Network Working Group   J. Rosenberg
Request for Comments: 5025   Cisco
Category: Standards Track   December 2007

                     Presence Authorization Rules

Abstract

   Authorization is a key function in presence systems. Authorization
   policies, also known as authorization rules, specify what presence
   information can be given to which watchers, and when. This
   specification defines an Extensible Markup Language (XML) document
   format for expressing presence authorization rules. Such a document
   can be manipulated by clients using the XML Configuration Access
   Protocol (XCAP), although other techniques are permitted.

Table of Contents

1. Introduction

    The Session Initiation Protocol (SIP) for Instant Messaging and
    Presence (SIMPLE) specifications allow a user, called a watcher, to
    subscribe to another user, called a presentity [17], in order to
    learn their presence information [18]. This subscription is handled
    by a presence agent. However, presence information is sensitive, and
    a presence agent needs authorization from the presentity prior to
    handing out presence information. As such, a presence authorization
    document format is needed. This specification defines a format for
    such a document, called a presence authorization document.

[8] specifies a framework for representing authorization policies,
and is applicable to systems such as geo-location and presence. This
framework is used as the basis for presence authorization documents.
In the framework, an authorization policy is a set of rules. Each
rule contains conditions, actions, and transformations. The
conditions specify under what conditions the rule is to be applied to
presence server processing. The actions element tells the server
what actions to take. The transformations element indicates how the
presence data is to be manipulated before being presented to that
watcher, and as such, defines a privacy filtering operation. [8]
identifies a small number of specific conditions common to presence
and location services, and leaves it to other specifications, such as
this one, to fill in usage specific details.

A presence authorization document can be manipulated by clients using
several means. One such mechanism is the XML Configuration Access
Protocol (XCAP) [2]. This specification defines the details
necessary for using XCAP to manage presence authorization documents.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED",
"SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY",
and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and
indicate requirement levels for compliant implementations.

3. Structure of Presence Authorization Documents

A presence authorization document is an XML document, formatted
according to the schema defined in [8]. Presence authorization
documents inherit the MIME type of common policy documents,
application/auth-policy+xml. As described in [8], this document is
composed of rules that contain three parts - conditions, actions, and
transformations. Each action or transformation, which is also called
a permission, has the property of being a positive grant of
information to the watcher. As a result, there is a well-defined
mechanism for combining actions and transformations obtained from
several sources. This mechanism is privacy safe, since the lack of
any action or transformation can only result in less information
being presented to a watcher.

This section defines the new conditions, actions, and transformations
defined by this specification.

3.1. Conditions

3.1.1. Identity

   Although the <identity> element is defined in [8], that specification
   indicates that the specific usages of the framework document need to
   define details that are protocol and usage specific. In particular,
   it is necessary for a usage of the common policy framework to:

   o Define acceptable means of authentication.

   o Define the procedure for representing the identity of the WR
      (Watcher/Requestor) as a URI or Internationalized Resource
      Identifier (IRI) [13].

   This sub-section defines those details for systems based on [18]. It
   does so in general terms, so that the recommendations defined here
   apply to existing and future authentication mechanisms in SIP.

3.1.1.1. Acceptable Forms of Authentication

   When used with SIP, a request is considered authenticated if one of
   the following is true:

   The watcher proves its identity to the server through a form of
   cryptographic authentication, including authentication based on a
   shared secret or a certificate, and that authentication yields an
   identity for the watcher.

   The request comes from a sender that is asserting the identity of the
   watcher, and:

   1. the assertion includes a claim that the asserting party used a
       form of cryptographic authentication (as defined above) to
       determine the identity of the watcher, and

   2. the server trusts that assertion, and

   3. the assertion provides an identity in the form of a URI.

   Based on this definition, examples of valid authentication techniques
   include SIP [5], digest authentication [4], cryptographically
   verified identity assertions (RFC 4474 [15]), and identity assertions
   made in closed network environments (RFC 3325 [16]).

   However, the anonymous authentication described on page 194 of RFC
   3261 [5] is not considered a valid mechanism for authentication

because it does not produce an identity for the watcher. However, an
anonymous From header field, when used in conjunction with RFC 4474
[15], is considered an acceptable mechanism for authentication, since
it still implies that the asserting node performed authentication
that produced the identity of the watcher.

3.1.1.2. Computing a URI for the Watcher

Computing the URI for the watcher depends on whether the identity is
being ascertained through authentication or through an asserted
identity.

If an identity assertion is being utilized, the asserted identity
itself (which is in the form of a URI for acceptable forms of
identity assertion) is utilized as the URI. If the identity
assertion mechanism asserts multiple URIs for the watcher, then each
of them is used for the comparisons outlined in [8], and if any of
them match a <one> or <except> element, the watcher is considered a
match.

If an identity is being determined directly by a cryptographic
authentication, that authentication must produce a URI, or must
produce some form of identifier that can be linked, through
provisioning, to a URI that is bound to that identifier.

For example, in the case of SIP Digest authentication, the
authentication process produces a username scoped within a realm.
That username and realm are bound to an Address of Record (AOR)
through provisioning, and the resulting AOR is used as the watcher
URI. Consider the following "user record" in a database:

SIP AOR: sip:alice@example.com
digest username: ali
digest password: f779ajvvh8a6s6
digest realm: example.com

If the presence server receives a SUBSCRIBE request, challenges it
with the realm set to "example.com", and the subsequent SUBSCRIBE
contains an Authorization header field with a username of "ali" and a
digest response generated with the password "f779ajvvh8a6s6", the
identity used in matching operations is "sip:alice@example.com".

