WEBDAV Working Group
INTERNET-DRAFT
<draft-ietf-webdav-acreq-00.txt>

H. Palmer, Netscape
November 17, 1997

Expires May 17, 1998

Requirements for Access Control within Distributed Authoring and Versioning Environments on the World Wide Web

Status of this Memo

Abstract

To provide a robust model for modifying documents and data within a distributed World Wide Web authoring environment, it is necessary to furnish a methodology which controls access to objects. Access control may include the ability to read an object, modify an object, or perform other more advanced functions upon an object. Access control is necessary to prevent unauthorized access or modification of objects within the authoring environment which could lead to unintended loss, damage or disclosure of data.

This document proposes requirements for the support of access control within a Web Distributed Authoring and Versioning (WebDAV) environment. It describes a model for the representation and interpretation of access control policies, and a set of operations needed to manage policies in

that form. It is intended that these requirements be supported within the framework of the proposed WebDAV extensions [WEBDAV1] to HTTP [HTTP], in the form of additional protocol operations and compliance requirements for clients and servers

1. Introduction

This document proposes WebDAV functionality to support access control in a Distributed Authoring and Versioning (DAV) environment, as defined by other specifications produced by the IETF WWW Distributed Authoring and Versioning working group [WebDAV1,WEBDAV2]. Specifically, this functionality would enable a Distributed Authoring Tool to discover access control policies associated with a given resource, to present those policies in a human-readable form to an end-user, and to set or modify such policies, as directed by an authorized user.

Although access control policies generally can be formulated in many ways, it is necessary to define a particular framework within which to discuss specific requirements for the expression and application of access control policy information. This document proposes such a framework, and then uses it to describe the requirements for DAV protocol support for the management of access control.

2. Rationale

The IDC report, "The Intranet's Many Faces" [IDC] points to "Central Administration of user access rights and restrictions" as an essential benefit of Web-based technology. Unfortunately, this fundamental requirement has not been standardized. Existing Web Server and Authoring Tool implementations do not have an interoperable mechanism for assigning access control information to a particular resource or requesting access control information about a particular resource.

Access control mechanisms in existing Web Servers, such the "htaccess" mechanism support in NCSA and Apache servers, suggest that Web-based access control requires a level of flexibility that is not common in the access control mechanisms of native operating systems. For example, Web-based access control often supports access control policies based on the client's IP address or DNS name. When user authentication is desired, Web-based access control allows the requirement for authentication to be placed selectively on particular resources or collections of resources, and sometimes supports a choice of user authentication mechanisms.

Users who authenticate through Web-based user authentication may or may not have user accounts on the native operating system on which the Web Server runs. In some cases, even when a user does have a native operating system account, it is not used. For example, the native operating system identity associated with the Web Server itself may not enable it to assume the identity of other users for purposes of accessing the native file system. In other cases, native operating system accounts are not created for Web Server users, with the intent to restrict these users to accessing the system only through the Web Server.

In spite of the flexibility often provided in formulating Web-based access control policies, mechanisms for viewing and setting access controls from Web clients have been limited to proprietary solutions. In order to realize a standard mechanism for supporting these operations

within the framework of the proposed HTTP standard for distributed authoring, there must be a protocol for conveying access control policy information between client and server, and for executing operations to view and set this information.

3. Terminology

Where there is overlap, usage is intended to be consistent with that in the WebDAV requirements specification [WEBDAV2].

ACE
: An Access Control Entry. This is the smallest unit of access control policy. It grants or denies a given set of access rights to a set of principals. An ACE is a component of an ACL, which is associated with a resource.

ACL
: An Access Control List. This contains all of the access control policies which are directly associated with a particular resource. These policies are expressed as ACEs.

Client
: A program which issues HTTP requests and accepts responses.

Collection
: A collection is a resource that contains other resources, either directly or by reference.

Distributed Authoring Tool
: A program which can retrieve a source entity via HTTP, allow editing of this entity, and then save/publish this entity to a server using HTTP.

Entity
: The information transferred in a request or response.

Hierarchical Collection
: A hierarchical organization of resources. A hierarchical collection is a resource that contains other resources, including collections, either directly or by reference.

Principal
: A loosely-defined term which is normally used to refer to a user identity that has been associated with a user agent as a result of some unspecified authentication protocol exchange between a server and the user agent.

