

XMPP
Internet-Draft
Obsoletes: 6122 (if approved)
Intended status: Standards Track
Expires: October 16, 2012

P. Saint-Andre
Cisco Systems, Inc.
April 14, 2012

Extensible Messaging and Presence Protocol (XMPP): Address Format
draft-ietf-xmpp-6122bis-02

Abstract

This document defines the address format for the Extensible Messaging and Presence Protocol (XMPP), including support for code points outside the US-ASCII range. This document obsoletes RFC 6122.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on October 16, 2012.

Copyright Notice

Copyright (c) 2012 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Overview	3
1.2. Terminology	3
2. Addresses	3
2.1. Fundamentals	3
2.2. Domainpart	5
2.3. Localpart	6
2.4. Resourcepart	7
3. Enforcement in JIDs and JID Parts	8
4. Internationalization Considerations	10
5. Security Considerations	10
5.1. Reuse of PRECIS	10
5.2. Reuse of Unicode	11
5.3. Address Spoofing	11
5.3.1. Address Forging	11
5.3.2. Address Mimicking	12
6. IANA Considerations	13
6.1. Use of NameClass	13
6.2. Use of FreeClass	13
7. Conformance Requirements	13
8. References	15
8.1. Normative References	15
8.2. Informative References	16
Appendix A. Differences from RFC 6122	19
Appendix B. Acknowledgements	20
Author's Address	20

1. Introduction

1.1. Overview

The Extensible Messaging and Presence Protocol (XMPP) [RFC6120] is an application profile of the Extensible Markup Language [XML] for streaming XML data in close to real time between any two or more network-aware entities. The address format for XMPP entities was originally developed in the Jabber open-source community in 1999, first described by [XEP-0029] in 2002, and then defined canonically by [RFC3920] in 2004 and [RFC6122] in 2011.

As specified in RFC 3920 and RFC 6122, the XMPP address format used the "stringprep" technology for preparation of non-ASCII characters [RFC3454]. Following the migration of internationalized domain names away from stringprep, this document defines the XMPP address format in a way that no longer depends on stringprep. Instead, this document builds upon the internationalization framework defined by the IETF's PRECIS Working Group [FRAMEWORK].

This document obsoletes RFC 6122.

1.2. Terminology

Many important terms used in this document are defined in [FRAMEWORK], [RFC5890], [RFC6120], [RFC6365], and [UNICODE].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Addresses

2.1. Fundamentals

An XMPP entity is anything that is network-addressable and that can communicate using XMPP. For historical reasons, the native address of an XMPP entity is called a Jabber Identifier ("JID"). A valid JID is a string of [UNICODE] code points, encoded using UTF-8 [RFC3629], and structured as an ordered sequence of localpart, domainpart, and resourcepart (where the first two parts are demarcated by the '@' character used as a separator, and the last two parts are similarly demarcated by the '/' character).

The syntax for a JID is defined as follows using the Augmented Backus-Naur Form (ABNF) as specified in [RFC5234].

```
jid          = [ localpart "@" ] domainpart [ "/" resourcepart ]
localpart    = 1*(localpoint)
              ;
              ; a "localpoint" is a UTF-8 encoded Unicode
              ; code point that conforms to the localpart
              ; subclass of the "NameClass" string class
              ; defined in draft-ietf-precis-framework
              ;
domainpart    = IP-literal / IPv4address / ifqdn
              ;
              ; the "IPv4address" and "IP-literal" rules are
              ; defined in RFC 3986, and the first-match-wins
              ; (a.k.a. "greedy") algorithm described in RFC
              ; 3986 applies to the matching process
              ;
              ; note well that reuse of the IP-literal rule
              ; from RFC 3986 implies that IPv6 addresses are
              ; enclosed in square brackets (i.e., beginning
              ; with '[' and ending with ']')
              ;
ifqdn         = 1*(domainpoint)
              ;
              ; a "domainpoint" is a UTF-8 encoded Unicode
              ; code point that conforms to the "domain name"
              ; string class effectively defined in RFC 5890
              ;
resourcepart  = 1*(resourcepoint)
              ;
              ; a "resourcepoint" is a UTF-8 encoded Unicode
              ; code point that conforms to the resourcepart
              ; subclass of the "FreeClass" string class
              ; defined in draft-ietf-precis-framework
              ;
```

All JIDs are based on the foregoing structure. However, note that the foregoing structure does not capture all of the rules and restrictions that apply to JIDs, which are described below.

Each allowable portion of a JID (localpart, domainpart, and resourcepart) MUST NOT be zero bytes in length and MUST NOT be more than 1023 bytes in length, resulting in a maximum total size (including the '@' and '/' separators) of 3071 bytes.

Implementation Note: When dividing a JID into its component parts, an implementation needs to match the separator characters '@' and '/' before applying any transformation algorithms, which might decompose certain Unicode code points to the separator characters (e.g., under Unicode Normalization Form KC U+FE6B SMALL COMMERCIAL

AT decomposes to U+0040 COMMERCIAL AT, although this is not true under Unicode Normalization C, which is used in this specification).

This document defines the native format for JIDs; see [RFC5122] for information about the representation of a JID as a Uniform Resource Identifier (URI) [RFC3986] or Internationalized Resource Identifier (IRI) [RFC3987] and the extraction of a JID from an XMPP URI or IRI.

2.2. Domainpart

The domainpart of a JID is that portion after the '@' character (if any) and before the '/' character (if any); it is the primary identifier and is the only REQUIRED element of a JID (a mere domainpart is a valid JID). Typically a domainpart identifies the "home" server to which clients connect for XML routing and data management functionality. However, it is not necessary for an XMPP domainpart to identify an entity that provides core XMPP server functionality (e.g., a domainpart can identify an entity such as a multi-user chat service [XEP-0045], a publish-subscribe service [XEP-0060], or a user directory).

The domainpart for every XMPP service MUST be a fully-qualified domain name (FQDN), an IPv4 address, an IPv6 address, or an unqualified hostname (i.e., a text label that is resolvable on a local network).

Informational Note: The term "fully-qualified domain name" is not well defined. In [RFC1034] it is also called an absolute domain name, and the two terms are associated in [RFC1535]. The earliest use of the term can be found in [RFC1123]. References to those older specifications ought not to be construed as limiting the characters of a fully-qualified domain name to the ASCII range; for example, [RFC5890] mentions that a fully-qualified domain name can contain one or more U-labels.

Interoperability Note: Domainparts that are IP addresses might not be accepted by other services for the sake of server-to-server communication, and domainparts that are unqualified hostnames cannot be used on public networks because they are resolvable only on a local network.

If the domainpart includes a final character considered to be a label separator (dot) by [RFC1034], this character MUST be stripped from the domainpart before the JID of which it is a part is used for the purpose of routing an XML stanza, comparing against another JID, or constructing an [RFC5122]. In particular, the character MUST be stripped before any other canonicalization steps are taken.

A domainpart MUST NOT be zero bytes in length and MUST NOT be more than 1023 bytes in length. This rule is to be enforced after any mapping or normalization of code points. Naturally, the length limits of [RFC1034] apply, and nothing in this document is to be interpreted as overriding those more fundamental limits.

In the terms of IDNA2008 [RFC5890], the domainpart of a JID is an "IDNA-aware domain name slot".