In SIP systems, it is possible for a user to have aliases - that is,
there are multiple SIP AORs "assigned" to a single user. In terms of
this specification, there is no relationship between those aliases.
Each would look like a different user. This will be the consequence
for systems where the watcher is in a different domain than the
presentity. However, even if the watcher and presentity are in the

same domain, and the presence server knows that there are aliases for
the watcher, these aliases are not mapped to each other or used in
any way.

SIP also allows for anonymous requests. If a request is anonymous
because the watcher utilized an authentication mechanism that does
not provide an identity to the presence server (such as the SIP
digest "anonymous" username), the request is considered
unauthenticated (as discussed above) and will match only an empty
<identity> element. If a request is anonymous because it contains a
Privacy header field [14], but still contains an asserted identity
meeting the criteria defined above, that identity is utilized, and
the fact that the request was anonymous has no impact on the identity
processing.

It is important to note that SIP frequently uses both SIP URI and tel
URI [12] as identifiers, and to make matters more confusing, a SIP
URI can contain a phone number in its user part, in the same format
used in a tel URI. A WR identity that is a SIP URI with a phone
number will NOT match the <one> and <except> conditions whose 'id' is
a tel URI with the same number. The same is true in the reverse. If
the WR identity is a tel URI, this will not match a SIP URI in the
<one> or <except> conditions whose user part is a phone number. URIs
of different schemes are never equivalent.

3.1.2. Sphere

The <sphere> element is defined in [8]. However, each application
making use of the common policy specification needs to determine how
the presence server computes the value of the <sphere> to be used in
the evaluation of the condition.

To compute the value of <sphere>, the presence agent examines all
published presence documents for the presentity. If at least one of
them includes the <sphere> element [9] as part of the person data
component [10], and all of those containing the element have the same
value for it, which is the value used for the <sphere> in presence
policy processing. If, however, the <sphere> element was not present
in any of the published documents, or it was present but had
inconsistent values, its value is considered undefined in terms of
presence policy processing.

Care must be taken in using <sphere> as a condition for determining
the subscription handling. Since the value of <sphere> changes
dynamically, a state change can cause a subscription to be suddenly
terminated. The watcher has no way to know, aside from polling, when
their subscription would be reinstated as the value of <sphere>

changes. For this reason, <sphere> is primarily useful for matching
on rules that define transformations.

3.2. Actions

3.2.1. Subscription Handling

The <sub-handling> element specifies the subscription authorization
decision that the server should make. It also specifies whether or
not the presence document for the watcher should be constructed using
"polite blocking". Usage of polite blocking and the subscription
authorization decision are specified jointly since proper privacy
handling requires a correlation between them. As discussed in [8],
since the combination algorithm runs independently for each
permission, if correlations exist between permissions, they must be
merged into a single variable. That is what is done here. The
<sub-handling> element is an enumerated Integer type. The defined
values are:

block: This action tells the server to reject the subscription,
   placing it in the "terminated" state. It has the value of zero,
   and it represents the default value. No value of the <sub-
   handling> element can ever be lower than this. Strictly speaking,
   it is not necessary for a rule to include an explicit block
   action, since the default in the absence of any action will be
   block. However, it is included for completeness.

confirm: This action tells the server to place the subscription in
   the "pending" state, and await input from the presentity to
   determine how to proceed. It has a value of ten.

polite-block: This action tells the server to place the subscription
   into the "active" state, and to produce a presence document that
   indicates that the presentity is unavailable. A reasonable
   document would exclude device and person information elements, and
   include only a single service whose basic status is set to closed
   [3]. This action has a value of twenty.

allow: This action tells the server to place the subscription into
   the "active" state. This action has a value of thirty.

   NOTE WELL: Placing a value of block for this element does not
   guarantee that a subscription is denied! If any matching rule has
   any other value for this element, the subscription will receive
   treatment based on the maximum of those other values. This is
   based on the combining rules defined in [8].

Future specifications that wish to define new types of actions MUST
define an entirely new action (separate from <sub-handling>), and
define their own set of values for that action. A document could
contain both <sub-handling> and a subscription handling action
defined by a future specification; in that case, since each action is
always a positive grant of information, the resulting action is the
least restrictive one across both elements.

The exact behavior of a presence server upon a change in the sub-
handling value can be described by utilizing the subscription
processing state machine in Figure 1 of RFC 3857 [6].

If the <sub-handling> permission changes value to "block", this
causes a "rejected" event to be generated into the subscription state
machine for all affected subscriptions. This will cause the state
machine to move into the "terminated" state, resulting in the
transmission of a NOTIFY to the watcher with a Subscription-State
header field with value "terminated" and a reason of "rejected" [7],
which terminates their subscription. If a new subscription arrives
later on, and the value of <sub-handling> that applies to that
subscription is "block", the subscription processing follows the
"subscribe, policy=reject" branch from the "init" state, and a 403
response to the SUBSCRIBE is generated.