Principal Attribute
: A characteristic of a user agent, or of a request to a server made by a user agent, which has a particular value that can be

determined by the server at the time of such a request. In this context, a principal attribute is assumed to be significant in the formulation of access control policy on the server.

Principal Description
: A description of a set of principal identities, formulated in terms of assertions about the values of the principal attributes. A given principal identity will match a given principal description, or not, depending on whether the principal attribute assertions in the principal description evaluate to "true" or "false" with respect to the principal attribute values of the principal identity, and also depending on how the Boolean results of such evaluations are logically combined in the principal description.

Principal Identity
: A specific set of corresponding values for a given set of principal attributes. A server determines these values at the time of an access, and a unique set of values defines a unique principal identity for purposes of access control.

Property
: Named descriptive information about a resource.

Resource
: A network data object or service that can be identified by a URI.

Server
: A program which receives and responds to HTTP requests.

User Agent
: The client that initiates a request.

4. Access Control Framework

This section proposes a conceptual framework to serve as a context for describing the operation of access control provisions of the WebDAV protocol. The main focus of this model are the abstractions used to describe access control policies, as the main protocol requirement is to convey access control policy specifications in a form that is meaningful to compliant clients and servers. The meaning of an access control policy specification is defined by how a server interprets it in determining whether to grant or deny a particular request. The model also describes how resources and access control policies are related.

4.1. Access Rights

Access rights are names for types of access to resources. A given request to a server may require access to one or more resources, and potentially different types of access to different resources. For example, a server-based "copy" operation would require "read" access to the resource being copied, and "modify" access to the collection in which the copy will be created. Access rights may be categorized as

generic, that is, those which apply to most or all types of resources, or specialized, that is, those which apply to a particular type of resource. For example, generic access rights might include "read", "modify", and "delete". For a hypothetical type of resource that

represents a bank account, specialized access rights, such as "deposit" and "withdraw", might be defined.

This specification defines a set of generic access rights that must be implemented by compliant clients and servers. WebDAV protocol specifications should include descriptions of which generic rights are required to which resources for each defined operation that may be requested of a server. The server evaluates the access control policies associated with referenced resources in order to determine whether the necessary access rights are granted or denied for a given access.

4.2. Principals

The term, "principal", is often used in security-related discussions to refer to the identity of an entity involved in some communication. It can refer to a person, a program acting as an agent for a person, or a program with its own associated identity. It usually seems to abstract the notion of "who".

In this context, we will normally use the term "principal identity" to express a related concept. Specifically, each access to a server has an associated principal identity, which is usually associated with the user agent, and which may require the user agent to provide the server with some kind of authentication credentials. However, unlike the normal usage of "principal", we use "principal identity" to refer to a set of attributes and associated values. There may be various mechanisms through which a server obtains values for these attributes. For example, a user identity attribute may be obtained through some authentication protocol, or an IP address of the client may be obtained from the server's connection state for the client, or some other attribute may be obtained from an HTTP header. That is, a principal identity abstracts not only "who" the client represents, but also other conditions that may exist at the time of an access, that is, conditions which are considered relevant to access control policy.

Exactly what is relevant to access control policy is expressed in terms of "principal attributes". Each principal attribute has a name that refers to a particular type of relevant information. For example, a principal attribute, "IP", might refer to the IP address of the user agent, or "user" might refer to an authenticated user name.

In some cases, the value of one principal attribute may be derived from the value of another. For example, the value of a principal attribute, "DNS", representing the DNS name of the user agent, might be determined via a reverse DNS lookup, using the value of the "IP" attribute. Similarly, the value of a principal attribute, "group", might be determined by using the value of the "user" attribute to access a database which defines the groups to which users belong.

A set of principal identities is described by a "principal description". The simplest form of a principal description would assert that a particular principal attribute must have a certain specified value, e.g.:

```
isEqual("user", "fred")
```

More complex principal descriptions may involve several principal attributes, with more sophisticated assertions about the values of these attributes, combined in a logical expression, e.g.:

```
OR(isMatch("DNS", "*.foo.com"),
   AND(isEqual("user", "fred"), isMatch("DNS", "*.bar.com")))
```

[Note that the syntax used here to represent principal descriptions is intended only to demonstrate the desired level of expressive capability, and nothing more.]

