Transparent Security for Collaborative Environments

Eva Hladká, Daniel Kouřil, Michal Procházka, Luděk Matyska, Petr Holub
CESNET z.s.p.o.,
Zikova 4, 160 00 Praha 6,
Czech Republic
{first.last}@cesnet.cz

***Abstract*—Current collaborative tools are often not able to profit from existing systems for user management. It is therefore necessary for collaborative systems to administrate their users using their own solutions, which may not be adequate in terms of scalability or security. Many users may also experience problems working with authentication credentials (e.g. digital certificates) employed by collaborative systems. In this paper, we propose a general framework to provide easy-to-use yet secure access to collaborative systems, which offers a general middleware layer to accommodate various types of collaborative tools. The framework utilizes the emerging model of federations, which allows to provide a user-friendly means of logging in to a collaborative system as well as a solid basis for specifying access control policies. The framework handles all security aspects in a transparent way without requiring the users to perform complicated tasks. Using user attributes maintained in the federation, it is also possible to implement efficient and dynamic group management of the collaborating users.**

I. Introduction

The need to efficiently collaborate is very crucial, especially for people distributed accross large geographical distances or organizational boundaries. Nowadays we have communication networks allowing us to overcome distances but we are still a long way from a full-fledged virtual collaborative environment. In recent years, most effort in this area has been targeted at different aspects of human–machine and human–human interactions, with the underlying middleware receiving little attention. However, the middleware has a decisive role in acceptance (or rejection) of the system, because difficulties in infrastructure set-up can preclude its use. In this paper, we present a federation-based approach to security middleware, which allows practically transparent set-up and use of a security layer beneath the collaborative environment. This work is based on our long experience with supporting collaborative environments for differing user groups with a rather conservative approach towards complex computerized solutions.

Let's start with a broad definition of a collaborative environment. The basic definition of such an environment is a set of tools supporting work processes and interactions among users. We will, however, use a more detailed definition, namely that a collaborative environment is a set of software/hardware system tools allowing people to communicate and share applications in real time synchronously with required security. This definition is still rather general, but it reflects the complexity of collaborative environments. From a technical point of view an ideal instance of a collaborative environment can be divided into two layers. The bottom layer is the middleware and the top layer consists of concrete client tools. The communication tools distinguish different collaborative environments and are specific to particular collaborating groups. The middleware, on the other hand, provides a functionality required by all collaborative systems and its principles (or even implementations) are shared among different collaborative environments.

The primary roles of the middleware in a collaborative environment is to realize multipoint connectivity, to provide user and group management and to guarantee security, i.e., authorization, authentication, and at least some basic accounting. Most of the functionality of middleware services should be transparent for users.

Multipoint connectivity would ideally be delegated to the underlying network. However, as multicast is not a reliable network service (it uses the unreliable UDP protocol) and is still not widely available, the collaborative systems usually provide their own overlay network to guarantee the multipoint connectivity. Since a full mesh, where all partners directly communicate with each other, is very inefficient in terms of consumed network capacity (and also client processing power), usually some star-like overlay is created. In the H.323 world MCUs (Multipoint Control Units) are used. Other systems use other (hardware or software) reflectors as well.

The security aspects may also be addressed at the network level. In fact, the network middleware is primarily interested in providing a security framework for applications running on top of the network. The major disadvantage of the classical network middleware approach is its lack of user friendliness—its use expects a rather high level of technical knowledge and experience. This is even truer for user and group management, where user orientation is necessary for acceptance of the system.

After several years of sometimes painful experience with supporting collaborative environments for mostly conservative user groups, we focused our research on their middleware, emphasizing transparency and user friendliness (the user-centric approach). Our main goal is to design and implement a middleware which will ensure easy set-up of collaborative sessions supporting secure communication among users. The middleware should also provide easy membership management functionality. Our proposed solution is based on a federative approach, which removes most of the problems encountered by more classical security and group management solutions.

In the next section we compare the most frequently used collaborative systems from this point of view.

