Management and Use of Client Certificates for the Extensible Messaging
and Presence Protocol (XMPP)
draft-meyer-xmpp-sasl-cert-management-01

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on September 9, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This document defines methods for managing and using client
certificates in the Extensible Messaging and Presence Protocol (XMPP). These methods, which make use of the EXTERNAL mechanism of the Simple Authentication and Security Layer (SASL) protocol, enable an XMPP client to log in to an XMPP server without providing a password.

Table of Contents

1. Introduction .................................................. 3
2. First Login .................................................... 4
3. Certificate Generation ........................................ 4
4. Uploading a Certificate ....................................... 4
5. Subsequent Login via SASL EXTERNAL .......................... 6
6. Requesting the List of Certificates ........................... 10
7. Removing a Certificate ......................................... 10
8. Revoking a Certificate ......................................... 11
9. Security Considerations ....................................... 11
   9.2. Stream Characteristics ................................. 12
   9.3. Check subjectAltName ................................. 12
   9.4. Changing the Password ................................. 12
10. References .................................................. 12
    10.1. Normative References .................................. 12
    10.2. Informative References ................................ 13
Appendix A. XML Schema ........................................... 13
Appendix B. Copying Conditions .................................. 15
Authors’ Addresses ............................................... 15
1. Introduction

An XMPP client typically needs a user name and a password to log in to an XMPP server. Many clients provide a mechanism to store these credentials so that a human user can automatically log in without being prompted for the password. While this practice is very user friendly, it can be a security risk, especially for some devices. Mobile devices like a mobile phone or a laptop might get stolen, providing the thief with the required password. Mobile phones are particularly insecure: providing the password on the keypad for each login is too complicated and the risk of losing the phone is high.

A solution to this problem is to allow a client to log in without knowing the password. XMPP as specified in [rfc3920bis] allows the use of any Simple Authentication and Security Layer [SASL] mechanism in the authentication of XMPP entities, including the SASL EXTERNAL mechanism. Therefore this document defines two methods that will enable password-free login for XMPP clients:

- How a client generates an X.509 certificate [X509], manages the list of client certificates, and informs the server of its authorized certificates.
- How a client presents a certificate during the Transport Layer Security [TLS] handshake and refers to it during SASL negotiation using the EXTERNAL mechanism.

The overall process is as follows:

1. Client logs in to server using standard password-based authentication methods (or a previously authorized certificate).
2. Client generates or obtains a certificate.
3. Client informs server of the certificate.
4. On subsequent login attempts, client can use the authorized certificate.

The client can also retrieve the list of authorized certificates, remove a certificate, or revoke a certificate.

These use cases are explained in the following sections.

Note: The following capitalized keywords are to be interpreted as described in [TERMS]: "MUST", "SHALL", "REQUIRED"; "MUST NOT", "SHALL NOT"; "SHOULD", "RECOMMENDED"; "SHOULD NOT", "NOT RECOMMENDED"; "MAY", "OPTIONAL".
2. First Login

On first login, the client has not yet authorized a certificate and therefore cannot use SASL EXTERNAL to authenticate. (There is a possible exception if the client already has a valid certificate issued by a certificate authority ("CA") that is recognized by the server, but we ignore that case here because it is relatively rare.) Therefore the client would authenticate using standard XMPP methods as described in [rfc3920bis]. If the client will attempt to upload and authorize a certificate for subsequent login attempts, it MUST protect the client-to-server stream using channel encryption via Transport Layer Security [TLS] as described in [rfc3920bis].

3. Certificate Generation

In order to upload and authorize a certificate, the client needs to generate or obtain a certificate. Here we assume that the client generates a self-signed certificate since this is also a requirement of [XTLS]; however, it is also possible for the client to obtain a CA-issued certificate. The client certificate MUST include a JID as described in section 15.2.1.2 of [rfc3920bis], where the JID will be represented as an XmppAddr. The JID can be either a bare JID of the form "user@domain.tld" or a full JID of the form "user@domain.tld/resource".

