XEP-0078: Non-SASL Authentication

Peter Saint-Andre
mailto:peter@andyet.net
xmpp:stpeter@stpeter.im
https://stpeter.im/

2008-10-29
Version 2.5

Status	Type	Short Name
Obsolete	Standards Track	iq-auth

This document specifies a protocol for authentication with Jabber servers and services using the jabber:iq:auth namespace. Note Well: The protocol specified herein has been superseded in favor of SASL authentication as specified in RFC 3920 / RFC 6120, and is now obsolete.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Note Well: The protocol specified herein has been superseded in favor of SASL authentication as specified in RFC 3920 [1] and RFC 6120 [2], and is now obsolete.

Jabber technologies have long included a wire protocol that enables a client to authenticate with a server. [3] The method originally used in the Jabber community makes use of the 'jabber:iq:auth' namespace and has been documented variously in Internet-Drafts and elsewhere. When the core Jabber protocols were formalized by the IETF, the 'jabber:iq:auth' protocol was replaced by the Simple Authentication and Security Layer (SASL) as specified in RFC 4422 [5]. SASL was incorporated into XMPP because it provides a more flexible approach to authentication by enabling XMPP entities to use a wide variety of authentication methods (e.g., PLAIN, DIGEST-MD5, EXTERNAL, and ANONYMOUS), some of which are more secure than the 'jabber:iq:auth' protocol.

The 'jabber:iq:auth' protocol specified herein is now obsolete. However, because it will take some time for existing implementations and deployments to be upgraded to SASL, client and server software implementations still need to include support for 'jabber:iq:auth' in order to interoperate, and this document provides canonical documentation of the 'jabber:iq:auth' protocol. Nevertheless, implementation and deployment of SASL authentication is strongly recommended, since the 'jabber:iq:auth' protocol will eventually be obsoleted entirely.

2 Requirements

The 'jabber:iq:auth' namespace must make it possible for a Jabber client to authenticate with a server. In particular, the client must provide a username and appropriate credentials for the specific authentication method used. The methods defined herein are:

1. plaintext
2. digest (using the SHA1 algorithm specified in RFC 3174 [6])

Note: This document does not include the so-called "zero-knowledge" method; that method did not provide stronger security than digest authentication and thus is unnecessary.

[1] RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc3920>.
[2] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[3] Component authentication is out of scope for this document, and is specified separately in Jabber Component Protocol (XEP-0114) [4].
[5] RFC 4422: Simple Authentication and Security Layer (SASL) <http://tools.ietf.org/html/rfc4422>.
[6] RFC 3174: US Secure Hash Algorithm 1 (SHA1) <http://tools.ietf.org/html/rfc3174>.

3 Use Cases

3.1 User Authenticates with Server

In order to determine which fields are required for authentication with a server, a client SHOULD first send an IQ get to the server. A client SHOULD NOT attempt to guess at the required fields, since the nature of the required data is subject to service provisioning.

Listing 1: Client Requests Authentication Fields from Server

```
<iq type='get' to='shakespeare.lit' id='auth1'>
  <query xmlns='jabber:iq:auth'/>
</iq>
```

Listing 2: Server Returns Authentication Fields to Client

```
<iq type='result' id='auth1'>
  <query xmlns='jabber:iq:auth'>
    <username/>
    <password/>
    <digest/>
    <resource/>
  </query>
</iq>
```

If the client included a username with the IQ-get but there is no such username, the server SHOULD NOT return an error, but instead SHOULD return the normal authentication fields (this helps to prevent unknown users from discovering which usernames are in use). If the server does not support non-SASL authentication (e.g., because it supports only SASL authentication as defined in RFC 6120), it MUST return a <service-unavailable/> error. If the client previously attempted SASL authentication but that attempt failed, the server MUST return a <policy-violation/> stream error (see RFC 6120 regarding stream error syntax).
Both the username and the resource are REQUIRED for client authentication using the 'jabber:iq:auth' namespace; if more flexible authentication and resource provisioning are desired, a server SHOULD implement SASL authentication and resource binding as defined in RFC 6120 (e.g., to enable the server to provide the resource). The <username/> and <resource/> elements MUST be included in the IQ result returned by the server in response to the initial IQ get, and also MUST be included in the IQ set sent by the client when providing authentication credentials.
The foregoing stanza shows that the server supports both plaintext authentication (via the <password/> element) and digest authentication with SHA1-encrypted passwords (via the <digest/> element).
Therefore, in order to successfully authenticate with the server in this example, a client MUST provide a username, a resource, and one of password or digest.