II. Related Work

Some form of authentication and authorization, even though sometimes fairly implicit, is present in the vast majority of collaborative systems. In this section, we will discuss the nearest competitors to our approach—AccessGrid—and also a few other widespread systems with less sophisticated authentication and authorization support: Skype, VRVS, VRVS EVO, and WebEx.

Skype [1] has become a very attractive option for many users, because of its capabilities to penetrate firewalls and work behind NATs. Though the internals of its security model are proprietary and very obscure [2], [3], the basic approach is pretty straightforward: the users register with the Skype server in order to get their account protected using a password. The authorization is handled intuitively in the sense of common telephony systems: the user calls another user (one or more depending if the calls are limited to point to point or if there is a multipoint call), and the user being called authorizes or refuses the calling party. This model is sufficient here as the Skype does not assume complex multi-party collaborative sessions.

Another well-known and widely used collaborative environment, created originally for the high-energy physics community, is the Virtual Room Videoconferencing System (VRVS)[1]. The successor of VRVS, called Enabling Virtual Organizations (EVO)[2], is based on the self-organization of a system of reflectors. Both use authentication based on user registration with the VRVS system, with a password as a credential. Users are organized in virtual rooms, the rooms are either public (i.e., available for anybody who is registered with the VRVS) or protected by a password shared among the room participants (the so called Meeting password in EVO).

WebEx[3] is a commercial portal-based system. It is also provided as a service, therefore users communicate through the service provider's servers. A simpler version of WebEx does not support complete group communication, as only one video and four audio streams are permitted simultaneously. Each group needs a moderator to assign video and audio channels (in the case of larger groups), the group is protected by a shared password. No client software is installed on users' machines, only a web browser is required.

On the other side of the collaborative tools spectrum, there is an experimental open extensible system called AccessGrid (AG) [4], [5]. Since its version 2, it features service-oriented architecture for collaborative environments based on Grid services [6]. AccessGrid uses X.509 certificates for authentication purposes. It supports both high-quality certificates from trusted CAs and also the so-called "anonymous certificates" from AccessGrid CA. The "anonymous certificates" were introduced after the first releases as it turned out that ordinary AG users were having serious problems with obtaining certificates from a trusted CA, thus being discouraged from using AccessGrid. The authorization is implemented as part of a Venue server using a role-based approach [7]: each user may have a number of roles, thus enabling him or her to take certain actions. The roles are assigned by the Venue server administrator(s).

In our previous work [8] we developed a collaborative environment based on MBone Tools, our own modular user-empowered UDP packet reflector and security layer based on a VPN solution. Users make secure connection to the VPN server and only data related to the collaborative session are transferred through the VPN tunnel to the server. Authentication is based on X.509 certificates provided for VPN authentication, and authorization is implemented using a list of allowed DNs listed at the VPN server. No advanced group management is supported, a group is simply a list of DNs.

III. Current models for management of collaborative groups

All the collaborative systems just described use a centralized service for user management. The service stores information about user accounts and credentials. In most cases, the credentials are username and password, in others—the AG and the VPN solution—they are based on the user's X.509 certificate [9], being part of a Public Key Infrastructure (PKI) [10]. The username/password credentials are usually unique for the particular collaborative environment—users are discouraged from using the same password for several different services. With the X.509 credentials, users must remember a passphrase that protects the private key file; sometimes users are encouraged to have separate X.509 credentials for these services, to lower the risk of compromising them. In either case, users are required to maintain and/or remember new specific credentials.

Group management is usually built on the top of individual user credentials. The group management defines who is allowed to participate in a particular collaborative session. Again, most systems impose their own group management system (if any).

A. Common drawbacks in used models

All the collaborative environments mentioned require user registration and some type of custom user credentials for access to the collaborative services. Skype, AccessGrid, VRVS and our VPN solution require installation by the users. Both the registration and the installation makes the collaborative environment hard to use by an ordinary user. Deployment in most institutions brings problems with firewalls, NATs, restriction policies and/or non-cooperative IT staff.

