Preliminary version; to appear in 8th IEEE/ACM International Conference on Grid Computing (Grid 2007), Sept 19-21,2007, Austin, TX

Fine-Grained Access Control for GridFTP using SecPAL

Marty Humphrey[#], Sang-Min Park[#], Jun Feng[#], Norm Beekwilder[#], Glenn Wasson[#], Jason Hogg[*], Brian LaMacchia[*], and Blair Dillaway[*]

[#]*Department of Computer Science, University of Virginia, Charlottesville, VA USA*

{ humphrey | sp2kn | jf4t | nfb5z | wasson }@cs.virginia.edu

[*]*Microsoft Corporation, 1 Microsoft Way, Redmond, WA USA*

{ jason.hogg | bal | blaird }@microsoft.com

Abstract – Grid access control policy languages today are generally one of two extremes: either extremely simplistic, or overly complex and challenging for even security experts to use. In this paper, we explicitly identify requirements for an access control policy language for Grid data and then consider six specific data access use-cases that have been problematic in today's Grids: attribute-based access, role-based access, "role-deny" access, impersonation-based access, delegation-based access, and capability-based access. We evaluate the Security Policy Assertion Language (SecPAL) against those requirements, specifically in the context of these six use-cases involving GridFTP.NET. We find that while some of these six use-cases are individually possible via existing Grid authorization systems, we believe that SecPAL uniquely offers a *single* approach that meets the requirements of a Grid access control policy language, thereby creating support for a wide range of expanded scenarios for Grid data access.

I. Introduction

A persistent challenge in Grid environments is access control. The *grid-mapfile* [1][2] approach is relatively simple, using fully-trusted authorities (e.g., the installed list of trusted Certificate Authorities) and a coarse-grained mapping from authenticated Grid users to specific accounts on the local resource. XACML [3] is representative of the other extreme, offering flexibility and explicitly separating what attributes an authority is allowed to issue vs. subsequently authorizing access based on those attributes. However, while the *grid-mapfile* is arguably too coarse-grained, XACML can be challenging to use [4][5][6][7], in part because of this flexibility. The Grid community would greatly benefit from the creation of an access control approach that features the broad capabilities of the XACML paradigm and the intuitive simplicity of the *grid-mapfile* approach.

In this paper, we report on new capabilities to express and implement fine-grained authorization policies for Grid data access. We focus on authorization in the context of the GridFTP protocol [8] (GridFTP.NET [9] in particular), but our results are applicable to the wide class of Grid data access mechanisms. We begin by identifying the requirements of an access control language and implementation for a Grid data server. In our requirements, we differentiate between the policy for data access and the policy for explicitly expressing trust relationships – the latter is used to express that a particular entity can issue authentication tokens (including what claims in such a token are acceptable) or the ability of one entity to "speak for" [10] another entity in a limited perspective. We then consider six specific data access use-cases that have been problematic in today's Grids: attribute-based access, role-based access, "role-deny" access, impersonation-based access, delegation-based access, and capability-based access. To express and implement these access control policies, we use a new security policy language from Microsoft Research (and the corresponding reference implementation in .NET) called SecPAL [11][12]. While some of these six use-cases are individually possible via existing Grid authorization systems, we believe SecPAL uniquely offers a *single* approach for all six use-cases, meeting the requirements for an access control language for Grid data access, thereby creating support for a wide range of expanded scenarios for Grid data access [13].

The rest of this paper is organized as follows. Section II introduces requirements, covers related work in current Grid access control systems, and gives an overview of SecPAL and GridFTP.NET. Section III describes how SecPAL can be used to meet the general requirements of a Grid access control policy language. Section IV includes a discussion and evaluation. Section V concludes.

II. Background

A. Requirements

Grid data access generally involves four basic entities. A *data owner* creates an original or derived set of data and wishes to make this data available to authorized parties in the Grid. The data owner entrusts one or more *resource providers* to make this data available, subject to the authorization policy of the data owner. Each resource provider can impose further authorization policies that can augment the policy of the data owner. The *virtual organization (VO)* in which the data owner and resource provider operate can mandate additional policy. For example, the virtual organization can define the set of trusted information providers utilized in the access control policies of the data owner and resource providers. The *data requester* attempts to access the data in question.

The requirements of the access control policy language for Grid data access can then be described both from a general perspective and from the perspective of each of the four entities. We identify requirements that are recognized in active practice today, as well as identify requirements that we argue will facilitate more robust and effective future Grid environments. Note that we limit our analysis to the policy *language* and try to avoid discussions regarding an implementation of the language.