subjectAltName=otherName:id-on-xmppAddr;UTF8:hamlet@example.com

4. Uploading a Certificate

After the client has logged in and generated a certificate, it shall upload the certificate to its XMPP server. This is done by sending an XMPP IQ stanza of type "set" containing an <upload/> element qualified by the ‘urn:xmpp:saslcert:0’ namespace; this element in turn MUST contain at least one <item/> element, which in turn MUST contain a <certificate/> child element and SHOULD contain a <name/> child element. The XML character of the <certificate/> element is the X.509 certificate in DER encoding, Base64-encoded as specified in Section 4 of [RFC4648] for sending over the XML stream. The XML character data of the <name/> element is a human-readable name for the certificate (thus making it easier for a human user to manage the different certificates); the name does not have to be unique, since the certificate’s fingerprint provides a truly unique identifier. A client can upload multiple certificates with each certificate defined in one individual <item/> element.
If the server can process the certificate, it returns an empty IQ result.

(Error cases will be described in a future version of this specification, although the normal XMPP stanza errors apply.)

Once the server has accepted the certificate, a client can use that certificate to authenticate the user using SASL EXTERNAL on subsequent logins. Therefore the client MUST NOT store the password for subsequent login attempts.

The client that uploads the certificate does not need to be the client that subsequently uses the certificate. For example, a user might use a full-featured client to upload a certificate for subsequent use by a "bot" (e.g., an automated service or a device such as a set-top box). The bot creates its certificate and private key, and the client uploads the certificate to the server with a different client. After that procedure the bot can log in to the server and even participate in secure end-to-end communication without ever knowing the user’s password.
An optional element `<no-cert-management/>` inside the `<item/>` element indicates that a client logged in with that certificate is not allowed to add or remove certificates from the list. A server MAY allow such a client to query the list of certificates.

```xml
<iq from='hamlet@example.com/denmark'
    id='hf65d4aq'
    type='set'>
    <upload xmlns='urn:xmpp:saslcert:0'>
        <item>
            <name>Simple Bot</name>
            <no-cert-management/>
            <certificate>
                Certificate-in-DER-format-Base64-encoded
            </certificate>
        </item>
    </upload>
</iq>
```

5. Subsequent Login via SASL EXTERNAL

The RECOMMENDED protocol flow for client-to-server use of SASL EXTERNAL with end-user certificates is as follows:

1. Client initiates stream to the server.

```xml
<stream:stream
    xmlns:stream='http://etherx.jabber.org/streams'
    xmlns='jabber:client'
    to='example.com'
    version='1.0'>
</stream:stream>
```

2. Server replies with stream header.

```xml
<stream:stream
    xmlns:stream='http://etherx.jabber.org/streams'
    xmlns='jabber:client'
    id='c2s_234'
    from='example.com'
    version='1.0'>
</stream:stream>
```


```xml
<stream:features>
    <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
        <required/>
    </starttls>
</stream:features>
```
4. Client sends STARTTLS command to the server.
   <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
5. Server tells the client to proceed.
   <proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
6. During TLS handshake, the server requests a certificate and the client presents its certificate.
7. TLS negotiation completes successfully.
8. Client initiates a new stream header to the server.
   <stream:stream
     xmlns:stream='http://etherx.jabber.org/streams'
     xmlns='jabber:client'
     to='example.com'
     version='1.0'>
   <stream:stream
     xmlns:stream='http://etherx.jabber.org/streams'
     xmlns='jabber:client'
     id='c2s_345'
     from='example.com'
     version='1.0'>
10. Server advertises SASL mechanisms. If the server expects that the client will be able to authenticate and authorize as the identity provided in the presented certificate, then the server SHOULD advertise the SASL EXTERNAL mechanism; otherwise, if presented certificate is unacceptable (e.g., because the certificate is expired, not yet valid, or revoked), the server MUST NOT offer the EXTERNAL mechanism.
   <stream:features>
     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
       <mechanism>EXTERNAL</mechanism>
       <mechanism>DIGEST-MD5</mechanism>
       <mechanism>ANONYMOUS</mechanism>
       <required/>
     </mechanisms>
   </stream:features>
11. Because the client presented a certificate, it SHOULD consider
EXTERNAL to be its preferred SASL mechanism. If the client
certificate includes only one XMPP address and the user wishes
to authorize as the identity that has been authenticated by the
TLS layer (i.e., the XMPP address that is contained in the
certicate), then the client SHOULD NOT include an
authorization identity (i.e., the XML character data for the
<auth/> element SHOULD be "="), indicating an empty response); if
the client certificate contains more than one XMPP address or if
the user wishes to authorize as another identity, then the
client MUST include an authorization identity; if the client
certificate contain no XMPP address, then the client SHOULD
include an authorization identity (but MAY omit the
authorization identity if it does not know its identity, instead
having it assigned by the server).

<auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
 mechanism='EXTERNAL'>=</auth>

12. Server determines whether to allow authentication and
authorization of user.
1. If (1) the certificate presented by the client contains only
one valid XMPP address that corresponds to a registered
account on the server and (2) the client did not pass an
authorization identity in the SASL exchange, then the server
SHOULD allow authentication and authorization of that JID.
For the purpose of client authentication and authorization
with a server, a valid XMPP address is a JID encapsulated as
a subjectAltName entity of type otherName with an ASN.1
Object Identifier of "id-on-xmppAddr" as specified in
Section 15.2.1.3 of [rfc3920bis].

<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

2. If the certificate contains more than one valid XMPP address
that corresponds to a registered account on the server
(e.g., because the server offers virtual hosting), then the
server SHOULD allow authentication and authorization of the
JID specified as the authorization identity.

<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

3. If no authorization identity is included, then the server
MUST return a SASL failure case of <invalid-authzid/> and
close the stream.
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <invalid-authzid/>
</failure>
</stream:stream>

4. If the certificate does not contain an XMPP address, then the server MAY attempt to determine if there is a registered account associated with the user, for example by performing an LDAP lookup based on the Common Name in the certificate; if such a JID mapping is successful and the mapped JID matches the authorization identity provided, then the server SHOULD allow authentication and authorization of that mapped JID.

<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

5. If JID mapping is unsuccessful, then the server MUST return a SASL failure case of <not-authorized/> and close the stream.

<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <not-authorized/>
</failure>
</stream:stream>

6. If JID mapping is successful but the mapped JID does not match the authorization identity provided (if any), then the server MUST return a SASL failure case of <invalid-authzid/> and close the stream.

<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <invalid-authzid/>
</failure>
</stream:stream>

13. If SASL authentication succeeded, the client opens a new stream, then the client and server proceed with resource binding as described in [rfc3920bis]. If the XmppAddr in the certificate is a full JID then the server MUST force the client to use the defined resource during resource binding. The client is only allowed to use the provided resource name. If a client with the same resource name is currently logged in and that client is not forced to use the specified resource name, it SHOULD be logged out by the server.
6. Requesting the List of Certificates

A client can request the list of all certificates that are authorized to authenticate for its bare JID using SASL EXTERNAL. This is done by sending an XMPP IQ stanza of type "get" containing a <list/> element qualified by the 'urn:xmpp:saslcert:0' namespace.

```xml
<iq from='hamlet@example.com/denmark'
    id='hf7435gj'
    type='get'>
    <list xmlns='urn:xmpp:saslcert:0'/>
</iq>
```

The server then returns the list of all known certificates, including the provided name. Each certificate is contained in a separate <item/> element and uniquely identified by the value of the 'id' attribute. In the following example the 'id' is the SHA1 value in hex of the certificate. The 'id' is used for the client to remove or revoke a certificate.

```xml
<iq id='hf7435gj'
    to='hamlet@example.com/denmark'
    type='result'>
    <list xmlns='urn:xmpp:saslcert:0'>
        <item id='428b1358a286430f628da23fb33ddaf6e474f5c5'>
            <name>Mobile Client</name>
            <certificate>
                Certificate-in-DER-format-Base64-encoded
            </certificate>
        </item>
        <item id='571b23d99892f4566017426e92c377288ed6c983'>
            <name>Simple Bot</name>
            <no-cert-management/>
            <certificate>
                Certificate-in-DER-format-Base64-encoded
            </certificate>
        </item>
    </list>
</iq>
```