This specification defines a set of principal attributes, types of principal attribute assertions, and the mechanisms for combining principal attribute assertions to form principal descriptions.

4.3. Access Control Entry (ACE)

An Access Control Entry is the most fundamental unit of access control policy. An ACE contains a list of access rights, a principal description, and an indication of whether the specified access rights are to be granted or denied for principal identities which match the principal description. An ACE has no applicability to principal identities which do not match its principal description.

4.4. Access Control List (ACL)

An Access Control List is an ordered list of ACEs which are directly associated with a given resource. When policies expressed by the ACEs in a given ACL are in conflict, the policies expressed by ACEs nearer the beginning of the list have precedence. Conflicts can easily arise when some principal descriptions are very specific, and others more general. For example, a principal description that matches a certain user name, and a principal description that matches a certain group name may both apply to that user name. In this case, one would expect the ACE for the user to be placed before the ACE for the group, on the assumption that the policy expressed by the user ACE is an exception to the policy expressed by the group ACE.

The relative specificity of ACEs may not always be well-defined, depending on the principal attributes being used in their principal descriptions. Even if some precedence were assigned to each principal attribute, the precedence of a principal description involving several attributes would be problematic to compute at best. Precedence based on simple ordering of the ACEs in an ACL is more intuitive, and thus is less likely to lead to erroneous policies.

Given that ACEs are ordered within an ACL, they are evaluated in that order. Evaluation stops either when all requested access rights have

been granted by one or more ACEs, or when any one of the requested access rights has been denied by one of the ACEs. ACEs containing principal descriptions that do not match the current principal identity have no effect on the outcome of the evaluation.

4.5. Inheritance

An ACL which is directly associated with a collection resource may also apply to the member resources of the collection, in various ways. The ACL may be applied only when a new member resource is created in the collection, not to control the operation of resource creation, but to assign an initial ACL to the new resource. This will be known as "static inheritance". Alternatively, a collection ACL may apply on every access to a member resource, supplementing any ACL associated directly with the member resource, in specifying access control policies for the resource. We will refer to this as "dynamic inheritance".

Static and dynamic inheritance are not mutually exclusive, although if both are supported, it would generally be more useful to allow a collection to have one statically inherited ACL and one dynamically inherited ACL, rather than a single ACL that is inherited both statically and dynamically. If neither of these ACLs applies to operations on the collection resource itself, there may be a third ACL associated with the collection. Since this third ACL would be like an ACL associated with a non-collection resource, it will be known as a "resource ACL". So a collection may have a "static ACL" and/or a "dynamic ACL", and all resources, including collections, may have a resource ACL. If a static or dynamic ACL is maintained separately from the resource ACL, it is said to be "inherit-only".

Inheritance may also have the property of being recursive with respect to the collection hierarchy, meaning that a member resource inherits ACLs from all collections up from the collection which contains it to the root of the collection hierarchy.

A server may also support "user-based" inheritance. That is, if the server is able to associate user identities with user agents, based on authentication or some other means, it may also allow an inherit-only ACL to be associated with a user identity. The most useful form of this, from the end-user's perspective, is an ACL that is inherited by any resource created by that user, that is, a static, inherit-only ACL. One could also conceive of dynamic, inherit-only ACLs being associated with user identities. These might be used to restrict a user's access rights to all resources, on a per user, rather than per resource basis.

Whether a compliant server supports inheritance (of any form) or not is beyond the scope of this standard. However, there are some requirements which must be met when inheritance is supported, described in the sections below.

4.5.1. Discovery

It must be possible for a client to discover if a server supports inheritance, and if so, what kinds of inheritance it supports.

Specifically, the discovery mechanism should make the distinctions in the type of inheritance supported that have been described above.

4.5.2. Conflict Resolution

Just as the ACEs within an ACL may specify conflicting policies, ACLs which are applied to a resource via inheritance may conflict with each other or with an ACL associated directly with the resource. As the potential conflict between ACEs within an ACL was resolved by specifying that ACEs are ordered, potential conflict between inherited ACLs, the resource ACL, and user-based ACLs is also resolved by specifying a logical ordering or precedence.