- **[GENERAL-R1]** *It must be possible to express both fine-grained access control policy (e.g., method-level, file-level, data-record-level) as well as coarse-grained access control policy (e.g., service-level, host-level, or VO-level).*
- **[GENERAL-R2]** *Authorization decisions must be provably correct and should be guaranteed to terminate.*
- **[GENERAL-R3]** *It must be agnostic to existing and future authentication policies (e.g., "no cleartext passwords on the wire") and mechanisms (e.g., SSH) as well as information providers.*
- **[GENERAL-R4]** *It must be possible to broadly express policies for information sources used in the evaluation of access control policies (e.g., to specify that a particular source is authorized to provide certain information).*
- **[GENERAL-R5]** *Policies must be composable and extensible without requiring centralized policy authoring.*
- **[GENERAL-R6]** *Users should not be required to be an expert in computer security to author or understand an access control policy.*
- **[GENERAL-R7]** *It must be possible to specify a lifetime on a policy and policies should be able to be modified during their lifetime.*
- **[GENERAL-R8]** *It must be possible to delegate a subset of a principal's rights.*
- **[DATA-OWNER-R1]** *It must be possible to specify role- and attribute-based authorization policies.* Roles and attributes are well-established approaches for scalability, an obvious requirement in Grid environments.
- **[DATA-OWNER-R2]** *It must be possible to specify policy specific to an access mode and purpose of the attempted access.* For example, part of the access policy might broadly address *why* the potential access is being requested, such as so that a particular Grid job can read an important input file (that the veracity of such a claim is a non-trivial concern but is largely orthogonal to this requirement). Note that a data owner might also require that access be granted over an encrypted channel.
- **[RESOURCE-OWNER-R1]** *It must be possible to specify policy based on time and access mode.* A resource owner might wish to restrict access according to local (site) policy. For example, access to local resources via Grid mechanisms might be curtailed during normal local business hours, instead giving priority to local users.
- **[VO-R1]** *It must be possible to define an acceptable set of authorities for the virtual organization as a whole.* The VO can often define a trusted set of authorities, such as for authentication and attributes.

We note that the data requester contributes no unique requirements for an access control policy language. We attribute this to the natural client/service decoupling.

B. Existing Grid Authorization Systems

In the majority of Grid deployments today, access control is based solely on the authenticated identity of the requestor. In this *grid-mapfile* [1][2] approach, the X.509 DN is mapped to a specific account on the target machine. This approach is coarse-grained and fairly static, and thereby not able to meet many of the requirements established in Section II.A (particularly GENERAL-R1, GENERAL-R4, and DATA-OWNER-R1).

More advanced approaches have been taken in recent research on VO authorization infrastructures such as CAS [14], VOMS [15], and PRIMA [16]. For example, the approach of VOMS is to embed an attribute certificate into an X.509 certificate. A VOMS-enabled service can base access decisions on these attributes. The VOMS-enabled service is similar to the grid-mapfile approach in that a user account is ultimately selected. Certain authorization systems have focused on particular applications (e.g., CAS, Akenti [17], and PERMIS [18]). In general, these systems more comprehensively meet the requirements in Section II.A (e.g., DATA-OWNER-R1) but do not sufficiently address all requirements (e.g., GENERAL-R3, DATA-OWNER-R2 and RESOURCE-OWNER-R1).

The Open Grid Forum has recently published an "experiences document" that specifies the use of SAML for Grid authorization [20], based in part on the more generic X.812 | ISO 10181-3 access control framework [21]. This document provides a profile on the SAML AuthorizationDecisionQuery, Extended Authorization Decision Query, and Simple Authorization Decision Statement. While this OGF work is an important step toward creating interoperable interfaces for authorization services, this document does not address how to author/compose policies nor any details of what processing occurs within the authorization service.

XACML [3], an XML-based access control policy language, defines a set of complex data types, functions and combining logic to create detailed, attribute-based access control policies. XACML has been identified as being challenging to use effectively [4][5][6][7]. For example, selecting the most appropriate set of terms in XACML by which to express a policy can be non-trivial (GENERAL-R6). In addition, XACML has a "first-applicable" algorithm in which the policy evaluation terminates upon finding the first matching rule, which can sometimes result in decisions that may be complicated for a user to understand/verify. There are issues with regard to composition in XACML [7] (GENERAL-R5). It is not clear to what extent policy-controlled delegation can be expressed (GENERAL-R8).

Shibboleth [23] is a Web-based authentication and authorization system being created by Internet2.It is a federated identity infrastructure based on SAML. A recent effort called GridShib [24] integrates the Globus Toolkit with Shibboleth. Policies are XML documents that reference SAML attributes. An open issue in Shibboleth is how to specify the conditions under which an attribute authority will release attributes (GENERAL-R4, VO-R1). More broadly it can be difficult to determine in the general case the source of authority for particular attributes (GENERAL-R6).

C. SecPAL

The Security Policy Assertion Language (SecPAL) is a logic-based security policy language developed by Microsoft for addressing access control requirements for distributed systems. SecPAL has been designed to support advanced requirements such as:

- Establishing fine-grained trusts both within and across organizational boundaries
- Distributed policy authoring and composition with enforced separation of duties
- Constrained (and unconstrained) delegation of rights
- Fine grained revocation
- Cryptographically strong, Internet scale, authentication information
- Proof graphs showing the logical reasoning behind an authorization decisions

The SecPAL formal model [11] describes the abstract types, logical expressions, language semantics and rules for evaluating authorization decisions in a deterministic and efficient manner. The more practical issues of implementing this formal model have been addressed through the definition of encodings for concrete types, variables and verbs [12][25]. A .NET implementation of SecPAL is used for the evaluation in this paper.

