Role-based AAA for Service Utilization in Federated Domains

Markus Hillenbrand, Joachim Götze, Jochen Müller and Paul Müller

University of Kaiserslautern
Department of Computer Science
http://www.icsy.de
{hillenbr,j_goetze,jmueller,pmueller}@informatik.uni-kl.de

Abstract: The Web service specifications managing authentication and authorization are still a work in progress. A large-scale application built today is in need of a solid authentication and authorization infrastructure. A role-based authentication, authorization and accounting (AAA) concept is able to provide the services needed, while it is possible to gain additional flexibility to handle the co-operation between different authentication and authorization infrastructures.

1 Motivation

In service-oriented architectures the utilization of services is the main focus. Services provide their functionality on typically public interfaces that are utilized by users directly or by software on behalf of the user. In general entities acting as *service requestor*, *service provider* and *service broker* are involved. These entities are building the core of a service-oriented architecture. Aspects like access control, data security, and accounting play an important role if a service is accessible via an open network like the Internet. The utilization of a provided service is often only possible for authorized users and hence authorization has to be included in an accounting strategy.

An essential part of a user-friendly design of service-oriented architectures includes a mechanism for single sign-on, allowing users to sign-on and access all possible services without additional or subsequent requests for sign-on. After establishing the simple access through single sign-on to all services of a security domain, it is possible to extend this strategy on multiple domains forming a federation of trust of security domains that mutually trust each other. User of a domain can – without an additional sign-on – access services of other domains in this federation of trust. This concept is based on a direct trust between these two security domains. Additionally it is possible to extend the understanding of trust beyond direct trust and allow a transitive trust relation (indirect trust) by allowing the access of services of a security domain via an intermediary security domain. The authentication mechanisms of the domains involved are responsible for the management and realization of the trust relations. Furthermore it is very convenient in a dynamic environment to allow a certain user to delegate his access rights for a short period of time to another user. Because of this a user is gaining access to services that are normally not at his disposal and makes it possible to concede a user access to services needed urgently in an exceptional situation without much effort.

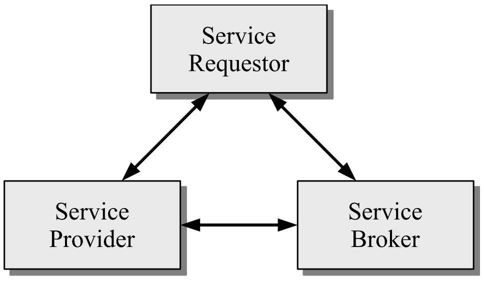

Figure 1: Service-oriented Architecture

Moreover the concept of joining security domains into a federation of trust and their services into a combined working environment is leading to an environment allowing users to act in different roles. These roles take effect on the utilization of services, e. g. a user is working in a domain as administrator and because of that he is allowed to manage the services of this domain, but if he accesses services in a federated domain, he will receive the access rights of a guest and by that only have access to minor functionality of the services. An additional scenario is the user working within multiple projects where he or she acts with different roles that have to be accounted for separately. In the context of a single sign-on environment this means that the authentication and authorization services have to know about the different roles and have to distinguish between them.

In the Venice[1] project [HMM04b, HZ04, HMM04a, ZHM05, ZH04] a general model for providing supplementary services for a Voice over IP (VoIP) environment has been realized. In this project necessary services realized as Web services are provided by corresponding service providers on the Internet and accessed by VoIP clients. The VoIP providers are forming the above mentioned federation of trust and the user and their booked supplementary services are mapped to roles according to their domain. After the sign-on in his security domain the user is able to access and use services of other VoIP providers and receive an accounting for all accrued costs via his own provider.

This paper provides an overview on the design of the role-based AAA framework beginning with a short overview on related work in section 2. In section 3 the basic concepts applied in the framework are explained. Section 4 describes the model in detail, focusing on the new extensions to basic Web service specification that have been made to achieve authentication, authorization, and user roles. Section 5 concludes this paper with a summary.

2 Related Work

The specification of Web services includes the mechanisms to handle authentication and authorization information. WS-Security [ADLHH02] has its focus on the secure transport of network messages, while the specifications WS-Trust [ABCDL04], which is mainly

[1] This project is funded by Siemens AG and performed at the University of Kaiserslautern, Germany.

oriented on secure network communications, and WS-Federation [BDLD$^+$03], which is enlarging the scope of secure communication across domain borders, are still work in progress.

