XEP-0070: Verifying HTTP Requests via XMPP

Matthew Miller
mailto:linuxwolf@outer-planes.net
xmpp:linuxwolf@outer-planes.net

Joe Hildebrand
mailto:jhildebr@cisco.com
xmpp:hildjj@jabber.org

Dave Smith
mailto:dizzyd@jabber.org
xmpp:dizzyd@jabber.org

Peter Saint-Andre
mailto:stpeter@stpeter.im
xmpp:stpeter@jabber.org
https://stpeter.im/

2016-12-09
Version 1.0.1

Status	Type	Short Name
Draft	Standards Track	http-auth

This specification defines an XMPP protocol extension that enables verification of an HTTP request via XMPP.

Legal

Copyright

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

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

HTTP (see RFC 2616 [1]) is a nearly-ubiquitous mechanism for the publication and retrieval of information over the Internet. Sometimes it is appropriate for an HTTP Server to allow access to that information only if the HTTP Client first provides authentication credentials. While there exist several standardized HTTP authentication schemes (see RFC 2617 [2]), it may be useful in some applications to enforce verification of an HTTP request by requiring an XMPP entity (normally an IM user) to confirm that it made the request. This request verification can be combined with native HTTP authentication to provide a stronger association between the request and a particular user, as well as to take advantage of the strong user authentication provided in XMPP (see XMPP Core [3]).

2 Terminology

2.1 HTTP Terms

This document inherits terminology about the HyperText Transfer Protocol from RFC 2616 and RFC 2617.

2.2 Entities

Term	Definition
HTTP Client	A client that implements the HyperText Transfer Protocol (HTTP)
HTTP Server	A server that implements the HyperText Transfer Protocol (HTTP)
XMPP Client	A client that implements the Extensible Messaging and Presence Protocol (XMPP) or its antecedents
XMPP Server	A server that implements the Extensible Messaging and Presence Protocol (XMPP) or its antecedents

Note well that an XMPP Client can simultaneously be an HTTP Client (or vice-versa), and that an XMPP Server can simultaneously be an HTTP Server (or vice-versa). However, for the purposes of this discussion, we assume that these entities are logically if not physically separate and distinct.

---

[1]RFC 2616: Hypertext Transport Protocol -- HTTP/1.1 <http://tools.ietf.org/html/rfc2616>.

[2]RFC 2617: HTTP Authentication: Basic and Digest Access Authentication <http://tools.ietf.org/html/rfc2617>.

[3]RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

3 Requirements

The motivations for this document are to:

- Use an existing XMPP connection to associate an HTTP request with an XMPP entity.

- Require confirmation of the request by the XMPP entity before allowing access.

- Ensure that the HTTP request was generated by the principal controlling the XMPP entity.

4 Use Case

The process flow for this protocol is as follows:

1. HTTP Client requests object via HTTP.

2. HTTP Server sends Authenticate Response via HTTP.

3. HTTP Client sends Authorization Request via HTTP (E1).

4. HTTP Server processes request and forwards it to XMPP Server.

5. XMPP Server requests confirmation via XMPP (E2).

6. XMPP Client confirms request via XMPP.

7. XMPP Server delivers confirmation to HTTP Server.

8. HTTP Server allows HTTP Client to access object (E3).

Error cases:

- **E1:** HTTP Client does not understand the presented authentication scheme.

- **E2:** HTTP Server does not recognize or understand the request.

- **E3:** HTTP Server denies access.

This process flow is described in more detail in the following sections.

4.1 HTTP Client Sends Request via HTTP

Let us stipulate that an XMPP user (say, <juliet@capulet.com>) learns of an HTTP URL (e.g., <https://files.shakespeare.lit:9345/missive.html>). The user then attempts to retrieve the URL using her HTTP Client, which opens a TCP connection to the appropriate port of the host and sends an HTTP request as defined in RFC 2616. The request method MAY be any valid HTTP request method, including user-defined methods.
An example is provided below:

Listing 1: HTTP Client Makes Request (No Credentials)

```
GET https://files.shakespeare.lit:9345/missive.html HTTP/1.1
```

In order to avoid a round trip, the initial request MAY contain HTTP authorization credentials as described below.

4.2 HTTP Server Returns Authenticate Response via HTTP

If the user did not provide authorization credentials in the initial request, the HTTP Server then MUST respond with a (401) Authenticate response as defined in RFC 2616. The response MUST contain an HTTP 401 error and one WWW-Authenticate header for each authentication scheme recognized by the HTTP Server. In order to provide verification via XMPP, at least one of these headers MUST specify a realm of "xmpp" (case-sensitive).