The fundamental SecPAL concept is the *security assertion.* An assertion is a statement made by a principal that may: define a binding between a principal and an attribute; specify a principal's permissions to operate on a resource; express a trust or delegation policy; express an authorization policy; revoke a prior assertion; or declare principal identifier alias relationships. These assertions are expressed using a uniform grammar which maps directly both to simple English sentences (for human consumption) as well as an XML schema (for machine consumption). Examples in this Section use the English grammar for the sake of readability and compactness.

```
LocalAuthority says TokenService can say
   %p possesses %a (from %t1 until %t2)
where
   %t2 - %t1 <= "366.00:00:00",
   %t1 <= CurrentTime() <= %t2,
   %a matches rfc822Name:”.*@fabrikam\.com”
```

Policy extract 1 – Sample policy showing delegation of rights to assert attributes

As an example of SecPAL's expressivity, Policy extract 1 illustrates a simple policy in which a local resource guard (or Policy Enforcement Point (PEP)), the LocalAuthority, delegates the rights to assert that principals have email names in the fabrikam.com domain where such assertions may be valid for up to 366 days. Note that this can also be interpreted as a policy expressing what information the LocalAuthority trusts the TokenService to provide. In this policy, variable %a represents an attribute which must match the given rfc822 email name pattern, the %p variable represents any principal, and the variables %t1 and %t2 represent datetime values which are constrained to represent a timespan of no more than 366 days. Based on the first policy, the token service could make the assertion specified in policy extract 2 which is valid with respect to the LocalAuthority policy.

```
TokenService says Joe possesses
        rfc822Name:"Joe@fabrikam.com"
(from "2007-01-01T00:00:00Z" until "2007-12-31T00:00:00Z")
```

Policy extract 2 – Attribute assertion

As shown in more detail in Section III, these *Policy Claims* and *Security Tokens* are used when SecPAL attempts to find a logical proof for a particular *Authorization Query.* That is, the logic-based nature of SecPAL allows a *SecPAL engine* to search for a collection (or collections) of assertions by which to conclude the validity of the authorization claim, and such collection(s) of assertions are returned to invoker of the engine, along with the bindings of variables in the assertions that make the query true. Proof graphs can be generated to show the result of a SecPAL authorization decision. Such proof graphs can be persisted to support auditing requirements.

D. GridFTP.NET

GridFTP [8] is a data transfer protocol for accessing distributed data on the Grid. Its first—and still major—implementation is in the Globus Toolkit. It is based on the RFC 959 “File Transfer Protocol”, RFC 2228 “FTP Security Extensions”, and RFC 2389 “Feature Negotiation Mechanism for the File Transfer Protocol”. The GridFTP protocol is optimized for high-performance, secure, and reliable data transfer in high-bandwidth wide-area networks by providing: parallel data transfer; authentication, data integrity and confidentiality; third party control of data transfer; striped data transfer; and partial data transfer.

The GridFTP.NET service [9] we use in these experiments is architecturally similar to the Globus Toolkit V4 (GT4) GridFTP implementation, both to facilitate interoperability and to reduce the learning curve for new .NET GridFTP users who already are familiar with the GT4 architecture and services. Our implementation of GSI [1][2] (including support for proxy certificates, delegation, and GSSAPI-style mutual authentication via SSL) utilizes the Microsoft Windows Security Service Provider Interface (SSPI). SSPI allows an application on Windows to use various security models (e.g., Kerberos, NTLM) available on a computer/network without changing the interface to the security system.

Performance evaluation shows our implementation is comparable to the GT4 GridFTP on both single and parallel stream transfers, and is better on transfer of large collection of small files. More specifically, on stream mode performance on LAN and WAN, .NET GridFTP is a little slower than the GT4 GridFTP for small files – experiments showed that when the file is bigger than 128M, the performance difference becomes very small. Parallel stream performance was comparable, showing little value in a LAN but significant value in the WAN. More details can be found in [9].

Prior to the research described in this paper, we had designed and implemented a modular authorization subsystem within the GridFTP.NET service, so that we could experiment with different authorization approaches. Strictly speaking, at the time we designed this modular architecture, the only requirement was to support *gridmap*-style authorization (so that, in effect, there was one consistent policy and mechanism across our UVa Campus Grid [19], which consists of our .NET-based Grid software and GT4). Soon after this implementation, we began to experience first-hand the limitations of the *gridmap* approach, so we designed and implemented two other authorization approaches: an LDAP approach, in which a dedicated LDAP server is queried on a per-service, per-DN, and per-method basis; and an XACML-based approach that was one of the first implementations of the Open Grid Forum SAML interface for authorization [20]. Although both approaches are appropriate for particular situations, we found that in general the LDAP approach was still limited in its expressiveness and we found writing even simple policies in XACML to be difficult. It is these experiences that in part motivated our need for a general set of requirements for Grid authorization and a subsequent exploration of the use of SecPAL to meet these requirements, discussed in the rest of this paper.