The Security Assertion Markup Language (SAML) [HM03], [CKPM03] defines syntax and processing semantics of assertions made about a subject by a system entity in order to create and exchange security information between online partners. The focus of SAML 1.1 has been on Web site security, while the specification of SAML 2.0 adds functionality for Web services.

Part of the Liberty Alliance Project is the specification of the Liberty Identity Web services Framework [Kem04]. Standardized messages to handle authentication, message protection, service discovery and policies are defined in this framework.

Within the current status of these specifications it is not possible to realize a role-based AAA as fine-grained as explained in section 3.

3 Basic Concepts

A role-based authentication, authorization, and accounting (AAA) environment is based on several basic definitions. These definitions are the building blocks of the whole environment and will be explained in the following.

3.1 Single Sign-On

A single sign-on (SSO) environment is an authentication infrastructure that provides a single authentication authority for all authentication processes by sourcing out authentication processes to a specialized infrastructure that handles authentication for all applications. Jan de Clercq defines SSO in [dC02] as follows:

> "Single sign-on is the ability for a user to authenticate once to a single authentication authority and then access other protected resources without re-authentication."

In a service-oriented architecture the management of authentication information has to be provided as a service. This service is located on a centralized authentication infrastructure to handle any authentication request in a standardized manner. From the user's point of view one authentication gives transparent access to all services that use this single sign-on service.

Fig. 2 shows the authentication process in a service-based SSO environment using tokens to proof successful authentication and to gain access to services. After the primary sign-on the user receives a temporary token that he can use to proof his identity while signing on to any authentication authority trusting the primary authentication authority.

3.2 Federation of Trust

Because of the diversity of SSO solutions, it is difficult to combine several infrastructures to enhance the range of a SSO domain. Especially if two companies are working together

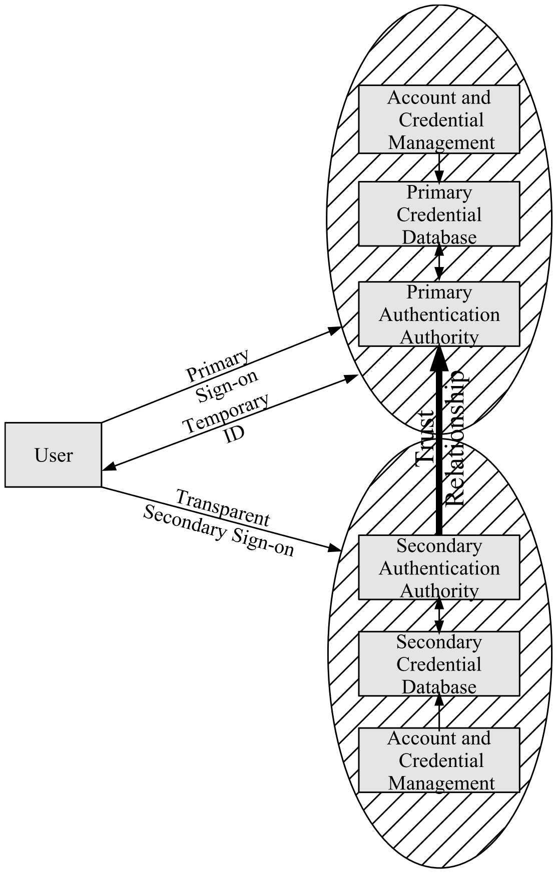
Account and Credential Management
Primary Credential Database
Primary Authentication Authority
Primary Sign-on
Temporary ID
User
Trust Relationship
Transparent Secondary Sign-on
Secondary Authentication Authority
Secondary Credential Database
Account and Credential Management

Figure 2: Service-based SSO

closely, the ability of SSO is highly desirable, but none of the companies would be willing to change their complete authentication infrastructure to match with the infrastructure of the other company. Additionally the trust relationship between two companies cannot be compared with the trust relationship between two authentication authorities inside a single company. Both companies would only be willing to give access to information needed for their joint venture, but this would not include the access to all business secrets. Also is it crucial that both companies have complete control of the access rights they are granting.

Although a federation of trust can be based on a long-term contract, especially in a service-based environment there is a high dynamism that has to be taken into consideration, too. In order to support short-term co-operations a trust relationship has to be installed quickly and it may not exist for a long time. A federation of trust between two authentication authorities can solve these problems by defining mechanisms to enable different security realms to federate using different or alike mechanisms by allowing and brokering trust of identities, attributes, authentication between participating authentication authorities.

