INTERNET-DRAFT K. Ito
Xythos
Expires April, 2002 October, 2001

Ticket-Based Access Control Extension to WebDAV

<draft-ito-dav-ticket-00.txt>

Status of this Memo

Abstract

This draft describes an extension to the WebDAV Distributed Authoring Protocol which provides a mechanism for transferring access privileges on a DAV resource through the use of a token shared between multiple principals. This allows a principal to transfer specific privileges to others in a time or usage-limited manner.

Notational Conventions

The augmented BNF used by this document to describe protocol elements is described in Section 2.1 of [RFC2616]. Because this augmented BNF uses the basic production rules provided in Section 2.2 of [RFC2616], those rules apply to this document as well.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Definitions of XML elements in this document use XML element type declarations (as found in XML Document Type Declarations), described in Section 3.2 of [REC-XML].

1. Overview

Tickets are a mechanism for the transfer of specific access

privileges for a limited time or number of uses from one principal to another.

Issuing a ticket on a resource provides a principal with a ticket identifier, a token which can then be distributed to other principals. A principal who presents this ticket ID transfers access privileges from the original principal to himself. This can be useful in situations where one wishes to grant access to individuals who are not easily defined within the access control scheme implemented by the server, for example, individuals who do not have accounts on the server on which the resource resides. It can also be used if a principal wishes to grant duration or usage limited access on a specific resource to other users.

In any series of transactions involving tickets, there are two roles played by one or more principals. The first role is that of the principal who "issues" the ticket by sending a MKTICKET request to the server, and receives a ticket in the response. This user will be referred to as the "ticket owner". The second role is that of the principal who "uses" the ticket, by sending the ticket as a parameter in another WebDAV request to gain access to the resource. This principal will be referred to as the "ticket user". Note that it is possible for both roles to be played by the same principal, and for multiple distinct principals to play the ticket user role on the same ticket.

1.1 Transfer of access privileges

Every ticket MUST define a set of permissions which it will pass from the ticket owner to the ticket user. This set of permissions acts as a mask on the resource permissions. Permissions that the ticket grants on a resource, but which the ticket owner has not been granted will not be granted to the ticket user. Similarly, permissions that the ticket owner has been granted on a resource, but which the ticket does not grant will not be granted to the ticket user.

1.2 Ticket ID Scheme

The only condition imposed on ticket IDs is that the ticket ID MUST be unique on a resource at any given time. However, since the ticket ID is used as proof that a principal is in possession of the ticket, a server SHOULD select a ticket ID scheme such that it would be sufficiently difficult for an adversary in a way to guess or predict a ticket ID.

1.3 Ticket Use

A ticket can be presented to the server either as a header in the

request for a resource or as a parameter in the query component of the Request-URI as described in [RFC2396]. Allowing the ticket to be presented in the URI makes it possible for a ticketed resource

to be accessed using the ticket with a single HTML link. When presented in the query component, the ticket ID should be given as the value of the "ticket" parameter. When presented as a header, the ticket ID should be given as the value of the "Ticket" header.

1.4 Ticket Invalidation

A ticket can be invalidated in one of three ways, any of which is sufficient to invalidate the ticket: (1) The ticket times out by existing for longer than the timeout interval determined at ticket creation, (2) the ticket is used to gain access to a resource more times than were specified at ticket creation, (3) the ticket is deleted using the DELTICKET method. In any of these three cases, the server MUST treat any further requests using the invalidated ticket as though the ticket did not exist.

1.5 Ticket Discovery

In order for client to determine the tickets that have been issued on a resource, the ticketdiscovery property is provided. The ticketdiscovery property lists all active tickets, their ticket ID's, the time remaining on each ticket, the number of uses remaining on each ticket, the permissions granted by each ticket, and the ticket owner. Any resource that supports the MKTICKET method MUST support the ticketdiscovery property.

As the ability to view a ticket's ID also gives the viewer the ability to use the ticket, servers SHOULD restrict the visibility of the ticketdiscovery property to principals that it has determined to have sufficient access privileges on the resource.

2. The MKTICKET Method

The following describes the MKTICKET method, which is used to request a ticket on a URI.