If the <sub-handling> permission changes value to "confirm", the
processing depends on the states of the affected subscriptions.
Unfortunately, the state machine in RFC 3857 does not define an event
corresponding to an authorization decision of "pending". If the
subscription is in the "active" state, it moves back into the
"pending" state. This causes a NOTIFY to be sent, updating the
Subscription-State [7] to "pending". No reason is included in the
Subscription-State header field (none are defined to handle this
case). No further documents are sent to this watcher. There is no
change in state if the subscription is in the "pending", "waiting",
or "terminated" states. If a new subscription arrives later on, and
the value of <sub-handling> that applies to that subscription is
"confirm", the subscription processing follows the "subscribe, no
policy" branch from the "init" state, and a 202 response to the
SUBSCRIBE is generated, followed by a NOTIFY with Subscription-State
of "pending". No presence document is included in that NOTIFY.

If the <sub-handling> permission changes value from "blocked" or
"confirm" to "polite-block" or "allow", this causes an "approved"
event to be generated into the state machine for all affected
subscriptions. If the subscription was in the "pending" state, the
state machine will move to the "active" state, resulting in the
transmission of a NOTIFY with a Subscription-State header field of
"active", and the inclusion of a presence document in that NOTIFY.

If the subscription was in the "waiting" state, it will move into the
"terminated" state. If a new subscription arrives later on, and the
value of <sub-handling> that applies to that subscription is
"polite-block" or "allow", the subscription processing follows the
"subscribe, policy=accept" branch from the "init" state, and a 200 OK
response to the SUBSCRIBE is generated, followed by a NOTIFY with
Subscription-State of "active" with a presence document in the body
of the NOTIFY.

3.3. Transformations

The transformations defined here are used to drive the behavior of
the privacy filtering operation. Each transformation defines the
visibility a watcher is granted to a particular component of the
presence document. One group of transformations grants visibility to
person, device, and service data elements based on identifying
information for those elements. Another group of transformations
provides access to particular data elements in the presence document.

3.3.1. Providing Access to Data Component Elements

The transformations in this section provide access to person, device,
and service data component elements. Once access has been granted to
such an element, access to specific presence attributes for that
element is controlled by the permissions defined in Section 3.3.2.

3.3.1.1. Device Information

The <provide-devices> permission allows a watcher to see <device>
information present in the presence document. It is a set variable.
Each member of the set provides a way to identify a device or group
of devices. This specification defines three types of elements in
the set - <class>, which identifies a device occurrence by class;
<deviceID>, which identifies a device occurrence by device ID; and
<occurrence-id>, which identifies a device occurrence by occurrence
ID. The device ID and occurrence ID are defined in [10]. Each
member of the set is identified by its type (class, deviceID, or
occurrence-id) and value (value of the class, value of the deviceID,
or value of the occurrence-id).

For example, consider the following <provide-devices> element:

```
<provide-devices>
  <deviceID>urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6</deviceID>
  <class>biz</class>
</provide-devices>
```

This set has two members. This is combined with a <provide-devices>
element from a different rule:

```
<provide-devices>
  <class>home</class>
  <class>biz</class>
</provide-devices>
```

The result of the set combination (using the union operation) is a
set with three elements:

```
<provide-devices>
  <class>home</class>
  <class>biz</class>
  <deviceID>urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6</deviceID>
</provide-devices>
```

The <provide-devices> element can also take on the special value
<all-devices>, which is a short-hand notation for all device
occurrences present in the presence document.

Permission is granted to see a particular device occurrence if one of
the device identifiers in the set identifies that device occurrence.
If a <class> permission is granted to the watcher, and the <class> of
the device occurrence matches the value of the <class> permission
based on case-sensitive equality, the device occurrence is included
in the presence document. If a <deviceID> permission is granted to
the watcher, and the <deviceID> of the device occurrence matches the
value of the <deviceID> permission based on URI equivalence, the
device occurrence is included in the presence document. If an
<occurrence-id> permission is granted to the watcher, and the
<occurrence-id> of the device occurrence matches the value of the
<occurrence-id> permission based on case-sensitive equality, the
device occurrence is included in the presence document. In addition,
a device occurrence is included in the presence document if the
<all-devices> permission was granted to the watcher.

3.3.1.2. Person Information

The <provide-persons> permission allows a watcher to see the <person>
information present in the presence document. It is a set variable.
Each member of the set provides a way to identify a person
occurrence. This specification defines two types of elements in the
set - <class>, which identifies a person occurrence by class, and
<occurrence-id>, which identifies an occurrence by its occurrence ID.
Each member of the set is identified by its type (class or
occurrence-id) and value (value of the class or value of the
occurrence-id). The <provide-persons> element can also take on the

special value <all-persons>, which is a short-hand notation for all
person occurrences present in the presence document. The set
combination is done identically to the <provide-devices> element.

Permission is granted to see a particular person occurrence if one of
the person identifiers in the set identifies that person occurrence.
If a <class> permission is granted to the watcher, and the <class> of
the person occurrence matches the value of the <class> permission
based on case-sensitive equality, the person occurrence is included
in the presence document. If an <occurrence-id> permission is
granted to the watcher, and the <occurrence-id> of the person
occurrence matches the value of the <occurrence-id> permission based
on case-sensitive equality, the person occurrence is included in the
presence document. In addition, a person occurrence is included in
the presence document if the <all-persons> permission was granted to
the watcher.