Listing 2: HTTP Server Returns Authenticate Response

```
401 Unauthorized HTTP/1.1
WWW-Authenticate: Basic realm="xmpp"
WWW-Authenticate: Digest realm="xmpp",
                  domain="files.shakespeare.lit",
                  stale=false,
                  nonce="ec2cc00f21f71acd35ab9be057970609",
                  qop="auth",
                  algorithm="MD5"
```

4.3 HTTP Client Sends Authorization Request via HTTP

The HTTP Client responds with an Authorization Request as defined in RFC 2616. The following rules apply:

1. The request MUST include the Jabber Identifier (JID) of the user making the request. This SHOULD be the full JID (<user@host/resource>) of a client that supports the protocol defined herein, although it MAY be the user's bare JID (<user@host>) instead.

2. The request MUST include a transaction identifier for the request. This identifier MUST be unique within the context of the HTTP Client's interaction with the HTTP Server. If the HTTP request is generated by the XMPP Client (e.g., because the HTTP URL was discovered via Out-of-Band Data (XEP-0066) [4]) then the transaction identifier SHOULD be generated by the client; if not, the transaction identifier SHOULD be provided by the human user who controls the HTTP Client.

The Authorization Request process is described in the following subsections.

4.3.1 Basic Access Authentication Scheme

The Basic Access Authentication scheme is defined in RFC 2617. This scheme specifies that the authorization information shall consist of a userid and password, separated by a ':' character and then encoded using Base64. When the realm is "xmpp", the profile defined herein further specifies that the userid MUST be a valid JID as described above, that the password entity MUST be a transaction identifier as described above, that any character in the JID or transaction identifier that is outside the range of the US-ASCII coded character set MUST be transformed into a percent-encoded octet as specified in Section 2.1 of RFC 3986 [5] prior to Base64 encoding, and that Base64 encoding MUST adhere to Section 4 of RFC 4648 [6].
(Refer to RFC 2617 for specification of the syntax of the Basic Access Authentication scheme; that information is not duplicated here.)

Listing 3: HTTP Client Makes Basic Authorization Request

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

4.3.2 Digest Access Authentication Scheme

The Digest Access Authentication scheme is defined in RFC 2617. This scheme specifies that the authorization information shall consist of the MD5 checksum of the username, a cnonce generated by the client, a nonce value provided in the challenge, the HTTP method, and the requested URL. When the realm is "xmpp", the profile defined herein further specifies that prior to creating the MD5 checksum the username MUST be a valid JID as described above, that the cnonce MUST be a transaction identifier as described above, and that any character in the JID or transaction identifier that is outside the range of the US-ASCII coded character set MUST be transformed into a percent-encoded octet as specified in Section 2.1 of RFC 3986.
(Refer to RFC 2617 for specification of the syntax of the Digest Access Authentication scheme; that information is not duplicated here.)

---

[4]XEP-0066: Out of Band Data <https://xmpp.org/extensions/xep-0066.html>.
[5]RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax <http://tools.ietf.org/html/rfc3986>.
[6]RFC 4648: The Base16, Base32, and Base64 Data Encodings <http://tools.ietf.org/html/rfc4648>.

