

Functional Specification for UCCAP Attribute Server and Attribute Query Protocol

Sal Gurnani
August 20, 1998

1. Background and Purpose

The payload of a UC Personal certificate, as defined in the July 1998 UCCAP Report [1], will not contain any timely attribute information appropriate for use in authorization and access control decisions commonly made by web applications. However, the payload will include the Netid or unique identifier for the owner of the certificate as well as a pointer to a Campus Attribute Server from which the attributes assigned to that individual may be queried. During a secure HTTP session between a web browser and server, a Personal certificate can be requested by a web application, and the Netid/Authorization pointer pair may be used to request attribute information. This document proposes a functional specification and an implementation plan for a Campus Attribute Server and an associated Attribute Query Protocol.

2. Functional Specification

Attribute Server

The Campus Attribute Server is defined as an online query gateway to database information assigned to selected individuals. This information could reside as data fields possibly stored in a variety of information sources including the University Directory[2], a local proprietary database, a campus LDAP directory, etc. The function of the Attribute Server is to receive and process queries on the values of those data fields while enforcing a desired access control policy. The Attribute Server may also support it's own Attributes, the values for which are determined by logical operations upon actual database fields. For example, a web application attempting to determine whether an individual is a registered graduate student could determine this via the following query: "What are the values for the fields *GraduateStatus*, *RegistrationStatus*, and *PaymentStatus* for the individual referred to by Netid *N* a registered graduate student?" These fields would most likely reside in a restricted online Registrar's database, and a Registrar's policy on student privacy could severely restrict publication of this information. Alternatively, the application could contact an Attribute Server supporting the Attribute *GraduateStudentStatus* and make the query: "What is the value of the Attribute Graduate Student Registration Status for the individual with Netid *N*?" The policy restricting publication of this Attribute would probably be less restrictive than the policy on individual fields used to determine its value. To store these Attributes an Attribute Server would have to contain an internal database. This database could also be used to maintain the required access control rules used to enforce a desired attribute access policy.

Attribute Query Protocol (AQP)

The Attribute Query Protocol (AQP) must provide support for authenticating to and querying the Campus Attribute Server over a secure encrypted transaction. Queries should consist of an initial authentication phase followed by a request for the value(s) of a given attribute(s) assigned to an individual designated by a Netid or similar unique identifier. The response should consist of one of the following: the information requested, a referral to another data source, or an error code. The protocol must necessarily be TCP based and should be founded upon accepted standards for authentication and encryption, query formulation, and formatted data exchange. Due effort should be taken to insure that the format in which queries are formulated and the subsequent results should conform to an IETF or ISO approved standard if applicable.

3. Discussion of Attributes and Roles

Attributes

Attributes allow for the greatest flexibility in determining and enforcing an access control policy. By restricting access to discrete database field information via Attributes, it becomes unnecessary to incorporate access control rules directly into legacy data sources. This also precludes the need to support multiple, possibly conflicting access control policies within each data source. Alternatively, the Attribute Server can be given access to a larger range of fields within the underlying data sources and trusted to enforce the appropriate access policy governing a smaller set of well defined fields called Attributes. To provide additional flexibility, it should be possible to create a one-to-one correspondence between an Attribute and an actual field/table within a given data source. It should also be possible to allow an Attribute to inherit the native access control restrictions of that database field if such a scenario is desirable.

Roles

A Role is defined as an Attribute that can be used to designate an individual's status, duties, responsibilities, and or privileges within a community[3]. Some examples of roles that would apply to UC community members are Student, Faculty, Staff, and Member. The last, Member, indicates that at least one of the first three applies to an individual. These roles do not in themselves imply any privileges or responsibilities, but do allow for grouping of individuals in a way analogous to how Campus specific and UC-wide services like the libraries commonly delineate levels of service. The value of the Attribute "UC Student" was probably determined by examining several fields in a Campus Registrar's database possibly including restricted fields like "Fee Payment Status", etc. However, by only divulging the value of the Attribute/Role "UC Student", enforcement of a policy that conforms to federal privacy regulations, i.e. FERPA, becomes much simpler and easier to implement.