III. ACHIEVING FINE-GRAINED ACCESS FOR GRID DATA: USING SECPAL WITH GRIDFTP.NET

Through our experiences with *gridmap*, LDAP, and XACML, we were able to establish the requirements of the access control language for GridFTP (these requirements are enumerated in Section II.A). Furthermore, our role as GridFTP.NET source-code maintainer and as GridFTP.NET deployer generated additional goals or requirements: Low overhead (relative to the data access operation being authorized); auditable decisions (for post-facto analysis of the correctness and integrity of the Grid data service); ability to provide to the client (in debugging mode) the justification for the authorization decision; integration with local attribute authorities; no mandatory modifications to clients; and policy that is retrievable by the client, so that the client can determine *a priori* the security tokens sufficient for access. The previous approaches (gridmap, LDAP, and XACML) failed to meet these challenges in many ways. In this section, we describe our architecture we created around SecPAL to meet these requirements. Section IV contains a discussion and evaluation of this approach.

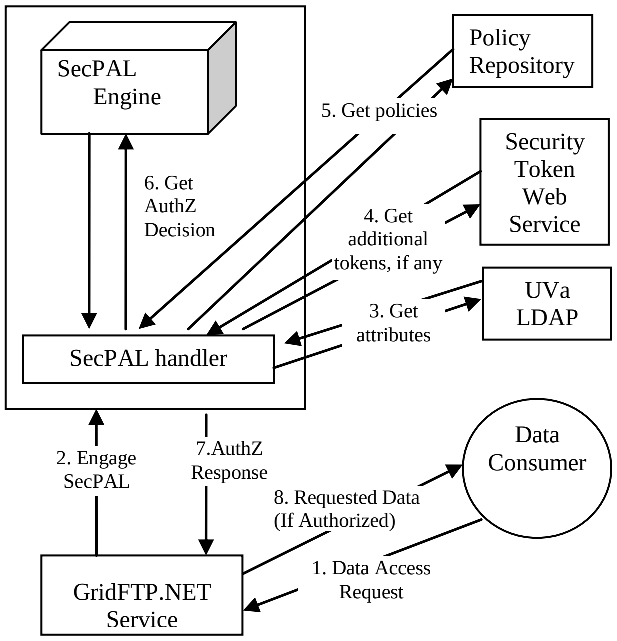

Figure 1: Using SecPAL for GridFTP.NET Authorization

Our architecture of GridFTP.NET that utilizes SecPAL is shown in Figure 1. A "SecPAL handler" was written to bridge the gap between Microsoft's SecPAL engine and our authorization call-out. A general flow of messages is:

1. The data consumer sends a request to access the data to the data custodian, which is the particular GridFTP.NET service.
2. The authorization enforcement point for GridFTP.NET has been configured to use SecPAL, so the GridFTP.NET-specific SecPAL handler will be engaged. At this point, the only information available

is the ID of the requestor (authenticated via the GSI handshake).

3. An external attribute authority is engaged to return the attributes associated with the authenticated user. The particular attribute authority we use is the UVa Central Computing LDAP server that contains information for all people on campus (not to be confused with the standalone LDAP server utilized previously). This architecture is our first attempt at using the information contained in this pre-existing UVa Central Computing LDAP server (because we did not have any mechanism prior to this project that could easily use such attributes). We were able to map several of the LDAP user properties to implemented SecPAL attributes with little effort.
4. We created a Web service to hold SecPAL tokens on a per-user basis. In Step 4, our SecPAL handler engages this Web service to retrieve any tokens relevant to the user and/or query. We discuss this in more detail in each of the use-cases, below.
5. Relevant policies are retrieved from the Policy Repository, in which they are stored as XML. In practice we stored these policies in the file system accessible by the SecPAL handler.
6. The information retrieved from Steps 3-5 are handed to the SecPAL engine, which returns an authorization decision to the SecPAL handler.
7. The SecPAL handler returns this decision to the GridFTP service.
8. The GridFTP.NET service returns the requested data to the consumer based on the authorization decision.

The Security Token Web Service and Policy Repository warrant further discussion. First, the Security Token Web Service was created because the additional information upon which we needed to make access control decisions could not be obtained from a legacy client. In other words, we needed a secure storage for such information or tokens. Our current instantiation of this service is only a proof of concept – in the future we plan to design a modified MyProxy service to hold such information. Second, the Policy Repository is currently fairly limited in its capabilities and must be expanded in the future. For example, we do not mean to imply that there is a *single* Policy Repository for the entire enterprise, but similarly it is unlikely that each service have its own Policy Repository. Such general policy management concerns (e.g., distribution, updating, locating) are problematic to *all* large-scale policy systems (not just SecPAL) and are largely outside of the scope of this work.