User management in current systems presents a large overhead and also poses a potential security risk. Since the collaborative environments mentioned employ their own user management systems they require users to register before they join any service in the collaboration. The registration consists of filling in a form in which users give information about themselves, but the registration system can do very little to verify that the information is correct. Also there is only a very

[1] http://www.vrvs.org/
[2] http://evo.caltech.edu/
[3] http://www.webex.com/

limited way to verify that the information remains current, leaving its maintenance to the users themselves. This arrangement could lead to unauthorized access to information or its leakage. Systems based on an existing PKI could leverage the proper identification of the users, but more information is usually needed to make appropriate access control decisions. Also, getting and using PKI credentials often present a high barrier for ordinary users.

As mentioned before, most collaborative systems utilize a username and password to authenticate users. Since the systems are not integrated with other environments, these credentials are unique for the systems and have to be distributed to the end users. Distribution of these credentials is not always properly secured, so they may be intercepted during the transmission by an attacker who could use the credentials to gain unauthorized access to the collaboration.

Collaborating users often need to establish a group to work together on a particular problem. It is highly desirable for the group maintainer to be able to assign a policy specifying who is a member of the group. Due to lack of other information about users, current systems usually only allow the group administrator to specify an exact list of authorized usernames or use a shared password. Both these methods have significant flaws. In the former case it is necessary to know the list of users in advance, which makes group management inflexible. The latter approach requires distribution of the password (which again might be subject to attack) and does not provide any easy method for revoking authorization once it has been distributed. These issues make it impossible in practice to create collaborative groups dynamically on demand in a controlled and secure way.

It is possible to address most of the issues described in this section using the federation model. In the rest of the paper we propose a general framework built upon an existing federation and describe how it allows us to cope with the issues.

IV. The federation model

A federation is an infrastructure connecting user management systems from different institutions to provide standardized access to information about users maintained by their systems. Federations provide a bus layer to which systems for user management and end applications can connect and share authentication and authorization data. Every organization participating in a federation manages its users by a local user management system. An *Identity Provider* (IdP) service is built on the top of each local user management system, providing a standardized interface to access authentication information and other attributes about the users. Any party in the federation can get this information by calling the IdP service using a standardized protocol. End services (*Service Providers*—SP) are able to process the data returned by the user's home IdP and use them to make access control decisions. Before users are allowed to use a service, they have to present a set of *attributes* issued by their home IdP. These attributes are provided to users or to a service working on their behalf upon proper authentication of the user with the IdP.

The major advantage of using the federation model lies in the fact that users authenticate to arbitrary services with their home institution's credentials (which may be a username and password, a digital certificate, a hardware token, or something else.). Every SP in the federation can use this mechanism transparently from the user's point of view. There is no need to introduce new credentials for every new service or to synchronize existing credentials (like passwords) among different services. Having no additional credentials also means there is no need to distribute them among the users. Such an arrangement not only eases credential management but also makes it more secure, as users are only required to maintain one piece of authentication information. Similar functionality is offered by PKI. However, it is too difficult and time consuming for most users, especially if their home institution does not already have an extensive network of registration authorities and properly trained user support. The federation model is undoubtedly more acceptable to the users, as it is not tied to any particular authentication method and institutions can decide the most appropriate method for their users.

Upon proper user authentication, the IdP provides a set of attributes that represent additional information about the user. The attributes are very often encoded using the Security Assertion Markup Language (SAML) [11]. In this way the home institution provides information that can be used, e.g. for specification of a group of users without explicitly naming them. For example, it is possible to create a collaborative session for students enrolled on particular course at different institutions (provided that information about the courses a person is enroled on is made available as part of the attributes). The whole communication can be logged for future auditing, so the session administrator can learn who participated in a session without needing to be given this information in advance, in order to authorize access.