4. UC-Wide Roles

Certain UC applications, most notably library applications, would greatly benefit from the definition of roles having UC-wide context and definition. These UC-wide applications could use these roles to delineate and assign different levels of service if applicable. After some discussions with CDL staff dealing with various library applications having UC-wide scope including external licensing and Patron Initiated Request (PIR), the following roles were mentioned and found to be important:

UC Student
 UC Faculty
 UC Staff
 UC Member

This is by no means an exhaustive list, but is meant to represent a good starting point for developing algorithms for assigning and using roles effectively as well as determining an appropriate access policy for these new Role Attributes. The data sources and field values which determine these UC-wide will need to be agreed upon by a group of representatives from the Campus Libraries, CDL, UCOP, and various cross-campus committees. These definitions will likely correspond to the current way the libraries, Campus Registrars or Payroll systems classify individuals.

5. Campus Specific Roles

Each UC campus will need to explore the needs of local service applications to determine the appropriate roles and groups that need to be defined. A likely set of roles is the parallel to the UC-wide roles mentioned:

UC Campus Student
 UC Campus Faculty
 UC Campus Staff
 UC Campus Member

The definitions of these three roles could be identical to the UC-wide definitions or differ based on local campus policy.

6. Implementation Proposal

Attribute Query Protocol: LDAP w/ SSL Authentication

The Lightweight Directory Access Protocol v3 [4] seems to be the most appropriate IETF standard for use as an Attribute Query Protocol. LDAP v3 supports client authentication, a flexible querying syntax, well defined response syntax, replication, as well as database entry modification, addition, and deletion.

The LDAP v3 specification currently supports simple password, kerberos credentials, and SASL, which is extensible enough to support authentication via SSL and X.509 digital certificates. The Directory Server from Netscape Communications, Inc. [5] and the eNetwork X.500 Directory from IBM [6] are two commercial LDAP server products supporting SSL based authentication. To achieve a non-commercial solution, it should be possible to add SSL support to the University of Michigan's reference implementation of an LDAP server using the freeware SSLEAY or similar SSL development package from RSA, etc. Alternatively, one of the commonly available HTTP-LDAP gateway clients could be used in combination with an HTTPS server and LDAP directory to achieve an SSL connection and proxy LDAP authentication and queries.

LDAP supports querying of directory entries, each of which is identified by a unique Distinguished Name (DN) and is composed of one or more named attributes. These attributes may be of varying data types and may have multiple values. An LDAP query can also be encoded as an LDAP URL, having the following form [7]:

```
ldap://server.domain/DN?attributelist?scope?filter
```

Where DN is the distinguished name of a database entry or branchpoint, attributelist is a comma delimited list of attributes to return, scope is the depth to search within the database, and filter is the attribute value requirements for the entries returned. It follows that the LDAP URL for the query "What is the value of the Attribute *GraduateStudentStatus* for the individual with Netid *N*?" would look like:

```
ldap://authorization.ucal.edu/ O=University of  
California,C=US?GraduateStudentStatus?sub?(Netid=N)
```

The response to an LDAP query can be a list of zero or more matching entries containing all the requested attributes minus those restricted under access control rules defined in the underlying database. Alternatively, the response could be a list of one or more referrals or an error code. A referral is essentially a URL which points to another online data source, LDAP or otherwise, and is returned in the case where the LDAP directory does not contain the requested information but is aware of a data source where the information may be obtained. An error code can be used to indicate an authentication failure or denial of service due to access control restrictions.

A powerful feature of LDAP is its support for modification, addition, and deletion of entries. With strong authentication and a good access control rules, these features can be used to perform real time incremental updates to entries and attributes as well as replication between LDAP capable databases. Since the entries and associated attributes will most likely be gleaned from multiple and disparate data sources, use of these functions would be necessary to ensure that the directory is kept current.