2.1. Operation

A MKTICKET request creates a ticket on the resource specified in the Request-URI. MKTICKET requests MUST have an XML request body which contains a ticketinfo XML element (see section 4 for XML tag definitions). The ticketinfo element contains subelements specifying how long the ticket is valid for, how many times the ticket may be used, and the permissions that the ticket transfers to the ticket user.

The response MUST contain the value of the resource's ticketdiscovery property in a prop XML element.

In order to indicate the ID associated with a newly created ticket,

a Ticket response header containing the newly created ticket's ID MUST be included in the response for every successful MKTICKET

request.

2.2. Status Codes

200 (OK) - The MKTICKET request succeeded and the value of the ticketdiscovery property for this resource is included in the body.

400 (Bad Request) - The XML body was omitted or not parsable.

403 (Forbidden) - The user does not have proper access privileges on this resource.

404 (Not Found) - The resource specified could not be found.

2.3. Example - Ticket request

```
>>Request
MKTICKET /test.txt HTTP/1.1
Host: www.foo.com
Content-length: xxx
Content-Type: text/xml; charset="utf-8"
Authorization: Basic dGVzdHVzZXI6dGVzdHVzZXI=

<?xml version="1.0" encoding="utf-8" ?>
<D:ticketinfo xmlns:D="DAV:" >
  <D:privilege><D:read/></D:privilege>
  <D:timeout>Second-3600</D:timeout>
  <D:visits>1</D:visits>
</D:ticketinfo>

>>Response
HTTP/1.0 200 OK
Ticket: A658B29924F9D39C
Content-Type: text/xml; charset=UTF-8
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<D:prop xmlns:D="DAV:">
  <D:ticketdiscovery>
    <D:ticketinfo>
      <D:id>A658B29924F9D39C</D:id>
      <D:owner>
        <D:href>http://www.foo.com/users/testuser</D:href>
      </D:owner>
      <D:timeout>Second-3600</D:timeout>
      <D:visits>1</D:visits>
      <D:privilege>
        <D:read/>
      </D:privilege>
```

```
      </D:ticketinfo>
    </D:ticketdiscovery>
```

```
</D:prop>
```

This example shows the successful creation of a one-time use ticket with a lifetime of one hour on the resource http://www.foo.com/test.txt. Since the ticket was requested by the principal identified by "http://www.foo.com/users/testuser", the ticket owner is listed as such. The ticket ID, listed both in the ticketdiscovery property and the Ticket header displays the ID assigned to this ticket, in this case a random 64-bit integer expressed in hexadecimal notation.

2.4. Example - Using a ticket

```
>>Request
GET /test.txt?ticket=A658B29924F9D39C HTTP/1.1
Host: www.foo.com

>>Response
HTTP/1.0 200 OK
Content-Type: text/plain
Content-Length: xxx

<content follows>
```

In this example, it is assumed that the ticket with ID A658B29924F9D39C is the ticket requested in example 2.3, and that it was valid at the time of use. It is also assumed that the principal http://www.foo.com/users/testuser who issued the ticket has read access privileges on the resource. Since the ticket grants read access, and since the ticket owner has read access on the resource, the request succeeds.

3. The DELTICKET Method

The DELTICKET method is used to delete a ticket on a resource. To identify the ticket that is to be deleted, the "Ticket" header MUST be passed with the DELTICKET request. Any resource which supports the MKTICKET method must also support the DELTICKET method.

3.1. Status Codes

204 (No Content) - The DELTICKET request succeeded; the ticket specified has been deleted.

404 (Not Found) - The resource specified could not be found.

412 (Precondition Failed) - The ticket specified does not exist.

3.2. Example - Ticket deletion

```
>>Request
DELTICKET /test.txt HTTP/1.1
Host: www.foo.com
Ticket: A658B29924F9D39C

>>Response
HTTP/1.1 204 No Content
```

4. XML Element Definitions

4.1. Previously defined XML elements

This draft makes use of a number of XML elements that have previously been defined in other documents. DAV:href, DAV:multistatus, DAV:prop, DAV:propfind, DAV:propstat, DAV:response and DAV:status are defined in the WebDAV Distributed Authoring Protocol [RFC2518]. DAV:owner, DAV:privilege, DAV:read, and DAV:write are defined in the WebDAV Access Control Protocol [WACP] internet draft. To avoid a dependency on that draft, the relevant definitions are reproduced in section 5 of this document.