In the case of static inheritance, the ACL for a new resource would be created by copying, in order, ACEs from any user-based ACL, and then from any static ACL on the collection in which the resource is created, and then from any static ACL on the collection containing that collection, and so on, up to the collection root. The result would be a single resource ACL for the new resource.

The order is different for dynamic inheritance. First, any dynamic, user-based ACL is evaluated, then any dynamic ACL on the collection root, and so on, down to any dynamic ACL on the collection containing the resource being access, and finally, any resource ACL on the resource itself.

The reason for using a different ordering of ACLs for static and dynamic inheritance is based on assumptions about who would be likely to be able to modify each type ACL. For static inheritance, the resulting resource ACL would probably be subject to subsequent modification by the user who created the resource. The ordering for static inheritance therefore attempts to reflect the most likely desires of that user. That is, it assumes that static ACLs on resources lower in the collection hierarchy are more likely to have been set or influenced by the user than ACLs higher towards the root of the collection hierarchy.

For dynamic inheritance, the assumption is that dynamic ACLs at different levels of the collection hierarchy may be administered by different people. It is further assumed that dynamic ACLs placed higher towards the root of the collection hierarchy are administered by the people who have the most authority to set access control policy, and therefore the ordering favors ACLs set on higher-level collections.

4.5.3. Distinction of Inherited ACLs

The protocol should include a mechanism for inherited ACLs to be retrieved and set separately from resource ACLs. A compliant server which supports inheritance should use these mechanisms. Servers should interpret operations to retrieve and set the ACL of a resource as applying only to the resource ACL on that resource, and not to any inherited ACLs.

In general, it should be possible for a compliant client to have no support for dealing with inheritance. In this case, the client would

have access only to resource ACLs. Nevertheless, the actual access control policies in effect for the client's accesses to server resources would include any policies expressed in the form of inherited ACLs.

4.6. Access to ACLs

A mechanism is required to enforce access control on the retrieval and modification of ACLs themselves. Whether this mechanism and a means to manage it should be within the scope of this specification is currently unresolved.

4.7. Other Access Control

A compliant server may support other kinds of access control, which are outside the scope of this protocol. For example, the access control policy in effect when no ACLs are present or no ACEs are applicable is a server implementation decision. Similarly, a server may implement other mechanisms which supplement the ACL-specified policies of this specification. These mechanisms may be implemented with higher or lower precedence than the ACL-specified mechanism. However, to the extent that such alternate mechanisms are used to exclusion of ACLs, the usefulness of this protocol with respect to managing access control on such a server will be diminished.

5. Requirements

This section describes the WebDAV requirements to enable a client to manage access control policies on a server.

5.1. Discovery

The protocol must provide a way for a client to discover any optional or extended functionality that may be implemented on a server, without side-effects. Compliant servers must be able to respond meaningfully to discovery operations.

5.1.1. Rationale

It is intended that the access control protocol be extensible with respect to types of ACEs, principal attributes, and access rights, at least. The discovery mechanism is needed so that a client and server can determine what extensions both of them support.

5.2. Operations

The protocol must enable a client to perform a number of operations with respect to access control policies which are stored on a server. These operations are described in the sections below.

If inheritance is supported, then any of these operations that refer to the ACL of a resource need to also specify whether the resource ACL, a static, inherit-only ACL, or a dynamic, inherit-only ACL is the target.

5.2.1. ACL Retrieval

An operation to retrieve an ACL for a given resource must be supported. This operation must return an ACL in such a way that the contents of the individual ACEs that comprise an ACL can be interpreted by the client,

and in such a way that the ordering of the ACEs within the ACL can be determined by client.

5.2.1.1. Rationale

Access control policies are represented in the form of ACLs, and clients need to be able to retrieve them, either to present the access control policies to a person, or in preparation for making a modification to them. It is not necessary to be able to retrieve individual ACEs from an ACL.

5.2.2. ACL Creation

An operation to create a new ACL for a given resource must be supported. This operation may allow the specification of an initial list of ACEs to be included in the new ACL.

5.2.2.1. Rationale

A separate operation is needed to explicitly create an ACL, because an empty ACL may have different semantics than no ACL.

5.2.3. ACL Deletion

An operation to delete an ACL for a given resource must be supported. The ACL and any ACEs it contains are deleted.