Enhanced privacy is a potential side effect of the abovementioned use of attributes. The IdP could provide only attributes, not a precise user identity (i.e., it could provide the information that a person is enrolled on course identified as CS102 without revealing his or her name or other unique identification). The attributes are sufficient for a service to make an authorization decision, however the precision of audit trace is lost (while the privacy of the user is enhanced). In the event of abuse, the home institution is still able to identify the user from their local logs. This approach is interesting if we do not want to reveal the individual user's identity to a collaborative session administrator, e.g., in some semi-anonymous survey.

V. Federation based collaborative environment management framework

We have designed a framework making use of the federation model, which allows us to set up and maintain a collaborative group in a secure and rapid manner. The framework resides in the middleware layer, providing a security infrastructure for any collaborative tools used by the end users. The framework is primarily aimed at our solution based on VPN tunnels and

MBone tools, but is general enough to be used with other collaborative tools, too.

The framework provides a solution to manage users and groups and allows users to authenticate using the standard authentication credentials that they use to access services at their home institution. The framework removes any need to independently maintain user records or to distribute new credentials across the collaborating community. It operates on the middleware level, making it possible to move the functionality for the user management and authentication and authorization from the application to the bottom layers, reducing the complexity of the applications.

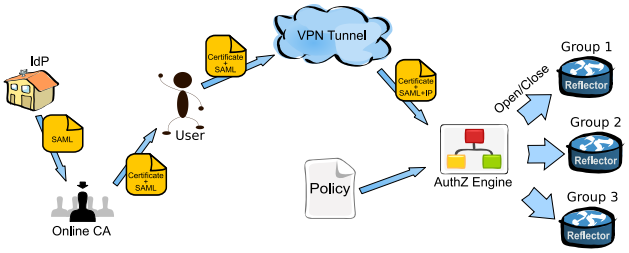

Fig. 1. Schema of the framework

An overall schema of the framework is depicted in Fig. 1. The framework addresses two very crucial aspects necessary to maintain a collaborative environment: access control of users and group management, both based on a strong authentication. Authentication must not add additional barriers for the users, while being as general as possible. A simple approach to authentication in such an environment would be to let users authenticate directly against the collaborative server using his or her home password. By pretending to be the users, the collaborative server could verify the credentials against the users' home IdP. This approach would be simple for the users as they would not need any additional tools on their side. On the other hand it suffers from severe shortcomings and weaknesses. Not all institutions use a username/password pair to authenticate their users, also adapting the collaborative server to support new authentication method is usually a complicated or even impossible task. Moreover, this approach requires the collaborative server to verify the passwords, which requires a much higher level of trust in the server and its administrators, because a malicious or compromised server could easily collect users' passwords and misuse them later. In view of these drawbacks we decided to introduce a new concept of *logging into the collaborative environment*, which makes the authentication step general without imposing additional barriers to the users. The logging phase is invoked explicitly by the user at the beginning of a work session and is responsible for primary authentication. After a user successfully logs in, he or she receives a short-term credential that can be used to access the collaborative environment. Management and usage of the credential is hidden to the user and is performed by our framework.

A. Logging into a collaboration

We have chosen PKI as the primary authentication mechanism for accessing a collaborative system. PKI is widely used and supported by a rich variety of tools and applications and it is a native authentication mechanism for AccessGrid and our VPN/MBone solution. We do not expect users to already have a certificate from some trusted certification authority (CA) or even any experience of the certificate use. Instead we make the certificate generation be part of the explicit log-in step. In order to allow easy access to certificates, we plan to establish a federative CA that works as a service provider in an existing federation, which issues certificates on the fly to users who can authenticate using their home credentials. Apart from the usual information stored in an ordinary public-key certificate, the certificates issued by the federative CA also contain a set of attributes assigned to the user by the identity provider. These attributes can be later used to perform access control decisions by the server, while the certificate provides a suitable container to transport them between the user and the server. We are preparing user tools that can be used to obtain such a certificate in a user-friendly manner.