We now describe six use-cases we encounter that have been difficult to make operational in today's Grid environments. We use the following three principal types: *LocalAuthority (LA)* represents the resource guard responsible for making policy assertions and issuing the authorization assertion; *TokenService* is a Security Token Service that can issue cryptographically verifiable tokens to users or services; *K-User* represents a user that might request data. An example of *LocalAuthority* is the combination of a particular data owner, resource provider and VO, collectively creating the policies contained in the "Policy Repository" (Figure 1). We used X.509 certificates issued by the UVa CA ("UVA-CA"). For each case, we present the policy claims (retrieved from the Policy Repository), the Security Tokens (constructed in the SecPAL handler and also retrieved from the Security Token Web service), and the authorization query that are all passed to the SecPAL engine for evaluation. For clarity of the presentation, we use a slightly different syntax than used in the overview of SecPAL (Section II.C). Note that in many cases we did not require any specific notion of time (e.g., in policies) because of the time validation check that took place as part of the standard X.509 certificate validation processing in the GSI/SSL handshake. Note, however, that all tokens contained in the Security Token Web service had explicit time bounds.

Case 1: Attribute-based Access. We required the ability to restrict certain content to only those people who had a valid UVa email address:

Policy Claims
LA says UVa-CA can say %p possesses %a (from %t1 until %t2) where %t2 - %t1 <= "365.00:00:00", %t1 <= CurrentTime() <= %t2, %a matches rfc822Name:".*@virginia\.edu"
LA says %p can read, write digitalContent:"file:///GridFTPRoot/" if %p possesses %a where %a matches rfc822Name:".*@virginia\.edu"
Security Tokens
UVa-CA says K-User possesses rfc822Name:"humphrey@virginia.edu" (from "2007-01-01T00:00:00Z" until "2007-12-31T00:00:00Z")
Authorization Query
LA says K-User can read digitalContent:"file:///GridFTPRoot/data1.txt"?

The SecPAL engine authorizes the reading of the Grid data access, because the UVA CA states that "K-User" has a valid UVa email address, anyone with a valid UVa email address is allowed to read the particular directory, *and* the UVA CA is *explicitly allowed* to state who has a valid UVa email address. Case 1 was easy to implement, given that the UVa certificates contain the email address of the subject (hence it was easy to create a SecPAL token). As with the other five cases, the GridFTP operations mapped easily onto the SecPAL action verbs, and GridFTP assumes a hierarchical file system structure, implying the read of the root and everything under it.

Case 2: Role-based Access. We needed to quantify access for certain classes of users (e.g., faculty):

Policy Claims
LA says UVa-CA can say %p possesses %a (from %t1 until %t2) where %t2 - %t1 <= "365.00:00:00", %t1 <= CurrentTime() <= %t2, %a matches rfc822Name:".*@virginia\.edu"
LA says UVa-LDAP can say %p possesses %a (from %t1 until %t2) if %p possesses %b where %t2 - %t1 <= "365.00:00:00", %t1 <= CurrentTime() <= %t2, %a matches roleName:"^(Faculty|Student)$" %b matches rfc822Name: ".*@virginia.edu"
LA says %p can read, write digitalContent:"file:///GridFTPRoot/" if %p possesses %a, where %a matches roleName:"Faculty"
Security Tokens
UVa-CA says K-User possesses rfc822Name:"humphrey@virginia.edu" (from "2007-01-01T00:00:00Z" until "2007-12-31T00:00:00Z")
UVa-LDAP says K-User possesses roleName:"Faculty" (from "2007-01-01T00:00:00Z" until "2007-12-31T00:00:00Z")
Authorization Query
LA says K-User can read digitalContent:"file:///GridFTPRoot/data1.txt"?

Case 2 was straightforward to implement, because the existing UVa LDAP server contains faculty/student information upon which to create a SecPAL token.

Case 3: "Role-Deny" Access. Similarly, we desired to deny access based on certain roles. For example, students are forbidden from reading certain content. In general, although human policy makers can tend to think in terms of "disallowing" certain activities and allowing all others, writing such "exclusion" policies to achieve the desired result is often difficult. Negative assertions or policies can lead to a situation where policies don't compose in the obvious way -- there might be a fact which is true in one policy, but false in a superset of that same policy! Additionally, depending on the situation, it can be possible that the requester simply not assert that he/she has the "deny-role" and thus gain access.