Listing 3: Client Provides Required Information (Plaintext)

```
<iq type='set' id='auth2'>
  <query xmlns='jabber:iq:auth'>
    <username>bill</username>
    <password>Calli0pe</password>
    <resource>globe</resource>
  </query>
</iq>
```

Plaintext passwords are straightforward (obviously, characters that map to predefined XML entities MUST be escaped according to the rules defined in section 4.6 of the XML specification, and any non-US-ASCII characters MUST be encoded according to the encoding of XML streams as specified in RFC 6120, i.e., UTF-8 as defined in RFC 3629 [7]).
The value of the <digest/> element MUST be computed according to the following algorithm:

1. Concatenate the Stream ID received from the server with the password. [8]
2. Hash the concatenated string according to the SHA1 algorithm, i.e., SHA1(concat(sid, password)).
3. Ensure that the hash output is in hexidecimal format, not binary or base64.
4. Convert the hash output to all lowercase characters.

Listing 4: Client Provides Required Information (Digest)

```
<iq type='set' id='auth2'>
  <query xmlns='jabber:iq:auth'>
    <username>bill</username>
    <digest>48fc78be9ec8f86d8ce1c39c320c97c21d62334d</digest>
    <resource>globe</resource>
  </query>
</iq>
```

The character data shown in the <digest/> element is the output produced as a result of following the algorithm defined above when the stream ID is '3EE948B0' and the password is 'Calli0pe'.
If the credentials provided match those known by the server, the client will be successfully authenticated.

Listing 5: Server Informs Client of Successful Authentication

```
<iq type='result' id='auth2'/>
```

[7] RFC 3629: UTF-8, a transformation format of ISO 10646 <http://tools.ietf.org/html/rfc3629>.
[8] In Digest authentication, password characters that map to predefined XML entities SHOULD NOT be escaped as they are for plaintext passwords, but non-US-ASCII characters MUST be encoded as UTF-8 since the SHA-1 hashing algorithm operates on byte arrays.

Alternatively, authentication may fail. Possible causes of failure include:

1. The user provided incorrect credentials.
2. There is a resource conflict (i.e., there is already an active session with that resource identifier associated with the same username). The RECOMMENDED behavior is for the server to terminate the existing session and create the new one; however, the server MAY provide the opposite behavior if desired, leading to a conflict error for the newly requested login.
3. The user did not provide all of the required information (e.g., did not provide a username or resource).

Although RFC 6120 specifies that error stanzas SHOULD include the original XML sent, error stanzas qualified by the 'jabber:iq:auth' namespace SHOULD NOT do so given the sensitive nature of the information being exchanged.

Listing 6: Server Informs Client of Failed Authentication (Incorrect Credentials)

```
<iq type='error' id='auth2'>
  <error code='401' type='auth'>
    <not-authorized xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

Listing 7: Server Informs Client of Failed Authentication (Resource Conflict)

```
<iq type='error' id='auth2'>
  <error code='409' type='cancel'>
    <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

Listing 8: Server Informs Client of Failed Authentication (Required Information Not Provided)

```
<iq type='error' id='auth2'>
  <error code='406' type='modify'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

4 Stream Feature

RFC 6120 defines methods for advertising feature support during stream negotiation. It may be desirable for a server to advertise support for non-SASL authentication as a stream feature. The namespace for reporting support within <stream:features/> is

"http://jabber.org/features/iq-auth". Upon receiving a stream header qualified by the 'jabber:client' namespace, a server that returns stream features SHOULD also announce support for non-SASL authentication by including the relevant stream feature. Exactly when a server advertises the iq-auth stream feature is up to the implementation or deployment (e.g., a server MAY advertise this feature only after successful TLS negotiation or if the channel is encrypted via the older SSL method). Obviously, this does not apply to servers that do not support stream features (e.g., older servers that do not comply with XMPP 1.0).

Listing 9: Advertising non-SASL authentication as a stream feature