Once received from the CA, the certificate, along with the corresponding private key, is stored on a local disk so that they could be taken by the application that arranges connection to a collaborative service. The private key is stored unencrypted and only secured by the file system permissions, so that only its owner and his or her applications are able to read the file containing the key. In order to achieve a single sign-on functionality, yet to retain a reasonable level of security, we suppose the certificates will only be short-lived, with a typical lifetime of one day. Thus, the user must log in every day to get a new valid certificate. According to our experience with other environments, such a precaution is usually acceptable for users provided that they can work for the full day without interruption.

The PKI credential (i.e., the public-key certificate and the corresponding private key) are used to establish an authenticated channel with the collaborative service. If the collaborative tools support PKI authentication they can just be pointed to the files storing the credentials and communication goes on as usual. For tools not supporting PKI we propose to build a dedicated tunnel between the client and server that encapsulates the whole application protocol. For the latter approach to be useful, it is also necessary to adapt the server and its current authentication routines. Since it is not acceptable to change the client tools, they would still need to pass on a username/password to the server. However if PKI authentication is in place, this is not necessary and the server can just skip the verification and rely on the pre-authentication performed by PKI only. The only potential drawback of this solution is the possibility to undermine some specific network-related functionality of the collaborative environment used, like passing through a NAT or firewall. Such functionality can be gradually added to our solution in the future when users demand it.

B. Access control

In order to authenticate a client using his or her PKI credential and make access-control decision the collaborative server has to perform standard operations to check validity of the certificate, i.e., to check that it is signed by a trusted CA, its lifetime has not expired, etc. Due to the short lifetime of the certificates we do not plan to check revocation information, but such checks could be easily enabled in the future if a proper mechanism for timely distribution of revocation data were developed. After authentication finishes, the server passes the client's attributes from the certificate along with the client's identity to the authorization engine. The engine evaluates the input data, checking whether it matches the requirements of the collaborative system. Being based on the user's attributes, the access control policy can be very flexible, e.g., a collaborative group could be defined that is only open for users possessing certain attributes in particular institutions. For example, a group of physicians from different medical schools could establish a collaboration to discuss a particular case of a patient they are currently treating. As privacy is obviously an important issue in such a case, only the doctors involved in the case should be allowed to join. Using the relevant attributes it is possible to specify a policy that opens the group for such users even if identity of individual users is not known in advance.

By means of the proposed framework it is possible to establish a collaborative group very quickly without additional overhead. Users can join a collaborative environment immediately after it is set up using their home authentication credentials only. So no additional user management is required on the part of the collaboration administrators. Using the access control engine the administrators can define policies prescribed for particular groups they manage. Being based on the federation attributes, the policies may not be linked to the users' identities.

C. Current implementation

In this section we describe the current status of implementation of the framework and present tools that have been chosen to achieve the needed functionality. The implementation is based on our VPN solution, to which we added a service to obtain a certificate to log in and attribute-based access control. Currently we are finishing initial implementation.

The key assumption is that the framework is deployed in an environment where a functional federation is already available. Our development is being performed within `czTestFed`[4], which provides a federation testbed for the academic community in the Czech Republic. It is based on the Shibboleth middleware [12], which is the most widely used solution for federations currently, so our results will be directly applicable for other Shibboleth-based federations, too. It would be even possible to use the framework in federations that are built upon different middleware sets, such as OpenID [13], Microsoft InfoCard [14], or Liberty [15], provided they allow to build an on-line CA server based on the same concepts. Then, the only change to our framework would be adaptations to the server authorization engine to support the format that the federation middleware uses to transport attributes (if SAML is not used already).

To support the logging phase we employed the CA from the GridShib project [16]. The CA operates as a service provider in a federation, allowing users to receive their public key certificates upon authenticating using their home credentials. The GridShib CA is implemented as a web service reachable by common web browsers that support Java applets, which is convenient for the users. The Subject name for the client certificate is constructed using the client's principal name as specified in the attributes returned by his or her identity provider. A SAML assertion containing the full set of attributes is stored as an X.509 extension in the certificate, so it can be used later by other services accessed by the client. If privacy were a concern and people did not want to reveal their names in the certificate, it would be easily possible to construct the Subject name using a meaningless unique identifier and omit all personal information from the attributes in the certificate.