A domainpart consisting of a fully-qualified domain name MUST be an "internationalized domain name" as defined in [RFC5890] and MUST consist only of Unicode code points that conform to the rules specified in [RFC5892].

The domainpart of a JID MUST be treated as follows, where the operations specified MUST be completed in the order shown:

1. Uppercase and titlecase characters MUST be mapped to their lowercase equivalents.
2. Additional mappings MAY be applied, such as those defined in [MAPPINGS].
3. All characters MUST be mapped using Unicode Normalization Form C (NFC).
4. Each A-label MUST be converted to a U-label.

With regard to directionality, applications MUST apply the "Bidi Rule" defined in [RFC5893] (i.e., each of the six conditions of the Bidi Rule must be satisfied).

2.3. Localpart

The localpart of a JID is an optional identifier placed before the domainpart and separated from the latter by the '@' character. Typically a localpart uniquely identifies the entity requesting and using network access provided by a server (i.e., a local account), although it can also represent other kinds of entities (e.g., a chat room associated with a multi-user chat service [XEP-0045]). The entity represented by an XMPP localpart is addressed within the context of a specific domain (i.e., <localpart@domainpart>).

A localpart MUST NOT be zero bytes in length and MUST NOT be more than 1023 bytes in length. This rule is to be enforced after any mapping or normalization of code points.

A localpart MUST consist only of Unicode code points that conform to

the "NameClass" base string class defined in [FRAMEWORK], with the exception of the following characters that are explicitly disallowed in XMPP localparts:

U+0022 (QUOTATION MARK), i.e., "
U+0026 (AMPERSAND), i.e., &
U+0027 (APOSTROPHE), i.e., '
U+002F (SOLIDUS), i.e., /
U+003A (COLON), i.e., :
U+003C (LESS-THAN SIGN), i.e., <
U+003E (GREATER-THAN SIGN), i.e., >
U+0040 (COMMERCIAL AT), i.e., @

The localpart of a JID MUST be treated as follows, where the operations specified MUST be completed in the order shown:

1. Uppercase and titlecase characters MUST be mapped to their lowercase equivalents.
2. Additional mappings MAY be applied, such as those defined in [MAPPINGS].
3. All characters MUST be mapped using Unicode Normalization Form C (NFC).

With regard to directionality, applications MUST apply the "Bidi Rule" defined in [RFC5893] (i.e., each of the six conditions of the Bidi Rule must be satisfied).

2.4. Resourcepart

The resourcepart of a JID is an optional identifier placed after the domainpart and separated from the latter by the '/' character. A resourcepart can modify either a <localpart@domainpart> address or a mere <domainpart> address. Typically a resourcepart uniquely identifies a specific connection (e.g., a device or location) or object (e.g., an occupant in a multi-user chat room [XEP-0045]) belonging to the entity associated with an XMPP localpart at a domain (i.e., <localpart@domainpart/resourcepart>).

A resourcepart MUST NOT be zero bytes in length and MUST NOT be more than 1023 bytes in length. This rule is to be enforced after any mapping or normalization of code points.

A resourcepart MUST consist only of Unicode code points that conform to the "FreeClass" base string class defined in [FRAMEWORK].

The localpart of a JID MUST be treated as follows, where the

operations specified MUST be completed in the order shown:

1. Uppercase and titlecase characters MAY be mapped to their lowercase equivalents.
2. Additional mappings MAY be applied, such as those defined in [MAPPINGS].
3. All characters MUST be mapped using Unicode Normalization Form C (NFC).

With regard to directionality, applications MUST apply the "Bidi Rule" defined in [RFC5893] (i.e., each of the six conditions of the Bidi Rule must be satisfied).

XMPP entities SHOULD consider resourceparts to be opaque strings and SHOULD NOT impute meaning to any given resourcepart. In particular:

- o Use of the '/' character as a separator between the domainpart and the resourcepart does not imply that XMPP addresses are hierarchical in the way that, say, HTTP addresses are hierarchical; thus for example an XMPP address of the form <localpart@domainpart/foo/bar> does not identify a resource "bar" that exists below a resource "foo" in a hierarchy of resources associated with the entity "localpart@domainpart".
- o The '@' character is allowed in the resourcepart and is often used in the "nick" shown in XMPP chatrooms [XEP-0045]. For example, the JID <room@chat.example.com/user@host> describes an entity who is an occupant of the room <room@chat.example.com> with an (asserted) nick of <user@host>. However, chatroom services do not necessarily check such an asserted nick against the occupant's real JID.

3. Enforcement in JIDs and JID Parts

Enforcement of the XMPP address format rules is the responsibility of XMPP servers. Although XMPP clients SHOULD prepare complete JIDs and parts of JIDs in accordance with these rules before including them in protocol slots within XMPP streams, XMPP servers MUST enforce the rules wherever possible.

Enforcement applies to complete JIDs and to parts of JIDs. To facilitate implementation, this document defines the concepts of "JID slot", "localpart slot", and "resourcepart slot" (similar to the concept of a "domain name slot" for IDNA2008 defined in Section 2.3.2.6 of [RFC5890]):

JID Slot: An XML element or attribute explicitly designated in XMPP or in XMPP extensions for carrying a complete JID.

Localpart Slot: An XML element or attribute explicitly designated in XMPP or in XMPP extensions for carrying the localpart of a JID.

Resourcepart Slot: An XML element or attribute explicitly designated in XMPP or in XMPP extensions for carrying the resourcepart of a JID.

In general, servers are responsible for enforcing the address format rules when receiving protocol elements from clients where the server is expected to process or act on such elements; two examples from [RFC6120] are the 'to' attribute on XML stanzas (which is a JID slot used by XMPP servers for routing of outbound stanzas) and the <resource/> child of the <bind/> element (which is a resourcepart slot used by XMPP servers for binding of a resource to an account for routing of stanzas between the server and a particular client). However, servers are not responsible for enforcing the rules when the protocol elements are intended for communication among other entities; two examples are the 'initiator' attribute in the Jingle extension [XEP-0166] (which is a JID slot used for client-to-client coordination of multimedia sessions) and the 'nick' attribute in the Multi-User Chat extension [XEP-0045] (which is a resourcepart slot used for administrative purposes in the context of XMPP chatrooms); in such cases, clients SHOULD enforce the rules, and client implementers need to understand that not enforcing the rules can lead to a degraded user experience or security vulnerabilities.

This document does not provide an exhaustive list of JID slots, localpart slots, or resourcepart slots. However, implementers of core XMPP servers are advised to consider as JID slots at least the following elements and attributes:

- o The 'from' and 'to' stream attributes and the 'from' and 'to' stanza attributes [RFC6120].
- o The 'jid' attribute of the roster <item/> element for contact list management [RFC6121].
- o The 'value' attribute of the <item/> element for Privacy Lists [RFC3921] [XEP-0016] when the value of the 'type' attribute is "jid".
- o The 'jid' attribute of the <item/> element for Service Discovery defined in [XEP-0030].
- o The <value/> element for Data Forms [XEP-0004] when the 'type' attribute is "jid-single" or "jid-multi".
- o The 'jid' attribute of the <conference/> element for Bookmark Storage [XEP-0048].