3.3 User Roles

In a large network environment it is very common that every user has multiple views on the services he wants to access. These roles are comprising from an administrator's point of view to a user's point of view, e. g. a user is member of two project teams that happen to work on behalf of two different clients. Obviously it is important to allow separate accounting and billing for both clients. As a result a service-oriented environment has to enable a user to work within a specified context. This specified view is a result of the user's role and the intention with which the user is going to use the environment. Such a view is called a *user role*. These user roles do not only allow enhanced accounting and billing, but it is also possible to switch the user role during work, e. g. to gain the rights of administration for the print servers if there has occurred a problem. An additional advantage of allowing users to switch easily between their roles is the improved consciousness that a user is working only with the role having the least but sufficient rights to fulfill a task and does not use the role with the most access rights all the time to minimize effort. Additionally user roles allow the installation of priorities for roles. These priorities can be assigned in certain degrees to any user role, so that e. g. it is possible to define precedence nuances for a frequently used service to allow a certain group of users preferential access, e. g. for printer queues.

4 Model for an Authentication and Authorization Infrastructure

The process of authentication in the model proposed here is closely related to the authentication protocol Kerberos [Opp96], but it is extended to fulfill the needs of a federated SSO environment on the basis of Web services. The Kerberos protocol has the advantage of providing an easy to use basis for a SSO environment utilizing the security token to prove a successful sign-on. Although having advantages, the protocol has certain drawbacks to overcome. Kerberos V4 and V5 both provide the possibility for federated authentication (interrealm authentication), but both versions are not suitable for a highly dynamic network.

Kerberos V4 uses only direct trust between certain authentication domains (in Kerberos terminology: realms). This leads to a maximum of n^2 interrealm security keys for n domains. Not only the number of security keys is huge, but the time to set up a domain needing many communication paths to different domains (Fig. 3).

Kerberos V5 was designed to improve the interrealm communication and introduced a tree-like hierarchy of domains. A domain only needs to store the interrealm security keys from the parent node and the child nodes. If a user needs to access a federated domain, the user has to request a security token from the authentication service in its home domain that is marked valid for remote authentication. This token could then be used to authenticate locally. Although this concept offers several advantages to the interrealm authentication of the V4 protocol, the V5 interrealm authentication is still focused on a static hierarchy of authentication domains.

Additionally both types of Kerberos protocols do not support the ability to provide authorization information for access control. Therefore the extension proposed here does

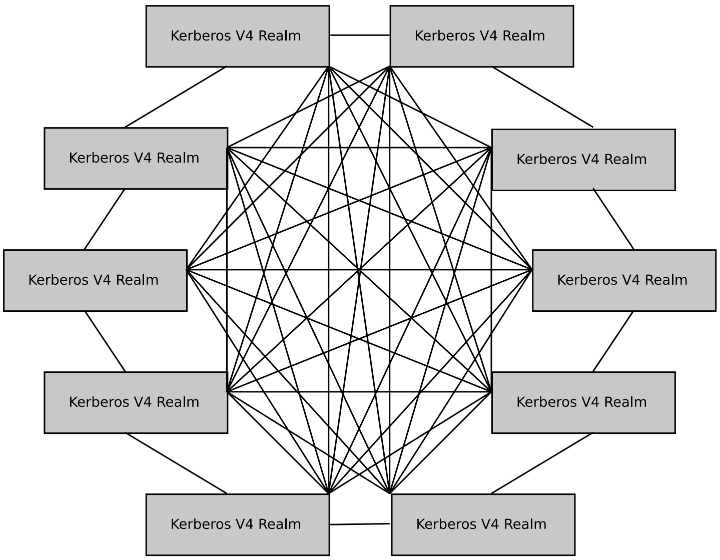

Figure 3: Full Interrealm connectivity with Kerberos V4

provide authorization information stored by a centralized authority. Furthermore it is possible to have a mapping of authorization information regarding different environments and operating systems utilizing user roles. This abstraction of authorization information based on a mapping allows every party belonging to a federation of trust to adjust the authorization of federated users as they seem fit.

4.1 Components of the authentication and authorization infrastructure

An authentication and authorization infrastructure consists of four basic components. The first component is the *user* or *client*. This component has to be integrated in the system of the client and handles any operation needed for authentication and authorization. The *authentication service* – providing everything needed to allow a user to claim an identity and proof that this claim is justified – is the second component. If the user has successfully claimed an identity, he receives a token to prove his identity. This token has to be used with the third component: the *authorization service*. This service provides the user with service tokens allowing him to access services, if he has the privileges to use them. Finally there is the *service component*. The task of this component is to ensure that the user of the service can provide a proper service token, i.e. he is successfully authenticated and has received authorization to access the service, and that the access rights are sufficient to utilize the service functionality requested.