4.2. ticketdiscovery property

Name:
ticketdiscovery
Namespace:
DAV:
Purpose:
Describes the tickets currently valid on a resource
Description:
The ticketdiscovery property returns a listing of all valid tickets on a resource. The server is free to withhold any or all of this information if the requesting principal does not have sufficient access privileges to see the data.

```
<!ELEMENT ticketdiscovery (ticket)*>
```

4.2.1. Example - Retrieving the ticketdiscovery Property

```
>>Request
PROPFIND /test.txt HTTP/1.1
Host: www.foo.com
Content-type: text/xml; charset=UTF-8
Content-length: xxx
Authorization: Basic dGVzdHVzZXI6dGVzdHVzZXI=

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop>
```

```
      <D:ticketdiscovery/>
    </D:prop>
```

```
</D:propfind>


>>Response
HTTP/1.0 207 MultiPart Response
Content-Type: text/xml; charset=UTF-8
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://www.foo.com/test1.txt</D:href>
    <D:propstat>
      <D:prop>
        <D:ticketdiscovery>
          <D:ticketinfo>
            <D:id>9818B9E6D119A488</D:id>
            <D:owner>
              <D:href>http://www.foo.com/users/testuser</D:href>
            </D:owner>
            <D:timeout>Second-941</D:timeout>
            <D:visits>1</D:visits>
            <D:privilege>
              <D:read/><D:write/>
            </D:privilege>
          </D:ticketinfo>

          <D:ticketinfo>
            <D:id>6832EAD024B10C97</D:id>
            <D:owner>
              <D:href>http://www.foo.com/users/testuser2</D:href>
            </D:owner>
            <D:timeout>Second-3600</D:timeout>
            <D:visits>100</D:visits>
            <D:privilege>
              <D:read/>
            </D:privilege>
          </D:ticketinfo>
        </D:ticketdiscovery>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

The request above is for the ticketdiscovery property on the resource http://www.foo.com/test.txt. Two tickets that are active on this resource are returned. Note that there may be more tickets defined on this resource, but the requesting user may not have

permission to view them.

4.3. ticketinfo XML Element

Name:
   ticketinfo
Namespace:
   DAV:
Purpose:
   Describes a single ticket
Description:
   Contains relevant information about a ticket, including the ticket ID, the owner of the ticket, the amount of time remaining before the ticket expires, the number of visits the ticket can be used for before it expires, and the access privileges defined on the ticket.
The ticketinfo element is used both in the MKTICKET request and as a child of the ticketdiscovery property in a PROPFIND request. The id and owner elements should be omitted in a MKTICKET request since the server is responsible for setting both of these properties of the ticket.

```
<!ELEMENT ticketinfo (id?, owner?, timeout, visits, privilege)>
```

4.4. id XML Element

Name:
   id
Namespace:
   DAV:
Purpose:
   Contains the ID of a ticket. See section 1.2 for recommendations on generating the ticket ID.

```
<!ELEMENT id (#PCDATA)>
```

4.5. visits XML Element

Name:
   visits
Namespace:
   DAV:
Purpose:
   Describes the number of visits that the ticket has remaining on it.
Value:
   1*DIGIT | "infinity"