- o The <JABBERID/> of the <vCard/> element for vCard 3.0 [XEP-0054] and the <uri/> child of the <impp/> element for vCard 4.0 [XEP-0292] when the XML character data identifies an XMPP URI [RFC5122].
- o The 'from' attribute of the <delay/> element for Delayed Delivery [XEP-0203].
- o The 'jid' attribute of the <item/> element for Simple Communications Blocking [XEP-0191].
- o The 'from' and 'to' attributes of the <result/> and <verify/> elements for Server Dialback [RFC3921], [XEP-0220].
- o The 'from' and 'to' attributes of the <amp/> element for Advanced Message Processing [XEP-0079].
- o The 'from' and 'to' attributes of the <iq/>, <message/>, and <presence/> elements for the Jabber Component Protocol [XEP-0114].

Developers of XMPP clients and specialized XMPP components are advised to check the appropriate specifications for JID slots, localpart slots, and resourcepart slots in XMPP protocol extensions such as Multi-User Chat [XEP-0045], Publish-Subscribe [XEP-0060], SOCKS5 Bytestreams [XEP-0065], In-Band Registration [XEP-0077], Roster Item Exchange [XEP-0144], and Jingle [XEP-0166].

4. Internationalization Considerations

XMPP applications MUST support IDNA2008 for domainparts, the "NameClass" string class from [FRAMEWORK] for localparts (with the exception of certain ASCII characters specified under Section 2.3), and the "FreeClass" string class from [FRAMEWORK] for resourceparts. This enables XMPP addresses to include a wide variety of characters outside the US-ASCII range. Rules for enforcement of the XMPP address format are provided in [RFC6120] and specifications for various XMPP extensions.

For backward compatibility, many XMPP applications support IDNA2003 [RFC3490] for domainparts, and the stringprep [RFC3454] profiles Nodeprep and Resourceprep [RFC3920] for localparts and resourceparts.

5. Security Considerations

5.1. Reuse of PRECIS

The security considerations described in [FRAMEWORK] apply to the "NameClass" and "FreeClass" base string classes used in this document for XMPP localparts and resourceparts. The security considerations described in [RFC5890] apply to internationalized domain names, which are used here for XMPP domainparts.

5.2. Reuse of Unicode

The security considerations described in [UTR39] apply to the use of Unicode characters in XMPP addresses.

5.3. Address Spoofing

There are two forms of address spoofing: forging and mimicking.

5.3.1. Address Forging

In the context of XMPP technologies, address forging occurs when an entity is able to generate an XML stanza whose 'from' address does not correspond to the account credentials with which the entity authenticated onto the network (or an authorization identity provided during negotiation of SASL authentication [RFC4422] as described in [RFC6120]). For example, address forging occurs if an entity that authenticated as "juliet@im.example.com" is able to send XML stanzas from "nurse@im.example.com" or "romeo@example.net".

Address forging is difficult in XMPP systems, given the requirement for sending servers to stamp 'from' addresses and for receiving servers to verify sending domains via server-to-server authentication (see [RFC6120]). However, address forging is possible if:

- o A poorly implemented server ignores the requirement for stamping the 'from' address. This would enable any entity that authenticated with the server to send stanzas from any localpart@domainpart as long as the domainpart matches the sending domain of the server.
- o An actively malicious server generates stanzas on behalf of any registered account at the domain or domains hosted at that server.

Therefore, an entity outside the security perimeter of a particular server cannot reliably distinguish between JIDs of the form <localpart@domainpart> at that server and thus can authenticate only the domainpart of such JIDs with any level of assurance. This specification does not define methods for discovering or counteracting the kind of poorly implemented or rogue servers just described. However, the end-to-end authentication or signing of XMPP stanzas could help to mitigate this risk, since it would require the rogue server to generate false credentials for signing or encryption of each stanza, in addition to modifying 'from' addresses.

Furthermore, it is possible for an attacker to forge JIDs at other domains by means of a DNS poisoning attack if DNS security extensions [RFC4033] are not used.

5.3.2. Address Mimicking

Address mimicking occurs when an entity provides legitimate authentication credentials for and sends XML stanzas from an account whose JID appears to a human user to be the same as another JID. Because many characters are visually similar, it is relatively easy to mimic JIDs in XMPP systems. As one simple example, the localpart "juliet" (using the Arabic numeral one as the third character) might appear the same as the localpart "juliet" (using lowercase "L" as the third character).

As explained in [RFC5890], [FRAMEWORK], [UTR36], and [UTR39], there is no straightforward solution to the problem of visually similar characters. Furthermore, IDNA and PRECIS technologies do not attempt to define such a solution. As a result, XMPP domainparts, localparts, and resourceparts could contain such characters, leading to security vulnerabilities such as the following:

- o A domainpart is always employed as one part of an entity's address in XMPP. One common usage is as the address of a server or server-side service, such as a multi-user chat service [XEP-0045]. The security of such services could be compromised based on different interpretations of the internationalized domainpart; for example, a user might authorize a malicious entity at a fake server to view the user's presence information, or a user could join chatrooms at a fake multi-user chat service.
- o A localpart can be employed as one part of an entity's address in XMPP. One common usage is as the username of an instant messaging user; another is as the name of a multi-user chat room; and many other kinds of entities could use localparts as part of their addresses. The security of such services could be compromised based on different interpretations of the internationalized localpart; for example, a user entering a single internationalized localpart could access another user's account information, or a user could gain access to a hidden or otherwise restricted chat room or service.
- o A resourcepart can be employed as one part of an entity's address in XMPP. One common usage is as the name for an instant messaging user's connected resource; another is as the nickname of a user in a multi-user chat room; and many other kinds of entities could use resourceparts as part of their addresses. The security of such services could be compromised based on different interpretations of the internationalized resourcepart; for example, two or more confusable resources could be bound at the same time to the same account (resulting in inconsistent authorization decisions in an XMPP application that uses full JIDs), or a user could send a

message to someone other than the intended recipient in a multi-user chat room.

XMPP services and clients are strongly encouraged to define and implement consistent policies regarding the registration, storage, and presentation of visually similar characters in XMPP systems. In particular, service providers and software implementers are strongly encouraged to use the policies recommended in [FRAMEWORK].

6. IANA Considerations

6.1. Use of NameClass

The IANA shall add an entry to the PRECIS Usage Registry for reuse of the PRECIS NameClass in XMPP, as follows:

Application Protocol: XMPP.
Base Class: NameClass.
Subclassing: Yes. See Section 2.3 of RFC XXXX.
Directionality: If the string contains at least one right-to-left code point, the entire string is considered to be right-to-left.
Casemapping: Uppercase and titlecase code points are mapped to their lowercase equivalents.
Normalization: NFC.
Specification: RFC XXXX.

6.2. Use of FreeClass

The IANA shall add an entry to the PRECIS Usage Registry for reuse of the PRECIS FreeClass in XMPP, as follows:

Application Protocol: XMPP.
Base Class: FreeClass
Subclassing: No.
Directionality: If the string contains at least one right-to-left code point, the entire string is considered to be right-to-left.
Casemapping: None.
Normalization: NFC.
Specification: RFC XXXX.

7. Conformance Requirements

This section describes a protocol feature set that summarizes the conformance requirements of this specification. This feature set is appropriate for use in software certification, interoperability testing, and implementation reports. For each feature, this section

provides the following information:

- o A human-readable name
- o An informational description
- o A reference to the particular section of this document that normatively defines the feature
- o Whether the feature applies to the Client role, the Server role, or both (where "N/A" signifies that the feature is not applicable to the specified role)
- o Whether the feature **MUST** or **SHOULD** be implemented, where the capitalized terms are to be understood as described in [RFC2119]

The feature set specified here provides a basis for interoperability testing and follows the spirit of a proposal made by Larry Masinter within the IETF's NEWTRK Working Group in 2005 [INTEROP].