4.2 Authentication in the local authentication domain

The authentication service has to handle two different types of service requests. At first there is a user belonging to the local authentication domain signing on and secondly there is a user from a federated domain signing on. The focus of this section is on the local sign-on process, while the next section explains the sign-on process in a federated domain.

During the authentication process the claimed identity of the requesting user is verified. During the following analysis of local and federated authentication processes, the transfer of encrypted data between communicating parties is needed many times and in order to provide a general notation, the notation defined in [Opp96] is used.

Referring to Table 1, in step 1 the client *C* transmits his username *U*, his domain D_U, the timestamp T and the lifetime L to the authentication service AS in order to claim his identity. The authentication service AS responds in step 2 by sending back the key K and the security token $T_{C,STS}$ which is encrypted with the password K_U of the user U. The term $T_{C,STS} = \{U, D_U, STS, T, L, K\}K_{STS}$ is used to refer to a security token issued by the authentication service AS to be used by the client C to authenticate to a service token server STS to request a service token for a service the client is going to use. The key K_{STS} is a shared secret between the authentication service AS and the service token server STS. Although these two services are regarded as separate services, it is possible to combine them to a single service. The key K is used for encryption when the client C communicates with the service token server STS.

During this authentication process the client never presents his credentials to the authentication service, because of that the authentication service is not able to compare the credentials with the stored credentials of the user. Nonetheless the authentication service replies to the user with a security token that enables the user to authenticate to the service token server, but this reply is encrypted with a key K_U, the password of the identity the user claimed to be. Whether the correct user has claimed that identity or not, only the correct user is able to provide the correct password, i.e. the key for decryption, to take possession of the security token contained in the reply message. Because of that a user's password is never transmitted across the network, but obviously it is important for the user to change his password regularly.

The process of using a service is separated into two consecutive operations. First the client has to request a service token from the service token service (step 3 + 4) and then present that token to the corresponding service to authenticate himself (step 5 + 6). After that the client is allowed to utilize the service.

In step 3 the client C sends his request for a service token to the service token service STS. For this request the client C presents the already acquired security token $T_{C,STS}$, the name S and domain D_S of the service the client wants to use and an authenticator $A_{C,STS} = \{U, D_U, T, L, N\}K$. An authenticator is a data record containing information that allows the receiver to check if the transfered message has been recently generated. The authenticator is encrypted using the session key K known only to the client and the server. The nonce N is a serial number used to prevent the replay of the message. The service token service STS has to store the last serial numbers used to validate the freshness of an authenticator. This communication is implicitly encrypted, because $T_{C,STS}$ and $A_{C,STS}$ are already encrypted.

In step 4 the service token service STS responds to the requesting client C with a service token $T_{C,S} = \{U, D_U, S, D_S, A, T, L, K'\}K_S$, that can only be used by the client C for

Table 1: Communication in the authentication process

Step	Sender	$\rightarrow$	Recipient	: Transmitted data
1	C	$\rightarrow$	AS	$: U, D_U, T, L$
2	AS	$\rightarrow$	C	$: \{K, T_{C,STS}\}K_U$
3	C	$\rightarrow$	STS	$: S, D_S, T_{C,STS}, A_{C,STS}$
4	STS	$\rightarrow$	C	$: \{T_{C,S}, K'\}K$
5	C	$\rightarrow$	S	$: T_{C,S}, A_{C,S}$
6	S	$\rightarrow$	C	$: A_{S,C}$

the service S (encrypted with the secret key K_S, that is shared between the service token service STS and the service S), and a session key K' for the communication between the client C and the service S. The reply is encrypted with the secret key K, which the client received during authentication (step 2). The service token service STS gets access to the secret key K, because the secret key K is encrypted in the security token $T_{C,STS}$ the client presents to authenticate.

After step 4 the first part of the process to use a service is complete. The client has acquired everything needed to access the service. In step 5 the client C sends the service token $T_{C,S}$ and the authenticator $A_{C,S} = \{U, D_U, T, L, N\}K'$ to the service S.