Attribute Server: LDAP Directory vs. LDAP Front-End

The Campus Attribute Server could be implemented as either an actual LDAP Directory Server or as a proprietary database (SQL, etc.) with an LDAP front end. A commercial LDAP Directory Server is normally implemented as an X.500 database. The appropriate

choice will depend on the technical resources available and the current policies surrounding access and use of the legacy data sources.

In the case where an actual LDAP Directory Server is desired, it is preferable to use the Netscape Directory Server product since it supports SSL and X.509 certificates natively and is available at a nominal cost to educational institutions. If source code is desired, the LDAP server reference implementation produced by the University of Michigan is an appropriate choice. However, this reference implementation does not support SSL, though an SSL layer could be added via the SSLeay libraries.

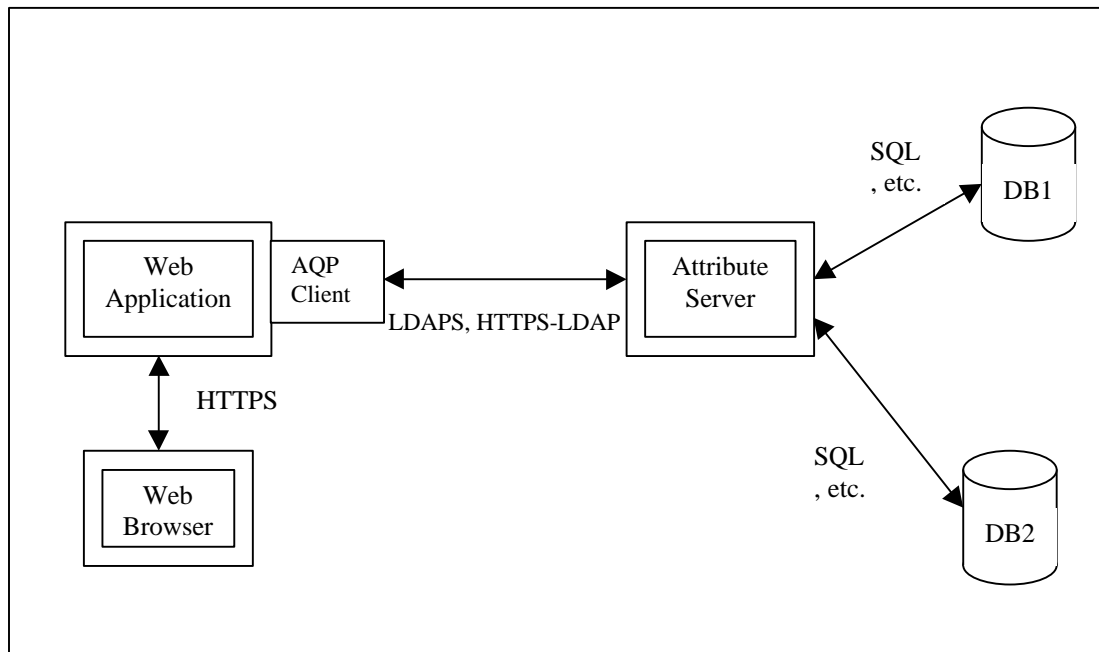
If it is desirable to maintain the attribute information within an existing directory or database that is not accessible via LDAP, it will be necessary to write or purchase an LDAP front-end or HTTPS-LDAP gateway. The reference implementation of LDAP from the University of Michigan currently supports a rudimentary Unix database based on DBM. If a commercial LDAP gateway is unavailable, it should be possible to modify this source code to support SQL, etc. A software development toolkit from the database vendor may be necessary to achieve this more easily. The LDAP front end would also need to be enabled for SSL using the SSLeay libraries or a similar toolkit. This scenario is the main incentive for using an HTTPS-LDAP gateway to access the Attribute Server, rather than LDAP over SSL.