Feature: address-domain-length

Description: Ensure that the domainpart of an XMPP address is at least one byte in length and at most 1023 bytes in length, and conforms to the underlying length limits of the DNS.

Section: Section 2.2

Roles: Server **MUST**, client **SHOULD**.

Feature: address-domain-prep

Description: Ensure that the domainpart of an XMPP address conforms to IDNA2008, mapped to lowercase and normalized using NFC.

Section: Section 2.2

Roles: Server **MUST**, client **SHOULD**.

Feature: address-localpart-length

Description: Ensure that the localpart of an XMPP address is at least one byte in length and at most 1023 bytes in length.

Section: Section 2.3

Roles: Server **MUST**, client **SHOULD**.

Feature: address-localpart-prep

Description: Ensure that the localpart of an XMPP address conforms to the "NameClass" base string class from the PRECIS framework, excluding the eight XMPP prohibited code points (U+0022, U+0026, U+0027, U+002F, U+003A, U+003C, U+003E, and U+0040), with all code points mapped to lowercase and normalized using NFC.

Section: Section 2.3

Roles: Server **MUST**, client **SHOULD**.

Feature: address-resource-length

Description: Ensure that the resourcepart of an XMPP address is at least one byte in length and at most 1023 bytes in length.

Section: Section 2.4

Roles: Server MUST, client SHOULD.

Feature: address-resource-prep

Description: Ensure that the resourcepart of an XMPP address conforms to the "FreeClass" base string class from the PRECIS framework, with all code points normalized using NFC.

Section: Section 2.4

Roles: Server MUST, client SHOULD.

8. References

8.1. Normative References

[FRAMEWORK]

Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Handling Internationalized Strings in Protocols", draft-ietf-precis-framework-02 (work in progress), March 2012.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[RFC5892] Faltstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.

[RFC5893] Alvestrand, H. and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, August 2010.

- [UNICODE] The Unicode Consortium, "The Unicode Standard, Version 3.2.0", 2000.
- The Unicode Standard, Version 3.2.0 is defined by The Unicode Standard, Version 3.0 (Reading, MA, Addison-Wesley, 2000. ISBN 0-201-61633-5), as amended by the Unicode Standard Annex #27: Unicode 3.1 (<http://www.unicode.org/reports/tr27/>) and by the Unicode Standard Annex #28: Unicode 3.2 (<http://www.unicode.org/reports/tr28/>).
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [UTR36] The Unicode Consortium, "Unicode Technical Report #36: Unicode Security Considerations", 2008, <<http://www.unicode.org/reports/tr36/>>.
- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

8.2. Informative References

- [INTEROP] Masinter, L., "Formalizing IETF Interoperability Reporting", Work in Progress, October 2005.
- [MAPPINGS] Yoneya, Y. and T. NEMOTO, "Mapping characters for PRECIS classes", draft-yoneya-precis-mappings-01 (work in progress), February 2012.
- [RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.
- [RFC1535] Gavron, E., "A Security Problem and Proposed Correction With Widely Deployed DNS Software", RFC 1535, October 1993.
- [RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.
- [RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

See Section 1 for an explanation of why the normative reference to an obsoleted specification is needed.

- [RFC3920] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, October 2004.
- [RFC3921] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 3921, October 2004.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.
- [RFC5894] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, August 2010.
- [RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, March 2011.
- [RFC6122] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 6122, March 2011.
- [RFC6365] Hoffman, P. and J. Klensin, "Terminology Used in Internationalization in the IETF", BCP 166, RFC 6365, September 2011.
- [UTR39] The Unicode Consortium, "Unicode Technical Report #39: Unicode Security Mechanisms", August 2010, <<http://unicode.org/reports/tr39/>>.
- [XEP-0004] Eatmon, R., Hildebrand, J., Miller, J., Muldowney, T., and P. Saint-Andre, "Data Forms", XSF XEP 0004, August 2007.
- [XEP-0016] Millard, P. and P. Saint-Andre, "Privacy Lists", XSF XEP 0016, February 2007.

- [XEP-0029] Kaes, C., "Definition of Jabber Identifiers (JIDs)", XSF XEP 0029, October 2003.
- [XEP-0030] Hildebrand, J., Millard, P., Eatmon, R., and P. Saint-Andre, "Service Discovery", XSF XEP 0030, June 2008.
- [XEP-0045] Saint-Andre, P., "Multi-User Chat", XSF XEP 0045, July 2008.
- [XEP-0048] Blackman, R., Millard, P., and P. Saint-Andre, "Bookmarks", XSF XEP 0048, November 2007.
- [XEP-0054] Saint-Andre, P., "vcard-temp", XSF XEP 0054, July 2008.
- [XEP-0060] Millard, P., Saint-Andre, P., and R. Meijer, "Publish-Subscribe", XSF XEP 0060, July 2010.
- [XEP-0065] Smith, D., Miller, M., Saint-Andre, P., and J. Karneges, "SOCKS5 Bytestreams", XSF XEP 0065, April in progress, last updated 2010.
- [XEP-0077] Saint-Andre, P., "In-Band Registration", XSF XEP 0077, September 2009.
- [XEP-0079] Miller, M. and P. Saint-Andre, "Advanced Message Processing", XSF XEP 0079, November 2005.
- [XEP-0114] Saint-Andre, P., "Jabber Component Protocol", XSF XEP 0114, March 2005.
- [XEP-0144] Saint-Andre, P., "Roster Item Exchange", XSF XEP 0144, August 2005.
- [XEP-0165] Saint-Andre, P., "Best Practices to Discourage JID Mimicking", XSF XEP 0165, December 2007.

- [XEP-0166] Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan, S., and J. Hildebrand, "Jingle", XSF XEP 0166, December 2009.
- [XEP-0191] Saint-Andre, P., "Simple Communications Blocking", XSF XEP 0191, February 2007.
- [XEP-0203] Saint-Andre, P., "Delayed Delivery", XSF XEP 0203, September 2009.
- [XEP-0220] Miller, J., Saint-Andre, P., and P. Hancke, "Server Dialback", XSF XEP 0220, March 2010.
- [XEP-0292] Saint-Andre, P. and S. Mizzi, "vCard4 Over XMPP", XSF XEP 0292, October 2011.
- [XML] Paoli, J., Maler, E., Sperberg-McQueen, C., Yergeau, F., and T. Bray, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", World Wide Web Consortium Recommendation REC-xml-20060816, August 2006, <<http://www.w3.org/TR/2006/REC-xml-20060816>>.
- [RFC5122] Saint-Andre, P., "Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)", RFC 5122, February 2008.

Appendix A. Differences from RFC 6122

Based on consensus derived from working group discussion, implementation and deployment experience, and formal interoperability testing, the following substantive modifications were made from RFC 6122.

- o Changed domainpart preparation to use IDNA2008 (instead of IDNA2003).
- o Changed localpart preparation to use the PRECIS NameClass (instead of the Nodeprep profile of Stringprep).
- o Changed resourcepart preparation to use the PRECIS FreeClass (instead of the Resourceprep profile of Stringprep).

