorization based only on D_s.s' identity and context information.

3.3 Example Evaluation

Figure 3 illustrates an example of a hospital collaboration network. The Chicago Hope Hospital (CH), the Sacred Heart Hospital (SH) and the Cook County General Hospital (CCG) form a federation that allows physicians to read medical records of patients that are housed in collaborating hospitals, e.g., the Chicago Hope Hospital. This privilege is represented through the permission policy CH.ReadMedicalRecords. Figure 3 shows an extract of the access control rules. For keeping the example concise, we disregarded further conditions, like the restriction that only medical records of patients that agreed to publish their data within the federation would be accessible. The policies contain rules for the organization spanning assignment of roles. Rule $CH_{1,2}$ states that chief physicians of SH are allowed to delegate the role CH.AttendingPhysician, which is applied by rule $CH_{1,3}$. Through $SH_{1,2}$ this role is assigned to chief physicians of the CCG and, according to $CCG_{1,1}$, also to Kerry Weaver. If Kerry Weaver requests to read a medical record of a patient housed at the Chicago Hope Hopital, the authorization chain $CCG_{1,1}$, $SH_{1,2}$, $CH_{1,3}$ to $CH_{1,1}$ has to be traversed in reverse order. Foremost, as local evaluation at CH fails, the roles that are granted CH.ReadMedicalRecords are determined. This applies to CH.AttendingPhysician. As Kerry Weaver is not granted the role based on the local policies, the Delegation Service of SH is invoked, querying whether she possesses the role SH.CoopPhysician. Again, this cannot be answered SH-locally, so that the Delegation Service of CCG is invoked, asking for the assignment of CCG.ChiefPhysician. Due to $CCG_{1,1}$ the request is granted.

3.4 Inter-organizational Assignments

Collaboration networks based on Web services technology can be highly dynamic. This especially applies to short-term interactions. ServiceGlobe as the underlying authorization middleware provides the required level of flexibility through the support of inter-organizational assignments. Privileges for the assingment or revocation of access rights or roles are expressed via permission policies in the meaning of assertions (2.3) and (2.4) in Section 2.1. If a subject D_{s1}.s1 intends to assign the privilege D.p (respectively role D.r) to a subject D_{s2}.s2, it invokes the Delegation Service of D. If D_{s1}.s1 is authorized, particular policies are generated. In case of a privilege assignment, a base policy representing $[D_{s2}.s2 \rightarrow_P D.p]\, D_{s1}.s1$ is created. A role assignment is represented through a role assignment policy $[D_{s2}.s2 \rightarrow_R D.r]\, D_{s1}.s1$.

Revocations of assignments lead to the deletions of the respective policies. We distinguish between conservative and destructive revocation. If, for example, D_{s1}.s1 revokes the role assignment $[D_{s2}.s2 \rightarrow_R D.r]\, D_{s1}.s1$, only the associated role assignment policy will be deleted in case of conservative revocation. If the destructive approach is applied, it is analyzed if D.r authorizes to delegate roles or privileges, and every applied assignment will be revoked, too. Consequently, the destructive approach cascades and involves the reorganization of the policy repositories of the cooperating organizations, in general. Beneath its complexity, destructive revocation might not be used depending on the application, e.g., if an administrative officer of a hospital, who was responsible for the arrangement of attending physicians, retires, his/her assignments shall not get lost.

A policy repository has to be modified exclusively, i.e., has to be locked in case of updates. Otherwise, concurrency can cause inconsistencies. Considering two requests with the first one performing a delegation of any role r, while the second initiates the revocation of the respective delegation permission, isolation of requests has to be ensured to avoid conflicts.

For some applications it might be required that (long-lasting) inter-organizational transactions perform a rollback if any required privilege is revoked in the meantime. This is realized by integrating Delegation Services into the transaction workflow. Prior to the start of a particular transaction, authorization proceeds as described above. Before the services commit, access control is performed once again, e.g., by verifying the authorization path that was found during the initial authorization check. If authorization fails, the Delegation Service sends an abort message to the coordinator, initiating a rollback of the transaction.