AQP Clients

In order for a web application to communicate with the Campus Attribute Server, an appropriate client program that understands the Attribute Query Protocol is required. In the case where the AQP is implemented as LDAP over SSL, an appropriate choice is the Secure LDAP client that ships with both the Netscape Certificate Server and the Netscape Directory Software Development Kit [8]. Alternatively, if the AQP is implemented as an HTTPS-LDAP gateway, any scriptable HTTPS client like Netscape Navigator, etc. may be employed to communicate with the Attribute Server.

Diagram

The following diagram depicts the interaction between a web application, and a Campus Attribute Server.



7. Conclusions and Proposed Prototype

The payload of the UC Personal Certificate issued to each member of the UC community will contain a pointer to a Campus Attribute Server, which contains information about that individual. When a web application is presented with a Personal Certificate, it may contact the designated Attribute, authenticate itself in a secure fashion, and query the server for the attribute information to which it is allowed access. The results of this query may contain consist of a list of attributes and their values, a referral to another Attribute Server, or a formatted error response. The web application may then use the information retrieved to help make local authorization and access control decisions. In addition this UC-wide Attribute Query Service must fulfill the following basic requirements as stated in the July 1998 UCCAP report:

“The authorization server can be very dynamic, with real-time updates if necessary... the service must be distributed and secure; it must be bootstrapped from the certificate; and it may be hierarchical.”

As recommended in the July UCCAP report, a prototype network of Attribute Servers should be established on each of the nine UC campuses to create an Attribute Query Service as specified in the July 1998 UCCAP Report. The Attribute Servers should be implemented

as either an LDAP directory product or as proprietary database with an LDAP front end. The appropriate choice will depend on the technical resources available and the current policies surrounding access and use of the actual data sources. The Attribute Server should also support SSL connections and authentication via X.509 certificates. . An HTTPS-LDAP gateway to the Attribute Server is an acceptable implementation. The API for this type of gateway will need to be defined by the Authentication Workgroup, and should be used by all UC Attribute Servers' wishing to make use of this option. For testing purposes, the initial data source should include the entries obtained from the University Directory extracts. Also, the following three UC-wide role attributes will be supported: UC Faculty, UC Staff, UC Student, and UC Member. The value returned for each of these attributes will be "Yes" or "No".

In addition, each UC campus may load the complete set of fields for those individuals whose records contain the campus location code. Also, for all other individuals in the University Directory, only the Netid and Location fields may be loaded. These entries will later be enhanced to include an Attribute Server pointer field, which will be used to generate a referral response. This will allow any web application to query it's local Attribute Server regarding any UC community member holding a Personal Certificate and receive a referral to the appropriate Attribute Server containing more information about that individual.

8. References

- [1] UCCAP Task Force Report, July 1998
UCCAP Task Force
<http://dcas.ucdavis.edu/authentication/archive.html>
- [2] University Directory Project (UDIR)
Vance Vaughan
UCCAP Task Force
<http://www.ucop.edu/~authuser/cap/udir.html>
- [3] Role-Based Access Control Models
Sandhu, R.S.; Coyne, E.J.; Feinstein, H.L.; Youman, C.E.
Computer, vol.29 (no. 2), IEEE Comput. Soc., Feb. 1996
- [4] RFC 2252, Lightweight Directory Access Protocol (v3). M. Wahl, T. Howes, S. Kille.
December 1997, IETF.
<http://info.internet.isi.edu/in-notes/rfc/files/rfc2251.txt>
- [5] Netscape Communications Directory Server Page
<http://home.netscape.com/directory/v3.0/index.html>
- [6] IBM eNetwork X.500 Directory Page
<http://www.software.ibm.com/enetwork/directory>
- [7] RFC 1959, An LDAP URL Format. T. Howes & M. Smith. June 1996. IETF
<http://info.internet.isi.edu/in-notes/rfc/files/rfc1959.txt>

[8] Netscape Directory Server SDK Page

<http://developer.netscape.com/software/sdks/index.html?content=/tech/directory/downloads.html>