

**Implementing Role-Based Authorization Capabilities in
Session Initiation Protocol (SIP)**

by

ANAND CHAVALI

B.E., University of Mumbai, 2001

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirement for the degree of
Master of Science
Interdisciplinary Telecommunications Program

2003

This thesis entitled:
Implementing role-based authorization capabilities in the
Session Initiation Protocol (SIP)
written by Anand Chavali
has been approved for the Interdisciplinary Telecommunications Program

Prof. Douglas Sicker

Prof. Ray Nettleton

Prof. Richard Han

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Abstract

Anand Chavali (M.S., Telecommunications)

Implementing Role-based Authorization capabilities in the Session Initiation Protocol (SIP)

Thesis directed by Professor Douglas Sicker

This thesis presents an approach to providing role-based authorization capabilities for the Session Initiation Protocol (SIP). SIP defines various methods for performing authentication. Authorization, however, is not defined explicitly and is generally based on identity. This does not scale well in multi-domain scenarios. In order to facilitate a greater level of granularity and scalability for authorization in SIP, new mechanisms need to be defined. This thesis describes the implementation of role-based authorization capabilities as part of a federation. Federation, or federated management, is an approach where domains share the responsibility of controlling access to their respective resources to users in similarly diverse domains. It is particularly suited to a multi-domain environment and is extremely scalable. Role-based authorization is a paradigm wherein authorization decisions are based on ‘role(s)’ asserted or assumed by a user rather than the identity of that user. Roles assigned to a user depend on the function performed by that user in the particular organization. For instance, a user may have the role of a faculty member of a particular department, or the manager of a

certain group. This arrangement enables easier management of authorization, expression of more sophisticated authorization policies, and affords some level of anonymity in certain scenarios.

The approach in this paper involves asserting user attributes across domains in a secure manner. Security Assertion Markup Language (SAML) is the protocol chosen for the purpose. User attributes are coded into SAML assertions which are then transported between the SIP entities in different domains. These attributes each describe a role of the user. The agreements that are part of the federation are responsible for negotiating the set of attributes that need to be transferred in any given context.

Bindings and profiles are essential components of the solution as they define ways to incorporate SAML in different communication protocols. This paper defines two profiles for using SAML in SIP; describing the transfer of SAML assertions by value or reference. An implementation of an authorization service is presented to execute these profiles. A security analysis of the threat model is also provided for each of the profiles.

**To Ma,
for everything**

Acknowledgements

It gives me great pleasure to thank the many people who have contributed to the completion of this thesis and to my education here at Boulder.

Firstly, I would like to thank my advisor, Professor Douglas Sicker, for his guidance and counsel. The creative freedom and flexibility which he allowed me in my work has been crucial in bringing out the best in me. He has also shown a lot of faith in my ability, for which I am extremely grateful. One of the factors that kept me motivated was my determination to not let him down. The most valuable lesson I have learned from him is the importance of keeping the big picture in mind, and always understanding how each little task or answer will contribute to the eventual goal.

I would also like to thank Professor Rick Han for collaborating with me on my project on sensing-zone signaling. He has given me a lot of advice about good research practices and helped me understand the importance of building systems, either to support or evaluate a good theory. My interactions with him have helped me refine my approach of looking at a research problem.

I am also extremely grateful to Professor Ray Nettleton for all his help and advice. He has had a very calming influence on me and was always available with

guidance when I needed it. His feedback on this thesis has been extremely valuable.

I would also like to thank fellow graduate students, Ameet Kulkarni and Mudassir Fajandar, for several useful and interesting discussions. Thanks are also due to John Giacomoni and David Keaton for their help in getting over sticky situations during the implementation phase.

I am also grateful to my uncle, Y. S. Rao, for his reassuring advice and to his lovely daughters, Mallika and Isha, for all the profound conversations.

Finally and most importantly, I would like to thank my parents for their constant support and encouragement that helped me tide over the most trying times and to my sister, Aparna for being the most wonderful person I know.

Contents

<u>1. Introduction.....</u>	<u>1</u>
1.1. <i>What is SIP?</i>	1
1.2. <i>Motivation & Problem Space</i>	2
1.3. <i>Context</i>	3
1.4. <i>Methodology</i>	3
1.5. <i>Contributions of this thesis</i>	4
<u>2. Background & Related Work</u>	<u>6</u>
2.1. <i>Session Initiation Protocol (SIP)</i>	6
2.1.1. <i>Overview</i>	6
2.1.2. <i>Architecture & Messaging</i>	6
2.1.3. <i>Authentication & Authorization</i>	8
2.2. <i>Security across domains & Federation</i>	10
2.2.1. <i>Security across domains</i>	10
2.2.2. <i>Federation</i>	11
2.2.3. <i>Issues and problems</i>	13
2.3. <i>SAML</i>	14
2.3.1. <i>Overview</i>	14
2.3.2. <i>Assertions</i>	16
2.3.3. <i>Bindings and Profiles</i>	17
2.4. <i>Role-based Authorization</i>	18
2.4.1. <i>Overview</i>	18
2.4.2. <i>Security Administration with RBA</i>	19
2.4.3. <i>Advantages</i>	19
2.4.4. <i>Application</i>	20
2.5. <i>Related Work</i>	20
2.5.1. <i>Video-Middleware group</i>	20
2.5.2. <i>Network convergence lab at CGU</i>	21
2.5.3. <i>Mudassir Fajandar's thesis</i>	21
2.5.4. <i>Ameet Kulkarni's thesis</i>	21
<u>3. Solution Rationale: Authorization in SIP</u>	<u>22</u>
3.1. <i>Objectives</i>	22
3.2. <i>Solution</i>	24
<u>4. Use of SAML in SIP</u>	<u>27</u>
4.1. <i>What are bindings and profiles?</i>	27
4.2. <i>Need for bindings and profiles?</i>	28

4.3.	Overview / SIP Profiles for SAML.....	28
4.4.	General Working.....	28
4.5.	Artifact profile.....	31
4.6.	Assertion profile	38
4.7.	UA – initiated profiles	42
4.8.	Bindings discussion.....	43
5.	<u>Implementation of Authorization Service.....</u>	44
5.1.	<i>Overview</i>	44
5.2.	<i>Functionality description.....</i>	45
5.3.	<i>Implementation details & challenges.....</i>	49
5.3.1.	<i>VOCAL SIP Implementation</i>	49
5.3.2.	<i>Local authentication</i>	51
5.3.3.	<i>Creating assertions.....</i>	53
5.3.4.	<i>Processing assertions</i>	57
6.	<u>Analysis.....</u>	58
6.1.	<i>Conformance of profiles to OASIS guidelines.....</i>	58
6.2.	Security Analysis	60
6.2.1.	<i>Artifact Profile Security Model</i>	60
6.2.2.	<i>Assertion Profile Security Model.....</i>	62
7.1.	Conclusions.....	64
7.2.	Future Work.....	65
	<u>Bibliography</u>	66
	<u>Appendix A.....</u>	69

1. Introduction

The objective of this thesis is to provide enhanced authorization capabilities in the Session Initiation Protocol (SIP). The current security mechanisms defined in the SIP specification, RFC 3261 [1], provide authentication, integrity and confidentiality. Authorization, however, hasn't been explicitly provided. It is implied as the result of successful authentication. This thesis defines authorization capabilities needed in SIP, defines the necessary mechanisms to provide them, and presents a sample implementation of an authorization service (AS) that implements these mechanisms as a proof-of-concept.

1.1. *What is SIP?*

Call setup information of any kind is carried by protocols known as signaling protocols. Signaling protocols are used to setup and control sessions for multimedia applications. Applications which require sessions to be setup include telephony, videoconferencing, and remote learning. Circuit-switched telephone systems employ a signaling protocol known as signaling system 7 (SS7). IP signaling protocols are used to setup sessions over packet-switched networks for IP-based multimedia applications such as voice-over-IP (VoIP) and IP-based videoconferencing. Session Initiation Protocol (SIP) and H.323 are the two most popular IP signaling protocols.

SIP is an application layer signaling protocol created by the Internet Engineering Task Force (IETF) that allows entities to locate one and other on a network and invite them to participate in a session. It is also responsible for modifying and terminating these sessions. SIP has been designed with the IETF's philosophy – “specify only what you need”. Unlike H.323, which is a suite of protocols that handle everything from the location of endpoints to negotiating media parameters, SIP is a lightweight protocol that is limited in its functionality. Once it has established the session, other protocols like the Session Description Protocol (SDP) take over and handle other tasks (media negotiation in the case of SDP). SIP has also been adopted by the 3GPP for call control and has also found applications in instant messaging and on-line gaming.

1.2. *Motivation & Problem Space*

The motivation for this research stems from the state of security currently provided in SIP. [1] defines very strong mechanisms for providing confidentiality, integrity, and authentication. However, there are no mechanisms explicitly defined for authorization. A user receiving a SIP request to set up a session has to make the decision based on the identity of the originating user, which it may authenticate. However, given the proliferation of IP-based services in recent times, the number of users is so large that it may not be possible to have pre-existing relationships between users. In such scenarios, the user receiving the request would like more information about the originating user in order to make

an authorization decision.[4] A sophisticated authorization framework is, therefore, required. It is also important that this framework be scalable, given the sheer number of users. This can be accomplished by providing federated, role-based authorization capabilities in SIP.

1.3. *Context*

This work has been done as part of Internet2's Video-Middleware (Vid-Mid) [10] working group's initiative to provide middleware functionalities in the areas of security, specifically authentication & authorization, and directory assistance to videoconferencing applications using the two most common IP signaling protocols – SIP and H.323. The author is part of a research group, led by Prof. Douglas Sicker, which is focusing on authentication and authorization in SIP. This thesis describes the necessary mechanisms to provide federated, role-based authorization in SIP. These mechanisms need modifications to the behavior of SIP, particularly two SIP entities known as the SIP user agent and the SIP proxy. These modifications are being developed as part of the other theses in the group. Together, the group attempts to create a complete working architecture that implements the authorization mechanisms presented in this thesis.