The certificate returned by the CA is stored on a local disk along with the corresponding private key. The credentials can be later used by all applications supporting PKI authentication. We modified the default configuration of our VPN clients so that they look at the location where the PKI credentials are stored by default. After the initial log-in (authentication, creation of the certificate) the user can initiate a VPN channel without any more authentication or configuration steps.

Using the federative CA, it is very straightforward for the ordinary user to get valid certificates using his or her home credentials. There is no need to distribute or explicitly generate and store new credentials. The federation certificate is simple to obtain but secure enough to be acceptable by the collaborative environment (as well as other services). The certificates offer a sufficient level of trust as they are not issued anonymously but their owner can be tracked down to her home institution when necessary. Since the certificates are only short-lived, private keys can be stored unencrypted, allowing the user's applications access to them without the user's assistance, which provides a true single sign-on environment.

On the server side of our solution we use an OpenVPN server [17] and a set of reflectors available from the VPNs managed by the server. The OpenVPN server allows us to specify various plugins that can be invoked to perform custom actions on different events. In order to perform access control decisions, we plan to use an "on-connect" plugin that gets called upon establishing a connection from a client, which ensures calling the authorization engine. Since the standard implementation of the OpenVPN server only provides access to the subject name from the client certificate and not the full certificate we were not able to read the SAML assertion (the attributes) from the certificate. Therefore we are working on modifications to the part of the OpenVPN server that invokes the plugin to pass a full certificate content so the assertion embedded in the certificate is available to the plugin.

[4]https://cztestfed.feld.cvut.cz/

The plugin will read the assertion and pass it to the authorization engine along with the IP address of the client endpoint of the VPN channel that was assigned to the client. Having evaluated the policy, the engine constructs a list of groups the client currently has access to and grants the client access to the appropriate reflectors. The Reflector Administrator Protocol [18] will be used to contact the reflectors and pass information about the client and his or her IP address that should be allowed to access. Once the engine library has finished evaluating the policy the plugin will return control to the master OpenVPN process to finish establishment of the VPN so the client could start communicating. When the client closes the connection an "on-close" plugin will be invoked, which ensures the IP address is unregistered from all the reflectors.

We have not yet finished an exact specification of the format of the access control policy. The current design uses an XML format, mapping a name of a group to a set of attributes (SAML AttributeStatements) that are required to join the group.

VI. Conclusion

Authentication and user and group management are important parts of collaborative environments. While often seen as just an underlying service, they can easily decide the fate of a new collaborative system. For example, these areas might be regarded as too complex or hard to use. Also, different authentication mechanisms make it difficult for users to freely choose a collaborative environment that best suits their immediate needs—to speak with family or friends, to work on a public project, or to do private work.

To make this situation easier, we have developed a framework based on the federative approach. The framework is part of the middleware layer and provides a common security solution that can be used by any collaborative environment. It is based on PKI, but it uses an on-line Certification Authority that provides an easy and transparent way of obtaining certificates. This CA is part of a federation; the users ask for certificate generation with their "home" credentials. The certificates are extended with attributes provided by the home institutions, which are used for the authorization process and group management in a very generic manner. The federation-based authorization mechanism allows for efficient group management, with support for dynamic group creation and highly secure access control (e.g., it is possible to specify that a particular collaborative session is open to everybody who is a physician at any of a set of listed institutions).

Extended with VPN tunnels, the framework provides a truly general layer—a middleware bus—that can be used behind any collaborative tools. When combined with the security services presented in this paper, the VPN-based solution provides a pre-authenticated secure channel that is used to transport the actual data for the collaborative environments. According to our previously published measurements [8], this solution does not introduce any significant impact on the traffic parameters. There is no need for any upper-layer authentication or user or group management, as all this functionality is encapsulated in the process of federated certificate generation and secure-channel establishment. Clients of the collaborative environments can remain unchanged, only the servers need to be modified to work with the established secure channels, skipping any upper-layer authentication or authorization.