In this case, we stress that the better solution is to express this as a set of "role-allow" access, leading to a provably correct implementation. But what can be done in the general case that the number of authorized roles is *large*? Enumerating the entire set in some situations can be tedious and thus prone to error. While SecPAL does not support a way to make negative authorization statements within policies (i.e., there is no way to conclude that "no access" to a resource is a valid fact), it is possible to encode desired exclusions as part of an AuthorizationQuery. We investigate this possibility:

Policy Claims
LA says UVa-CA can say %p possesses %a (from %t1 until %t2) where %t2 - %t1 <= "365.00:00:00", %t1 <= CurrentTime() <= %t2, %a matches rfc822Name:".*@virginia\.edu"
LA says UVa-LDAP can say %p possesses %a (from %t1 until %t2) where %t2 - %t1 <= "365.00:00:00", %t1 <= CurrentTime() <= %t2, %a matches roleName:"^(Admin|Faculty|Student)$"
LA says %p can read, write digitalContent:"file:///GridFTPRoot/" if %p possesses %a where %a matches rfc822Name:".*@virginia\.edu"
Security Tokens
UVa-CA says K-User possesses rfc822Name:"sangmin@virginia.edu" (from "2007-01-01T00:00:00Z" until "2007-12-31T00:00:00Z")
UVa-LDAP says K-User possesses roleName:"Student" (from "2007-01-01T00:00:00Z" until "2007-12-31T00:00:00Z")
Authorization Query
(LA says K-User can read digitalContent:"file:///GridFTPRoot/data1.txt" AND NOT LA says K-User possesses roleName:"Student")?

By implementing it as shown, K-User was indeed denied access in accordance with our intended behavior (note that it is not possible for K-User to *not* tell the authorization engine that he is a student, because this assertion originates in server-side processing). However, there are a number of problems with this approach. First, the potential lack of composability in policy is a significant issue. Second, we had to special-code this Authorization Query in the GridFTP server, which is clearly undesirable. Third, ultimately we judged this as written to be *not* significantly easier to encode as compared to the enumeration of those roles that were allowed access (i.e., the number of possible roles asserted by the UVa LDAP server is small, currently eight). Ultimately, we reformulated this in terms of "positive policies" (not shown here, but similar to Case 2).

Case 4: Impersonation-based Access. We define impersonation as the ability of one principal to unconditionally act as another principal, assertable by a Token Service. For example, if a particular person is temporarily out of contact, perhaps another person can "fill in" for a short time. Another example is the ability of a user to delegate to a proxy server (PS), which could then delegate further (e.g., "K-User1 says PS can say K-User2 canActAs K-User1 [t1,t2] if (t2-t1)<5days and t2<12Apr2007"). Note that the first principal is not available to make the impersonation assertion himself (which we will refer to as "delegation" – Case 5).

Policy Claims
LA says UVa-CA can say %p possesses %a (from %t1 until %t2) where %t2 - %t1 <= "365.00:00:00", %t1 <= CurrentTime() <= %t2, %a matches rfc822Name:".*@virginia\.edu"
LA says %p can read, write digitalContent:"file:///GridFTPRoot/" if %p possesses %a where %a matches rfc822Name:"humphrey@virginia\.edu"
LA says UVa-CA can say %x can act as %y (from %t1 until %t2) where %t2 - %t1 <= "365.00:00:00", %t1 <= CurrentTime() <= %t2
Security Tokens
UVa-CA says K-User possesses rfc822Name:"humphrey@virginia.edu" (from "2007-01-01T00:00:00Z" until "2007-12-31T00:00:00Z")
UVa-CA says K-User2 can act as K-User
Authorization Query
LA says K-User2 can read digitalContent:"file:///GridFTPRoot/data1.txt"?

We believe that SecPAL is uniquely valuable for this situation, because SecPAL generates the chain of deductions that lead to impersonation (viewable via the proof graph) and that with other solutions, once the impersonation occurs, the principal that was originally authenticated is forgotten or ignored. However, our implementation of this requirement has limitations. A certificate authority (e.g., UVa-CA) is generally not able/willing to make assertions that one user is able to act as another, so we had to artificially make such assertions and store them in a specially-created Security Token Web Service. We could not achieve this with existing Token Services (e.g., UVa-CA) in part because they don't currently support a mechanism (e.g., SecPAL) by which to control use of such security tokens. We hope that the work contained in this paper creates new investigations into mechanisms and policies for such assertions.

Case 5: Delegation-based Access. In contrast to Case 4, in which the Token Service asserts the transfer of rights, a user might need to be able to delegate rights to another user. Note that this is, in fact, still unconstrained delegation (impersonation) as opposed to constrained delegation.

Policy Claims
LA says UVa-CA can say %p possesses %a (from %t1 until %t2) where %t2 - %t1 <= "365.00:00:00", %t1 <= CurrentTime() <= %t2, %a matches rfc822Name:".*@virginia\.edu"
LA says %p can read, write digitalContent:"file:///GridFTPRoot/" if %p possesses %a where %a matches rfc822Name:".*@virginia\.edu"
LA says %p can say %x can %v digitalContent:"file:///GridFTPRoot/" if %p can %v digitalContent:"file:///GridFTPRoot/"
Security Tokens
UVa-CA says K-User1 possesses rfc822Name:"humphrey@virginia.edu" (from "2007-01-01T00:00:00Z" until "2007-12-31T00:00:00Z")
K-User1 says K-User2 can read digitalContent:"file:///GridFTPRoot/"
Authorization Query
LA says K-User2 can read digitalContent:"file:///GridFTPRoot/data1.txt"?