1.4. *Methodology*

In order to provide federated, role-based authorization in SIP, we make use of the Security Assertion Markup Language (SAML). SAML defines an XML

framework for exchanging security information in the form of XML constructs. This thesis describes how SAML can be used to achieve federated, role-based authorization in SIP. It describes the bindings and profiles to be used for the purpose. Bindings and profiles are tools necessary in order to use SAML in any communication protocol or framework. This thesis also develops a basic authorization service (AS) that will implement the mechanisms defined here and will serve as a proof-of-concept.

This thesis will be evaluated on three fronts. The proof-of-concept will serve as the first. A details threat model analysis, describing the various possible threats and their countermeasures, for the authorization mechanisms will serve as the second. The third will be a compliance analysis of the profiles with certain standard guidelines.

1.5. *Contributions of this thesis*

As mentioned before, SIP has an extremely rudimentary authorization mechanism. This thesis describes a way to enhance it to a federated, role-based mechanism by the use of the Security Assertion Markup Language (SAML). The topics described as part of the background in Chapter 2 are all well-known; however, they have not been applied before in the real-time applications space. Chapter 3 presents a brief discussion on the rationale behind the decision to bring federated management and role-based authorization to SIP.

Chapter 4 is an adapted version of a yet unfinished Internet-Draft, “Session Initiation Protocol (SIP) Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)” by Sicker, Chavali and Kulkarni. As part of the essential modifications required for the profiles, an abstract entity called an Authorization Service (AS) is defined that is responsible for creating, processing and, in certain cases, transferring SAML data. Chapter 5 describes a sample implementation for it. Chapter 6 presents an analysis of the profiles and the AS. Finally, chapter 7 presents conclusions and future work.

2. Background & Related Work

This chapter provides a brief description of each of the important protocols and concepts used in this thesis.

2.1. *Session Initiation Protocol (SIP)*

This section describes SIP in terms of its functionality, architecture, messaging and security procedures. It is largely paraphrased from [1].

2.1.1. Overview

The Session Initiation Protocol (SIP) is an application layer signaling protocol created by the Internet Engineering Task Force (IETF). SIP allows entities to locate one and other on a network and invite them to participate in a session. It is also responsible for modifying and terminating these sessions. The details of this protocol are specified in RFC 3261 [1]. Its relevant details are encapsulated in this section

2.1.2. Architecture & Messaging

A SIP-based network architecture primarily consists of SIP endpoints, SIP servers, and gateways for interoperation with other signaling protocols and legacy systems. The SIP endpoints are known as user agents (UA). The UA resides in an IP telephone or as a piece of software on a user's computer. It is responsible for

initiating requests for call setup and termination. The calling UA is generally referred to as the user agent client (UAC) and the called UA as the UA server (UAS). SIP servers can be of three types – proxy servers, redirect servers and registration servers (or registrars). Proxy servers (referred to simply as proxies in this document) receive requests and forward them on behalf of the requesting agent. Redirect servers are responsible for redirecting requests to alternate locations. Registrars receive requests for registration and store the information obtained from these requests in a location service. The proxy and redirect servers use the information contained in this location service to locate a particular user agent.

The messaging in SIP consists of requests called methods, and HTTP-like responses. A combination of a SIP request and at least one response is called a transaction. A SIP message consists of a number of header fields and occasionally one or more attachments. A message exchange to set up a session is shown in fig. 2.1.

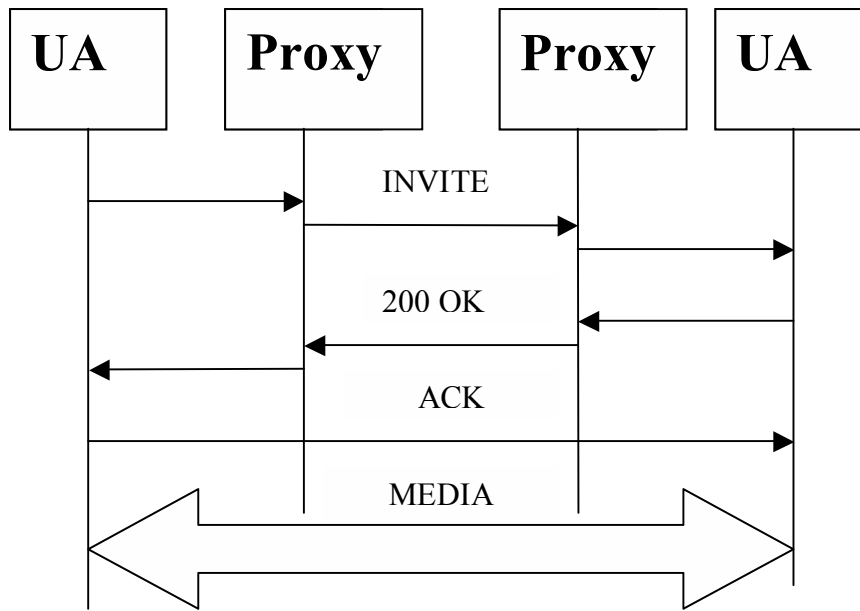


Fig 2.1: SIP message exchange for call setup

The call setup is requested by the use of the INVITE method. If the called UA wished to set up the session, it responds with a 200 OK success message, which is acknowledged by the called UA with the ACK method. The session is now established. The protocol responsible for media transfer (for e.g. the Real-time Transport Protocol, RTP) then takes over.

2.1.3. Authentication & Authorization

Authorization in SIP is performed only by the recipient of a request and is based entirely on authenticated identity. In other words, once a user has successfully authenticated, its identity is the only information available for the authorization

decision to be made. The authentication in SIP is essentially of two types, user-to-user authentication and proxy-to-user authentication.

User-to-user authentication

On receiving a request from a UAC, a UAS may choose to authenticate the user before processing the request. To do this, it first checks the Authorization header field for credentials. If none are provided, it can challenge the UAC to do so by rejecting the request with a 401 Unauthorized status code. This response must contain the WWW-Authenticate header field, which must be populated by at least one challenge. This challenge must indicate the authentication scheme(s), the realm, and the concerned parameters.

When the originating UAC receives this response, it should re-originate the request with the proper credentials. These credentials can be included in the Authorization header field in the request, which is resent after incrementing the CSeq header and retaining all other header fields.

Proxy-to-user authentication

Similar to user-to-user authentication, the proxy may also, on receiving a request, choose to authenticate the originator of a request, before processing it. In order to this, it initially checks the request to see if any credentials are provided. If there are no credentials provided with the request, the proxy can challenge the

originator to do so by rejecting the request with a 407 Proxy Authentication Required status code. This response must be populated with a Proxy-Authenticate header field that contains the appropriate value for that particular proxy.

When the UAC receives the 407 response, it should include the proper credentials and re-originate the request. The credentials can be contained in the Proxy-Authorization header field and the CSeq header value should be incremented.

2.2. Security across domains & Federation

This section examines the issues in multi-domain security and the notion of federated management of security.

2.2.1. Security across domains

Network resources exist as islands, controlled and maintained by a network authority, typically a network administrator. This control of resources includes access control mechanisms based on authentication and authorization. Resources may be perceived as ranging from public to highly restricted, which suggests the need for fine granularity of access control. [11]

An essential ingredient of any access control mechanism is an access control matrix, comprising of several access control lists (ACL) [12], [13], [14]. The rows in an access control matrix list the various users in the system and the columns mention the protected resources. An access control matrix, therefore, describes

what resources are accessible by a particular user or, conversely, who is allowed to access a particular resource. Each column in the matrix is a single ACL, while each row describes the privileges, i.e. access levels or permissions to different resources, for each user.

In the modern world, access to information is becoming all-important and the information that a user needs is very often distributed in many different locations. This can lead to unmanageably huge access control matrices, as each user's information would need to be stored in that many more locations, and would contribute an additional row in the access control matrices in each case. An additional burden is also induced by authentication, which requires usernames and passwords to be stored for every user in each location.

2.2.2. Federation

Federated administration, or federation, is a concept that is being increasingly talked about in the web services' security space [15]. It describes scenarios in which no single domain or organization is responsible for managing all users and resources in a distributed environment. Instead, each domain must manage its own local security policies that support mutually beneficial transactions between the various domains.

Federated administration assumes local middleware infrastructure deployed in a consistent fashion among those domains that permit their individual users to collaborate with each other, communicate, transact, and access protected resources across networks.

To work, a federation of enterprises typically needs to agree on two sets of topics:

- technical specifications (e.g. interoperable software, data types to be exchanged)
- policy specifications (e.g. practices for producing trustworthy data, handling of privacy)

Federations could be structures within academia, financial service partners, medical collaboratives such as hospitals and clinics, government agencies and commercial service providers, etc. [16]

The most common form of federation currently is the identity federation, or federated identity management, which is responsible for managing identities between domain boundaries. The main driver of identity federation is an application called Single (or Shared) Sign-On (SSO). SSO allows applications, systems, and domains to share authentication information. This allows an identity to be usable, without repeated authentications, across a bunch of domains as long as they share authentication information about the identity. [17]

The most common form of identity federation implementation involves a domain authenticating all the identities in its domain. When an identity attempts to access a resource in another domain, the local domain vouches for the authenticity of this identity. There is a trust agreement between the two domains, wherein the remote domain trusts the local domain to have reliably authenticated the concerned identity.[18] Since information about the identity is shared between domains, the remote domain is aware of what resources this identity is allowed to access.

2.2.3. Issues and problems

The following are the chief concerns with federated management of security.

Implementation

The main issue in federated management is the challenges involved in its implementation. The first challenge in establishing trust agreements between domains is to know exactly which domains to include. This becomes all the more difficult when domain boundaries are not very clear. Also, the when one domain trusts another domain to authenticate identities, it is beneficial only if the authenticating domain has solid security policies and systems in place [18]. This necessitates a very high level of standardization and compatibility, which will make implementation all the more difficult.