- o Specified that internationalized labels within domainparts must be U-labels (instead of should be U-labels).
- o Specified that servers must enforce the address formatting rules.

Appendix B. Acknowledgements

Thanks to Joe Hildebrand and Florian Zeitz for their feedback.

Some text in this document was borrowed or adapted from [RFC5890], [RFC5891], [RFC5894], and [XEP-0165].

Author's Address

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3282
Email: psaintan@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2012

M. Miller
P. Saint-Andre
Cisco Systems, Inc.
June 27, 2012

Using DNS Security Extensions (DNSSEC) and DNS-based Authentication of
Named Entities (DANE) as a Proofotype for XMPP Domain Name Associations
draft-miller-xmpp-dnssec-proofotype-02

Abstract

This document defines a proofotype that uses DNS-based Authentication of Named Entities (DANE) for associating a domain name with an XML stream in the Extensible Messaging and Presence Protocol (XMPP). It also defines a method that uses DNS Security (DNSSEC) for securely delegating a source domain to a derived domain in XMPP.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on December 29, 2012.

Copyright Notice

Copyright (c) 2012 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Requirements	3
4. Secure Delegation	4
5. Proofotype	4
5.1. No Service Records	4
5.2. Insecure Delegation	5
5.3. Secure Delegation	5
6. Internationalization Considerations	5
7. Security Considerations	5
8. IANA Considerations	5
9. Normative References	6
Authors' Addresses	7

1. Introduction

The [XMPP-DNA] specification defines a framework for secure delegation and authenticated domain name associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP). This document defines a a secure delegation method that uses DNS Security (DNSSEC) [RFC4033] in conjunction with the standard DNS SRV records [RFC2782] employed in domain name resolution in XMPP, with the result that a client or peer server that inititates an XMPP stream can legitimately treat a derived domain as a reference identifier during stream negotiation. This document also defines a proofotype for DNA that uses DNS-based Authentication of Named Entities [DANE] to verify TLS certificates containing source domains or derived domains during stream negotiation.

2. Terminology

This document inherits XMPP-related terminology from [RFC6120], DNS-related terminology from [RFC1034], [RFC1035], [RFC2782] and [RFC4033], and security-related terminology from [RFC4949] and [RFC5280]. The terms "source domain", "derived domain", "reference identifier", and "presented identifier" are used as defined in the "CertID" specification [RFC6125].

This document is applicable to connections made from an XMPP client to an XMPP server ("_xmpp-client._tcp") or between XMPP servers ("_xmpp-server._tcp"). In both cases, the XMPP initiating entity acts as a TLS client and the XMPP receiving entity acts as a TLS server. Therefore, to simplify discussion this document uses "_xmpp-client._tcp" to describe to both cases, unless otherwise indicated.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Requirements

An XMPP initiating entity (TLS client) that wishes to use this proofotype MUST do so before exchanging stanzas addressed to the source domain. In general, this means that the proof MUST be completed before the XMPP stream is restarted following STARTTLS negotiation (as specified in [RFC6120]). However, connections between XMPP servers MAY also use this proofotype to verify the addition of new source domains onto an existing connection, such as multiplexing or "piggybacking" via [XEP-0220].

4. Secure Delegation

An XMPP initiating entity (TLS client) that wishes to use this proofotype performs the following actions:

1. Query for the appropriate SRV resource record for the source domain (e.g. "_xmpp-client._tcp.im.example.com").
2. If there is no SRV resource record, pursue the fallback methods described in [RFC6120].
3. If there is an SRV resource record, validate that the SRV record answer is secure according to [RFC4033]. If the answer is insecure, then delegation to the derived domain(s), as indicated by the "target host" field, is insecure and the TLS client MUST treat only the source domain as a reference identifier during certificate verification, as described in [RFC6120]; if the answer is bogus, the TLS client MUST abort.
4. If the answer is secure, the TLS client SHOULD consider any derived domain(s) in the answer as securely delegated; during certificate verification, the TLS client MUST treat both the source domain and the derived domain to which it has connected as reference identifiers.

5. Proofotype

[DANE] provides additional tools to verify the keys used in TLS connections. A TLS client MAY use [DANE] for TLS certificate verification; its use depends on the delegation status of the source domain, as described in the following sections.

5.1. No Service Records

If no SRV records are found for the source domain, then the TLS client MUST query for a TLSA resource record as described in [DANE], where the prepared domain name MUST contain the source domain and the IANA-registered port 5222 for client-to-server streams (e.g. "_5222._tcp.im.example.com") or the IANA-registered port 5269 for server-to-server streams (e.g. "_5269._tcp.im.example.com").

In this case, the TLS client MUST treat only the source domain as its reference identifier during certificate verification, as described in [RFC6120].

5.2. Insecure Delegation

If the delegation of a source domain to a derived domain is not secure, then the TLS client MUST NOT make a TLSA record query to the derived domain as described in [DANE]. Instead, the TLS client MUST treat only the source domain as its reference identifier during certificate verification, as described in [RFC6120], and MUST NOT use [DANE].

5.3. Secure Delegation

If the source domain has been delegated to a derived domain in a secure manner as described under Section 4, then the TLS client MUST query for a TLSA resource record as described in [DANE], where the prepared domain name MUST contain the derived domain and a port obtained from the SRV answer (e.g., "_5555._tcp/hosting.example.net" for an SRV record such as "_xmpp-client._tcp.im.example.com IN TLSA 1 1 5555 hosting.example.net").

If no TLSA resource records exist for the specified service, then the TLS client MUST perform certificate verification as described under Section 4.

If TLSA resource records exist for the specified service, then the TLS client MUST treat the derived domain(s) as its reference identifier during certificate verification, using the information from the TLSA answer as the basis for verification as described in [DANE].

6. Internationalization Considerations

If the SRV, A/AAAA, and TLSA record queries are for an internationalized domain name, then they need to use the A-label form as defined in [RFC5890].

7. Security Considerations

This document supplements but does not supersede the security considerations provided in [RFC4033], [RFC6120], [RFC6125], and [DANE].

8. IANA Considerations

This document has no actions for the IANA.

9. Normative References

- [DANE] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", draft-ietf-dane-protocol-23 (work in progress), June 2012.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, May 2005.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.
- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.
- [XEP-0220] Miller, J., Saint-Andre, P., and P. Hancke, "Server Dialback", XSF XEP 0220, August 2011.

[XMPP-DNA]

Saint-Andre, P. and M. Miller, "Domain Name Associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP)", draft-saintandre-xmpp-dna-00 (work in progress), June 2012.

Authors' Addresses

Matthew Miller
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Email: mamille2@cisco.com

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Email: psaintan@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2013

M. Miller
Cisco Systems, Inc.
July 13, 2012

End-to-End Object Encryption for the Extensible Messaging and Presence
Protocol (XMPP)
draft-miller-xmpp-e2e-02

Abstract

This document defines a method of end-to-end object encryption for the Extensible Messaging and Presence Protocol (XMPP).

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on January 14, 2013.