3.3.1.3. Service Information

The <provide-services> permission allows a watcher to see service
information present in <tuple> elements in the presence document.
Like <provide-devices>, it is a set variable. Each member of the set
provides a way to identify a service occurrence. This specification
defines four types of elements in the set - <class>, which identifies
a service occurrence by class; <occurrence-id>, which identifies a
service by its occurrence ID; <service-uri>, which identifies a
service by its service URI; and <service-uri-scheme>, which
identifies a service by its service URI scheme. Each member of the
set is identified by its type (class, occurrence-id, service-uri, or
service-uri-scheme) and value (value of the class, value of the
occurrence-id, value of the service-uri, or value of the service-
uri-scheme). The <provide-services> element can also take on the
special value <all-services>, which is a short-hand notation for all
service occurrences present in the presence document. The set
combination is done identically to the <provide-persons> element.

Permission is granted to see a particular service occurrence if one
of the service identifiers in the set identifies that service
occurrence. If a <class> permission is granted to the watcher, and
the <class> of the service occurrence matches the value of the
<class> permission based on case-sensitive equality, the service
occurrence is included in the presence document. If a <service-uri>
permission is granted to the watcher, and the <service-uri> of the
service occurrence matches the value of the <service-uri> permission
based on URI equivalence, the service occurrence is included in the
presence document. If an <occurrence-id> permission is granted to
the watcher, and the <occurrence-id> of the service occurrence
matches the value of the <occurrence-id> permission based on case-

sensitive equality, the service occurrence is included in the
presence document. If a <service-uri-scheme> permission is granted
to the watcher, and the scheme of the service URI for the service
occurrence matches the value of <service-uri-scheme> based on case-
sensitive equality, the service occurrence is included in the
presence document. In addition, a service occurrence is included in
the presence document if the <all-services> permission was granted to
the watcher.

3.3.2. Providing Access to Presence Attributes

The permissions of Section 3.3.1 provide coarse-grained access to
presence data by allowing or blocking specific services or devices,
and allowing or blocking person information.

Once person, device, or service information is included in the
document, the permissions in this section define which presence
attributes are reported there. Certain information is always
reported. In particular, the <contact>, <service-class> [9], <basic>
status, and <timestamp> elements in all <tuple> elements, if present,
are provided to watchers. The <timestamp> element in all <person>
elements, if present, is provided to watchers. The <timestamp> and
<deviceID> elements in all <device> elements, if present, are
provided to all watchers.

3.3.2.1. Provide Activities

This permission controls access to the <activities> element defined
in [9]. The name of the element providing this permission is
<provide-activities>, and it is a Boolean type. If its value is
TRUE, then the <activities> element in the person data element is
reported to the watcher. If FALSE, this presence attribute is
removed if present.

3.3.2.2. Provide Class

This permission controls access to the <class> element defined in
[9]. The name of the element providing this permission is <provide-
class>, and it is a Boolean type. If its value is TRUE, then any
<class> element in a person, service, or device data element is
reported to the watcher. If FALSE, this presence attribute is
removed if present.

3.3.2.3. Provide DeviceID

   This permission controls access to the <deviceID> element in a
   <tuple> element, as defined in [9]. The name of the element
   providing this permission is <provide-deviceID>, and it is a Boolean
   type. If its value is TRUE, then the <deviceID> element in the
   service data element is reported to the watcher. If FALSE, this
   presence attribute is removed if present. Note that the <deviceID>
   in a device data element is always included, and not controlled by
   this permission.

3.3.2.4. Provide Mood

   This permission controls access to the <mood> element defined in [9].
   The name of the element providing this permission is <provide-mood>,
   and it is a Boolean type. If its value is TRUE, then the <mood>
   element in the person data element is reported to the watcher. If
   FALSE, this presence attribute is removed if present.

3.3.2.5. Provide Place-is

   This permission controls access to the <place-is> element defined in
   [9]. The name of the element providing this permission is <provide-
   place-is>, and it is a Boolean type. If its value is TRUE, then the
   <place-is> element in the person data element is reported to the
   watcher. If FALSE, this presence attribute is removed if present.

3.3.2.6. Provide Place-type

   This permission controls access to the <place-type> element defined
   in [9]. The name of the element providing this permission is
   <provide-place-type>, and it is a Boolean type. If its value is
   TRUE, then the <place-type> element in the person data element is
   reported to the watcher. If FALSE, this presence attribute is
   removed if present.

3.3.2.7. Provide Privacy

   This permission controls access to the <privacy> element defined in
   [9]. The name of the element providing this permission is <provide-
   privacy>, and it is a Boolean type. If its value is TRUE, then the
   <privacy> element in the person or service data element is reported
   to the watcher. If FALSE, this presence attribute is removed if
   present.

3.3.2.8. Provide Relationship

   This permission controls access to the <relationship> element defined
   in [9]. The name of the element providing this permission is
   <provide-relationship>, and it is a Boolean type. If its value is
   TRUE, then the <relationship> element in the service data element is
   reported to the watcher. If FALSE, this presence attribute is
   removed if present.

3.3.2.9. Provide Sphere

   This permission controls access to the <sphere> element defined in
   [9]. The name of the element providing this permission is <provide-
   sphere>, and it is a Boolean type. If its value is TRUE, then the
   <sphere> element in the person data element is reported to the
   watcher. If FALSE, this presence attribute is removed if present.

3.3.2.10. Provide Status-Icon

   This permission controls access to the <status-icon> element defined
   in [9]. The name of the element providing this permission is
   <provide-status-icon>, and it is a Boolean type. If its value is
   TRUE, then any <status-icon> element in the person or service data
   element is reported to the watcher. If FALSE, this presence
   attribute is removed if present.