Privacy

A very important issue in identity federation is the fact that some identity information will exit beyond the domain's firewall, and is consequently at least partially beyond the domain's control. This raises serious privacy concerns on the part of the user. Privacy legislation compels domains, especially corporate ones, to be cognizant of user's privacy rights and preferences. It would not be appropriate for sensitive user information at many different locations, where the user is not assured of the same quality of privacy protection. Moreover, the privacy policies in these different domains may differ substantially from each other. [17]

Liability

The third concern arising from this distribution of user information and functionality is the question of liability. Liability is a big issue in today's legal environment and most organizations engaged in any kind of web-based transactions are very specific about limiting or refusing to incur any liability as a result of their representations. With increasing inter-domain dependence as a result of federation, this becomes an even more complex and difficult issue.

2.3. SAML

This section gives a brief background on SAML.

2.3.1. Overview

The security assertion markup language (SAML) is a framework for exchanging security information in the form of XML constructs. It was created by the Security Services Technical Committee (SSTC) of the Organization for the Advancement of Structured Information Standards (OASIS) group.

The OASIS group defines SAML as follows:

“SAML is an XML-based framework for exchanging security information. This security information is expressed in the form of assertions about subjects, where a subject is an entity (either human or computer) that has an identity in some security domain. Assertions can convey information about authentication acts performed by subjects, attributes of subjects, and authorization decisions (already made) about whether subjects are allowed to access certain resources. The protocol, consisting of XML-based request and response message formats, can be bound to many different underlying communications and transport. OASIS currently defines one binding; SOAP* over HTTP.” [6]

SAML was created specifically to provide a standard implementation of federation which would ensure interoperability between the various domains in a federation. Thus when a user in one domain attempts to access a resource in another domain, the local domain sends a SAML assertion to the remote domain describing the local authentication performed by this particular user. Another problem that SAML tries to solve is the proprietary nature of the way permissions

* Simple Object Access Protocol (SOAP)

management data is shared and the huge amount of new code normally required to integrate new security features. [19]

2.3.2. Assertions

An assertion is a package of information that supplies one or more statements issued by a SAML issuer. SAML allows issuers to create three different kinds of assertions: authentication assertion, authorization decision assertion and attribute assertion.

An authentication assertion is a statement by an issuer about an act of authentication performed by a particular user. It also mentions the method of authentication and the instant the authentication took place. It is important to note that an authentication assertion is merely reporting an act of authentication and is independent of the method that was actually used. An attribute assertion states that a particular user is associated with a certain set of attributes. This information generally tends to be persistent and is normally obtained from a directory or some other form of data repository. An authorization assertion states the authorization decision for a particular user, at a certain level, for a particular resource, given some specific information. This assertion is always sent in response to a request for authorization that contains the required information. [19]

A single assertion may contain many individual statements about authentication, attributes, or authorization. [20]

Assertions are issued by SAML authorities, which could be authentication authorities, attribute authorities, or authorization authorities (also called policy decision points). These authorities can use various sources of information to create the assertions, such as data repositories, external policy stores, or even other assertions. Thus SAML authorities can both produce and consume assertions.

2.3.3. Bindings and Profiles

SAML assertions/artifacts, protocol requests and protocol responses can be embedded in other structures for transport. The specification for the bindings and profiles [7] provides a framework for this embedding and transport in SOAP messages over HTTP. [20] No such specification exists for any other framework or protocol.

SAML protocol bindings define the mappings from SAML request-response messaging into standard communication protocols. SAML profiles, on the other hand, are sets of rules describing how to embed and extract SAML assertions from a standard communication protocol. [7]

We will discuss bindings and profiles in more detail in chapter 4.

2.4. Role-based Authorization

This section describes the relevant aspects of role-based authorization.

2.4.1. Overview

Role-based Authorization (RBA) or Access Control (RBAC) is a concept that was introduced in 1992 by David Ferraiolo and Rick Kuhn. It involves making authorization decisions based on roles that individual users have or assume as part of an organization instead of their identity. An example of such a role could be a doctor or specialist in a particular hospital or a professor in a specific research group. [5]

Roles are for the most part group-oriented as they tend to be associated with a set of individual members. [2] However, such roles could also be context specific so as to exclude part of the group depending on the context. For instance, the role of ‘doctor on duty’ would be assigned only to the doctors working on that shift and not to the other doctors. This subset doesn’t have to be constant, but the set of privileges and their mappings can be.

There are generally three rules involved in role-based authorization –

Role authentication: A user cannot access any resources unless it has been assigned or has selected a role. The identification and authentication processes are necessary for the user to assume a role.

Role authorization: A user must be authorized to take on a role. This rule ensures that a user assumes only those roles it is authorized for in the association between roles and sets of users.

Transaction authorization: The user assuming a particular role can access only those resources that are authorized for that user's active role. [2]

2.4.2. Security Administration with RBA

Administration using RBA involves essentially administering roles and setting privileges for those roles. Role administration consists chiefly of associating users with roles according to the functions they perform in the domain. When the function of the user changes, its membership to existing roles is modified accordingly. Finally, when a user is no longer belongs to a domain, its memberships to the roles are simply deleted. Since the privileges associated with a particular role (and therefore function) are enterprise-specific, they generally tend to be fairly constant over time. [2]

2.4.3. Advantages

The obvious advantage from such a scheme is the streamlining of the security management process and the simplification of the authorization databases. Another advantage stemming from the use of roles is the potential to provide

some level of user anonymity by using the role as a pseudonym. Role-based authorization decisions also provide an effective means for developing and enforcing enterprise-specific security policies. Finally, the various roles assumed by a particular user could be used in combination to provide highly granular authorization. [2], [5]

Moreover, a comparison of a simple RBAC model and a group ACL mechanism by Barkley [21] shows that even the simplest RBAC model is as effective in its ability to express access control policy. An RBAC system with special features (which are not possible with ACLs) will be even more effective.

2.4.4. Application

While it is true that many vendors and researchers are applying RBA to various problems in network security, it has not yet been applied to the area of real-time IP based systems, e.g., voice and video over IP. In real-time communication using SIP an assertion, describing one or more roles, can be presented to the UAS or a proxy instead of the identity of the user initiating the session.

2.5. Related Work

This section describes research related to the material in this thesis

2.5.1. Video-Middleware group

The Video Middleware (VidMid) working group at Internet2 [middleware.internet2.edu/video] is working on creating middleware services for

digital video and related areas. The specific middleware functionalities that VidMid is looking to provide are primarily resource discovery and authentication & authorization for videoconferencing, video on demand, data collaboration, and voice over IP. They have defined H.350, a directory services architecture for multimedia conferencing with H.323, H.320, and SIP. This thesis will serve to provide a part of the services to meet the authentication & authorization requirements. More information is available at [10].

2.5.2. Network convergence lab at CGU

The network convergence lab (NCL) lab at Claremont Graduate University (CGU) is conducting research on building a videoconferencing client that is “secured and H.350 enabled” [22].

2.5.3. Mudassir Fajandar’s thesis

Mudassir Fajandar is working on developing the modifications necessary to the SIP UA for implementing the mechanisms described in this thesis.

2.5.4. Ameet Kulkarni’s thesis

Ameet Kulkarni is working on developing the modifications necessary to the SIP proxy server for implementing the mechanisms described in this thesis.

3. Solution Rationale: Authorization in SIP

This chapter describes the requirements of the authorization mechanisms this thesis attempts to provide and the reasoning behind some of the decisions taken.

3.1. Objectives

As mentioned in section 2.1.3, authorization in SIP is done solely on the basis of an authenticated identity, i.e. the UAS or proxy must make an ‘accept’ or ‘reject’ decision based on the UAC’s identity. The principal objective of this thesis is to enhance this authorization process for SIP requests. This thesis confines itself to the authorization of INVITE requests alone, as it is the request responsible for call setup.

It is desirable to have authorization decisions that are somewhat more sophisticated than simple accept/reject decisions. Authorization decisions that are context-specific would be a significant enhancement. For instance, a policy that is sensitive to the time of the day may authorize a user only during a certain period during the day (say during business hours). For such a policy to be implemented, some form of access control mechanism is needed that can express such a policy.

One of the problems with authorization systems based on identity is that there needs to be an *a priori* establishment of privileges associated with that identity,

especially when the access control policies have at least a moderate level of sophistication. Providing the authorizing entity with as much information as possible (and allowed) about the calling user will, therefore, enable a much better authorization process.

Given the proliferation of IP-based services in recent times, it is not realistic to expect that a SIP request would originate in the same domain as its recipient. This raises manageability and security issues. It is, therefore, a very important requirement for the authorization mechanism to be managed in a federated manner.

However, the federation described in section 2.2 is an identity federation. The problem with this is as follows. Consider an identity federation for the purpose described above, implemented using SAML. When a user, authenticated in one domain, wants to send an INVITE to a user in another domain, the local domain of this user will send a SAML authentication assertion to the remote domain stating the details of the authentication. The remote domain now has an assurance from the local domain about the identity of the calling user. The requirement to have more information about the user can also be met, using SAML attribute assertions. But, in order to authorize this user, the remote domain still needs to know the set of privileges associated with this user. Maintaining a set of

privileges for every such user in every domain can quickly become burdensome and, consequently, infeasible.

3.2. *Solution*

The solution for this problem is to employ a role-based authorization mechanism instead of a collection of privileges for every possible user (in other words a conventional access control matrix). This will provide much more simplified management of authorization.

The two requirements for administration with a role-based authorization mechanism are the administration of roles and the setting of privileges for those roles. The latter should be done in the remote domain and the former in the originating domain. This ensures that each domain is in charge of administering its users – both for outgoing and incoming requests. The appendix illustrates some key ideas in this type of management.

There are two more requirements to address. Firstly, the roles need to be conveyed to the remote domain. This can be done using attribute assertions. The attributes will each express a role that the user is associated with. As such the attribute authority will use the data repository containing the association of users and roles to create attributes.