Copyright Notice

Copyright (c) 2012 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Determining Support	3
4. Encrypting XMPP Stanzas	4
4.1. Prerequisites	4
4.2. Process	4
4.3. Example - Securing a Message	6
5. Decrypting XMPP Stanzas	9
5.1. Protocol Not Understood	9
5.2. Process	10
5.3. Insufficient Information	10
5.4. Failed Decryption	11
5.5. Timestamp Not Acceptable	12
5.6. Successful Decryption	12
6. Requesting Session Keys	12
6.1. Request Process	13
6.2. Accept Process	13
6.3. Error Conditions	14
6.4. Example of Successful Key Request	14
7. Inclusion and Checking of Timestamps	17
8. Interaction with Stanza Semantics	18
9. Mandatory-to-Implement Cryptographic Algorithms	19
10. Security Considerations	19
10.1. Storage of Encrypted Stanzas	19
10.2. Re-use of Session Master Keys	19
11. IANA Considerations	19
11.1. XML Namespace Name for e2e Data in XMPP	19
12. References	20
12.1. Normative References	20
12.2. Informative References	21
Appendix A. Schema for urn:ietf:params:xml:ns:xmpp-e2e:3	21
Author's Address	23

1. Introduction

End-to-end encryption of traffic sent over the Extensible Messaging and Presence Protocol [RFC6120] is a desirable goal. Requirements and a threat analysis for XMPP encryption are provided in [E2E-REQ]. Many possible approaches to meet those (or similar) requirements have been proposed over the years, including methods based on PGP, S/MIME, SIGMA, and TLS.

Most proposals have not been able to support multiple end-points for a given recipient. As more devices support XMPP, it becomes more desirable to allow an entity to communicate with another in a more secure manner, regardless of the number of agents the entity is employing. This document specifies an approach for encrypting communications between two entities which each might have multiple end-points.

2. Terminology

This document inherits XMPP-related terminology from [RFC6120], JSON Web Algorithms (JWA)-related terminology from [JOSE-JWA], JSON Web Encryption (JWE)-related terminology from [JOSE-JWE], and JSON Web Key (JWK)-related terminology from [JOSE-JWK]. Security-related terms are to be understood in the sense defined in [RFC4949].

The capitalized key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Determining Support

If an agent supports end-to-end object encryption, it MUST advertise that fact in its responses to [XEP-0030] information ("disco#info") requests by returning a feature of "urn:ietf:params:xml:ns:xmpp-e2e:3:encrypt".

```
<iq xmlns='jabber:client'
  id='disco1'
  to='romeo@montegue.lit/garden'
  type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3:encrypt' />
    ...
  </query>
```


</iq>

To help facilitate discovery, an agent SHOULD also include [XEP-0115] information in any directed or broadcast presence updates.

4. Encrypting XMPP Stanzas

The process that a sending agent follows for securing stanzas is the same regardless of the form of stanza (i.e., <iq/>, <message/>, or <presence/>).

4.1. Prerequisites

First, the sending agent prepares and retains the following:

- o The JID of the sender (i.e. its own JID). This SHOULD be the bare JID (localpart@domainpart).
- o The JID of the recipient. This SHOULD be the bare JID (localpart@domainpart).
- o A Session Master Key (SMK). The SMK MUST have a length at least equal to that required by the key wrapping algorithm in use and MUST be generated randomly. See [RFC4086] for considerations on generating random values.
- o A SMK identifier (SID). The SID MUST be unique for a given (sender, recipient, SMK) tuple, and MUST NOT be derived from SMK itself.

4.2. Process

For a given plaintext stanza (S), the sending agent performs the following:

1. Ensures the plaintext stanza is fully qualified, including the proper namespace declarations (e.g. contains the attribute 'xmlns' set to the value "jabber:client" for 'jabber:client' stanzas defined in [RFC6120]).
2. Notes the current UTC date and time N when this stanza is constructed, formatted as described under Section 7.
3. Constructs a forwarding envelope M using a <forwarded/> element qualified by the "urn:xmpp:forward:0" namespace (as defined in [XEP-0297]) as follows:

- * The child element <delay/> qualified by the "urn:xmpp:delay" namespace (as defined in [XEP-0203]) with the attribute 'stamp' set to the UTC date and time value N
 - * The plaintext stanza S
4. Convert the forwarding envelope M to a UTF-8 encoded string (M'), optionally removing line breaks and other insignificant whitespace between elements and attributes, i.e. M' = UTF8-encode(M). We call M' a "stanza-string" because for purposes of encryption and decryption it is treated not as XML but as an opaque string (this avoids the need for complex canonicalization of the XML input).
 5. Generates a Content Master Key (CMK). The CMK MUST have a length at least equal to that required by the content encryption algorithm in use and MUST be generated randomly. See [RFC4086] for considerations on generating random values.
 6. Generates any additional unprotected block cipher factors (IV); e.g. initialization vector/nonce. A sending agent MUST ensure that no two sets of factors are used with the same CMK, and SHOULD NOT reuse such factors for other stanzas.
 7. Performs the message encryption steps from [JOSE-JWE] to generate the JWE Header H, JWE Encrypted Key E, JWE Ciphertext C, and JWE Integrity Value I; using the following inputs:
 - * The 'alg' property is set to an appropriate key wrapping algorithm (e.g. "A256KW" or "A128KW"); recipients use 'keyreq' in Section 6 to obtain the SMK.
 - * The 'enc' property is set to the intended content encryption algorithm.
 - * SMK as the key for CMK Encryption.
 - * CMK as the Content Master Key.
 - * M' as the plaintext content to encrypt.
 8. Constructs an <e2e/> element qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:3" namespace as follows:
 - * The attribute 'id' set to the identifier value SID.

- * The child element <header/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:3" namespace and with XML character data as H, encoded base64url as per [RFC4684].
 - * The child element <cmk/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:3" namespace and with XML character as E, encoded base64url as per [RFC4684].
 - * The child element <data/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:3" namespace and with XML character data as C, encoded base64url as per [RFC4684].
 - * The child element <mac/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:3" namespace and with XML character data as I, encoded base64url as per [RFC4684].
9. Sends the <e2e/> element as the payload of a stanza that SHOULD match the stanza from step 1 in kind (e.g., <message/>), type (e.g., "chat"), and addressing (e.g. to="romeo@montague.net" from="juliet@capulet.net/balcony"). If the original stanza (S) has a value for the "id" attribute, this stanza MUST NOT use the same value for its "id" attribute.

4.3. Example - Securing a Message

NOTE: unless otherwise indicated, all line breaks are included for readability.

The sending agent begins with the plaintext version of the <message/> stanza 'S':

```
<message xmlns='jabber:client'
  from='juliet@capulet.lit/balcony'
  to='romeo@montague.lit'
  type='chat'>
  <thread>35740be5-b5a4-4c4e-962a-a03b14ed92f4</thread>
  <body>
    But to be frank, and give it thee again.
    And yet I wish but for the thing I have.
    My bounty is as boundless as the sea,
    My love as deep; the more I give to thee,
    The more I have, for both are infinite.
  </body>
</message>
```

and the following prerequisites:

- o Sender JID as "juliet@capulet.lit/balcony"
- o Recipient JID as "romeo@montegue.lit"
- o Session Master Key 'SMK' as (base64 encoded)
"xWtdjhYsH4Va_9SfYSeFsJfZu03m5RrbXo_UavxxeU8="
- o CMK identifier SID as "835c92a8-94cd-4e96-b3f3-b2e75a438f92"