An important aspect of this scenario is that LA can precisely restrict what any particular user can transfer, and it is up to the user to decide if/when to make this transfer of rights. This case also uses the Token Web Service, whereby K-User1 records and stores that K-User2 can read the particular file.

Case 6: Capability-based Access. A capability is an unforgeable token that, when presented to a resource guard, permits authorization based solely on the possession of the token and irrespective of the identity of the presenter. To date, capabilities have not been extensively utilized in Grid computing (e.g., the *gridmap-file* is essentially an access control list).

Policy Claims
LA says K-STS can say %p can read, write digitalContent:"file:///GridFTPRoot/"
Security Tokens
K-STS says K-User can read, write digitalContent:"file:///GridFTPRoot/"
Authorization Query
LA says K-User can read digitalContent:"file:///GridFTPRoot/data1.txt"

Similar to Case 4, there is not an easy way to represent, store, and convey such capabilities, so we chose to store capabilities in the Security Token Web Service. Note that the UVa CA and the UVa LDAP are not directly used in this scenario.

IV. Discussion and Evaluation

Through our implementation experiences, and as shown through the six use-cases, above, we have been able to qualitatively assess the ability of SecPAL to meet the requirements identified in Section II.C. We believe that most of the requirements are clearly met (uniquely strong support for GENERAL-R2, GENERAL-R4, and GENERAL-R8) although some of the requirements warrant further study. For example, our studies have not shown conclusively the ability to author and compose policies (GENERAL-R5, GENERAL-R6), although separation of trust policy authoring and file system access policy authoring duties is easily supported. We believe that we satisfied most of the more practical considerations enumerated in the first paragraph of Section III. However, we do not present to the client any indication *why* it was unauthorized, both because a legacy client (e.g., *globus-url-copy*) is not expecting such a response and it is not easy to generate such information from any logic-based language. We will continue to assess SecPAL through further development and expanded deployment of this modified GridFTP.NET service.

To assess the overhead of this new access control system, we measured the performance of our legacy GUI client when it first contacts the GridFTP server via a "Connect". The GridFTP.NET service ran on a commodity single-processor 3700+ AMD x64 server with 2G RAM running Windows Server 2003 (GigE NIC). Our client machine was a comparable AMD Opteron, connected over GigE LAN, except that the client machine had only a 100Mbps NIC. The two authorization policies we used in this experiment were:

1. *@virginia.edu can access the service
2. role == faculty can write in gridFTPRoot

In Table 1 we show a comparison of different authorization techniques, averaged over 500 cases.

TABLE I
DURATIONS FOR LEGACY GUI CLIENT "CONNECT"

gridmap	LDAP	SecPAL-based system
652.57 ms	893.06 ms	712.38 ms

The large majority of the time of each of the "Connect"s is to perform the SSL handshake. We observe that the most expensive of the three systems is the LDAP authorization system, which requires an additional SSL connection (between GridFTP.NET and the standalone LDAP server). In general we see that SecPAL does not incur substantial additional cost. Furthermore, separate measurements indicate that the SSL handshake between the GridFTP client and GridFTP service is the dominant factor, *irrespective of the authorization system*, requiring approximately 650 ms in our tests. In other words, the time taken by SecPAL and our information system architecture is small compared to the SSL handshake time. This means that SecPAL can be used (for the scenarios discussed here) with only a minimal impact on system performance.

To better understand and assess performance of our system built around the SecPAL engine, we measured the duration of the individual components of Figure 1. Table 2 shows these results, averaged over 500 cases.

TABLE II
DURATIONS FOR SECPAL-BASED SYSTEM OF FIGURE 1

SecPAL Token Gen.	UVa LDAP	Security Token Web Service	Policy Repo.	SecPAL Engine	Total
12.89 ms	3.96 ms X3	2.72 ms X4	2.50 ms	11.58 ms	49.95 ms

In the UVa PKI, each user certificate chain is of length 4, so we interacted with the UVa LDAP and Security Token Web Service for each cert (not the root cert in the latter case). We conclude that the SecPAL engine is efficient -- over 75% of the overall cost is incurred in information-gathering prior to the SecPAL engine invocation.

Our final test was to assess this SecPAL-based system "in the large", i.e., will the average client notice the performance difference? To test this, we invoked our GridFTP.NET command-line application to acquire a file from the service. Averaged over 100 cases, transferring a 10M file via *gridmap*-based access control took 2.505 sec vs. 2.61 sec for SecPal-based access control, or 4.2% longer. For 100M files, the difference is 10.73 sec (gridmap) vs. 10.84 sec (SecPAL), or 1.0% longer. We believe this difference is unlikely to impact most clients.

V. Conclusion and Future Work

Although some very capable Grid access control systems are used in practice today, many situations cannot be easily expressed and made operational. In this paper, we identified a set of requirements for an access control language for Grid data access and evaluated SecPAL in the context of GridFTP.NET against those requirements. These requirements were more complex that can be expressed by gridmap-like approaches, but were straightforward to implement using SecPALs grammar. We find that qualitatively and quantitatively SecPAL meets these requirements, although we recognize that further study and evaluation is necessary for a few of the requirements. The University of Virginia team is currently working to utilize this system via the GT4 GridFTP through its OGSA authorization call-out.

