XEP-0074: Simple Access Control

Justin Kirby
mailto:justin@openaether.org
xmpp:zion@openaether.org

2003-10-20
Version 0.2

Status	Type	Short Name
Retracted	Standards Track	sac

A simple protocol for querying information for permissions.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2024 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the ”Specification”), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an ”AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction 1

2 A More Formal Tone 1

3 Query List of ACL operations 3

4 Integrating with Service Discovery 3

5 Security Considerations 3

6 IANA Considerations 4

7 XMPP Registrar Considerations 4

8 Open Issues 4

1 Introduction

A Jabber travels further into the Jabber-as-Middleware world, it needs a protocol to determine ”who can do what to whom”. This proposal defines a protocol that enables any Jabber entity to determine the permissions of another Jabber entity, whether the context is a user JID querying a component for access, a client-to-client conversation, or a component asking another component about a request from a user.
All access control lists (ACLs) boil down to three aspects: Actor, Operation, and Target/Sink. With this in mind it becomes almost trivial to implement the basic query/response mechanism.

Listing 1: A simple query

```
<iq to="security.capulet.com"
            from="inventory.capulet.com"
            type="get" id="1234">
          <acl xmlns="http://jabber.org/protocol/sac"
               actor="juliet@capulet.com/church"
               oper="uri://capulet.com/inventory#obtain"
               target="poison"/>
        </iq>
```

Here we have the inventory.capulet.com component querying the security component as to whether juliet@ may obtain the requested poison.

Listing 2: A response to the above query

```
<iq to="inventory.capulet.com"
            from="security.capulet.com"
            type="result" id="1234">
          <acl xmlns="http://jabber.org/protocol/sac"
               actor="juliet@capulet.com/church"
               oper="uri://capulet.com/inventory#obtain"
               target="poison">
            <allowed/>
          </acl>
        </iq>
```

Unfortunately, the response is in the affirmative and the romantic tragedy follows.

2 A More Formal Tone

The <acl> element provides the container for the query. It MUST have the three attributes: actor, oper, and target.
The actor attribute is set to the Jabber ID which is attempting to perform an operation. This need not be the JID sending the acl, although it may be. Remember this is to allow for the

question, ”Can X do Y to Z?”, which is a question anyone can ask.
The oper attribute is application-specific and refers to the operation for which a permission check is required (e.g., read/write operations). This also defines the scope of the ACL check, so the implementation is responsible for interpreting the actor and target values based on the value of the 'oper' attribute.
The target is the object which the actor is trying to perform the operation on. This MUST be a node queryable via Service Discovery (XEP-0030) [1].
Requests MUST be in the form of an empty <acl/> element with ALL the attributes specified. If not all attributes are specified, the request is incomplete and ambiguities arise; therefore the entity receving the request MUST return a ”bad request” error to the sender.
Responses MUST be in one of three forms: allowed, denied, error.
The response is inserted into the <acl/> as a child element. If the response is allowed, then <allowed/> is inserted. If the JID is denied then <denied/> is returned. If there is inadequate information then <error/> is used following the standard Jabber error scheme.

Listing 3: A positive response

```
<iq to="inventory.capulet.com"
            from="security.capulet.com"
            type="result" id="1234">
          <acl xmlns="http://jabber.org/protocol/sac"
               actor="juliet@capulet.com/church"
               oper="uri://capulet.com/inventory#obtain"
               target="poison">
            <allowed/>
          </acl>
        </iq>
```

Listing 4: Negative response (denied)

```
<iq to="inventory.capulet.com"
            from="security.capulet.com"
            type="result" id="1234">
          <acl xmlns="http://jabber.org/protocol/sac"
               actor="juliet@capulet.com/church"
               oper="uri://capulet.com/inventory#obtain"
               target="poison">
            <denied/>
          </acl>
        </iq>
```

Listing 5: Error response

```
<iq to="inventory.capulet.com"
            from="security.capulet.com"
            type="error" id="1234">
```

[1] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

```
          <acl xmlns="http://jabber.org/protocol/sac"
               actor="juliet@capulet.com/church"
               oper="uri://capulet.com/inventory#obtain"
               target="poison"/>
          <error code="404">No information available</error>
        </iq>
```

3 Query List of ACL operations

To obtain a list of acl operations that a jid supports, you must send an empty <query/>

Listing 6: querying for list of acl operations

```
<iq to="security.capulet.com"
            from="inventory.capulet.com"
            type="get" id="1234">
          <query xmlns="http://jabber.org/protocol/sac"/>
        </iq>
```

The to jid must then respond with a list of operations, if the jid supports SAC.

Listing 7: response to the query

```
<iq to="inventory.capulet.com"
            from="security.capulet.com"
            type="result" id="1234">
          <query xmlns="http://jabber.org/protocol/sac">
            <oper uri="uri://capulet.com/inventory#obtain"/>
            <oper uri="uri://capulet.com/inventory#add"/>
            <oper uri="uri://capulet.com/inventory#remove"/>
          </query>
        </iq>
```

4 Integrating with Service Discovery

To follow.

5 Security Considerations

To follow.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [2].

7 XMPP Registrar Considerations

As a result of this document, the XMPP Registrar [3] will need to register the 'http://jabber.org/protocol/sac' namespace.

8 Open Issues

1. Add disco integration section.
2. Fill out error codes.
3. Add DTD and Schema.
4. Allow for query of all allowable operations for actor in relation to a target.
5. Investigate possible integration with pubsub.

[2] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

[3] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.