3.3.2.11. Provide Time-Offset

   This permission controls access to the <time-offset> element defined
   in [9]. The name of the element providing this permission is
   <provide-time-offset>, and it is a Boolean type. If its value is
   TRUE, then the <time-offset> element in the person data element is
   reported to the watcher. If FALSE, this presence attribute is
   removed if present.

3.3.2.12. Provide User-Input

   This permission controls access to the <user-input> element defined
   in [9]. The name of the element providing this permission is
   <provide-user-input>, and it is an enumerated integer type. Its
   value defines what information is provided to watchers in person,
   device, or service data elements:

   false: This value indicates that the <user-input> element is removed
      from the document. It is assigned the numeric value of 0.

bare: This value indicates that the <user-input> element is to be
   retained. However, any "idle-threshold" and "since" attributes
   are to be removed. This value is assigned the numeric value of
   10.

thresholds: This value indicates that the <user-input> element is to
   be retained. However, only the "idle-threshold" attribute is to
   be retained. This value is assigned the numeric value of 20.

full: This value indicates that the <user-input> element is to be
   retained, including any attributes. This value is assigned the
   numeric value of 30.

3.3.2.13. Provide Note

   This permission controls access to the <note> element defined in [3]
   for <tuple> and [10] for <person> and <device>. The name of the
   element providing this permission is <provide-note>, and it is a
   Boolean type. If its value is TRUE, then any <note> elements in the
   person, service, or device data elements are reported to the watcher.
   If FALSE, this presence attribute is removed if present.

   This permission has no bearing on any <note> values present within
   <activities>, <mood>, <place-is>, <place-type>, <privacy>,
   <relationship>, or <service-class> elements. Notes within these
   elements are essentially values for their respective elements, and
   are present if the respective element is permitted in the presence
   document. For example, if an <activities> element is present in a
   presence document, and there is a <note> value for it, that note is
   present in the document sent to the watcher if the <provide-
   activities> permission is given, regardless of whether the <provide-
   note> permission is given.

3.3.2.14. Provide Unknown Attribute

   It is important that systems be allowed to include proprietary or new
   presence information and that users be able to set permissions for
   that information, without requiring an upgrade of the presence server
   and authorization system. For this reason, the <provide-unknown-
   attribute> permission is defined. This permission indicates that the
   unknown presence attribute with the given name and namespace
   (supplied as mandatory attributes of the <provide-unknown-attribute>
   element) should be included in the document. Its type is Boolean.

   The value of the name attribute MUST be an unqualified element name
   (meaning that a namespace prefix MUST NOT be included), and the value
   of the ns attribute MUST be a namespace URI. The two are combined to
   form a qualified element name, which will be matched to all unknown

child elements of the Presence Information Data Format (PIDF)
<tuple>, <device>, or <person> elements with the same qualified name.
In this context, "unknown" means that the presence server is not
aware of any schemas that define authorization policies for that
element. By definition, this will exclude the <provide-unknown-
attribute> rule from being applied to any of the presence status
extensions defined by RPID, since authorization policies for those
are defined here.

Another consequence of this definition is that the interpretation of
the <provide-unknown-attribute> element can change should the
presence server be upgraded. For example, consider a server that,
prior to the upgrade, had an authorization document that used
<provide-unknown-attribute> with a value of TRUE for some attribute,
say foo. This attribute was from a namespace and schema unknown to
the server, and so the attribute was provided to watchers. However,
after upgrade, the server is now aware of a new namespace and schema
for a permission that grants access to the foo attribute. Now, the
<provide-unknown-attribute> permission for the foo attribute will be
ignored, resulting in a removal of those elements from presence
documents sent to watchers. The system remains privacy safe, but
behavior might not be as expected. Developers of systems that allow
clients to set policies are advised to check the capabilities of the
server (using the mechanism described in Section 8) before uploading
a new authorization document, to make sure that the behavior will be
as expected.

3.3.2.15. Provide All Attributes

This permission grants access to all presence attributes in all of
the person, device, and tuple elements that are present in the
document (the ones present in the document are determined by the
<provide-persons>, <provide-devices>, and <provide-services>
permissions). It is effectively a macro that expands into a set of
provide-activities, provide-class, provide-deviceID, provide-mood,
provide-place-is, provide-place-type, provide-privacy, provide-
relationship, provide-sphere, provide-status-icon, provide-time-
offset, provide-user-input, provide-note, and provide-unknown-
attribute permissions such that each presence attribute in the
document has a permission for it. This implies that, so long as an
entire person, service, or device occurrence is provided, every
single presence attribute, including ones not known to the server
and/or defined in future presence document extensions, is granted to
the watcher.

4. When to Apply the Authorization Policies

   This specification does not mandate at what point in the processing
   of presence data the privacy filtering aspects of the authorization
   policy are applied. However, they must be applied such that the
   final presence document sent to the watcher is compliant to the
   privacy policy described in the authorization documents that apply to
   the user (there can be more than one; the rules for combining them
   are described in [8]). More concretely, if the presence document
   sent to a watcher is D, and the privacy filtering operation applied
   do a presence document x is F(x), then D MUST have the property that
   D = F(D). In other words, further applications of the privacy
   filtering operation would not result in any further changes of the
   presence document, making further application of the filtering
   operation a no-op. A corollary of this is that F(F(D)) = D for all
   D.

