XEP-0250: C2C Authentication Using TLS

Dirk Meyer

mailto:dmeyer@tzi.de
xmpp:dmeyer@jabber.org

2008-09-08
Version 0.2

Status	Type	Short Name
Deferred	Standards Track	NOT_YET_ASSIGNED

This document describes how to negotiate TLS extensions when using TLS for end-to-end XML streams between two clients. It covers X.509 certificates with an without CA, the use of OpenPGP, Shared Remote Passwords (SRP) and how to use one extension to bootstrap a trust relationship.

Legal

Copyright

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

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

For secure client-to-client (C2C) communication the clients can use Link-Local Messaging (XEP-0174) [1] or Jingle XML Streams (XEP-0247) [2] to open a connection between the two clients. To open an XMPP connection End-to-End XML Streams (XEP-0246) [3] defines a stream setup similar to the setup used by client-server communications. To secure the communication the extension defines the use of Transport Layer Security as defined in RFC 5246 [4] for encryption and authentication. XEP-0246 suggest to use the OpenPGP TLS extension but does not specify how to negotiate if both peers support the extension and if they are able to verify the OpenPGP key. It makes no sense to use OpenPGP instead of X.509 certificates if there is also no trust on OpenPGP level. This document describes how to negotiate how to use TLS to exchange possible extensions and key fingerprints before the actual TLS handshake.

After the TLS handshake **both** communication partners MUST be sure that they are communicating with the correct person without a man-in-the-middle.

2 Supported TLS Handshake Mechanisms

A client requires support for X.509 certificates or one or more TLS extension that can be used to verify the end-to-end character of the stream. The following list defines authentication mechanisms a client MAY support. The list depends on TLS extensions defined by the Transport Layer Security working group of the IETF. A future version of the document may include additional extension like Short Authentication String (SAS) or Kerberos. A client MUST ignore XML stanzas defining an authentication method it does not understand.

2.1 X.509 Certificate

The classic usage of TLS is done with X.509 certificates. The certificate is the end of a certificate chain. A certificate should be either signed by another certificate from a third party or a well known certification authority installed on the client machine. In an enterprise scenario all client would be signed by a certificate from the company making it possible for all clients to verify the identity of the other.

Getting a signed certificate is a complex and expensive task unsuitable for normal users. It also raises the question why a client should trust a CA its user does not know. Instead they create a self-signed certificate. The certificate is signed by itself and can not be verified through a certificate chain. To use these certificates in a real scenario the user must know the fingerprint of the peer certificate to verify it. But comparing fingerprints is not very userfriendly and the user may not have the fingerprint before starting the communication.

---

[1]XEP-0174: Link-Local Messaging <https://xmpp.org/extensions/xep-0174.html>.
[2]XEP-0247: Jingle XML Streams <https://xmpp.org/extensions/xep-0247.html>.
[3]XEP-0246: End-to-End XML Streams <https://xmpp.org/extensions/xep-0246.html>.
[4]RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 <http://tools.ietf.org/html/rfc5246>.

2.2 OpenPGP Key

Instead of relying on a certificate chain the users may rely on a web-of-trust as defined by RFC 4880 [5]. The TLS extension "OpenPGP Keys for Transport Layer Security" (RFC 5081 [6]) describes how to use OpenPGP keys in the TLS handshake. If users share a trusted friend who has signed both their keys they can open a secure connection based on that trust. If there is no trust relationship between the two peers, OpenPGP is as useless as self-signed certificates. The users can verify the fingerprint and communicate that fingerprint over a different medium (e.g. a real-life meeting), but the same problem that occurs with X.509 certificates exists: comparing fingerprints is far away from being userfriendly.

2.3 Shared Secret

A third way to verify the identify of the peer is a shared secret. This secret could be exchanged on a personal meeting or could be described as riddle only the other person can answer before opening the c2c link. It is up to the user and a good user interface to make the secret exchange as painless as possible.

This document uses the Secure Remote Password (SRP) extension from RFC 5054 [7] when dealing with shared secrets and not the Pre-Shared Key Ciphersuites as defined in RFC4279. SRP allows the users to choose a much smaller password and it still verifies both clients to the other. The password (shared secret) is required by both to calculate the premaster secret which means both will notice if the peer is not who it should be. SRP requires a user name; since it is transmitted in plain text between client and server it has no value in this context. Both clients must use the same username for the calculations. For the TLS handshake using SRP both clients MUST use the bare JID of the initiator as username.