4 Caching of Authorization Paths

The described policy enforcement technique performs a DFS on the distributed role assignment graph. Thus, in the worst case the complete collaboration network has to be analyzed sequentially. While long execution times of unsuccessful policy enforcements might be tolerable, successful authorizations have to proceed as quickly as possible. Most collaborations can be assumed to span only

few organizations, i.e., role assignment chains are rather short. However, as outlined before, BFS can hardly be applied as the distributed role assignment graph cannot be provided at a central authority in order to preserve autonomy of authorization and ensure scalability. But the response times of the DFS variant can be reduced substantially through parallelizing the search, i.e., asynchronously querying the Delegation Services of cooperating organizations. Unfortunately, lower response times then have to be traded for an increase of network traffic. Both objectives, i.e., low response times and low network traffic can be obtained by caching authorization paths of frequently and/or recently used requests. More precisely, paths cached at a domain D look as follows:

$$\left\langle \mathrm{D_s.s} \xrightarrow{\mathrm{c_n}} \mathrm{D_n.r_n} \xrightarrow{\mathrm{c_{n-1}}} \mathrm{D_{n-1}.r_{n-1}} \ldots \xrightarrow{\mathrm{c_1}} \mathrm{D_1.r_1} \xrightarrow{\mathrm{c}} \mathrm{D.r} \right\rangle$$

This represents the delegation chain that asserts that the role D.r is assigned to the subject $\mathrm{D_s.s}$ – if the conditions $\{\mathrm{c_n}, \mathrm{c_{n\text{-}1}}, \ldots, \mathrm{c_1}, \mathrm{c}\}$ are fulfilled. Such a cache entry is created when $\mathrm{D_s.s}$ invokes a Web service of D for which the privileges of D.r are required and $\mathrm{D_s.s}$ is granted D.r. We refer to a Delegation Service as a client, in case it consumes information, i.e., caches results of authorization queries. If a Delegation Service returns successful evaluation results that can be cached at collaborating organizations, the service is characterized as a server.

Caches are evaluated prior to a distributed policy evaluation query. If $\mathrm{D_s.s}$ requires privileges that are granted to D.r, the cache entries starting with $\mathrm{D_s.s}$ and ending with $\mathrm{D.r'}$ are determined. In this regard, $\mathrm{D.r'}$ has to be a senior role of D.r or equal to D.r, i.e., $\mathrm{D.r'} \succeq \mathrm{D.r}$. Thus, not only exact matches can be handled. If no applicable path is found, distributed policy evaluation takes place as described above. Caching of access control information requires strong cache consistency [6], meaning that authorization must not succeed based on outdated, i.e., invalidated access control information. In the following we describe three strategies that ensure strong cache consistency and are thus applicable for caching authorization paths.

4.1 Client Validation

Client validation denotes that client-Delegation Services have to ensure the validity of cached entries. Therefore, a cache hit, i.e., an applicable authorization path, is validated before authorization succeeds. Considering the introduced formal representation of a delegation path, the Delegation Service of D first checks whether $\mathrm{D_1.r_1} \xrightarrow{\mathrm{c}} \mathrm{D.r}$ is still valid by evaluating the D-local policies and the requestor's context (D.r is assigned to $\mathrm{D_1.r_1}$ via a D-local role assignment policy). Subsequently, the evaluation is continued at the Delegation Service of $\mathrm{D_1}$ for verifying the next extract of the authorization path ($\mathrm{D_2.r_2} \xrightarrow{\mathrm{c_1}} \mathrm{D_1.r_1}$) and so on. Verification succeeds, if every assertion of the delegation path holds. Otherwise, the authorization path is removed from the cache and further applicable entries of the cache are evaluated, i.e., validated. Common distributed policy evaluation proceeds if no applicable entry could be validated.

Compared to the introduced DFS-like distributed policy enforcement, the validation of authorization paths significantly reduces run time. Client validation in the authorization context differs from common Web caching scenarios: In the general case, the location of the requested Web content remains unchanged and performance there is enhanced by reducing data transfer. Here, execution time is saved as the validation of authorization paths corresponds to a goal-oriented "walk" through the distributed role assignment graph, rather than a complete browsing of the graph in the worst case.