In step 6 the service S responds with the authenticator $A_{S,C} = \{S, D_S, T, L, N - 1\}K'$ to the client C. The nonce N received by the service S is decreased by 1 and returned in step 6. This allows the service to ensure the freshness of the message by comparing the nonce with the already received nonces and it allows the client to check the correctness of the reply by the service, who had to successfully decrypt the authenticator $A_{C,S}$ received from the client to present the correctly decreased nonce in the authenticator $A_{S,C}$. This decreased nonce proves that the service is the expected service and not a malicious service impersonating the intended service. After that the client and the service are mutually authenticated and the client can start to use the service.

4.3 Authentication in a federated authentication domain

In this section the authentication process of a user belonging to a federated authentication domain is explained in detail. This process only differs from the authentication process explained in the previous section in its first two steps. After that the user has proven his identity and can provide the security token ($T_{C,STS}$).

The first two steps change transparently for the user. From the client's point of view in step 1 the client C transmits his username U, his domain D_U, the timestamp T and the lifetime L to the authentication service AS in order to claim his identity. The authentication service AS responds in step 2 by sending back the key K and the security token $T_{C,STS}$ which is encrypted with the password K_U of the user U.

The difference in the authentication process has its origin in the usage of the user's password to encrypt the response of the authentication server. This password cannot be found in the local credential database, because the user does not belong to the local authentication domain. Therefore the data has to be requested from the federated authentication service as is explained in the following paragraph while the communication is shown in Table 2.

Table 2: Authentication utilizing direct trust

Step	Sender	$\rightarrow$	Recipient	: Transmitted data
1	C	$\rightarrow$	AS	$: U, D_U, T, L$
$1'$	AS	$\rightarrow$	ASR	$: \{U, D_U, T, L\}K_{fed}$
$2'$	ASR	$\rightarrow$	AS	
			$: \{K, T_{C,STS}, \{K, T_{C,STS}\}K_U\}K_{fed}$	
2	AS	$\rightarrow$	C	$: \{K, T_{C,STS}\}K_U$

The local authentication service AS transmits the sign-on request encrypted with the secret key K_{fed} to the remote authentication service ASR handling the authentication for the domain the client C belongs to (step 1'). The secret key K_{fed} is specified during the establishment of the Federation of Trust. In step 2' the authentication service ASR supplies the authentication service AS with the requested security token $T_{C,STS}$. The security token $T_{C,STS} = \{U, D_U, STS, T, L, K\}K_{fed}$ is encrypted with K_{fed} instead of K_{STS}, because the local authentication service AS needs to access the token. This security token $T_{C,STS}$ is returned to the client, who has to provide the correct key K_U to decrypt the data received and extract the secret key K that has to be used for the communication with the local service token service STS. After that the local service token service STS can handle any further requests of the client C by himself.

If the authentication request is from a user that does not belong to a domain with which the local domain has a direct trust relationship, the protocol must be extended in order to bridge the intermediary domains (refer to Table 3).

The extension of the protocol can be seen in the steps $1', \ldots, 1^{(n)}$ and $2', \ldots, 2^{(n)}$. In order to indicate that there can be more than one intermediary authentication service $ASI_i, i = 1, .., n$, these steps are separated by dots. In order to bridge these intermediary communication nodes, the data transmitted is encrypted with the corresponding secret key K_{fed}, that these intermediary communication nodes share with their predecessors respectively successors. All intermediary communication nodes have a direct trust relationship with their predecessors respectively successors which is the basis of an indirect trust relationship, i.e. trust is federated across the network of authentication domains.

The model does not define how a local authentication service can find the federated authentication service that can process the sign-on request, but if an indirect trust relationship exists and because a successful communication does only have a finite number of intermediaries, it is possible to find a path in the network graph. Algorithms to find a routing table for efficient communication can be found in [Tan96].

Table 3: Authentication utilizing indirect trust

Step	Sender	$\rightarrow$	Recipient	: Transmitted data
1	C	$\rightarrow$	AS	$: U, D_U, T, L$
$1'$	AS	$\rightarrow$	ASI_1	$: \{U, D_U, T, L\}K_{fed}$
$\vdots$		$\vdots$		
$1^{(n)}$	ASI_n	$\rightarrow$	ASR	$: \{U, D_U, T, L\}K_{fed}$
$2^{(n)}$	ASR	$\rightarrow$	ASI_n	
				$: \{K, T_{C,STS}, \{K, T_{C,STS}\}K_U\}K_{fed}$
$\vdots$		$\vdots$		
$2'$	ASI_1	$\rightarrow$	AS	
				$: \{K, T_{C,STS}, \{K, T_{C,STS}\}K_U\}K_{fed}$
2	AS	$\rightarrow$	C	$: \{K, T_{C,STS}\}K_U$

After the security token has been sent to the requesting user, the authentication process is finished. Any further requests of this user to receive any service tokens needed can then be sent directly to the local service token service STS, that handles these requests without further interaction with the remote authentication service ASR, but may need to contact the remote service token service to request authorization information of the user.