3 Protocol

3.1 Extension Negotiation

The main problem is what TLS extension to choose. It makes no sense to use OpenPGP if the clients have no trust-relationship or SRP if the users did not exchange a secret. To resolve this problem a client describes its extension in an offer to the peer. The offer lists all supported authentication methods including additional parameter like the certificate that will be used. Public Key Publishing (XEP-0189) [8] can be used to look-up the keys or the client can look in its OpenPGP keyring. A client could also know a certificate or password from an earlier connection. The following example contains the X.509 certificate and the OpenPGP key from

---

[5]RFC 4880: OpenPGP Message Format <http://tools.ietf.org/html/rfc4880>.

[6]RFC 5081: Using OpenPGP Keys for Transport Layer Security (TLS) Authentication <http://tools.ietf.org/html/rfc5081>.

[7]RFC 5054: Using the Secure Remote Password (SRP) Protocol for TLS Authentication <http://tools.ietf.org/html/rfc5054>.

[8]XEP-0189: Public Key Publishing <https://xmpp.org/extensions/xep-0189.html>.

the examples in XEP-0189.

Listing 1: Client Offer Supporting X.509, OpenPGP and SRP

```
<offer xmlns='urn:xmpp:tmp:c2ctls'>
  <keyinfo xmlns='urn:xmpp:tmp:pubkey'>
    <name>428b1358a286430f628da23fb33ddaf6e474f5c5</name>
  </keyinfo>
  <keyinfo xmlns='urn:xmpp:tmp:pubkey'>
    <name>89d099a3428481cc63fe3fa44e7df2d002b4ce44</name>
  </keyinfo>
  <srp/>
</offer>
```

Note: the keyinfo only contains the name and not the public key itself. If the peer does not know the key, it can not determine if it is an X.509 certificate or OpenPGP key. This does not matter because if the client does not know the key, it can not use it for secure communication. If both clients know the offer of the other, they can determine what method to use to complete the TLS handshake without an error. Note: this information will be exchanged over an insecure communication channel and may be forged. If the information is altered it will be detected when doing the TLS handshake.

3.2 Extension Probing

I client MAY want to know the supported extensions of the peer before opening the client-to-client stream. For serverless messaging this is not possible but for server based communication a client can exchange offers with the peer without using Jingle to open a client-to-client stream. The methods the peer supports depend on additional information from the client. Depending on the X.509 certificate of the client the peer may not support this extension because it can not verify the certificate. To receive the offer from the peer the client sends an IQ stanza with its own offer.

Listing 2: Client Sends Extension Probing IQ

```
<iq type='get'
    from='romeo@montague.net/b345687ba7607d3ddf401a0257464843a0a1c0b7'
    to='juliet@capulet.com/da39a3ee5e6b4b0d3255bfef95601890afd80709'
    id='info'>
  <offer xmlns='urn:xmpp:tmp:c2ctls'>
    <keyinfo xmlns='urn:xmpp:tmp:pubkey'>
      <name>RomeoX509CertificateHash</name>
    </keyinfo>
    <keyinfo xmlns='urn:xmpp:tmp:pubkey'>
      <name>RomeoOpenPGPFingerprint</name>
    </keyinfo>
    <srp/>
```

```
    </offer>
</iq>
```

The receiver sends its offer back to the client. It MUST only send the keyinfo elements that match a keyinfo of the offer. E.g. if the responder does not find an X.509 certificate it can verify in the offer it MUST NOT send its own X.509 certificate. In the following example the receiver supports SRP and X.509. OpenPGP is either not supported or skipped because the key can not be verified.

Listing 3: Peer Sends Extension Probing Result

```
<iq type='result'
    from='juliet@capulet.com/da39a3ee5e6b4b0d3255bfef95601890afd80709'
    to='romeo@montague.net/b345687ba7607d3ddf401a0257464843a0a1c0b7'
    id='info'>
  <offer xmlns='urn:xmpp:tmp:c2ctls'>
    <keyinfo xmlns='urn:xmpp:tmp:pubkey'>
      <name>JulietX509CertificateHash</name>
    </keyinfo>
    <srp/>
  </offer>
</iq>
```

A client MAY support X.509 certificates it can not verify to be verified later. In that case it adds the authentication method <insecure>. In the following example the receiver supports SRP and X.509, but can not verify the certificate from the offer.