5.2.3.1. Rationale

A separate operation to delete an ACL is needed because empty ACLs can exist.

5.2.4. ACE Insertion

An operation to insert a specified ACE into the ACL of a given resource must be supported. This operation must allow the ACE to inserted at any position relative to any ACEs already present in the ACL.

5.2.4.1. Rationale

An operation to insert an ACE into an ACL is needed because the access control policies which apply to operations on ACLs will quite likely be at a granularity that specifies for what access rights the current principal identity is permitted to change access control policy. This would effectively restrict the current principal to inserting or deleting ACEs that contain only the access rights they are allowed to modify. If the only way to modify an ACL was to rewrite the entire ACL, it would be difficult to apply access control at that level of granularity.

The order of the ACE in the ACL must be specified, because it is significant in resolving conflicts between ACEs.

5.2.5. ACE Deletion

An operation to delete a specified ACE from the ACL of a given resource must be supported. The ACE must be specified in a way that uniquely identifies it. For example, such a specification might consist of the complete contents of the ACE and its position in the ACL. This is necessary to avoid deleting the wrong ACE in the case where an ACE insertion operation is performed between the time a client retrieves an ACL and initiates an ACE deletion operation.

5.2.5.1. Rationale

See 5.2.4.1.

5.2.6. Test Access

An operation to test the access of a specified principal identity to a given resource must be supported. The operation includes a list of access rights, a set of principal attributes and values, and the resource for which the access rights are to be tested. The results of the operation indicate, for each specified access right, whether the right is granted, denied, or unspecified. If inheritance is supported, the results of this operation include its effects.

5.3. ACE Contents

The contents of an ACE must include a list of access rights, a principal description, and an indication of whether the rights are granted or denied for matching principal identities. The protocol may provide for defining additional types of ACEs.

5.3.1. Rationale

ACEs allow flexibility in describing a principal identity because this has proven useful in many existing Web servers.

ACEs may contain multiple access rights because it is common to grant or deny a multiple access rights to the same set of principal identities.

ACEs do not combine granting and denying because that can be achieved more simply by taking advantage of ACE ordering.

5.4. Principal Description Semantics

The form in which the protocol conveys a principal description must be capable of expressing arbitrary Boolean expressions involving terms in the form of principal attribute assertions. The Boolean operators, AND, OR, and NOT, must be supported. The arguments of these operators are ordered for evaluation purposes, and only as many as necessary are evaluated in order to determine the result of the operator. That is, Boolean operators explicitly employ short-circuiting.

5.4.1. Rationale

Many existing Web servers enable principal attribute assertions to be combined in making access control policy. Using Boolean operators provides a uniform, flexible, and predictable way to combine principal attribute assertions.

Short-circuiting of Boolean operators improves efficiency, and also helps support deferred authentication. This concept, which many existing servers support, is that user authentication is postponed until it has been established that the access control policy which applies to the current principal cannot be determined without an authenticated user identity. It is a common practice to create access control policies based solely on IP addresses or DNS names, and thus avoid the need for user authentication.

5.5. Principal Attribute Assertions

The protocol must provide a way to represent assertions about principal attributes in the context of a principal description. The types of assertions that must be supported for each of the required principal attributes are described below. The protocol specification should define a mechanism for extending the protocol with additional types of principal attribute assertions.

5.5.1. Equality Assertion

It must be possible to encode an assertion that any given principal attribute is equal to a specified value. The exact interpretation of what constitutes "equality" may vary with the type of principal attribute involved. For example, string-valued attributes may be specified to be case-sensitive or case-insensitive.

5.5.2. Matching Assertion

It must be possible to encode an assertion that any given principal attribute matches a specified pattern. The format of a valid pattern may vary with the type of principal attribute involved. The protocol may define several different types of patterns, such as lists or regular expressions. Possibly the functionality of asserting equality and of asserting a match could be captured in a single mechanism.

5.6. Principal Attributes

The protocol must be able to encode attribute assertions involving the principal attributes specified in the following sections. It should provide a mechanism for defining additional principal attributes.

In all cases, it should be possible for a compliant client to present attribute assertions in a form in which they can be reasonably understood by a person. It should also be possible for a client to encode attribute assertions in the protocol, based on input entered by a

person.