Listing 4: HTTP Client Makes Digest Authorization Request

```
Authorization: Digest username="juliet@capulet.com",
               realm="xmpp",
               nonce="ec2cc00f21f71acd35ab9be057970609",
               uri="missive.html",
               qop=auth,
               nc=00000001,
               cnonce="0a4f113b",
               response="6629fae49393a05397450978507c4ef1",
               opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

4.3.3 Additional Authentication Schemes

The HTTP Server MAY offer any other valid authentication scheme, instead of or in addition to the Basic and Digest schemes mentioned above, as long as the scheme makes it possible to specify a userid (JID) and transaction identifier as described above. However, it is RECOMMENDED to implement both the Basic and Digest authentication schemes.

4.4 HTTP Server Processes Request

Once the HTTP Client has communicated the JID and transaction identifier to the HTTP Server, the HTTP Server MUST verify that the JID is authorized to access the HTTP resource. This may involve JID-level or domain-level access checks, or (depending on local service policies) potentially no access checks at all if only verification is required.

If the JID is authorized to access the HTTP resource, the HTTP Server MUST pass the URL, method, JID, and transaction identifier to the XMPP Server for confirmation. Exactly how this is done is up to the implementation. It is RECOMMENDED for the HTTP Server to connect to the XMPP Server as a trusted server component and to itself generate the confirmation request as described below.

4.5 XMPP Server Requests Confirmation via XMPP

Upon receiving the JID and transaction identifier from the HTTP Server, the XMPP Server MUST send a confirmation request (via XMPP) to the XMPP Client (or route the confirmation request generated by the HTTP Server acting as a trusted XMPP server component).

The confirmation request shall consist of an empty <confirm/> element qualified by the "http://jabber.org/protocol/http-auth' namespace. This element MUST possess a 'method' attribute whose value is the method of the HTTP request, MUST possess a 'url' attribute whose value is the full HTTP URL that was requested, and MUST possess an 'id' attribute whose value is the transaction identifier provided in the HTTP Authorization Request.

If the JID provided was a full JID, the confirmation request SHOULD be sent in an <iq/> stanza of type "get" whose 'to' attribute is set to the full JID, but MAY be sent in a <message/> stanza.

If the JID provided was a bare JID, the confirmation request MUST be sent in a <message/> stanza whose 'to' attribute is set to the bare JID; this enables delivery to the "most available" resource for the user (however "most available" is determined by the XMPP Server). The <message/> stanza SHOULD include a <thread/> element for tracking purposes and MAY include a <body/> element that provides human-readable information or instructions. If it however provides a <body/>, the server SHOULD be able to handle a plaintext reply from the client, in the case where it does not support this XEP.

Listing 5: Confirmation Request Sent via IQ

```
<iq type='get'
    from='files.shakespeare.lit'
    to='juliet@capulet.com/balcony'
    id='ha000'>
  <confirm xmlns='http://jabber.org/protocol/http-auth'
           id='a7374jnjlalasdf82'
           method='GET'
           url='https://files.shakespeare.lit:9345/missive.html'/>
</iq>
```

Listing 6: Confirmation Request Sent via Message

```
<message type='normal'
         from='files.shakespeare.lit'
         to='juliet@capulet.com'>
  <thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
  <body>
    Someone (maybe you) has requested the following file:

        https://files.shakespeare.lit:9345/missive.html.

    The transaction identifier is:

        a7374jnjlalasdf82

    If you wish to confirm the request, please reply
    to this message by typing "OK". If not, please
    reply with "No".
  </body>
  <confirm xmlns='http://jabber.org/protocol/http-auth'
           id='a7374jnjlalasdf82'
           method='GET'
           url='https://files.shakespeare.lit:9345/missive.html'/>
</message>
```

4.6 XMPP Client Confirms Request via XMPP

If the confirmation request was provided via an <iq/> stanza, the XMPP Client MUST respond to the request by sending an <iq/> stanza back to the XMPP Server. If the user wishes to confirm the request, the <iq/> response stanza MUST be of type "result" and MAY contain the original <confirm/> child element (although this is not necessary since the XMPP 'id' attribute can be used for tracking purposes):

Listing 7: XMPP Client Confirms Request via IQ

```
<iq type='result'
    from='juliet@capulet.com/balcony'
    to='files.shakespeare.lit'
    id='ha000'/>
```

If the user wishes to deny the request, the <iq/> response stanza MUST be of type "error", MAY contain the original <confirm/> child element (although this is not necessary since the XMPP 'id' attribute can be used for tracking purposes), and MUST specify an error, which SHOULD be <not-authorized/>:

Listing 8: XMPP Client Denies Request via IQ

```
<iq type='error'
    from='juliet@capulet.com/balcony'
    to='files.shakespeare.lit'
    id='ha000'>
  <confirm xmlns='http://jabber.org/protocol/http-auth'
           id='a7374jnjlalasdf82'
           method='GET'
           url='https://files.shakespeare.lit:9345/missive.html'/>
  <error code='401' type='auth'>
    <not-authorized xmlns='urn:ietf:params:xml:xmpp-stanzas'/>
  </error>
</iq>
```

If the confirmation request was provided via a <message/> stanza and the <message/> contains a human-readable <body/> or does not contain a <body/> but the XMPP Client understands the 'http://jabber.org/protocol/http-auth' namespace, the XMPP Client SHOULD respond to the request by sending a <message/> stanza back to the XMPP Server. If the user wishes to confirm the request, the <message/> response stanza SHOULD be of type "normal", MUST mirror the <thread/> ID (if provided by the XMPP Server), and MUST contain the original <confirm/> child element:

Listing 9: XMPP Client Confirms Request via Message