   The subscription processing aspects of the document get applied by
   the server when it decides to accept or reject the subscription.

5. Implementation Requirements

   The rules defined by the document in this specification form a
   "contract" of sorts between a client that creates this document and
   the server that executes the policies it contains. Consequently,
   presence servers implementing this specification MUST support all of
   the conditions, actions, and transformations defined in this
   specification. If servers were to implement a subset of these,
   clients would need a mechanism to discover which subset is supported.
   No such mechanism is defined.

   It is RECOMMENDED that clients support all of the actions,
   transformations, and conditions defined in this specification. If a
   client supports a subset, it is possible that a user might manipulate
   their authorization rules from a different client, supporting a
   different subset, and store those results on the server. When the
   user goes back to the first client and views their presence
   authorization rules there, the client may not be able to properly
   render or manipulate the document retrieved from the server, since it
   may contain conditions, actions, or transformations not supported by
   the client. The only reason that this normative requirement is not a
   MUST is that there are valid conditions in which a user manipulates
   their presence authorization rules from a single client, in which
   case this problem does not occur.

   This specification makes no normative recommendations on the
   mechanism used to transport presence authorization documents from

   clients to their servers. Although Section 9 defines how to utilize
   XCAP, XCAP is not normatively required by this specification.

6. Example Document

   The following presence authorization document specifies permissions
   for the user "user@example.com". The watcher is allowed to access
   presence information (the 'allow' value for <sub-handling>). They
   will be granted access to the presence data of all services whose
   contact URI schemes are sip and mailto. Person information is also
   provided. However, since there is no <provide-devices>, no device
   information will be given to the watcher. Within the service and
   person information provided to the watcher, the <activities> element
   will be shown, as will the <user-input> element. However, any
   "idle-threshold" and "since" attributes in the <user-input> element
   will be removed. Finally, the presence attribute <foo> will be shown
   to the watcher. Any other presence attributes will be removed.

```
<?xml version="1.0" encoding="UTF-8"?>
<cr:ruleset xmlns="urn:ietf:params:xml:ns:pres-rules"
 xmlns:pr="urn:ietf:params:xml:ns:pres-rules"
 xmlns:cr="urn:ietf:params:xml:ns:common-policy">
 <cr:rule id="a">
  <cr:conditions>
   <cr:identity>
    <cr:one id="sip:user@example.com"/>
   </cr:identity>
  </cr:conditions>
  <cr:actions>
   <pr:sub-handling>allow</pr:sub-handling>
  </cr:actions>
  <cr:transformations>
   <pr:provide-services>
     <pr:service-uri-scheme>sip</pr:service-uri-scheme>
     <pr:service-uri-scheme>mailto</pr:service-uri-scheme>
   </pr:provide-services>
   <pr:provide-persons>
     <pr:all-persons/>
   </pr:provide-persons>
   <pr:provide-activities>true</pr:provide-activities>
   <pr:provide-user-input>bare</pr:provide-user-input>
    <pr:provide-unknown-attribute
     ns="urn:vendor-specific:foo-namespace"
     name="foo">true</pr:provide-unknown-attribute>
  </cr:transformations>
 </cr:rule>
</cr:ruleset>
```

7. XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:pres-rules"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:cr="urn:ietf:params:xml:ns:common-policy"
 xmlns:pr="urn:ietf:params:xml:ns:pres-rules"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:import namespace="urn:ietf:params:xml:ns:common-policy"/>
 <xs:simpleType name="booleanPermission">
  <xs:restriction base="xs:boolean"/>
 </xs:simpleType>
 <xs:element name="service-uri-scheme" type="xs:token"/>
 <xs:element name="class" type="xs:token"/>
 <xs:element name="occurrence-id" type="xs:token"/>
 <xs:element name="service-uri" type="xs:anyURI"/>
 <xs:complexType name="provideServicePermission">
  <xs:choice>
   <xs:element name="all-services">
    <xs:complexType/>
   </xs:element>
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
     <xs:element ref="pr:service-uri"/>
     <xs:element ref="pr:service-uri-scheme"/>
     <xs:element ref="pr:occurrence-id"/>
     <xs:element ref="pr:class"/>
     <xs:any namespace="##other" processContents="lax"/>
    </xs:choice>
   </xs:sequence>
  </xs:choice>
 </xs:complexType>
 <xs:element name="provide-services"
  type="pr:provideServicePermission"/>
 <xs:element name="deviceID" type="xs:anyURI"/>
 <xs:complexType name="provideDevicePermission">
  <xs:choice>
   <xs:element name="all-devices">
    <xs:complexType/>
   </xs:element>
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
     <xs:element ref="pr:deviceID"/>
     <xs:element ref="pr:occurrence-id"/>
     <xs:element ref="pr:class"/>
     <xs:any namespace="##other" processContents="lax"/>
    </xs:choice>
   </xs:sequence>
```