4.4 User Roles as a Basis for Authorization

In addition to a successful authentication process the authorization associated with the specific user is of great importance. This authorization information provided by the authorization service defines the type of access a user has during a service request. This section describes how the authorization information is organized and assigned to the user.

The authorization of a user is gained by user roles assigned to a user. These user roles specify a top-level view, e. g. administrator, member of a project team or guest. One of the roles assigned to a user has to be selected in order to define the context the user is working in.

A user role is refined by user permissions. A user permission defines the access rights to a service or a group of services. As a result a user role typically consists of multiple user permissions.

This concept allows the easy assignment of access rights to users, but typically there are small adjustments needed to give or revoke special access rights for certain users without changing the access rights of the whole user role. For example with these special adjustment it is possible to give / revoke the access to a certain service for just one user although all other users in this role cannot / can access this service. It is important to mention that this special adjustment must not influence all roles assigned to a user, e. g.

if a user lost the right to access a service in one role, he may still be able to access this service with another role if this role includes the rights to access this service.

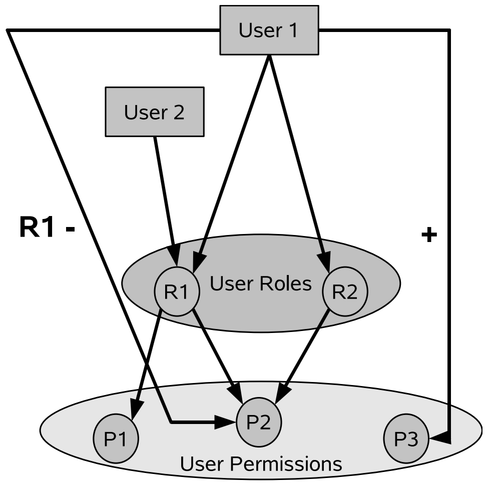

Figure 4: Association between user roles and user permissions

Fig. 4 shows an example of the association between user roles and user permissions. In this example the user roles *R1* and *R2* are assigned to *User1*, while *User2* only belongs to *R1*. While *User2* has the permissions *P1* and *P2* assigned to role *R1*, *User1* has additional permission assignments. *User1* received the additional permission *P3* and lost permission *P2* in the context of role *R1*. As a result *User1* is assigned to the same user role as *User2*, but does not have the same access rights, because his permissions have been adjusted so that he is able to use the services of permission *P1* and *P3*, but not the services of *P2*. This does not include an adjustment of permissions of role *R2* with respect to permission *P2*, but *User1* is still allowed to use the services of permission *P3*, because the granted permission was not restricted to any user role, in addition to the services of permission *P2*, which is the default configuration.

This combination of user roles, user permissions, and permission assignment or revocation builds a flexible access control mechanism. In the concrete VoIP environment of the Venice project where this authentication and authorization infrastructure is being used, these permissions can be identified with the permissions to use the system for private calls, international calls or certain supplementary services.

4.5 Authorization

During the authentication process the authorization information A is received by the service S as part of the token $T_{C,S} = \{U, D_U, S, D_S, A, T, L, K'\}K_S$. This information A includes everything the service needs to decide whether or not a client has access to the requested functionality of the service. Additionally the authorization information cannot be changed by the user to obtain extra privileges, because the whole token $T_{C,S}$ is encrypted with a shared secret K_S of the service token service and the service.

As there are many different kinds of services, e. g. print service or ftp service, with different types of restrictions for their functionalities, e. g. a user is only allowed to print in black and white or a user may only download one file simultaneously per session from the ftp service, the appearance of the authorization information cannot have universal definition. Therefore the appearance of the authorization information is not defined at all. Only the service itself can generate and understand the authorization information of a user. This is sufficient, because the service token service has only the task to provide the authorization information in a centralized way for its security domain, but does not need to understand the provided information.