```
<message from='juliet@capulet.com/balcony'
         to='files.shakespeare.lit'>
```

```
  <thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
  <confirm xmlns='http://jabber.org/protocol/http-auth'
           id='a7374jnjlalasdf82'
           method='GET'
           url='https://files.shakespeare.lit:9345/missive.html'/>
</message>
```

If the user wishes to deny the request, the <message/> response stanza MUST be of type "error", MUST mirror the <thread/> ID (if provided by the XMPP Server), MUST contain the original <confirm/> child element, and MUST specify an error, which SHOULD be <not-authorized/>:

Listing 10: XMPP Client Denies Request via Message

```
<message type='error'
         from='juliet@capulet.com/balcony'
         to='files.shakespeare.lit'>
  <thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
  <confirm xmlns='http://jabber.org/protocol/http-auth'
           id='a7374jnjlalasdf82'
           method='GET'
           url='https://files.shakespeare.lit:9345/missive.html'/>
  <error code='401' type='auth'>
    <not-authorized xmlns='urn:ietf:params:xml:xmpp-stanzas'/>
  </error>
</message>
```

4.7 HTTP Server Allows HTTP Client to Access Object

Once the XMPP Client has successfully confirmed the request, the XMPP Server forwards that confirmation to the HTTP Server, which allows access:

Listing 11: HTTP Server Allows Access to Object

```
200 OK HTTP/1.1
Content-Type: text/html
Content-Length: 3032


...
```

If the XMPP Client denied the request, the HTTP Server MUST return a Forbidden error:

Listing 12: HTTP Server Denies Access to Object

```
403 Forbidden HTTP/1.1
```

5 Implementation Notes

5.1 Interaction of HTTP methods

For the HEAD and OPTIONS methods, the credentials SHOULD be usable for a subsequent request from the same entity. This enables an entity to both determine support for the mechanism defined herein and start the authentication process.
For the POST and PUT methods (or any method containing a message body), clients MUST send all data with each request (if needed, the client should obtain credentials with a previous HEAD or OPTIONS method).

6 Security Considerations

6.1 Association of Request

In order to associate the HTTP request with the XMPP confirmation, a transaction identifier MUST be provided by the user in the HTTP Authorization Request, then passed unchanged to the XMPP Client as the value of the <confirm/> element's 'id' attribute. If the XMPP Client generated the HTTP request, it MUST check the transaction identifier against the requests it has made to verify that the request has not yet been confirmed. If the XMPP Client did not generate the HTTP request, it MUST present the transaction identifier to the user for confirmation. If the XMPP Client or User confirms the request, the XMPP Client MUST then return a confirmation to the XMPP Server for delivery to the HTTP Server.

6.2 Channel Encryption

To reduce the likelihood of man-in-the-middle attacks, channel encryption SHOULD be used for both the XMPP channel and the HTTP channel. In particular:

1. The channel used for HTTP requests and responses SHOULD be encrypted via SSL (secure HTTP via https: URLs) or TLS (RFC 2817 [7]).

2. If the standard binding of XMPP to TCP is used, TLS SHOULD be negotiated for the XMPP channel in accordance with RFC 6120.

3. If a binding of XMPP to HTTP is used (e.g., as specified in XEP-0124), exchanges between the XMPP Client and XMPP Server (connection manager) SHOULD be sent over a channel that is encrypted using SSL or TLS.

---

[7]RFC 2817: Upgrading to TLS Within HTTP/1.1 <http://tools.ietf.org/html/rfc2817>.

6.3 End-to-End Encryption

For added security, the XMPP Server and XMPP Client may wish to communicate using end-to-end encryption. Methods for doing so are outside the scope of this proposal.

7 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [8].

8 XMPP Registrar Considerations

8.1 Protocol Namespaces

The XMPP Registrar [9] includes "http://jabber.org/protocol/http-auth" in its registry of protocol namespaces.

9 XML Schema

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

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

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0070: http://www.xmpp.org/extensions/xep-0070.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='confirm'>
    <xs:complexType>
```

---

[8]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/>.

[9]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/>.

```
    <xs:simpleContent>
      <xs:extension base='empty'>
        <xs:attribute name='id' use='required' type='xs:string'/>
        <xs:attribute name='method' use='required' type='xs:NCName'/
          >
        <xs:attribute name='url' use='required' type='xs:anyURI'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

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

</xs:schema>
```