Secondly, an inherent assumption here is that the roles assignment in the originating domain is consistent with the roles that are assigned privileges in the remote domain. Ensuring this consistency is a cost, though not an expensive one for two reasons. The first reason is that the principal cost in role-based authorization is in the initial setup of privileges for roles and the administration of roles for existing users. The recurring costs are not too frequent given the general stability of access control policies and the trivial cost of role administration. The second reason is that such inter-domain collaborations and interactions are most common between similar organizations (for e.g. hospitals, universities, financial institutions, etc) which are quite likely to have similar functions, and therefore similar roles. In such cases, ensuring consistency of roles may not need much action.

The three rules of role-based authorization can be satisfied as follows. Role authentication & authorization are taken care of in the originating domain. A user who authenticates with its domain activates all the roles that it is associated with and is therefore authorized to assume. Transaction authorization is taken care of at the remote domain. Since the privileges are set according to roles, a user will not be able to access anything that is not authorized for its active roles(s).

Thus, role-based authorization when implemented with federation can provide the necessary information without causing storage concerns. Another advantage is

that using role-based authorization will allow the expression of much more complex and sophisticated access control policies.

In conclusion, authorization in SIP can be improved by employing federated, role-based authorization in SIP using SAML. The next chapter defines the necessary mechanisms for this.

4. Use of SAML in SIP

This chapter describes the use of SAML in SIP. The first section defines bindings and profiles. The next sections present the profiles. Two profiles are defined – artifact and assertion. A different approach to develop profiles is also described. Finally, a discussion on bindings is presented.

4.1. What are bindings and profiles?

OASIS defines SAML protocol bindings as “mappings from SAML request-response message exchanges into standard messaging or communication protocols” [7]. For instance, a mapping of SAML request-response message exchanges into HTTP message exchanges would be called an HTTP binding for SAML or a SAML HTTP binding.

SAML profiles are defined as “sets of rules describing how to embed and extract SAML assertions into a framework or protocol” [7]. An HTTP profile for SAML (also called a SAML HTTP profile), for instance, would describe how SAML assertions are attached to HTTP messages, communicated to the destination site from the origin site, and subsequently processed at the destination. [7]

4.2. Need for bindings and profiles?

The principal need for bindings and profiles, in general, is to provide a standard to ensure interoperability among various vendor implementations. [7] However, the only set of bindings and profiles currently defined are for the Simple Object Access Protocol (SOAP). In order to use SAML in SIP, an important first step is therefore to define bindings and profiles for the same.

4.3. Overview / SIP Profiles for SAML

The following sections define proposed SIP profiles for SAML.

Two profiles described are:

The SIP/Artifact profile of SAML

The SIP/Assertion profile of SAML

For each profile, a section describing the threat model and relevant countermeasures is also included.

4.4. General Working

In this scenario, a user authenticates herself within her local domain. The user then attempts to establish a session with another user at a target domain, without having to re-authenticate with the target domain. The objective of the profile is to transfer information about this local authentication, along with any other necessary information, to the target domain in secure manner. This information

allows the target domain to make an authorization decision without any explicit action required on the part of the user.

The local authentication can take place at any time. A suggested option is during UA registration. The methods used for this authentication could be anything, so long as they conform to [1].

Once the local authentication is successfully performed, this information is logged by the authorization service (AS) and the corresponding assertions are created. The assertions will contain authentication statements, which will each describe various facts about the local authentication, and attribute statements, which will each describe more persistent attributes of the user. The assertions may, therefore, be a combination of transient information (authentication statements) and persistent information (attribute statements). Since these attributes can specify 'roles', it is the attribute statements that provide the basis for role-based authorization.

The UA can then send out an INVITE addressed to the UA it wants to establish a session with. This INVITE is routed through the outgoing proxy server for that domain. The proxy server initiates a sequence of messages that would result in the transfer of the assertions to target domain.

This approach requires the identification of Policy Decision Points (PDP) and Policy Enforcement Points (PEP). As the names suggest, PDPs are where authorization decisions are made and PEPs are where they are executed. It is

proposed that the AS's function as the PDPs while the proxy servers function as the PEPs in their respective domains.

The Policy Decision Point (PDP) at the target domain examines these assertions obtained directly or by request of the assertions based on the artifact, and makes a decision on whether to allow the INVITE to go through to the destination user (called party). Since these decisions are based on the SAML assertions received from the origin domain they do not require a re-authentication by the originating user, with the target domain.

Figure 3.1 illustrates this basic model.

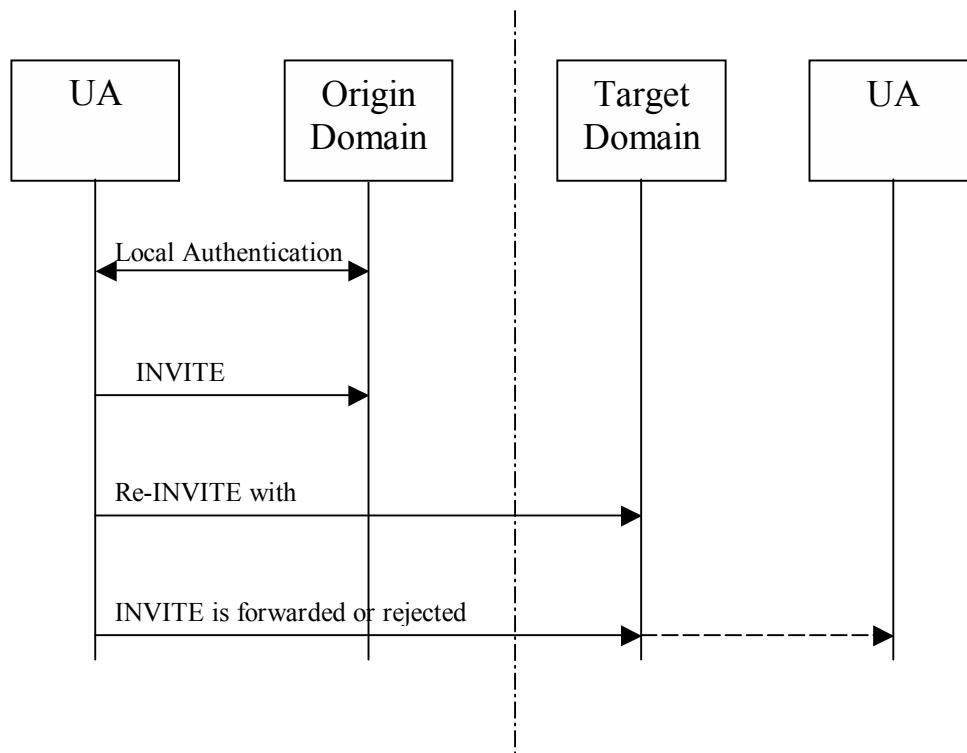


Fig. 3.1: General model for assertion transfer

Similar to the profiles defined for SOAP in [7], two techniques are proposed to carry the assertions between domains:

- SAML artifact – A SAML artifact of limited size is carried by the INVITE method as an attached MIME body or in one of the headers, such that when the artifact is conveyed back to the originating domain, the artifact unambiguously references an assertion. The destination PDP, on receiving the artifact, acquires the referenced assertion by some further steps. Typically, this involves the use of a registered SAML protocol binding. This technique is used in the Artifact profile of SAML.
- Message body assertion – A SAML assertions are transferred to the SIP UA as a MIME body and conveyed to the destination site as part of a MIME payload when the user makes a request. This technique is used in the Assertion profile of SAML.

4.5. Artifact profile

The artifact profile of SAML relies on a reference to the needed assertion traveling in a SAML artifact, which the target domain must dereference from the originating domain in order to determine the user's security information. The artifact **MUST** be carried as an attached MIME body or in a SIP header.

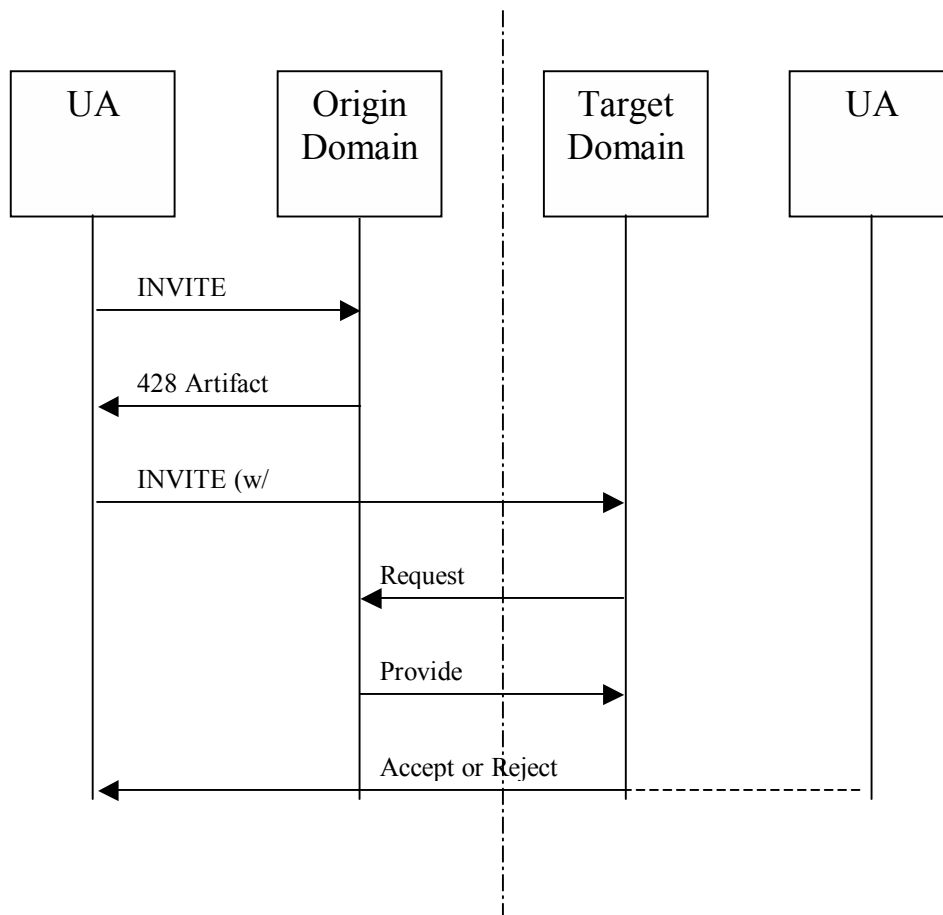


Fig. 3.2: Message exchanges for artifact model

The user agent/artifact profile consists of a single interaction among three parties; a user agent, an originating domain, and a target domain, with a nested sub-interaction between two parties (the originating domain and the target domain). Figure 3.2 shows the message sequence for this process.