According to the above description, a cache entry represents a complete delegation path. This design enables the efficient validation of cache entries, but it requires that all cooperating organizations agree in the exchange of access control information. In many cases, this assumption holds, because the cooperating organizations are supposed to trust each other. Nevertheless, the information flow is kept at minimum by caching authorization path fragments of the form $\langle \mathrm{D_s.s} \twoheadrightarrow \mathrm{D_1.r_1} \xrightarrow{c} \mathrm{D.r} \rangle$ at D. As the assignment $\mathrm{D_1.r_1} \xrightarrow{c} \mathrm{D.r}$ is contained in the repository of D, no security relevant data disperses to other organizations. The complete authorization path is restored by starting a goal-oriented search at $\mathrm{D_1}$. The Delegation Service of $\mathrm{D_1}$ either determines the subsequent fragment of the authorization path in its local cache, or – in the worst case – triggers distributed policy enforcement. With this modification, caching within dynamic collaboration networks, with trust being existent only temporarily, is enabled.

4.2 Server Invalidation

When using server invalidation, Delegation Services inform adjacent clients in case of policy updates that lead to an invalidation of cache entries. Therefore, a Delegation Service has to log the requests of clients that received positive authorization responses. A modification of policies must not proceed before all affected cache entries have been invalidated and the invalidation has been acknowledged by the clients. Consequently, this approach is vulnerable regarding the unreachability of services, e.g., because of network failures. This clearly disqualifies this caching technique for highly dynamic federations.

One further disadvantage of server invalidation is its limited scalability. A Delegation Service has to log requests of clients, in order to notify them of policy updates. But as in most use cases the size of collaboration networks is restricted, this is not considered to be a crucial drawback.

4.3 Lease-based Approach

Lease-based caching [7] is situated between client validation and server invalidation. A lease is a contract between client and server-Delegation Service. The server asserts not to modify the administered access control policies as long as the lease is valid. After the lease has expired, this task is shifted to the client. When updating policies, a server has to wait until each client has acknowledged the invalidation of cache entries, or in case of any of them being unreachable,

until the respective leases have expired. Consequently, the lease-based approach is parameterizable by the validity periods of leases. Setting them close to zero, it behaves quite similar to client validation. On the other hand, server invalidation is approximated by setting the periods near infinity. Thus, depending on the parametrization, the pros and cons of client validation and server invalidation more or less apply to this caching technique, too.

4.4 Experimental Results

To the best of our knowledge, there do not exist any standardized benchmarks for measuring and comparing the performance of caching access control information. Therefore, we developed several test cases varying the dimensions (i.e., width and depth) of collaboration networks, the frequency of policy updates, and the request characteristics. Due to space limitations, we only present experimental results regarding the variation of the request characteristics. Figure 4 illustrates the results of a benchmark that was performed varying the relation of local and distributed policy enforcement. p is the proportion of requests that are handled by local policy evaluation. We simulated a static collaboration network with branching degree 2 and maximum delegation depth 5. That means, every domain represented by a Delegation Service delegates trust to 2 other domains and the maximum length of role delegations is 5. Thus, a network of 63 collaborating organizations was simulated.

The benchmark was performed on ServiceGlobe installations running on a cluster of 2.8GHz Intel Xeon systems with up to 4GB of main memory. For the test scenario we measured the performance of authorization when no caching, client validation, and server invalidation were used. The performance of the lease-based approach depends on the expiration period of leases and resides between the performance of client validation and server invalidation. Therefore, results for this kind of caching technique are not listed for the following experiments. On average, the execution of a Delegation Service lasted 650ms. About 30% of this time are required for local policy evaluation and cache examination (with policies and caches being realized as XML documents), while the predominant amount is needed for service loading and initialization.

In the experiment, 50 different request types were simulated with $p{\cdot}50$ positive access rules being inserted in the topmost policy repository, i.e., a portion of p requests require local policy evaluation. The $(1-p)\cdot 50$ access rules that apply to the remaining part of request types were inserted in the other repositories according to a Zipf distribution. When sorting the levels of the collaboration network according to the frequency with which access rights are assigned to them, Zipf's law states that the frequency of rights being assigned to a level l ($\mathrm{frq}(l)$) is inversely proportional to its ranking following a power law: $\mathrm{frq}(l) \sim 1/l^{\alpha}$. Typically, $\alpha \in [0,1]$. A uniform distribution is modeled through $\alpha = 0$, while a highly skewed distribution is achieved at $\alpha = 1$. By setting $\alpha = 0.85$, we simulated a scenario with the predominant part of requests being authorized after a few delegation steps, while only some require the enforcement of policies of the undermost policy repositories. For each value of p, 2000 requests were posted