The sending agent performs steps 1, 2, and 3 to generate the envelope:

```
<forwarded xmlns='urn:xmpp:forward:0'>
  <delay xmlns='urn:xmpp:delay'
    stamp='1492-05-12T20:07:37.012Z' />
  <message xmlns='jabber:client'
    from='juliet@capulet.lit/balcony'
    to='romeo@montegue.lit'
    type='chat'>
    <thread>35740be5-b5a4-4c4e-962a-a03b14ed92f4</thread>
    <body>
      But to be frank, and give it thee again.
      And yet I wish but for the thing I have.
      My bounty is as boundless as the sea,
      My love as deep; the more I give to thee,
      The more I have, for both are infinite.
    </body>
  </message>
</forwarded>
```

Then the sending agent performs steps 4 through 7 (with Content Master Key as "-ElMo6FndEkMxWP3TIkpldDfVKqmqAAgrlcVnUVpOc=", base64 encoded) to generate the [JOSE-JWE] outputs:

JWE Header

```
{
  "alg": "A256KW",
  "enc": "A256CCM",
  "iv": "B7waCj2vF_sLaJfe-1GHrA=="
}
```

JWE Content Encryption Key

XvySjpkqv6m-hUrG2VSEwIM9wZqVcdP037trOMpqFbK_i2AvHLqiNA==

JWE Ciphertext

7LlMXd-qQPAQ_LZm6u9AR2csyDgT09z5DWdn8K5GLr_qbWRDKw2ufZrm09YZ-jHl
lIDeXeQ9azbNNViv8gpa-prDYkXOo3QoqYOJiA0RAkPU-UjN4lwqqVvV62gad_OB
Dd9q2xsNnK1PI5frIGTCZSexOIeSD3EcP0cDI_0MzMEKqpVnPbYQDkWQNrtxPs2b
lE15KcQXbHVxA9rEz7y0a-ITXruV_fOXGftkRVDuiFlyVh2xNRPa-TQxDegZh1D_
u_c2mwPLO6ED_1ZlvL7075_VL0DT01YGtDDQeyzrQWnQNEBJ4G5jFpCyqtCszbgx
9kjWjxLYNLLGxbM0twF45OLCd8JFUIQAHoLeKp4aIW5yp7aATX8dKvQm5_TFICt
nLonaM1e5mRPnRgg5zNeMERx6FFkqowOI1h7hVl_QHF8Of0y99CmWKiVT3Nq5ngL
74Xm4CtiJMHnAEn2Q-10-fWuIHIEA0u4GwBaXo0ToBw4uCM4ZhG1SFSKyCKkjXmr
7TL-5jwSuuQHx6efS8Hhi7fujmqw9VXeKeubsY2bt15put0SWT8_0S8ZBDMjrKXj
y7iI5NUOhQMms3oulr01NZdzNVKcqiX5q3z1eB1FLWmGymnnj_gE_HalWUL0HoqL
93FrlnfFLNhXLCZYhZ7By6T9NN8omp4ZYE92HMPzgo-eCGP

NOTE: For CCM [RFC3610], the integrity value is encoded as part of the ciphertext, therefore the resulting Integrity Value is an octet string of length 0.

Then the sending agent performs steps 8 and 9, and sends the following:

```

<message xmlns='jabber:client'
  from='juliet@capulet.lit/balcony'
  id='fJZd9WFIIwNjFctT'
  to='romeo@montegue.lit'
  type='chat'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3'
    id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <header>
      eyJlbmMiOiJBMjU2Q0NNIiwiaXYiOiJCN3dhQ2oydkZfc0xhSmZlLTFHS
      HJBPT0iLCJhZGF0YSI6IjQ5Mi0wNS0xMlQyMDowNzozNy4wMTJaIiwibX
      NpemUiOiIxNiJ9
    </header>
    <cmk></cmk>
    <data>
      7LlMXd-qqPAQ_LZm6u9AR2csyDgT09z5DWdn8K5GLr_qbWRDKw2ufZrmO
      9YZ-jHl1IDeXeQ9azbNNViv8gpa-prDYkXOo3QoqYOJiA0RAkPU-UjN41
      wqqVvV62gad_OBDd9q2xsNnK1PI5frIGTCZSexOIeSD3EcP0cDI_0MzME
      KqpVnPbYQDkWQNrtxPs2blE15KcQXbHVxA9rEz7y0a-ITXruV_fOXGftk
      RVDuiFlyVh2xNRPa-TQxDegZh1D_u_c2mwPLO6ED_1ZlvL7075_VL0DT0
      1YGtDDQeyzrQWnQNEBJ4G5jFpCyqtCszbgx9kjWjxLYNLLGxbM0twF450
      LCd8JFUIQAHoLeKp4aIW5yp7aATX8dKvQm5_TFICtnLonaMle5mRPnRg
      g5zNeMERx6FFkqowOI1h7hVl_QHF8Ofoy99CmWKiVT3Nq5ngL74Xm4Cti
      JMHnAEn2Q-10-fWuIHIEA0u4GwBaXo0ToBw4uCM4ZhG1SFSKyCKkjXmr7
      TL-5jwSuuQHX6efS8Hhi7fujmqw9VXekeubsY2bt15put0SWT8_0S8ZBD
      MjrkXjy7iI5NUOhQMms3oulr01NZdzNVKcqiX5q3z1eB1FLWmGymnnj_g
      E_HalWUL0HoqL93FrlnfFLNhXLCZYhZ7By6T9NN8omp4ZYE92HMPpZgo-
      eCGP
    </data>
    <mac/>
  </e2e>
</message>

```

5. Decrypting XMPP Stanzas

5.1. Protocol Not Understood

If the receiving agent does not understand the protocol, it MUST do one and only one of the following: (1) ignore the <e2e/> extension, (2) ignore the entire stanza, or (3) return a <service-unavailable/> error to the sender, as described in [RFC6120].

NOTE: If the inbound stanza is an <iq/>, the receiving agent MUST return an error to the sending agent, to comply with the exchanging of IQ stanzas in [RFC6121].

5.2. Process

Upon receipt of an encrypted stanza, the receiving agent performs the following:

1. Determines if a valid SMK is available, associated with the SID specified by the 'id' attribute value of the <e2e/> element and the sending agent JID specified by the 'from' attribute of the wrapping stanza. If the receiving agent does not already have the CMK, it requests it according to Section 6.
2. Performs the message decryption steps from [JOSE-JWE] to generate the plaintext forwarding envelope string M', using the following inputs:
 - * The JWE Header H from the <header/> element's character data content.
 - * The JWE Content Encryption Key from the <cmk/> element's character data content.
 - * The JWE Ciphertext C from the <data/> element's character data content.
 - * The JWE Integrity Value I from the <mac/> element's character data content.
3. Converts the forwarding envelope UTF-8 encoded string M' into XML element M.
4. Obtains the UTC date and time N from the <delay/> child element, and verifies it is within the accepted range, as specified in Section 7.
5. Obtains the plaintext stanza S, which is a child element node of M; the stanza MUST be fully qualified with proper namespace declarations for XMPP stanzas, to help distinguish it from other content within M.

.