Step 1: INVITE (Initiating an inter-domain session):

In step 1, the user agent at the origin domain sends an INVITE to the user agent at the target domain. This is routed through the outbound proxy in the origin domain.

The SIP request takes the following form, as defined in [1].

```
INVITE sip:originhost@origindomain.com SIP/2.0
Via: SIP/2.0/UDP proxyserver.origindomain.com
To: Target <sip:targethost@targetdomain.com>
From: Origin <sip:originhost@origindomain.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Contact: <sip:originhost@origindomain.com>
Content-Type: application/sdp
Content-Length: 142

--boundary
(SDP body)
--boundary
```

The “To:” header conveys information about the desired target user in the target domain. “Confidentiality and message integrity MUST be maintained in step 1.”

[7]

Step 2: 428 Artifact/Assertion Required:

In step 2, the outbound proxy server of the originating domain intercepts the INVITE and respond with a “428” response adding the artifact/s to the response message as a MIME body. The SIP response MUST take the following form:

```
SIP/2.0 428 Artifact/Assertion required
Via: SIP/2.0/UDP proxyserver.origindomain.com
;branch=z9hG4bKnashds8;received=192.0.2.3
From: Origin proxy <proxyserver.origindomain.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:targethost@192.0.2.4>

Content-Type: message/SAMLart
Content-Length: 60

--boundary
<SAML string>
--boundary
```

Where:

<SAML string> = ...TARGET=<Target>...SAMLart=<SAML artifact>...

The target description specifies the target PDP which will use the received artifact to request the corresponding assertion(s).

The SAML <SAML string> component must contain at least one SAML artifact.

If more than one artifact is carried within <SAML string>, all the artifacts must have the same source ID. [7]

The use of status code 428 is used to indicate that a MIME attachment containing SAML artifact/assertion is required. The response should contain the necessary artifact as part of the MIME attachment. The use of the 428 status code for this purpose was proposed by Peterson in [8].

There is an important decision that needs to be made here. We need to decide where the MIME body containing assertions is attached – at the SIP user agent or the SIP proxy (or both) It is attractive from the SIP standpoint to push as much control as possible out to the endpoint. However, given the nature of a federated administration, we require some participation by a local authentication entity. The solution that we tentatively adopt is for a local authentication entity to pass the body back to the UA, where a new INVITE (including the additional MIME body) will be created. This approach is a variation of the method described in [8]. The specifics are beyond the scope of this document.

“Confidentiality and message integrity **MUST** be maintained in step 2. It is **RECOMMENDED** that the message transfer be protected by SSL 3.0 or TLS 1.0. Otherwise, the one or more artifacts returned in step 2 will be available in plain text to an attacker who might then be able to impersonate the assertion subject.”
[7]

Step 3: Re-INVITE with Artifact

In step 3, the artifact is extracted from the MIME type attachment of the “428 Artifact/Assertion Required” response by the UA and added to the INVITE as a MIME attachment or in a SIP header, which is re-sent.

The SIP request **MUST** take the following form:

```
INVITE sip:originhost@origindomain.com SIP/2.0
Via: SIP/2.0/UDP proxyserver.origindomain.com
To: Target <sip:targethost@targetdomain.com>
From: Origin <sip:originhost@origindomain.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE

Max-Forwards: 70
Contact: <sip:originhost@origindomain.com>

Content-Type: multipart/mixed; boundary=unique-boundary-1
--unique-boundary-1
(SDP body)
--unique-boundary-1

<SAML string>

--unique-boundary-1
```

This message is routed through the inbound target SIP proxy server, which forwards it to the PDP for its domain. The PDP will use the received artifact to request the corresponding assertions.

In case of the artifact carried in the SIP header, the <SAML string> component will be contained in a special SIP header.

“Confidentiality and message integrity MUST be maintained in step 3. It is RECOMMENDED that the message transfer be protected by SSL 3.0 or TLS 1.0. Otherwise, the artifacts transmitted in step 3 will be available in plain text to any attacker who might then be able to impersonate the assertion subject.” [7]

Steps 4 & 5: Transferring the corresponding assertions

In steps 4 and 5, the PDP in the target domain dereferences the one or more SAML artifacts in its possession in order to acquire the SAML assertions that correspond to each artifact.

A registered SAML protocol binding must be used for a SAML request-response message exchange between the destination and source sites. The target domain functions as a SAML requester and the originating domain functions as a SAML responder.

“The target domain MUST send a <samlp:Request> message to the originating domain, requesting assertions by supplying assertion artifacts in the <samlp:AssertionArtifact> element. If the originating domain is able to find or

construct the requested assertions, it responds with a <samlp:Response> message with the requested assertions. Otherwise, it returns an appropriate error code, as defined within the selected SAML binding.

In the case where the originating domain returns assertions within <samlp:Response>, it MUST return exactly one assertion for each SAML artifact found in the corresponding <samlp:Request> element. The case where fewer or greater number of assertions is returned within the <samlp:Response> element MUST be treated as an error state by the target domain.

The originating domain MUST implement a “one-time request” property for each SAML artifact. Many simple implementations meet this constraint by an action such as deleting the relevant assertion from persistent storage at the source site after one lookup. If a SAML artifact is presented to the originating domain again, the originating domain MUST return the same message as it would if it were queried with an unknown artifact.” [7]

The selected SAML protocol binding MUST provide confidentiality, message integrity and bilateral authentication. [7]

“The originating domain MUST return a response with no assertions if it receives a <samlp:Request> message from an authenticated target domain X containing an artifact issued by the source site to some other target domain Y , where $X \triangleleft Y$. One

way to implement this feature is to have originating domain maintain a list of artifact and target domain pairs.” [7]

Step 6: Responding to the INVITE

In step 6, the origin user agent is sent a SIP response that either allows or denies access to the target user. If the user is authorized to initiate the session, then the INVITE is forwarded by the inbound proxy server to the target user’s UA, which responds to the INVITE in accordance with [1].

The target domain PEP SHOULD provide some form of helpful error message in the case where the originating user is not authorized to initiate a session with the target user.

4.6. Assertion profile

The Assertion profile of SAML allows authentication information to be supplied to the target domain without the use of an artifact and the need for a binding. It consists of a series of two interactions, the first between a user equipped with a user agent and the originating domain, and the second directly between the user and the target domain. Figure 3.3 illustrates the interactions between parties in the assertion profile.

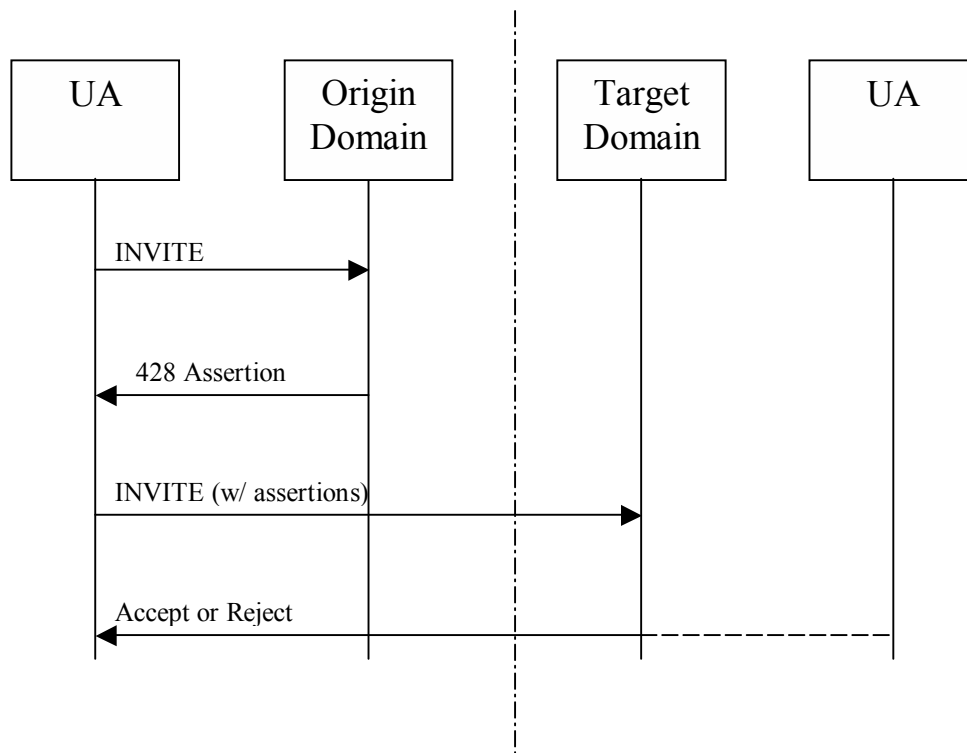


Fig. 3.3: Message exchange for assertion profile

Step 1: INVITE (Initiating an inter-domain session):

In step 1, the calling party's User Agent sends an INVITE to the target user

The SIP request takes the following form (as described in [1]):

```

INVITE sip:originhost@origindomain.com SIP/2.0
Via: SIP/2.0/UDP proxyserver.origindomain.com
To: Target <sip:targethost@targetdomain.com>
From: Origin <sip:originhost@origindomain.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Contact: <sip:originhost@origindomain.com>
Content-Type: application/sdp
Content-Length: 142

--boundary
(SDP body)
--boundary
  
```

The “To:” header conveys information about the desired target user in the target domain.

“Confidentiality and message integrity MUST be maintained in step 1.” [7]

Step 2: 428 Artifact/Assertion Required:

In step 2, the outbound proxy server of the originating domain intercepts the INVITE and sends back a “428 Artifact/Assertion Required” with a MIME body attached that contains a SAML Assertion.