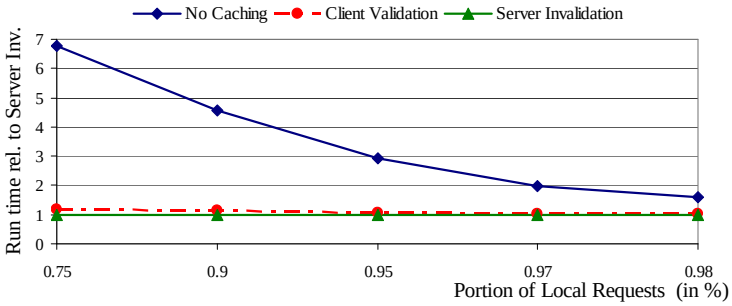

Fig. 4. Variation of the request characteristic

to the root Delegation Service and the mean evaluation time was determined. The 2000 requests are separated into clusters of those requiring local policy evaluation and those requiring distributed policy evaluation with a ratio of p to $(1 - p)$ – analogously to the distribution of access rights. For each cluster, the request types were chosen according to a Zipf distribution with $\alpha = 0.85$, as the authors of [8] showed that Zipf-like distributions are useful for modeling the request characteristics of many Web applications.

In contrast to the caching of arbitrary Web content, the space requirements for the caching of authorization paths can be estimated quite well in advance. Thus, cache replacement strategies are of minor interest and were not considered in these experiments. In many real-world applications, p is assumed to be quite close to 1. Figure 4 illustrates the performance measurements relative to the results for server invalidation. As the presented experimental results show, response times for requests requiring distributed policy enforcement can be reduced significantly, thus justifying the use of authorization caches.

5 Related Work

There has been substantial effort in the research community for providing security for distributed applications. The Community Authorization Service (CAS) [9] of the Globus Project manages access control for resources that are available within larger communities. The CAS plays the part of a central authority. In contrast to this, our framework is suitable for both, centralized and decentralized authorization. We presented our algorithm for distributed policy evaluation, supporting loosely coupled systems. Tightly coupled federations building upon centralized authorization can seamlessly be realized by shifting policy enforcement to a trusted authority, thus breaking authorization down to local policy enforcement. The CAS uses a push model for inferring granted access rights, while we use a pull model to determine the roles and privileges that are granted to users. A pull model is also used in Akenti [10] that is related to X.509 certification technique. Policies, conditions, and attribute statements can be encapsulated into certificates. Akenti allows access control for one resource to be administered by multiple authorities. Our framework supports this through delegation privileges, but policy evaluation remains the task of the organizations the

respective resource belongs to, thus preserving autonomy of authorization. Our approach for distributed role assignment is based on concepts similar to those of the RT-framework [11] and the dRBAC [12] approach. We adapted the syntax of [12] for the representation of privilege and role assignments. In [12] complexity of distributed policy enforcement was countered through a publish-and-subscribe algorithm related to server invalidation. As our comparison of caching techniques shows, server invalidation is not well applicable for dynamic federations. Instead, client validation is recommended.

Jajodia et. al. [13] present a flexible and capable security framework supporting positive and negative authorization. They introduce conflict resolution techniques as they are also provided by XACML [4, 5]. Both, XACML and the Web services technology are based upon XML. Thus, its integration into the security system of a service platform that is based on SAML [14] and/or WS-Security [15] like the one of ServiceGlobe is supported. As our formal notation can seamlessly be realized in XACML we chose it as policy language and extended it by integrating database specific object types and functions for attribute comparisons and the evaluation of conditions. In earlier work [1] we presented a technique for comparing XACML policies. More details and a definition of a partial order on policies are presented there.