Listing 4: Peer Sends Extension Probing Result

```
<iq type='result'
    from='juliet@capulet.com/da39a3ee5e6b4b0d3255bfef95601890afd80709'
    to='romeo@montague.net/b345687ba7607d3ddf401a0257464843a0a1c0b7'
    id='info'>
  <offer xmlns='urn:xmpp:tmp:c2ctls'>
    <insecure/>
    <srp/>
  </offer>
</iq>
```

3.3   STARTTLS Enhancements

The offer information described above will be embedded into the XML based part of the STARTTLS handshake described in the XMPP core.

3.3.1 TLS Stream Feature

The recipient in the role of the XMPP server is the first one active in the XMPP TLS handshake. It offers the STARTTLS feature to the client. Instead of just offering the feature, it also SHOULD describe what TLS extensions it supports and what keys to expect. In the following example the recipient supports all possible identification mechanisms describe before.

Listing 5: Recipient Sends Stream Features

```
<stream:features>
  <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
    <required/>
    <offer xmlns='urn:xmpp:tmp:c2ctls'>
      <keyinfo xmlns='urn:xmpp:tmp:pubkey'>
        <name>RecipientX509CertificateHash</name>
      </keyinfo>
      <keyinfo xmlns='urn:xmpp:tmp:pubkey'>
        <name>RecipientOpenPGPFingerprint</name>
      </keyinfo>
      <srp/>
    </offer>
  </starttls>
</stream:features>
```

If the recipient does not supply this additional information it is assumed that it does not support this extension and the TLS handshake continues as described in the XMPP core. If and how the users trust each other in that case is out of the scope of this document.

3.3.2 Choosing the STARTTLS Feature

When the initiator starts the TLS feature it also enhances the STARTTLS with its supported extensions and additional key information based on the recipient's offer. In the following example the initiator can only verify one keyinfo. It also supports SRP.

Listing 6: Initiator Starts TLS Feature

```
<starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
  <offer xmlns='urn:xmpp:tmp:c2ctls'>
    <keyinfo xmlns='urn:xmpp:tmp:pubkey'>
      <name>InitiatorX509CertificateHash</name>
    </keyinfo>
    <srp/>
  </offer>
</starttls>
```

Similar to the feature announcement, if the initiator does not add an offer the recipient MUST assume the initiator does not support this XMPP extension and continue normal TLS

handshake and hope it will work.

3.3.3 Accepting STARTTLS

The next step for an recipient is to send an proceed message so the initiator can start with the TLS handshake. At this point the recipient knows what certificates or OpenPGP keys to expect from the peer and what the peer supports. The method that the initiator SHOULD use is added to the proceed.

Listing 7: Recipient Accepts TLS

```
<proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
  <offer xmlns='urn:xmpp:tmp:c2ctls'>
    <srp/>
  </offer>
</proceed>
```

The initiator now starts the TLS handshake.
If only SRP is possible the client should ask the user if they exchanged a shared secret. If this is not the case no suitable methods are left and the recipient MUST sends a failure and close the stream.

Listing 8: Recipient Aborts TLS

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
</stream:stream>
```

4 Bootstrapping Trust

After the c2c connection is secure the clients MAY use this secure and trusted connection to update their information on each other. Using XEP-0189 "Requesting Public Keys Directly From Another Entity" or "Sending Public Keys Directly To Another Entity" they can bootstrap a different mechanisms for the next time. A SRP-based connection can bootstrap the trust of a self-signed certificate which will used the next time these two clients connect to each other. It can also be used to automatically sign an OpenPGP key with a minimum level of trust.

5 Usability Considerations

Usability Considerations will be provided in a future version of this document.

6 Implementation Notes

It is RECOMMENDED to always create a self-signed X.509 certificate for a client. It is of less value than an OpenPGP key also used by other applications and connected to a web-of-trust. It MAY also be stored on the users PC unencrypted like the XMPP password. This makes it possible to open secure communications without entering the OpenPGP password or the shared secret. Of cause, storing the X.509 certificate in plain text reduces the security. It all depends on how much the user trusts the PC.

7 Security Considerations

Additional security considerations will be provided in a later version of this document.

8 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [9].

9 XMPP Registrar Considerations

XMPP Registrar considerations will be provided in a later version of this document.

10 Acknowledgements

The author would like to thank Eric Rescorla for suggesting TLS-SRP and others on the xmpp-security mailing list for the discussion.

11 XML Schema

The XML schema will be provided in a later version of this document.

---

[9]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/>.