The SIP response MUST take the form:

```
SIP/2.0 428 Assertion required
Via: SIP/2.0/UDP proxyserver.origindomain.com
;branch=z9hG4bKnashds8;received=192.0.2.3
From: Origin proxy <proxyserver.origindomain.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:targethost@192.0.2.4>
Content-Type: application/pkcs7-signature; name=smime.p7s;boundary=boundary-42

Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=assertion.p7s;handling=required

--boundary42

<NAME="TARGET" VALUE="<Target">>
<NAME="SAMLResponse" VALUE="<SAMLResponse">>

--boundary42--
```

The MIME body MUST include at least one SAML response within the control name SAMLResponse. The SAML response MUST be digitally signed following the guidelines given in [20]. Included assertions MAY be digitally signed.

“Confidentiality and message integrity MUST be maintained for step 2. It is RECOMMENDED that the message transfer be protected by SSL 3.0 or TLS 1.0. Otherwise, the assertions returned will be available in plain text to any attacker who might then be able to impersonate the assertion subject.” [7]

Step 3: Re-INVITE with Assertions:

In step 3, the User Agent extracts the MIME body that contains the SAML response and again sends an INVITE, with the extracted assertion as a MIME body, as follows:

```
INVITE sip:originhost@origindomain.com SIP/2.0
Via: SIP/2.0/UDP proxyserver.origindomain.com
To: Target <sip:targethost@targetdomain.com>
From: Origin <sip:originhost@origindomain.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Contact: <sip:originhost@origindomain.com>
Content-Type: multipart/mixed; boundary=unique-boundary-1

--unique-boundary-1

(SDP body)

--unique-boundary-1

Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;handling=required

<NAME="TARGET" VALUE="<Target>">
<NAME="SAMLResponse" VALUE="<SAMLResponse>">

--boundary42--

--unique-boundary-1
```

Confidentiality and message integrity **MUST** be maintained for the SIP request in step 3. It is **RECOMMENDED** that the message transfer be protected by SSL 3.0 or TLS 1.0. Otherwise, the assertions transmitted in step 3 will be available in plain text to any attacker who might then impersonate the assertion subject. [7]

Step 4: Responding to the INVITE:

In step 4, assuming the user is authorized to initiate the session, the INVITE is forwarded by the inbound proxy server to the target user's UA. The SIP response **MUST** conform to [1]. The target domain PEP **SHOULD** provide some form of helpful error message in the case where the originating user is not authorized to initiate a session with the target user.

4.7. UA – initiated profiles

The profiles discussed so far have both been proxy-initiated, i.e. have required explicit action on the part of the outgoing SIP proxy in order to transfer assertions. It is possible to create a version of the profiles that does not depend on the proxy for any kind of intervention. The responsibility of invoking the profile would rest with the UA.

The key to defining these profiles is to have the UAC examine its request for the presence of necessary artifacts/assertions, and request the AS for them if needed. It would then attach the artifacts/assertions itself and send the request across. The

UAS, on receiving the artifacts/assertions, would forward them to its AS for processing and would receive user-readable attributes to display as caller information. This effectively converts the UAS into the PEP.

The relative merits of this approach vis-à-vis the proxy-initiated approach is that requests do not have to be routed through the proxies and it is more in consonance with the peer-to-peer philosophy of SIP. It also allows the UAS to function as a PEP or a PDP (depending on the amount of information given to it by the AS).

A drawback is that it may not be very practical, given that most implementations of SIP tend to route requests through proxies.

4.8. Bindings discussion

[7] defines SAML protocol bindings as “mappings from SAML request-response message exchanges into standard messaging or communication protocols”. A SIP binding for SAML would therefore describe the structure of a SIP message that would carry SAML requests and response messages. We find, however, that a binding is needed only when the artifact profile is used. In such a case, the binding would be used in steps 4 & 5. Since this communication would be between two AS’s (which aren’t SIP entities), the messaging protocol between them doesn’t have to be SIP. Thus, we conclude that a SIP binding for SAML is not needed. The existing binding, described in [7], can be re-used for the purpose of the AS-AS messaging. [11]

5. Implementation of Authorization Service

This chapter describes a sample implementation of the authorization service (AS) as defined in the previous chapter. This implementation serves as a proof-of-concept.

5.1. Overview

The idea of an authorization service was first introduced by Peterson, et al in a document on role-based authorization requirements for SIP [4]. He describes a framework where a UAC will send a request to an AS for an assertion, which it will receive once it has authenticated with this AS. Any suitable mechanism can be used for this authentication. Once the assertion has been provided to the UAC, it will need a mechanism of some sort to carry it. This mechanism is provided by the profiles, which are described in the previous chapter. This chapter addresses the details of the authorization service and a specific implementation of it.

The authorization service (AS) is a logical entity defined for the purpose of performing role-based authorization. As we have seen in earlier chapters, role-based authorization in SIP involves local authentication of the UA, transferring of information about this authentication to the target/remote domain with a request by the UAC, and an authorization decision by the remote domain on the basis of this information.

The AS implementation developed as part of this thesis, is essentially a software module that resides on the same machine as the proxy server. It runs as part of the SIP environment of the Vovida Open Communication Application Library (VOCAL) project, which is “an open-source project targeted at facilitating the adoption of VoIP in the marketplace” developed by Vovida.[23] It uses the OpenSAML libraries created by Scott Cantor of Ohio State University as part of the Shibboleth project [24]. OpenSAML is a set of open-source libraries that can be used to create, transport, and parse SAML messages.

It has been shown that role-based authorization using SAML can be achieved by using attributes to define the roles of a user. The attributes to be exchanged will depend on the trust agreement between the domains. Since this implementation of the AS is a very basic one, it considers only the attributes required to describe local authentication. Thus, from a SAML perspective, the AS is only going to handle authentication assertions. The extensions required to support a full-fledged RBA service are simply extensions to a data repository that stores the role mappings for each user, and an interface to it. This AS has been created to implement the assertion profile alone. As a result, it does not attempt to create artifacts or use any bindings.

5.2. Functionality description

The principal functions of the AS implementation are – log relevant authentication information, create SAML authentication assertions using it, and process it to make authorization decisions based on existing policies.

Local authentication: The objective here is to perform local authentication and record the necessary information for use in creating the assertions. There are two UAs available with VOCAL – `gua`, the command-line version, and `sipset`, the GUI version. The latter version was used in this consideration. As soon as `sipset` is invoked, it attempts to register with the registrar whose details it is configured with. In the VOCAL system, a request from a UAC is always routed through a Marshal server (VOCAL's implementation of a proxy). A Marshal server acts as a proxy for a certain set of UAs. In case the user account has been configured with a passphrase, it is challenged by the Marshal when it attempts to register. The request is forwarded to the registrar only when the challenge is successfully answered with the passphrase.

The above exchange explains the reason for locating the AS with the proxy. Logging authentication information in this case involves tracking the appropriate registration request and extracting the pertinent header information. Since the proxy is performing the process of local authentication in this case, locating the AS here simplifies this process considerably (especially since the other functions of the AS are fairly independent of the SIP architecture). Also, since this thesis is

concerned with the most basic implementation, the headers required for the simplest authentication assertion are chosen to be – subject, issuer, authentication method, and instant of authentication.

Creating assertions: The next step is to transcode the logged information into XML strings, using the Xerces library [25]. Xerces is a tool for converting to and from XML documents and is installed as a dependency along with OpenSAML. These strings can then be passed to OpenSAML library functions to create the assertions.

However, the OpenSAML library is a work in progress and does not represent a full implementation. Moreover, certain critical functions cannot be used as they are. This is due to the fact that this AS implementation creates a set of parameters that do not map exactly with the parameters that need to be passed to these functions. Therefore, slightly different versions of the required functions from OpenSAML need to be written.

The classes that need to be re-created are the ones responsible for the authentication statements and the assertions. At a high-level, this is done by creating a Document Object Model (DOM) element, *SAMLAuthenticationStatement*, and giving it the (DOM) attributes of subject, authentication method, and authentication instant. The statement is then passed as

a parameter to a class that constructs the assertion by creating a DOM element, *SAMLAAssertion*, and giving it the attributes of issuer, ‘not before’ and ‘not on or after’ (which specify the lifetime of the assertion and are calculated from the instant of authentication), and the authentication statement as a child.

Processing assertions: Once the assertions are created, the next step involves successfully parsing them to specify decisions, which could be sent to the proxy in a variety of ways, depending on the policies in place. For instance, if the policy mandated only a clear accept or reject decision, these could be wired to the response triggers in the proxy to accordingly enforce the authorization decision. If the UAS needs more information (which could be a subset of the total information contained in the assertion), it could be provided with the INVITE in, possibly, a displayable format.

In order to implement the profiles described in the previous chapter, there are certain changes required to the SIP architecture. The development of an AS is one of them. Other changes include modification of the SIP proxy to be able to transfer control to the AS on various junctures during a call setup process – immediately after successful local authentication with the appropriate information, on receiving an outgoing INVITE without any assertion attached, and on receiving an incoming INVITE with assertions attached. These modifications are being developed by Ameet Kulkarni and will be presented as

his thesis. There are also certain requirements on the UA – the ability to attach the assertions carried by the 428 response. These modifications are being developed by Mudassir Fajandar and are presented as part of his thesis.

5.3. Implementation details & challenges

This section describes the implementation details of the AS and the challenges faced.

5.3.1. VOCAL SIP Implementation

VOCAL's SIP stack is described in [26]. The two most relevant portions – the architecture and the state machine – are paraphrased here.

Architecture

VOCAL implements two types of software UAs – *gua*, a command-line version, and *sipset*, a GUI version. Hardware endpoints such as IP phones and gateways can also interoperate with VOCAL. VOCAL servers, on the other hand, are of three main types – Marshal servers, Feature servers, and Redirect servers.