REFERENCES

[1] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids. *Proc. 5th ACM Conference on Computer and Communications Security Conference*, pp. 83-92, 1998.

[2] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch. A National-Scale Authentication Infrastructure. *IEEE Computer*, 33(12):60-66, 2000.

[3] Oasis Access Control TC, "XACML 2.0 Specification". 2005. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

[4] Markus Lorch, Seth Proctor, Rebekah Lepro, Dennis Kafur and Sumit Shah, "First experiences using XACML for access control in distributed systems", In Proceedings of the 2003 ACM workshop on XML security , 2003.

[5] Hommel, W., Using XACML for Privacy Control in SAML–based Identity Federations. In 9th IFIP TC–6 TC–11 Conference on Communications and Multimedia Security (CMS 2005), Springer, September, 2005.

[6] Peter Lamb, Robert Power, Gavin Walker, Michael Compton. Role-based access control for data service integration. Proceedings of the 3rd ACM workshop on Secure web services. Alexandria, VA, Nov 2006.

[7] M.C. Tschantz, S. Krishnamurthi. Towards reasonability properties for access-control policy languages. 2006 Access Control Models and Technologies Symp. Lake Tahoe, CA.

[8] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke, "GridFTP: Protocol extensions to ftp for the Grid," 2001. [Online]. Available: http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf

[9] J. Feng, L. Cui, G. Wasson, and M. Humphrey. Toward Seamless Grid Data Access: Design and Implementation of GridFTP on .NET. *2005 Grid Workshop (Associated with Supercomputing 2005)*. Nov 2005. Seattle, WA.

[10] B. Lampson, M. Abadi, M. Burrows, and E.Wobber. Authentication in distributed systems: theory and practice. ACM Trans. on Computer Systems, 10(4):265–310, 1992.

[11] Moritz Y. Becker, Cedric Fournet, Andrew D. Gordon, SecPAL: Design and Semantics of a Decentralized Authorization Language Technical Report In Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF), 2007.

[12] Blair Dillaway, A Unified Approach to Trust, Delegation, and Authorization in Large-Scale Grids, Technical Paper, Microsoft Corporation, September 2006.

[13] V. Welch, I. Foster, T. Scavo, F. Siebenlist, and C. Catlett. Scaling TeraGrid access: A roadmap for attribute-based authorization for a large cyberinfrastructure (draft August 24). 2006. http://gridshib.globus.org/docs/tg-paper/TG-Attribute-Authz-Roadmap-draft-aug24.pdf.

[14] Foster, I., Kesselman, C., Pearlman, L., Tuecke, S., and Welch, V. The Community Authorization Service: Status and Future. In Proceedings of Computing in High Energy Physics 03 (CHEP '03), 2003.

[15] Alfieri, R., et al. VOMS, an Authorization System for Virtual Organizations. First European Across Grid Conferences. Santiago de Compostela, Spain, Feb. 2003.

[16] Lorch, M., Kafura, D., Fisk, I., Keahey, K., Carcassi, G., Freeman, T. Authorization and Account Management in the Open Science Grid. Proceedings of the Sixth International Workshop on Grid Computing (GRID'05)

[17] Thompson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K., Essiari, A. Certificate-based Access Control for Widely Distributed Resources. Proceedings of the Eighth USENIX Security Symposium (Security `99), Washington, D.C., August 23-26, 1999, pp 215-227.

[18] Chadwick, D., and Otenko, O. The PERMIS X.509 role based privilege management infrastructure. Future Generation Computer Systems, 19(2):277-289, Feb 2003.

[19] M. Humphrey and G. Wasson. The University of Virginia Campus Grid: Integrating Grid Technologies with the Campus Information Infrastructure. *2005 European Grid Conference (ECG 2005)*, Amsterdam, Feb 14-16, 2005.

[20] V. Welch, R. Ananthakrishnan, F. Siebenlist, D. Chadwick, S. Meder, L. Pearlman. "Use of SAML for OGSI Authorization". Open Grid Forum Proposed Standard. GFD-E.066. March 26 2006. http://www.ogf.org/documents/GFD.66.pdf

[21] ITU-T Rec X.812 (1995) | ISO/IEC 10181-3:1996, Security Frameworks for open systems: Access control framework.

[22] B. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The Globus striped GridFTP framework and server," 2005.

[23] Marlena Erdos and Scott Cantor, "Shibboleth Architecture v5", Internet2/MACE, May 2002

[24] V. Welch, T. Barton, K. Keahey, and F. Siebenlist, "Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration," 4th Public Key Infrastructure R&D Workshop, 2005

[25] Blair Dillaway, Jason Hogg, Security Policy Assertion Language (SecPAL) Specification, Version 1.0, 15 February 2007, http://research.microsoft.com/projects/secpal/downloadSecPALSpecification.aspx