```
   </xs:choice>
  </xs:complexType>
  <xs:element name="provide-devices"
   type="pr:provideDevicePermission"/>
  <xs:complexType name="providePersonPermission">
   <xs:choice>
    <xs:element name="all-persons">
     <xs:complexType/>
    </xs:element>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
     <xs:choice>
      <xs:element ref="pr:occurrence-id"/>
      <xs:element ref="pr:class"/>
      <xs:any namespace="##other" processContents="lax"/>
     </xs:choice>
    </xs:sequence>
   </xs:choice>
  </xs:complexType>
  <xs:element name="provide-persons"
   type="pr:providePersonPermission"/>
  <xs:element name="provide-activities"
   type="pr:booleanPermission"/>
  <xs:element name="provide-class"
   type="pr:booleanPermission"/>
  <xs:element name="provide-deviceID"
   type="pr:booleanPermission"/>
  <xs:element name="provide-mood"
   type="pr:booleanPermission"/>
  <xs:element name="provide-place-is"
   type="pr:booleanPermission"/>
  <xs:element name="provide-place-type"
   type="pr:booleanPermission"/>
  <xs:element name="provide-privacy"
   type="pr:booleanPermission"/>
  <xs:element name="provide-relationship"
   type="pr:booleanPermission"/>
  <xs:element name="provide-status-icon"
   type="pr:booleanPermission"/>
  <xs:element name="provide-sphere"
   type="pr:booleanPermission"/>
  <xs:element name="provide-time-offset"
   type="pr:booleanPermission"/>
  <xs:element name="provide-user-input">
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:enumeration value="false"/>
     <xs:enumeration value="bare"/>
     <xs:enumeration value="thresholds"/>
```