As the authorization information does not have one single shape, it is possible to introduce more sophisticated access rights. The access rights can now not only consist of a simple "yes" or "no", it is now possible to define access rights in any way the service may support restrictions. Especially services that use a concept of access rights that is not very common can be supported.

Beyond the access rights resulting from a single user permission, multiple user permissions can be assigned to a user, therefore the service token service has to provide a service token, that contains all authorization information for every related user permission, because the service token service is not able to decide which user permission can provide enough authorization to fulfill the desired request of the user. Additionally it is possible that user permissions provide disjoint authorization for the service, so that the request of the user can only be fulfilled by using more than one user permission. Because of that the appearance of the authorization information is: $A = \{A_{perm_1}, \ldots, A_{perm_N}\}$ where N is the number of user permissions assigned to the user in the context of the service the user is going to use.

4.6 Delegation

An additional feature of authorization, that can be very useful, is delegation. Delegation means that a user can give some of his access rights (or all of them) to another user for some period of time. In this model delegation is realized by the ability of the service token service to store a delegation order of the user who wants to transfer some of his access rights to another user. If the other user is requesting a service token not only his own user permissions are taken into consideration to generate the authorization information included in the service token, but also the transfered access rights of the delegating user. In order to delegate rights to another user, the originating user must have the right to delegate, because this option can have serious impact on domain security if it is not used reasonably.

4.7 Mapping of user roles

The authorization information of a local user is provided by the service token service by issuing an encrypted service token. If the user does not belong to the local domain, additional problems arise.

The service token service has to issue a service token without access to the authorization information of the user, because he belongs to a federated domain. Additionally there is the problem, that the service token service can not utilize this information from a federated domain as the remote operating system might vary from the local operating system, i.e. the access control information is not comparable with the access control information in the local domain, and the user permission assigned to the user does not exist in the local domain, i.e. no access rights can be identified with this user.

The solution proposed here uses a mapping of user roles. This mapping assigns a user role of a federated domain to a user role in the local domain. The mapping only works in one direction: from the federated domain to the local domain.

The advantages of this solution are as follows:

- A user impersonating a user role of a federated domain not being mapped, does only acquire access privileges on a guest level in the local domain.
- The access rights of federated users in the local domain are set and controlled by the local domain.
- Because of the unidirectional mapping each domain of a federation can control the access rights autonomously.

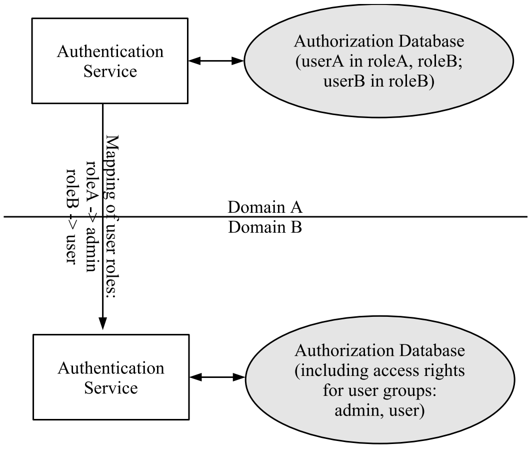
Authentication Service
Authorization Database (userA in roleA, roleB; userB in roleB)
Mapping of user roles:
roleA -> admin
roleB -> user
Domain A
Domain B
Authentication Service
Authorization Database (including access rights for user groups: admin, user)

Figure 5: Mapping of user roles

Fig. 5 shows an example of a user role mapping. The *domainA* and *domainB* have a direct trust connection and a mapping of user roles has been defined from *domainA* to *domainB*. The user role *roleA* is mapped to the user role *admin* and user role *roleB* is mapped to the user role *user*. The *userA* can impersonate the user roles *roleA* and *roleB*, while *userB* can only impersonate the user role *roleB*. Because of this mapping the *userA* can impersonate the user role *admin* in *domainB*, while *userB* has only a *user* status in *domainB*. Obviously the users from *domainA* will get access to *domainB* without the need to transfer the native access control information from *domainA* to *domainB*. Instead the native access control information of *domainB* can be used. It is important to mention that a user may gain access to a type of service the user does not have in his home domain and may loose access to a type of service the user has in his home domain because of the configuration decisions the local administrator has made while defining the user role mapping.

4.8 Rejection of trust

Although federated access to services is highly desirable, the model of an authentication and authorization infrastructure described so far may lead to some unwanted access. There are situations conceivable where only access of users from security domains having a direct trust relationship is desired. In order to prevent indirect trust it would be possible to define a new federation of trust including only the security domains having a direct trust relationship. This approach results in distributed responsibility among all participating security domains and cannot be controlled by every domain autonomously.