```
<stream:features>
    <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
      <mechanism>DIGEST-MD5</mechanism>
      <mechanism>PLAIN</mechanism>
    </mechanisms>
    <auth xmlns='http://jabber.org/features/iq-auth'/>
  </stream:features>
```

A server SHOULD NOT advertise non-SASL authentication to another server (i.e., if the initial stream header was qualified by the 'jabber:server' namespace).

5 Error Handling

As defined herein, the 'jabber:iq:auth' namespace supports both the old (HTTP-style) error codes and the extensible error classes and conditions specified in RFC 6120. A compliant server or service implementation MUST support both old-style and new-style error handling. A compliant client implementation SHOULD support both.

6 Expiration Date

In accordance with Section 8 of XMPP Extension Protocols (XEP-0001) [9], on 2004-10-20 this document was advanced to a status of Final with the understanding that it would expire in six months. On 2006-09-13, the Jabber Council (now XMPP Council) changed the status of this document to Deprecated. The Jabber Council will review this document every six months to determine whether to change its status to Obsolete or to extend the expiration date for an additional six months; this process will continue until the document is obsoleted. For the latest expiration date, refer to the XEP Information block at the beginning of this document.

[9] XEP-0001: XMPP Extension Protocols <https://xmpp.org/extensions/xep-0001.html>.

7 Security Considerations

Use of SASL authentication can be more secure than the 'jabber:iq:auth' protocol, depending on which SASL mechanism is used. Because SASL provides greater flexibility and can provide stronger security, the 'jabber:iq:auth' protocol is now obsolete. If both client and server implement SASL, they MUST prefer SASL over the 'jabber:iq:auth' protocol. If a client attempts to authenticate using the 'jabber:iq:auth' namespace after an attempt at SASL authentication fails, the server MUST refuse the 'jabber:iq:auth' attempt by returning a <policy-violation/> stream error to the client.
Client implementations MUST NOT make plaintext the default mechanism, and SHOULD warn the user that the plaintext mechanism is insecure. The plaintext mechanism SHOULD NOT be used unless the underlying stream is encrypted (using SSL or TLS) and the client has verified that the server certificate is signed by a trusted certification authority. A given domain MAY choose to disable plaintext logins if the stream is not properly encrypted, or disable them entirely. If a client implements the plaintext mechanism and a server allows both the digest mechanism and the plaintext mechanism when channel encryption is not in use, a downgrade attack is possible, in which a man-in-the-middle tricks the client into revealing the user's plaintext password.

8 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [10].

9 XMPP Registrar Considerations

9.1 Protocol Namespaces

The XMPP Registrar [11] includes the 'jabber:iq:auth' namespace in its registry of protocol namespaces.

9.2 Stream Features

The XMPP Registrar includes the 'http://jabber.org/features/iq-auth' namespace in its registry of stream feature namespaces.

[10] 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/>.

[11] 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/>.

10 XML Schemas

10.1 jabber:iq:auth

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='jabber:iq:auth'
    xmlns='jabber:iq:auth'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      NOTE WELL: Non-SASL Authentication via the jabber:iq:auth
      protocol has been superseded by SASL Authentication as
      defined in RFC 3920 and RFC 6120, and is now obsolete.

      For historical purposes, the protocol documented by this
      schema is defined in XEP-0078:

      http://www.xmpp.org/extensions/xep-0078.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='query'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='username' type='xs:string' minOccurs='0'/>
        <xs:choice>
          <xs:element name='password' type='xs:string' minOccurs='0'/>
          <xs:element name='digest' type='xs:string' minOccurs='0'/>
        </xs:choice>
        <xs:element name='resource' type='xs:string' minOccurs='0'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

10.2 Stream Feature

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
```

```
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='http://jabber.org/features/iq-auth'
    xmlns='http://jabber.org/features/iq-auth'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      NOTE WELL: Non-SASL Authentication via the jabber:iq:auth
      protocol has been superseded by SASL Authentication as
      defined in RFC 3920 and RFC 6120, and is now obsolete.

      For historical purposes, the protocol documented by this
      schema is defined in XEP-0078:

      http://www.xmpp.org/extensions/xep-0078.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='auth' type='empty'/>

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```