5.6.1. IP Address

It must be possible to express an attribute assertion using the client IP address as the principal attribute. Such an assertion might involve a pattern consisting of an IP address and subnet mask, a list, or a regular expression.

5.6.1.1. Rationale

IP addresses are supported by many Web servers as a way to describe principals for access control purposes, and this feature is widely used.

5.6.2. DNS Name

It must be possible to express an attribute assertion using the client DNS name as the principal attribute. Such an assertion might involve a pattern consisting of a list or a regular expression.

5.6.2.1. Rationale

DNS names are supported by many Web servers as a way to describe principals for access control purposes, and this feature is widely used.

5.6.3. User Name

It must be possible to express an attribute assertion using a user name as the principal attribute. It must be possible for the server to interpret values or patterns which are part of such an assertion in terms of authenticated user identities.

All compliant servers must support an encoding of a user name in a simple text form that can be directly presented to, or entered by, a person. The protocol may also provide for other encodings of a user name that assume that the client has access to a user database or directory, in which case it should also provide a mechanism to ensure that a client and server are using a given such encoding in a consistent manner.

5.6.3.1. Rationale

User agents may or may not have access to the Directory service (or user database) used by a server, but if a server supports user authentication, it always has access to some kind of Directory service, or at least to an authentication authority which has such access. If a client does not have access to the server's Directory service, it is likely that a person would have to enter text in order to specify a user identity in a principal description. All compliant servers need to be able to handle such text, as literally entered by a person, as a specification of a user identity. Likewise, regardless of how a user identity is represented internally, a compliant server must be able to send user identities in a principal description in a form suitable for presentation to a person, since the client may have no way to translate

an opaque user identifier into such a form.

On the other hand, some clients and servers may share a common Directory service, and should have some way to discover that. Assuming that the

Directory service can translate user identifiers into a form suitable for human presentation, the client and server could communicate user identifiers in whatever form is suitable for accessing the Directory. For example, a client and server which are both LDAP-capable might convey user identities in the form of LDAP Distinguished Names or LDAP URLs.

5.7. Access Rights

The protocol should provide for an extensible set of access rights. In particular, some resource types or operations may define access rights which are specialized in applicability to those resource types or operations. However, the formulation of access control policies can often be simplified through the use of a generic access rights, which are applicable to a broad range of resource types and operations. Therefore the following sections define a required set of generic access rights that must be support by compliant clients and server.

It is acceptable if some implementations wish to treat different access rights as synonymous (e.g., a change to the right controlling "list" access may simultaneously change both "list" and "read"). However, this may be done only as specified in the following descriptions of the generic access rights.

5.7.1. Rationale

The set of access rights needs to be extensible because generic access rights will not necessarily apply in any intuitive way to all types of resources and operations.

A generic set of access rights is necessary because it would too burdensome to require a separate set of access rights to be defined and used for each new resource type or operation. Since there is a standard set of operations which can be applied to many different resource types, it is consistent to have a generic set of access rights which control those operations.

Some server implementations may perform mappings between WebDAV access rights and native file system access rights. However, in some cases, not all of the generic rights described here will have distinct, intuitive mappings to access rights used in the native file system. Therefore, it is permitted to merge some generic rights with others in order to facilitate such mappings.

5.7.2. List

A generic access right, known as the "list" access right, determines whether a particular request to list the contents of a collection resource will be allowed. More generally, the "list" right determines whether a particular request to discover whether a particular resource

exists will be allowed.

The "list" access right may be merged with the "read" access right, defined below. When a server does this, the granting or denying of

"read" access also grants or denies "list" access. If a client attempts to create an ACE containing the "list" right, but not the "read" right, such a server should return an error indicating that "list" is not supported as an independent right.

5.7.2.1. Rationale

Experience with file systems has shown that unauthorized access or attempts at circumventing security policies increase when users have more information about the contents of the file system. Therefore, it is helpful to define an access right that controls whether a user can obtain this information about a particular resource.

5.7.3. Read

A generic access right, known as the "read" access right, determines whether a particular request to view the contents of a particular resource will be allowed.

The "read" access right may also subsume the meaning of the "list" access right, as specified in section 5.7.2.