Acknowledgment

The work has been supported by the research intent "Optical Network of National Research and Its New Applications" (MSM 6383917201) of the Ministry of Education of the Czech Republic.

References

[1] N. Zennström and J. Friis, "Skype," 2003-2007, http://www.skype.com/.
[2] P. Biondi and F. Desclaux, "Silver needle in the skype," in *BlackHat Europe*, Mar. 2006, http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-biondi/bh-eu-06-biondi-up.pdf.
[3] S. A. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," in *INFOCOM 2006*, Barcelona, Spain, May 2006, http://www1.cs.columbia.edu/~salman/publications/skype1_4.pdf.
[4] L. Childers, T. Disz, M. Hereld, R. Hudson, I. Judson, R. Olson, M. E. Papka, J. Paris, and R. Stevens, "ActiveSpaces on the Grid: The construction of advanced visualization and interaction environments," in *Simulation and Visualization on the Grid*, 2000.
[5] L. Childers, T. Disz, R. Olson, M. E. Papka, R. Stevens, and T. Udeshi, "Access grid: Immersive group-to-group collaborative visualization," in *Proceedings of Immersive Projection Technology*, Ames, Iowa, 2000.
[6] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," Jun. 2002. [Online]. Available: citeseer.ist.psu.edu/foster02physiology.html
[7] T. Uram, "Access Grid authorization," in *Access Grid Workshop–APAC'05*, 2005, see also Section 2.9 of Virtual Venue Management User Manual. [Online]. Available: http://www-unix.mcs.anl.gov/fl/research/accessgrid/documentation/tutorial/AGTk_2.4/Authorization/Authorization.ppt
[8] P. Holub, E. Hladká, M. Procházka, and M. Liška, "Secure and Pervasive Collaborative Platform for Medical Applications," in *Studies in Health Technology and Informatics*, vol. 126. The Netherlands, Amsterdam, IOS Press, 2007, pp. 229–238.
[9] "ITU-T Recommendation X.509: Information technology—Open Systems Interconnection—The Directory: Public-key and attribute certificate frameworks," 2005, http://www.itu.int/rec/T-REC-X.509/e.
[10] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 Public Key Infrastructure—Certificate and Certificate Revocation List (CRL) Profile," IETF RFC 3280, 2002.
[11] E. Maler and et al, "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1," Sep. 2003, OASIS.
[12] S. Cantor, "Shibboleth Architecture—Protocols and Profiles," http://shibboleth.internet2.edu/shibboleth-documents.html.
[13] D. Recordon and D. Reed, "Openid 2.0: a platform for user-centric identity management," in *13th ACM Conference on Computer and Communications Security Co-Located Workshops*, Nov. 2006, pp. 11–16.
[14] K. Brown, "A first look at infocard," *MSDN Magazine*, vol. 21, no. 5, Apr. 2006.
[15] *ID-WSF Advanced Client 1.0 Specifications - Draft Release 2*, 2007. [Online]. Available: http://www.projectliberty.org/resource_center/specifications/liberty_alliance_complete_specifications_zip_package_01_august_2007
[16] T. Barton, J. Basney, T. Freeman, T. Scavo, F. Siebenlist, V. Welch, R. Ananthakrishnan, B. Baker, M. Goode, and K. Keahey, "Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, GridShib, and MyProxy," in *5th Annual PKI R&D Workshop*, 2006.
[17] C. Hosner, *OpenVPN and the SSL VPN Revolution*, Aug. 2004, The SANS Institute, http://www.sans.org, http://www.sans.org/reading_room/whitepapers/vpns/1459.php.
[18] J. Denemark, P. Holub, and E. Hladká, "RAP – Reflector Administration Protocol," CESNET, Tech. Rep. 9/2003, 2003.