6 Conclusion and Future Work

The Web services technology provides the basis for sharing data in organization-spanning collaboration networks and future database related applications are likely to be realized as distributed Web service federations. But the sharing of data requires a capable and flexible security system coping with authorization within tightly and especially loosely coupled collaboration networks. This is offered by our authorization framework that can be applied for both types of federation systems. By employing a central authority responsible for administering the resources within a virtual organization through performing access control in a centralized manner, tightly coupled federations are realized. We also presented an algorithm for distributed, i.e., decentralized policy enforcement. This allows the set-up of loosely coupled federations with trust relationships being established only temporarily. Thereby, the flow of information within the federation is restricted and the collaborating organizations preserve their authorization autonomy by policies being administered and evaluated locally. In order to optimize distributed policy enforcement, we devised caching strategies that allow the efficient evaluation of frequently and repeatedly occurring authorization requests – again by ensuring a minimum flow of information.

Future research proceeds in the following directions: On the one hand, caching of authorization paths is examined further, e.g., by the evaluation of different cache replacement strategies and usage of cryptography for securing cache entries. On the other hand, we intend to examine negative authorization, i.e., the evaluation of prohibitions in the context of loosely coupled federations.

References

1. M. Wimmer, D. Eberhardt, P. Ehrnlechner, and A. Kemper, "Reliable and Adaptable Security Engineering for Database-Web Services," in *Proceedings of the Fourth International Conference on Web Engineering*, vol. 3140 of *Lecture Notes in Computer Science (LNCS)*, (Munich, Germany), pp. 502–515, July 2004.
2. M. Keidl, S. Seltzsam, K. Stocker, and A. Kemper, "ServiceGlobe: Distributing E-Services across the Internet (Demonstration)," in *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, (Hong Kong, China), pp. 1047–1050, Aug. 2002.
3. 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, 2001.
4. T. Moses, A. Anderson, A. Nadalin, B. Parducci, D. Engovatov, *et al.*, "eXtensible Access Control Markup Language (XACML) version 2.0." `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml` (last visited 06/20/05), Dec. 2004.
5. A. Anderson, "Core and Hierarchical Role Based Access Control RBAC Profile of XACML version 2.0." `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml` (last visited 06/20/05), Sept. 2004.
6. L. Y. Cao and M. T. Özsu, "Evaluation of Strong Consistency Web Caching Techniques," *World Wide Web*, vol. 5, no. 2, pp. 95–124, 2002.
7. C. Gray and D. Cheriton, "Leases: An Efficient Fault-tolerant Mechanism for Distributed File Cache Consistency," in *Proceedings of the Twelfth ACM Symposium on Operating Systems Srinciples*, pp. 202–210, ACM Press, 1989.
8. L. A. Adamic and B. A. Huberman, "Zipf's Law and the Internet," *Glottometrics*, vol. 3, pp. 143–150, 2002.
9. L. Pearlman, I. F. V. Welch, C. Kesselman, and S. Tuecke, "A Community Authorization Service for Group Collaboration," in *3rd International Workshop on Policies for Distributed Systems and Networks (POLICY)*, (Monterey, CA, USA), pp. 50–59, IEEE Computer Society, June 2002.
10. M. R. Thompson, A. Essiari, and S. Mudumbai, "Certificate-based Authorization Policy in a PKI Environment," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 4, pp. 566–588, 2003.
11. N. Li, J. C. Mitchell, and W. H. Winsborough, "Design of a Role-based Trust Management Framework," in *Proc. IEEE Symposium on Security and Privacy*, (Oakland), pp. 114–130, May 2002.
12. E. Freudenthal, T. Pesin, L. Port, E. Keenan, and V. Karamcheti, "dRBAC: Distributed Role-Based Access Control for Dynamic Coalition Environments," in *Proceedings of the Twenty-second IEEE International Conference on Distributed Computing Systems (ICDCS)*, (Vienna, Austria), pp. 411–420, July 2002.
13. S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian, "Flexible Support for Multiple Access Control Policies," *ACM Trans. Database Syst.*, vol. 26, no. 2, pp. 214–260, 2001.
14. S. Cantor, J. Kemp, R. Philpott, and E. Maler, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML)." `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security` (last visited 06/20/05), Mar. 2005.
15. A. Nadalin, C. Kahler, P. Hallam-Baker, R.Monzillo, *et al.*, "Web Services Security (WS-Security)." `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss` (last visited 06/20/05), Mar. 2004.