Therefore the rejection of trust has been included in the model of an authentication and authorization infrastructure introduced in this paper. Rejection of trust is the possibility to define a list of security domains that must not at all have access to the local security domain. This prevents users of unwanted domains from gaining access to the local security domain by an indirect trust relationship via a security domain that maintains a direct trust relationship with the local domain.

5 Conclusion

This paper described the model of an role-based authentication and authorization infrastructure supporting single sign-on and federations of trust on the basis of Web services. In order to reach this aim, the focus has been at first on the design of the authentication protocol closely related and extending the well understood Kerberos protocol. The ideas of this protocol have been extended with methods to transport authorization information and to delegate access priviliges.

In order to support federation of trust, the protocol allows users from remote domains to access local services by federating authentication and authorization to trusted domains. The authorization information is transported to the local domain by the definition of user role mappings allowing to gain access to local services without explicit specification of access rights for users from trusted domains. This concept thereby circumvents the problem of incomparable authorization information and non-existing user roles by assigning local access rights to federated users.

Support of delegation allows a user to receive additional access rights quickly on a temporary basis without much administrative effort.

The model of an role-based authentication and authorization infrastructure proposed here, has been implemented as a prototype for the Venice project using the Java programming language with the SOAP implementation Apache Axis 1.1. The server and service modules of the infrastructure utilize Tomcat 5.0 as a servlet container. Authentication and authorization information are stored in a PostgreSQL database.

References

[ABCDL04] Steve Anderson, Jeff Bohren, Marc Chanliau, and Giovanni Della-Libera. Web Services Trust Language (WS-Trust), Version 1.1, 2004. http://www-106.ibm.com/developerworks/library/specification/ws-trust/.

[ADLHH02] Bob Atkinson, Giovanni Della-Libera, Satoshi Hada, and Maryann Hondo. Web Services Security (WS-Security), Version 1.0, 2002. http://www-106.ibm.com/developerworks/webservices/library/ws-secure/.

[BDLD$^+$03] Siddharth Bajaj, Giovanni Della-Libera, Brendan Dixon, Mike Dusche, and Matt Hur. Web Services Federation Language (WS-Federation), 2003. http://www-106.ibm.com/developerworks/webservices/library/ws-fed/.

[CKPM03] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, 2003. http://www.oasis-open.org/committees/documents.php?wg_abbrev=security.

[dC02] Jan de Clercq. Single sign-on architectures. In *Proceedings of Infrastructure Security, International Conference, InfraSec (Bristol, UK)*, pages 40–58, 10 2002.

[HM03] John Hughes and Eve Maler. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1, 2003. http://www.oasis-open.org/committees/documents.php?wg_abbrev=security.

[HMM04a] Markus Hillenbrand, Paul Müller, and Kristian Mihajloski. A Software Deployment Service for Autonomous Computing Environments. In *Proceedings of IAWTIC 2004 (Gold Coast, Australia)*, 7 2004.

[HMM04b] Markus Hillenbrand, Paul Müller, and Harald Müller. Voice over IP als Web Service (Voice over IP as Web Service). In *Proceedings of 18. DFN Arbeitstagung (Düsseldorf, Germany)*, 6 2004.

[HZ04] Markus Hillenbrand and Ge Zhang. A Web Services Based Framework for Voice over IP. In *Proceedings of the 30th Euromico Conference 2004, Rennes, France*, 8,9 2004.

[Kem04] John Kemp. Liberty ID-WSF – a Web Services Framework, 2004. http://www.projectliberty.org/resources/whitepapers/Liberty_ID-WSF_Web_Services_Framework.pdf.

[Opp96] Rolf Opplinger. *Authentication Systems for Secure Networks*. Artech House, Inc., 1996.

[Tan96] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall International, 1996.

[ZH04] Ge Zhang and Markus Hillenbrand. Implementing SIP and H.323 Signaling as Web Services. In *Proceedings of the 30th Euromico Conference 2004 (Rennes, France)*, 8,9 2004.

[ZHM05] Ge Zhang, Markus Hillenbrand, and Paul Müller. Facilitating the Interoperability among Different VoIP Protocols with VoIP Web Services. In *Proceedings of the 1st International Conference on Distributed Frameworks for Multimedia Applications DFMA (Besançon, France)*, 2 2005.