Marshal servers are devices which, essentially, receive all incoming requests, authenticate the users sending them, and forward these requests to the appropriate entity. They can be either gateway Marshal servers (that work with signals coming from and going to PSTN gateways), Internetwork Marshal servers (that work with signals coming from and going to a known SIP proxy on another IP network), UA Marshal servers (that work with signals coming from or going to

UAs connected to the network), or Conference Bridge Marshal servers (that work with signals destined for an ad hoc conference call).

Feature servers provide enhanced telephony services such as call blocking and call forwarding. In essence, the functionality of a SIP proxy as defined in [1] has been split into the Marshal and Feature servers.

The Redirect server receives SIP registration requests, maintains a record of registered users and their locations, and provides routing information. Again, this is essentially a combination of the SIP registrar, redirect server, and location service as described in [1].

State Machine

The state machine is at the heart of all the VOCAL components. The base code, which all the components are built on, is based on the state machine. The general model provides a class for each state. The constructor for each of these classes lists operators that are instantiated at the same time as the class. Each of the operators, in turn, is written as a separate class.

When an event is sent to a state machine, it passes the event to the current state and then to each of the operators listed in the constructor for that state. Each

operator executes for the events it is enabled to work with and returns the next state for the state machine.

The main types of classes are the operator, state, feature, and builder classes. The operator classes all tend to have a function called *process* that does most of the work. The first task performed by this function is to check the relevance of the event, returning a zero if it is not. A zero is also returned if the operator is not responsible for a conversion to the next state. The main features of the state class include the name function, which enables printing a state name representation for debugging purposes, and the constructor, which uses an add operator call to create a list of operators that must be executed in this state. Feature classes are provided for developers to implement alternate state machines based on the same messages. Builder classes are used to build all the other components.

5.3.2. Local authentication

Local authentication in this case is tied-in to the registration process. The reason for this is very simple – it is necessary for local authentication to be performed before a user can attempt to send any kind of request. It is also necessary in SIP for a UA to be registered before it sends out an INVITE. Performing local authentication during registration would, therefore, ensure that local authentication has been performed prior to any INVITE being sent.

The registration process basically consists of a UA sending a REGISTER request to the registrar, which creates a mapping between the address of record (for e.g. an IP address) and the SIP URI. [1] In order to force local authentication in this process, the UA must be challenged before the REGISTER request can be successfully processed.

In VOCAL, this authentication is performed by the UA Marshal server (referred to as Marshal server from hereon). In the VOCAL architecture, all UA requests have to be proxied by a Marshal, which also authenticates the request. A REGISTER request from the UA will, therefore, be received first by the Marshal server. This request is rejected with a 401 Unauthorized[†], containing a nonce, and requesting a password. The UA then sends another REGISTER request and this time includes the password. The request is then forwarded to the Redirect server (that contains the registrar server).

The Marshal operator that operates on the REGISTER message is the `MrshlOpRegister` class. Apart from performing the functions described in the previous paragraph, this class also stores the messages received and sent in the transaction.

[†] While the SIP proxy normally authenticates using a 407 Proxy Authentication Required, in this case it is authenticating on behalf of the registrar and therefore uses a 401.

The first module of the AS, consequently, is a UNIX shell script that pulls out the second REGISTER message (the one that has successfully authenticated) and extracts the appropriate headers, i.e. the subject from the 'From' header, the method from the 'Authorization' header, and the domain name from the 'Request-URI' header. The script also records the time of this information being created as the instant of authentication. This information is then formatted appropriately and stored into a file that is labeled with a timestamp.

5.3.3. Creating assertions

The first step here is to transcode the stored authentication information into XML strings of type XMLCh*, using the transcode function from the Xerces library. Xerces is an XML parser that allows an application to read and write XML data. This transcoding is done in order to use the library functions provided by OpenSAML, which takes in XMLCh* parameters for most of its functions.

While this process involved simply reusing the transcode function (XMLString::transcode()), there were some challenges encountered in getting this to work. The main issue was the use of the g++3 compiler, which necessitated that the namespace be declared along with the class of the function. The XML Platform Utilities had to be instantiated with a locale. The default, "C" locale, was chosen.

Once the XMLCh* strings were created, they were ready to be used with the OpenSAML library functions. However, as mentioned in the previous section, the OpenSAML library functions, particularly those for creating authentication statements and assertions, have a very specific format with little flexibility as to their contents. Some of these contents or parameters are not available to the AS and some are not applicable to SIP. Extensions are, therefore, needed to the OpenSAML libraries. While extending OpenSAML libraries was not originally part of the scope of this thesis, it is necessary for the working of the implementation and will therefore be pursued as necessary future work. The rest of this section, therefore, describes the necessary code specific to this implementation, i.e. the creation of authentication statements and assertions.

Document Object Model

The concept of a Document Object Model (DOM) is briefly described here. The DOM is an application programming interface for XML documents [27]. It defines a logical structure for navigating XML documents and modifying them. XML documents in the DOM tend to have the structure of one or more trees and are modeled using objects. This model describes the structure, behavior, and objects contained in the document.

Authentication Statement

The authentication parameters to be conveyed are subject, authentication method, authentication instant, and issuer. Of these, the first three will be contained in the authentication statement, and the issuer will be contained in the assertion.

The new `SAMLAuthenticationStatement` class will contain the usual constructors, destructor, a function to examine an authentication statement and a function to create one. The constructor will instantiate the class using the three parameters mentioned above.

In the function `toDOM()` that will create the authentication statement, a DOM document is first instantiated, and a DOM element (Xerces class `DOMElement*`) representing the authentication statement is created in it. The attributes of this element are created using the `setAttributeNS()` method. These attributes are authentication method, authentication instant, and subject. While subject is a SAML object in the OpenSAML libraries and consequently appended as a child, for this implementation it will suffice to define it as an `XMLCh*` string and an attribute of the authentication statement element. The DOM tree for authentication statement is shown in fig 5.1.

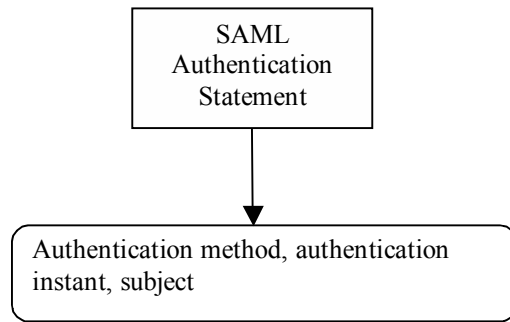


Fig 5.1: DOM tree for SAML Authentication Statement

Authentication Assertion

A SAML authentication assertion can be composed of any number of authentication statements created by a particular domain, identified by 'issuer'. In this implementation, it will contain just the one described above. No conditions other than 'not before' and 'not on or after' will be applied.

The new SAMLAssertion class will contain the usual constructors, destructor, a function to examine an assertion, and a function to create one. The constructor will be instantiated using these parameters – issuer, not before, not on or after, and the authentication statement.

In the function toDOM() that will create the assertion, a DOM document is first instantiated, and a DOM element (Xerces class DOMELEMENT*) representing the assertion is created in it. The attributes of this element are created using the

setAttributeNS() method. Issuer will be the only attribute of this element. Two children, the authentication statement and the conditions, would be appended. The DOM statement for assertion is given in fig. 5.2.

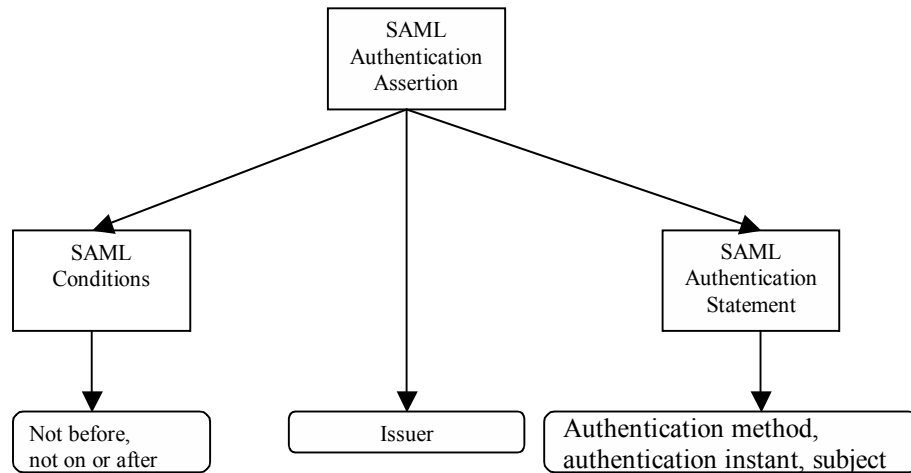


Fig 5.2: DOM tree for SAML Assertion

5.3.4. Processing assertions

For processing assertions, the parser available with the OpenSAML libraries will be used.

This section examined the authorization service and a sample implementation of it. The AS implementation is responsible for logging information pertaining to local authentication, create SAML assertions, transfer them and process them to make authorization decisions. The next chapter analyses the profiles presented in chapter 4 for conformance and security performance.

6. Analysis

This section analyses the work presented in this thesis. The analysis is done on two fronts. First, the profiles are analyzed for their conformance to the OASIS guidelines. Secondly, an analysis of the threats and countermeasures is presented.

6.1. Conformance of profiles to OASIS guidelines

The following guidelines have been specified in the protocol bindings and profiles draft for new bindings and profiles [7]. The compliance of the profiles described in Chapter 4 is given alongside in each case.

1. Describe the set of interactions between parties involved in the binding or profile. Any restriction on applications used by each party and the protocols involved in each interaction must be explicitly called out

Compliance: - All message flows have been clearly laid out in detail describing the interactions between the various parties.

2. Identify the parties involved in each interaction, including: how many parties are involved, and whether intermediaries may be involved.

Compliance: - The parties involved have been identified in the message flows.

3. Specify the method of authentication of parties involved in each interaction, including whether authentication is required and acceptable authentication types.

Compliance: - Parties exchanging artifacts or assertions must authenticate each other. However, no method of authentication is specified as part of the profiles. This is because a trust agreement is assumed between the domains involved that specifies the mode of authentication.