```
      <xs:enumeration value="full"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:element>
   <xs:element name="provide-note" type="pr:booleanPermission"/>
   <xs:element name="sub-handling">
    <xs:simpleType>
     <xs:restriction base="xs:token">
      <xs:enumeration value="block"/>
      <xs:enumeration value="confirm"/>
      <xs:enumeration value="polite-block"/>
      <xs:enumeration value="allow"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:element>
   <xs:complexType name="unknownBooleanPermission">
    <xs:simpleContent>
     <xs:extension base="pr:booleanPermission">
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="ns" type="xs:string" use="required"/>
     </xs:extension>
    </xs:simpleContent>
   </xs:complexType>
   <xs:element name="provide-unknown-attribute"
    type="pr:unknownBooleanPermission"/>
   <xs:element name="provide-all-attributes">
    <xs:complexType/>
   </xs:element>
  </xs:schema>
```

8. Schema Extensibility

    It is anticipated that future changes to this specification are
    accomplished through extensions that define new types of permissions.
    These extensions MUST exist within a different namespace.
    Furthermore, the schema defined above and the namespace for elements
    defined within it MUST NOT be altered by future specifications.
    Changes in the basic schema, or in the interpretation of elements
    within that schema, may result in violations of user privacy due to
    misinterpretation of documents.

    When extensions are made to the set of permissions, it becomes
    necessary for the agent constructing the permission document
    (typically a SIP user agent, though not necessarily) to know which
    permissions are supported by the server. This allows the agent to
    know how to build a document that results in the desired behavior,
    since unknown permissions would be ignored by the server. To handle
    this, when presence authorization documents are transported using

XCAP, the XCAP capabilities document stored at the server SHOULD
contain the namespaces for the permissions supported by the presence
server. This way, an agent can query for this list prior to
constructing a document.

9. XCAP Usage

The following section defines the details necessary for clients to
manipulate presence authorization documents from a server using XCAP.

9.1. Application Unique ID

XCAP requires application usages to define a unique application usage
ID (AUID) in either the IETF tree or a vendor tree. This
specification defines the "pres-rules" AUID within the IETF tree, via
the IANA registration in Section 11.

9.2. XML Schema

XCAP requires application usages to define a schema for their
documents. The schema for presence authorization documents is in
Section 7.

9.3. Default Namespace

XCAP requires application usages to define the default namespace for
their URIs. The default namespace is urn:ietf:params:xml:ns:pres-
rules.

9.4. MIME Type

XCAP requires application usages to define the MIME type for
documents they carry. Presence authorization documents inherit the
MIME type of common policy documents, application/auth-policy+xml.

9.5. Validation Constraints

There are no additional constraints defined by this specification.

9.6. Data Semantics

Semantics of a presence authorization document are discussed in
Section 3.

9.7. Naming Conventions

   When a presence agent receives a subscription for some user foo
   within a domain, it will look for all documents within http://[xcap
   root]/pres-rules/users/foo, and use all documents found beneath that
   point to guide authorization policy. If only a single document is
   used, it SHOULD be called "index".

9.8. Resource Interdependencies

   There are no additional resource interdependencies defined by this
   application usage.

9.9. Authorization Policies

   This application usage does not modify the default XCAP authorization
   policy, which is that only a user can read, write, or modify their
   own documents. A server can allow privileged users to modify
   documents that they don't own, but the establishment and indication
   of such policies are outside the scope of this document.

10. Security Considerations

   Presence authorization policies contain very sensitive information.
   They indicate which other users are "liked" or "disliked" by a user.
   As such, when these documents are transported over a network, they
   SHOULD be encrypted.

   Modification of these documents by an attacker can disrupt the
   service seen by a user, often in subtle ways. As a result, when
   these documents are transported, the transport SHOULD provide
   authenticity and message integrity.

   In the case where XCAP is used to transfer the document, both clients
   and servers MUST implement HTTP over Transport Layer Security (TLS)
   and HTTP Digest authentication. Sites SHOULD authenticate clients
   using digest authentication over TLS, and sites SHOULD define the
   root services URI as an https URI.

   Authorization documents themselves exist for the purposes of
   providing a security function - privacy. The SIP presence
   specifications [18] require the usage of an authorization function
   prior to the granting of presence information, and this specification
   meets that need. Presence authorization documents inherit the
   privacy properties of the common policy format on which they are
   based. This format has been designed to be privacy-safe, which means
   that failure of the presence server to obtain or understand an
   authorization document can never reveal more information than is

desired about the user, only less. This is a consequence of the fact
that all permissions are positive grants of information, and not
negative grants.

A consequence of this design is that the results of combining several
authorization documents can be non-obvious to end users. For
example, if one authorization document grants permission for all
users from the example.com domain to see their presence, and another
document blocks joe@example.com, the combination of these will still
provide presence to joe@example.com. Designers of user interfaces
are encouraged to carefully pay attention to the results of combining
multiple rules.

Another concern is cases where a user sets their privacy preferences
from one client, uploads their presence authorization document to a
server, and then modifies them from a different client. If the
clients support different subsets of the document format, users may
be confused about what information is being revealed. Clients
retrieving presence authorization documents from a server SHOULD
render, to the users, information about rules that they do not
understand, so that users can be certain what rules are in place.

11. IANA Considerations

There are several IANA considerations associated with this
specification.

11.1. XCAP Application Usage ID

This section registers an XCAP Application Usage ID (AUID) according
to the IANA procedures defined in [2].

   Name of the AUID: pres-rules

   Description: Presence rules are documents that describe the
   permissions that a presentity [17] has granted to users that seek
   to watch their presence.

11.2. URN Sub-Namespace Registration

   This section registers a new XML namespace, per the guidelines in
   [11]

      URI: The URI for this namespace is
      urn:ietf:params:xml:ns:pres-rules.

      Registrant Contact: IETF, SIMPLE working group (simple@ietf.org),
      Jonathan Rosenberg (jdrosen@jdrosen.net).

      XML:

      BEGIN

      <?xml version="1.0"?>
      <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
       "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
      <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <meta http-equiv="content-type"
          content="text/html;charset=iso-8859-1"/>
        <title>Presence Rules Namespace</title>
      </head>
      <body>
        <h1>Namespace for Permission Statements</h1>
        <h2>urn:ietf:params:xml:ns:pres-rules</h2>
      <p>See
      RFC5025.</p>
      </body>
      </html>
      END

11.3. XML Schema Registrations

   This section registers an XML schema per the procedures in [11].

      URI: urn:ietf:params:xml:schema:pres-rules.

      Registrant Contact: IETF, SIMPLE working group (simple@ietf.org),
      Jonathan Rosenberg (jdrosen@jdrosen.net).

      The XML for this schema can be found as the sole content of
      Section 7.

12. Acknowledgements

   The author would like to thank Richard Barnes, Jari Urpalainen, Jon
   Peterson, and Martin Hynar for their comments.

13. References

13.1. Normative References

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

   [2]   Rosenberg, J., "The Extensible Markup Language (XML)
         Configuration Access Protocol (XCAP)", RFC 4825, May 2007.

   [3]   Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and
         J. Peterson, "Presence Information Data Format (PIDF)", RFC
         3863, August 2004.

   [4]   Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S.,
         Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication:
         Basic and Digest Access Authentication", RFC 2617, June 1999.

   [5]   Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A.,
         Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP:
         Session Initiation Protocol", RFC 3261, June 2002.

   [6]   Rosenberg, J., "A Watcher Information Event Template-Package for
         the Session Initiation Protocol (SIP)", RFC 3857, August 2004.

   [7]   Roach, A., "Session Initiation Protocol (SIP)-Specific Event
         Notification", RFC 3265, June 2002.

   [8]   Schulzrinne, H., Tschofenig, H., Morris, J., Cuellar, J., Polk,
         J., and J. Rosenberg, "Common Policy: A Document Format for
         Expressing Privacy Preferences", RFC 4745, February 2007.

   [9]   Schulzrinne, H., Gurbani, V., Kyzivat, P., and J. Rosenberg,
         "RPID: Rich Presence Extensions to the Presence Information Data
         Format (PIDF)", RFC 4480, July 2006.

   [10] Rosenberg, J., "A Data Model for Presence", RFC 4479, July 2006.

   [11] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January
        2004.

   [12] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966,
        December 2004.

   [13] Duerst, M. and M. Suignard, "Internationalized Resource
        Identifiers (IRIs)", RFC 3987, January 2005.

   [14] Peterson, J., "A Privacy Mechanism for the Session Initiation
        Protocol (SIP)", RFC 3323, November 2002.

13.2. Informative References

   [15] Peterson, J. and C. Jennings, "Enhancements for Authenticated
        Identity Management in the Session Initiation Protocol (SIP)",
        RFC 4474, August 2006.

   [16] Jennings, C., Peterson, J., and M. Watson, "Private Extensions
        to the Session Initiation Protocol (SIP) for Asserted Identity
        within Trusted Networks", RFC 3325, November 2002.

   [17] Day, M., Rosenberg, J., and H. Sugano, "A Model for Presence and
        Instant Messaging", RFC 2778, February 2000.

   [18] Rosenberg, J., "A Presence Event Package for the Session
        Initiation Protocol (SIP)", RFC 3856, August 2004.

Author's Address

   Jonathan Rosenberg
   Cisco
   Edison, NJ
   US

   EMail: jdrosen@cisco.com
   URI: http://www.jdrosen.net