7. Removing a Certificate

A client needs to create a new certificate before its current one expires. After the new certificate is uploaded to the server, it might want to remove the old certificate to keep the list of certificates short (otherwise the list will grow indefinitely, making certificate handling more difficult for the user). The client
removes a certificate by sending an XMPP IQ stanza of type "set" containing a <remove/> element that in turn contains an empty <item/> whose 'id' attribute uniquely identifies the certificate as retrieved from the server with the <list/> IQ stanza. Similar to the upload procedure a client can remove multiple certificates by adding more than one <item/> element.

```xml
<iq from='hamlet@example.com/denmark' id='di5rshyy' type='set'>
  <remove xmlns='urn:xmpp:saslcert:0'/>
  <item id='428b1358a286430f628da23fb33ddaf6e474f5c5'/>
</iq>
```

Once a certificate has been removed it can no longer be used for SASL EXTERNAL. A server MAY automatically remove expired certificates from the list.

8. Revoking a Certificate

The user can revoke a certificate for a stolen or compromised device. The mechanism is similar to removing a certificate. The difference is that if a client is logged in with the compromised certificate using SASL EXTERNAL, the server SHOULD close the stream to that client thus forcing that client to log out. The client revokes a certificate by sending an XMPP IQ stanza of type "set" containing a <revoke/> element that in turn contains an empty <item/> whose 'id' attribute uniquely identified the certificate.

```xml
<iq from='hamlet@example.com/denmark' id='rev9gewf' type='set'>
  <revoke xmlns='urn:xmpp:saslcert:0'/>
  <item id='428b1358a286430f628da23fb33ddaf6e474f5c5'/>
</iq>
```

9. Security Considerations


This specification defines a method whereby a user can authorize self-signed certificates for login. In accordance with local security policies, a given XMPP deployment can refuse to support this feature, can allow only clients that have authenticated with CA-
issued certificates to upload self-signed certificates, can accept self-signed certificates only for full JIDs, etc.

9.2. Stream Characteristics

This specification allows the user to manipulate an alternative way to log into the server. The certificates are not required to be signed and any certificate can be used. Therefore the server MUST reject any communication described in this document if the link between client and server is not secured with both STARTTLS and SASL.

9.3. Check subjectAltName

The server MUST check if the JID in the subjectAltName of the certificate matches the bare JID of the user. A user MUST NOT be allowed to upload certificates for a different user.

9.4. Changing the Password

[XEP-0077] defines a mechanism to change the password without knowing the current one. If the server supports password change it MUST return not-authorized for clients logged in using SASL EXTERNAL and MAY include a password change form requiring the old password. If the client has logged in with the current password, the server MAY change the password without a form as specified in XEP-0077.

If a client is allowed to change the password without knowing the current password, the additional security provided by this document is compromised.

10. References

10.1. Normative References


Informative References


Appendix A. XML Schema

The following schema is not normative.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
 xmlns:xs='http://www.w3.org/2001/XMLSchema'
 targetNamespace='urn:xmpp:saslcert:0'
 xmlns='urn:xmpp:jingle:saslcert:0'
 elementFormDefault='qualified'>
  <xs:element name='list'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='item'
                    type='itemElementType'
                    minOccurs='0'
                    maxOccurs='unbounded'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='remove'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='item'
                    type='itemElementType' />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
<xs:complexType name="itemElementType">
  <xs:sequence>
    <xs:element name="name"
      type="xs:string"
      minOccurs='0'
      maxOccurs='1'/>
    <xs:element name="no-cert-management"
      type="empty"
      minOccurs='0'
      maxOccurs='1'/>
    <xs:element name="certificate"
      type="xs:string"
      minOccurs='0'
      maxOccurs='1'/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="optional"/>
</xs:complexType>

<xs:simpleType name="empty">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:complexType>
Appendix B. Copying Conditions

Regarding this entire document or any portion of it, the authors make no guarantees and are not responsible for any damage resulting from its use. The authors grant irrevocable permission to anyone to use, modify, and distribute it in any way that does not diminish the rights of anyone else to use, modify, and distribute it, provided that redistributed derivative works do not contain misleading author or version information. Derivative works need not be licensed under similar terms.

Authors’ Addresses

Dirk Meyer
Universitaet Bremen T2I

Email: dmeyer@tzi.de

Peter Saint-Andre
Cisco

Email: psaintan@cisco.com