5.7.3.1. Rationale

Some resources may contain confidential or sensitive information. It should be possible to limit whether a particular user is allowed to read the contents of a resource.

5.7.4. Modify

A generic access right, known as the "modify" access right, determines whether a particular request to modify the contents of a particular resource will be allowed.

The "modify" access right may also subsume the meaning of the "delete" access right, as specified in section 5.7.5.

5.7.4.1. Rationale

Experience with a wide number of information systems has shown that different users need the ability to modify different resources.

5.7.5. Delete

A generic access right, known as the "delete" access right, determines whether a particular request to delete a particular resource will be allowed.

The "delete" access right may be merged with the "modify" access right, defined above. When a server does this, the granting or denying of

"modify" access also grants or denies "delete" access. If a client attempts to create an ACE containing the "delete" right, but not the "modify" right, such a server should return an error indicating that "delete" is not supported as an independent right.

5.7.5.1. Rationale

Experience with file systems has shown that it is sometimes preferable to permit certain users to modify a particular resource without allowing them to delete it.

5.7.6. Change Access

A generic access right, known as the "change access" access right, determines whether a particular request to change an access control policy will be allowed.

5.7.6.1. Rationale

Experience with file systems has shown that there is a significant desire to separate the management of information content (what is contained within the resources when a user reads it) from the manage of access control structure. Often, different people in different roles are responsible for these capabilities, and it may compromise the intended security plan to allow users to change access control information about a resource even if they are normally allowed to change or delete it.

5.8. Security Considerations

Transfer of access control policies between a client and server over an open network creates the potential for those policies to be modified or disclosed without proper authorization. The requirements for the access control protocol discussed in this document do not include cryptographic protection of access control policy information, because it is assumed that this protection can be provided through an implementation of HTTP over TLS 1.0 or SSL V3.

The use of client IP addresses or DNS names in formulating access control policies is subject to spoofing attacks. This risk should be carefully considered when implementing such policies. The Web presents somewhat of a dilemma in that most users have an expectation of relatively anonymous read access to servers, leaving IP addresses and DNS names as the only information about clients available to servers to be used for controlling read access.

It may also be necessary to require other WebDAV protocol operations to utilize HTTP over a secure transport protocol, in order to fully enforce access control policies. If entities are transferred between a client and server over an open network without cryptographic protection, those entities are subject to unauthorized disclosure or modification, regardless of what access control policies are in effect for the associated resources, and regardless of whether the access control policies themselves are protected when in transit over the network.

6. Copyright

7. Acknowledgments

Parts of this draft were taken from a working draft edited by Jon Radoff. Input from the following people has been instrumental in bringing this document to its current state of completion, although it does not necessarily reflect a consensus at this time:

Jim Davis, Xerox PARC, jdavis@parc.xerox.com
Yaron Y. Goland, Microsoft, yarong@microsoft.com
Paul Leach, Microsoft, paulle@microsoft.com
Larry Masinter, Xerox PARC, masinter@parc.xerox.com
Jon Radoff, NovaLink, jradoff@novalink.com
Judith Slein, Xerox, slein@wrc.xerox.com
Jim Whitehead, UC Irvine, ejw@ics.uci.edu

8. References

[HTTP] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, and T. Berners-Lee, **"Hypertext Transfer Protocol -- HTTP/1.1"**, RFC 2068, U.C. Irvine, DEC, MIT/LCS, January 1997.

[IDC] T. Julian, R. Villars, "The Intranet's Many Faces,"

Report 11344, International Data Corporation, April 1996.

[WEBDAV1] Y. Y. Goland, E. J. Whitehead, Jr., A. Faizi, S. R. Carter, D. Jensen. "Extensions for Distributed Authoring and Versioning on the World Wide Web - WEBDAV", Internet Draft, work-in-progress, draft-ietf-webdav-protocol-04.txt, October 1997.

[WEBDAV2] J. A. Slein, F. Vitali, E. J. Whitehead, Jr., D. Durand, "Requirements for Distributed Authoring and Versioning on the World Wide Web." Internet-draft, work-in-progress, draft-ietf-webdav-requirements-03.txt, September 1997.

8. Author's Address

Howard Palmer
M/S MV061
Netscape Communications Corp.
501 E. Middlefield Rd.
Mountain View, CA. 94043

EMail: hep@acm.org