```
<!ELEMENT visits (#PCDATA)>
```

5. Definitions From ACL Draft

This draft makes use of a number of terms and XML property tags that

are defined in the internet draft "WebDAV Access Control Protocol" [WACP]. To avoid making this draft dependent on [WACP], the

following definitions have been copied verbatim from that document, with portions not used in this draft omitted.

5.1 Principals

A principal is a network resource that represents a distinct human or computational actor that initiates access to network resources. On many implementations, users and groups are represented as principals; other types of principals are also possible. A URI of any scheme MAY be used to identify a principal resource. However, servers implementing this specification MUST expose principal resources at an http(s) URL, which is a privileged scheme that points to resources that have additional properties. So, a principal resource can have multiple URI identifiers, one of which has to be an http(s) scheme URL. Although an implementation SHOULD support PROPFIND and MAY support PROPPATCH to access and modify information about a principal, it is not required to do so.

A principal resource may or may not be a collection. If a person or computational agent matches a principal resource that is contained by a collection principal, they also match the collection principal. This definition is recursive, and hence if a person or computational agent matches a collection principal that is the child of another collection principal, they also match the parent collection principal. Membership in a collection principal is also recursive, so a principal in a collection principal GRPA contained by collection principal GRPB is a member of both GRPA and GRPB.

Implementations not supporting recursive membership in principal collections can return an error if the client attempts to bind collection principals into other collection principals.

Servers that support aggregation of principals (e.g. groups of users or other groups) MUST manifest them as collection principals. At minimum, principals and collection principals MUST support the OPTIONS and PROPFIND methods.

5.2 Privileges

Ability to perform a given method on a resource SHOULD be controlled by one or more privileges. Authors of protocol extensions that define new HTTP methods SHOULD specify which privileges (by defining new privileges, or mapping to ones below) are required to perform the method. A principal with no privileges to a resource SHOULD be denied any HTTP access to that resource.

5.2.1 DAV:read Privilege

The read privilege controls methods that return information about

the state of the resource, including the resource's properties.
Affected methods include GET and PROPFIND. Additionally, the read

privilege MAY control the OPTIONS method.

<!ELEMENT read EMPTY>

5.2.2 DAV:write Privilege

The write privilege controls methods that modify the content, dead properties, or (in the case of a collection) membership of the resource, such as PUT and PROPPATCH. Note that state modification is also controlled via locking (see section 5.3 of [RFC2518]), so effective write access requires that both write privileges and write locking requirements are satisfied.

<!ELEMENT write EMPTY>

5.3 Access Control Properties

5.3.1 DAV:owner

This protected property identifies a particular principal as being the "owner" of the resource. Since the owner of a resource often has special access control capabilities (e.g., the owner frequently has permanent DAV:write-acl privilege), clients might display the resource owner in their user interface.

<!ELEMENT owner (href)>

6. Security Considerations

By providing another means for granting access privileges on resources, tickets increase the risk of unauthorized access to the particular resources that they are issued on. Since the ticket ID is sent in both the response from the server following a MKTICKET request and in the request for a resource using the ticket, there is the risk of the ticket ID being intercepted by an attacker who can then present it to the server and use it to maliciously gain access to the resource on which the ticket is issued.

However, this risk is mitigated by several factors. First, as long as the channels used to request, distribute, and use the ticket are secure, there is no added risk in using tickets. Second, since a ticket transfers a limited set of access privileges on a single resource for a limited duration and limited number of accesses, the scope of potential security violations is limited.

7. References

[RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396. MIT/LCS,

U.C. Irvine, Xerox, August, 1998.

[RFC2026] S.Bradner, "The Internet Standards Process - Revision 3." RFC 2026, BCP 9. Harvard, October, 1996.

[RFC2119] S.Bradner, "Key words for use in RFCs to Indicate Requirement Levels." RFC 2119, BCP 14, Harvard, March, 1997.

[REC-XML] T. Bray, J. Paoli, C.M. Sperberg-McQueen, "Extensible Markup Language (XML)." World Wide Web Consortium Recommendation REC-xml-19980210. http://www.w3.org/TR/REC-xml-19980210.

[RFC2616] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1." RFC 2616. U.C. Irvine, Compaq, Xerox, Microsoft, MIT/LCS, June, 1999.

[RFC2518] Y. Goland, E. Whitehead, A. Faizi, S. R. Carter, D. Jensen, "HTTP Extensions for Distributed Authoring -- WEBDAV." RFC 2518. Microsoft, U.C. Irvine, Netscape, Novell, February, 1999.

[WACP] G. Clemm, A. Hopkins, E. Sedlar, J. Whitehead, "WebDAV Access Control Protocol." IETF Internet Draft draft-ietf-webdav-acl-06. http://www.ietf.org/internet-drafts/draft-ietf-webdav-acl-06.txt Rational, Microsoft, Oracle, U.C. Santa Cruz, June, 2001.

8. Author's Address

Keith Ito
Xythos
77 Maiden Lane, Suite 200
San Francisco, CA, USA
kito@xythos.com

This internet draft expires April, 2002.