4. Identify the level of support for message integrity. What mechanisms are used to ensure message integrity?

Compliance: - Message integrity is a must for most of the interactions. Where needed, SSL 3.0 or TLS 1.0 will be used to guarantee message integrity.

5. Identify the level of support for confidentiality, including whether a third party may view the contents of SAML messages and assertions, whether the binding or profile requires confidentiality and the mechanisms recommended for achieving confidentiality.

Compliance: - Confidentiality is a must for most of the interactions. Where needed, SSL 3.0 or TLS 1.0 will be used to guarantee message integrity.

6. Identify the error states, including the error states at each participant, especially those that receive and process SAML assertions or messages.

Compliance: - The error states have been explicitly identified for each applicable step.

7. Identify security considerations, including analysis of threats and description of countermeasures.

Compliance: - These issues are addressed in section 6.4

8. Identify SAML confirmation method identifiers defined and/or utilized by the binding or profile.

Compliance: - No such identifiers are used by the profiles. No binding has been defined.

6.2. Security Analysis

This section provides a threat model and countermeasures analysis of the profiles.

6.2.1. Artifact Profile Security Model

This subsection considers the threat model and counter measures for the artifact model described in section 4.5

6.2.1.1. Stolen or Modified Artifact

The threat is that an eavesdropper could intercept and reuse an artifact. In general, confidentiality, bilateral authentication and message integrity must be

applied across this exchange. Much of this protection comes from the use of underlying protocol security measures. Furthermore, within the message exchanges the use of accurate time stamps to provide a time-to-live feature could assist in preventing reuse. These countermeasures would also address the potential for man-in-the-middle attacks.

6.2.1.2. Malicious Destination Domain

The threat is that a malicious intended user could attempt to reuse the artifact. The countermeasure relies on authentication of a source to a destination in order to obtain the assertions associated with an artifact. In other words, the site would be unable use the artifact since it does not have the appropriate origin.

6.2.1.3. Forged Artifact

The threat is that a malicious user could forge an artifact. This threat is addressed by the design of artifacts wherein it is infeasible to guess the value. The particular design is outside of the scope of this document but considered in [28]. In addition, measures could be taken to establish alarm thresholds on repeat requests to prevent brutal force guessing attacks.

6.2.1.4. User Agent State Exposure

The threat involves the storage of the artifact in some persistent memory associated with the user agent. The artifact could be stolen and reused. The countermeasure is the one-use property of the artifact as expressed in associated time stamps.

6.2.2. Assertion Profile Security Model

This subsection considers the threat model and counter measures for the assertion model described in section 4.6.

6.2.2.1. Stolen or Modified Assertions

The threat is that an eavesdropper could intercept and reuse an assertion. In general, confidentiality, bilateral authentication and message integrity must be applied across this exchange. Much of this protection comes from the use of underlying protocol security measures. Furthermore, within the message exchanges the use of accurate time stamps to provide a time-to-live feature could assist in preventing reuse. These countermeasures would also address the potential for man-in-the-middle attacks.

6.2.2.2. Forged Assertion

The threat is that a malicious intended user could attempt to reuse the assertion. The countermeasure relies on the assertion to be signed, thereby providing

authentication and integrity. The destination site is therefore required to check the signature.

6.2.2.3. User Agent State Exposure

The threat involves the storage of the assertion in some persistent memory associated with the user agent. The assertion could be stolen and reused. The countermeasure is to include a short lifetime with the assertion, one that would limit probably of reuse.

7. Conclusions & Future Work

This chapter presents the conclusions for this thesis and future work

7.1. Conclusions

This thesis has presented mechanisms for providing federated, role-based authorization capabilities to SIP. The present specification for authorization in SIP is an extremely rudimentary. We have seen that a better and more sophisticated authorization framework can be achieved by the usage of federated, role-based authorization. This framework allows the expression of highly granular authorization policies and is also very scalable. SAML is the protocol chosen to help provide the necessary security information to implement this framework. This thesis has defined the mechanisms necessary to use SAML in SIP.

Along with the mechanisms for role-based authorization in SIP, this thesis also described a sample implementation of an authorization service to use these mechanisms. An analysis of all the possible threats and their countermeasures was conducted. The compliance of the profiles to the guidelines defined by OASIS was also examined.

7.2. *Future Work*

The implementation of the AS described in this thesis is a partial implementation, on account of the need for extensions to the OpenSAML libraries. The necessary future work is to complete the implementation by extending the OpenSAML libraries and using them.

Other future work includes extending this AS to interface with a data repository containing user-role mappings for the creation of appropriate attribute assertions. Integrating this AS with the modified SIP UA and proxy would yield a complete architecture that provides enhanced authorization capabilities to SIP.

Bibliography

- [1] Rosenberg J., Schulzrinne H., Camarillo G., Johnston A., Peterson J., Sparks R., Handley M., Schooler E, "SIP: Session Initiation Protocol", Internet RFC 3261, <http://www.ietf.org/rfc/rfc3261.txt>.

- [2] Ferraiolo D., Kuhn R., "Role-Based Access Controls", Proceedings of the 15th National Computer Security Conference, Baltimore MD, October 1992.

- [3] Sicker, D., Kulkarni, A., Chavali, A., Fajandar, M., "A Federated Model for Secure Web-Based Videoconferencing", IEEE Computer Society Press Proceedings of the International Conference on Information Technology: Coding and Computing, 2003.

- [4] Peterson J., Polk J., Sicker D, "Role-based Authorization Requirements for the Session Initiation Protocol", SIPPING-WG Internet Draft, March 2003.

- [5] "Role Based Access Control", National Institute of Standards and Technology. <http://csrc.nist.gov/rbac>.

- [6] SAML 1.0 Specification Set (31 May 2002): Committee Specifications (OASIS Standard as of 5-Nov-2002)
<http://www.oasis-open.org/committees/security/>.

- [7] Mishra P., et al., "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)", OASIS, May 2002,
<http://www.oasis-open.org/committees/security/>.

- [8] Peterson, J., " SIP Authenticated Identity Body (AIB) Format", draft-ietf-sip-authid-body-01, February 2003.

- [9] Sicker D., Chavali A., Kulkarni A., "SIP Bindings and Profiles for SAML", Internet Draft (Work in progress).

- [10] Home page, Video-Middleware Working group at Internet2,
<http://middleware.internet2.edu/video/>.

- [11] Sicker D., Chavali A., Kulkarni A., “Role Based Authorization of Session Initiation Protocol based on SAML”, Submitted to ITCC 2004.
- [12] Anderson R., “Security Engineering: A Guide to Building Dependable Distributed Systems”, John Wiley & Sons, 2001.
- [13] Proctor P. E., Byrnes F., C., “The Secured Enterprise: Protecting your Information Assets”, Prentice Hall PTR, 2002.
- [14] Kaufman C., Perlman R., Speciner M., “Network Security: Private Communication in a Public World”, 2nd edition, Prentice Hall TR, 2002.
- [15] Kobiellus J., “Federation key to web services”, Network World, April 2002,
<http://www.nwfusion.com/columnists/2002/0429kobiellus.html>.
- [16] Home page, Federating Organizations Organization (FOO), Internet2,
<http://middleware.internet2.edu/foo/>.
- [17] Norlin E., Durand A., “Federated Identity Management”, PingID Whitepaper, 2002,
http://www.pingid.com/downloads/Whitepaper_Identity_Federation.pdf.
- [18] Carr D. F., “What’s Federated Identity Management”, eWeek, November 2003,
<http://www.eweek.com/article2/0,4149,1378431,00.asp>.
- [19] Rouault J., “Introduction to SAML”, Presentation, January 2002,
http://www.simc-inc.org/archive0002/February02/devwed1015_rouault.pdf.
- [20] Maler E., et al., “Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1”, OASIS, September 2003.
- [21] Barkley J., “Comparing Simple Role Based Access Control Models and Access Control Lists”, Second ACM workshop on Role-based Access Controls, 1997.
- [22] Home page, Network Convergence Lab, Claremont Graduate University

<http://ncl.cgu.edu/>.

[23] Home page, Vovida, <http://www.vovida.org>.

[24] Home page, Shibboleth working group, Internet2, <http://shibboleth.internet2.edu>.

[25] Home page, Xerces C++, version 2.4.0, <http://xml.apache.org/xerces-c/index.html>.

[26] Dang L., Jennings C., Kelly D., “Practical VoIP Using VOCAL”, O’Reilly & Associates, 2002.

[27] Wood L., et al., “Document Object Model (DOM) Level 1 Specification, Version 1.0”, W3C, October 1998.

[28] Maler E., et al., “Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V1.1”, OASIS, September 2003.

Appendix A

Key ideas in the management of role-based authorization

This section provides an example of how role-based authorization can be managed in a particular domain. The diagrams shown here are oversimplifications and are intended to convey just the key points of managing role-based authorization.

Example of a role-user mapping

Role	Users
Computer Science affiliation	Alice, Bob, Eve, Mallory, ...
Faculty Member	Adam, Eve, Dave, ...
XYZ research group members	Eve, Alice, Joe

The above table illustrates associations of users to each role. Role administration involves essentially adding and removing users from such a table.

Example of a role-role mapping for authorization

Role	Computer Science affiliation	Faculty Member	XYZ research group members
Computer Science affiliation			
Faculty Member		X	
XYZ research group members	X	X	X

The above table illustrates a role-role mapping for authorizing requests. For instance, members of the XYZ research group will only be allowed to receive requests from users that have the roles of ‘Computer Science affiliation’, ‘Faculty Member’, and ‘XYZ research group member’. Faculty members can receive requests from users with the role ‘Faculty member’. Users with just a computer science affiliation cannot receive requests from anyone.

We see from the first diagram that Eve has three roles. A very restrictive authorization policy would apply the authorization requirements of all the roles. A less restrictive policy would permit the requirements of any of the roles to be applied. So, in the earlier scenario, Eve can receive requests only from users which have all three roles, while in the latter case, it can receive requests from users that are just faculty members.