5.3. Insufficient Information

At step 1, if the receiving agent is unable to obtain the CMK, or the receiving agent could not otherwise determine the additional information, it MAY return a <bad-request/> error to the sending agent (as described in [RFC6120]), optionally supplemented by an application-specific error condition element of <insufficient-information/>:

```

<message xmlns='jabber:client'
  from='juliet@capulet.lit/balcony'
  id='fJZd9WFIIwNjFctT'
  to='romeo@montegue.lit/garden'
  type='chat'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3'
    id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <header>[XML character data]</header>
    <data>[XML character data]</header>
  </e2e>
  <error type='modify'>
    <bad-request
      xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' />
    <insufficient-information
      xmlns='urn:ietf:params:xml:ns:xmp-e2e:3' />
  </error>
</message>

```

In addition to returning an error, the receiving agent SHOULD NOT present the stanza to the intended recipient (human or application) and SHOULD provide some explicit alternate processing of the stanza (which MAY be to display a message informing the recipient that it has received a stanza that cannot be decrypted).

5.4. Failed Decryption

At step 2, if the receiving agent is unable to successfully decrypt the stanza, the receiving agent SHOULD return a <bad-request/> error to the sending agent (as described in [RFC6120]), optionally supplemented by an application-specific error condition element of <decryption-failed/> (previously defined in [RFC3923]):

```

<message xmlns='jabber:client'
  from='juliet@capulet.lit/balcony'
  id='fJZd9WFIIwNjFctT'
  to='romeo@montegue.lit/garden'
  type='chat'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3'
    id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <header>[XML character data]</header>
    <data>[XML character data]</header>
  </e2e>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' />
    <decryption-failed xmlns='urn:ietf:params:xml:ns:xmp-e2e:3' />
  </error>
</message>

```


In addition to returning an error, the receiving agent SHOULD NOT present the stanza to the intended recipient (human or application) and SHOULD provide some explicit alternate processing of the stanza (which MAY be to display a message informing the recipient that it has received a stanza that cannot be decrypted).

5.5. Timestamp Not Acceptable

At step 4, if the stanza is successfully decrypted but the timestamp fails the checks outlined in Section 7, the receiving agent MAY return a <not-acceptable/> error to the sender (as described in [RFC6120]), optionally supplemented by an application-specific error condition element of <bad-timestamp/> (previously defined in [RFC3923]):

```
<message xmlns='jabber:client'
  from='juliet@capulet.lit/balcony'
  id='fJZd9WFIIwNjFctT'
  to='romeo@montague.lit/garden'
  type='chat'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3'
    id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <header>[XML character data]</header>
    <data>[XML character data]</header>
  </e2e>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' />
    <bad-timestamp xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3' />
  </error>
</message>
```

5.6. Successful Decryption

If the receiving agent successfully decrypted the payload, it MUST NOT return a stanza error.

If the payload is an <iq/> of type "get" or "set", and the response to this <iq/> is of type "error", the receiving agent MUST send the encrypted response wrapped in an <iq/> of type "result", to prevent exposing information about the payload.

6. Requesting Session Keys

Because of the dynamic nature of XMPP stanza routing, the protocol does not exchange session keys as part of the encrypted stanza. Instead, a separate protocol is used by receiving agents to request a particular session key from the sending agent.

6.1. Request Process

Before a SMK can be requested, the receiving agent MUST have at least one public key for which it also has the private key.

To request a SMK, the receiving agent performs the following:

1. Constructs a [JOSE-JWK] JWK Set (KS), containing information about each public key the requesting agent wishes to use. Each key SHOULD include a value for the property 'kid' which uniquely identifies it within the context of all provided keys. Each key MUST include a value for the property 'kid' if any two keys use the same algorithm.
2. Constructs a <keyreq/> element qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:3" namespace as follows:
 - * The attribute 'id' set to the SMK identifier value SID.
 - * The child element <pkey/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:3" namespace and with XML character data as KS, encoded base64url as pre [RFC4684].
3. Sends the <keyreq/> element as the payload of an <iq/> stanza with the attribute 'type' set to "get", the attribute 'to' set to the full JID of the original encrypted stanza's sender, and the attribute 'id' set to an opaque string value the receiving agent uses to track the <iq/> response.

6.2. Accept Process

If the sending agent approves the request, it performs the following:

1. Chooses a key (PK) from the keys provided via KS, and notes its identifier value 'kid'.
2. Constructs a partial [JOSE-JWE] header (H) as follows:
 - * The property 'alg' set to the cryptographic algorithm for PK, which is used to secure the content master key SMK.
 - * The property 'kid' set to the identifier matching PK.
3. Encrypts the content master key SMK using the key PK, i.e. SMK' = pki-encrypt(PK, SMK).

4. Constructs a <keyreq/> element qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:3" namespace as follows:
 - * The attribute 'id' set to the SMK identifier SID.
 - * The child element <header/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:3" namespace and with XML character data as H, encoded base64url as per [RFC4684].
 - * The child element <cmk/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:3" namespace and with XML character data as SMK', encoded base64url as per [RFC4684].
5. Sends the <keyreq/> element as the payload of an <iq/> stanza with the attribute 'type' set to "result", the attribute 'to' set to the full JID from the request <iq/>'s 'from' attribute, and the attribute 'id' set to the value of the request <iq/>'s 'id' attribute.

6.3. Error Conditions

If the sending agent does not approve the request, it sends an <iq/> stanza of type "error" and containing the reason for denying the request:

- o <forbidden/>: the key request is made by an entity that is not authorized to decrypt stanzas from the sending agent and/or for the indicated SID.
- o <item-not-found/>: the requested SID is no longer valid.
- o <not-acceptable/>: the key request did not contain any keys the sending agent understands.

6.4. Example of Successful Key Request

NOTE: unless otherwise indicated, all line breaks are included for readability.

To begin a key request, the receiving agent performs step 1 from Section 6.1 to generate the [JOSE-JWK]:

```
{
  "keys": [{
    "alg": "RSA-OAEP",
    "mod": "vtqejkMF01h8oKEaHfHEY00C2jM7eISbbSvNs0SNItYW06GbjpJf
N4ldXw2vpVRdysnwU3zk6o2_SD0YCH1WgeuI0QK1knMTDdNSXx52e1c4BTw
h1A8iHuutTWmpBgesn1GNZmqB3jYsJOkVBYwCJtkB9APaBvk0itlRtizjCf
1HHnau7nGStyshgu8-srxi_d8rC5TTLSB_zT1i6fP8fwdloemXOtC0U65by
5P-1ZHXaf_bD8fpjps6gwSgdkZKMJAI0bOWZWuMpp2ntqa0wLB7Ndx2Ijr
eog_s5ssAoSiXDVdoswSbp36ZP-1lnCk2j-vZ4qbhaFg5bZtgt-gwQ==",
    "exp": "AQAB",
    "kid": "romeo@montegue.lit/garden"
  }]
}
```

Then the receiving agent performs step 2 to generate the <keyreq/>:

```
<keyreq xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3'
  id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
  <pkey>
    W3siYWxnIjoUlNBIIiwibW9kIjoIQUw3YW5vNURCZE5ZZktDaEdoM3h4R0R
    0QXRvek8zaUVtMjByemJORWpTTFdGanVobTQ2U1h6ZUpYVjhOc jZWVvhjck
    o4Rk44NU9xTnYwZzlHQWg5Vm9IcmlORUN0Wkp6RXczVFVsOGVkbnRYT0FVO
    ElaUVBJaDdyclUxcHFRYW5ySjlSaldacWdkNDJMQ1RwRlFXTUFpYlBZlFE
    MmdiNU5JclpVY1l1zNHduOVJ4NTJydTV4a3JjcklZTHZQcks4WXYzZkt3dVU
    weTBnZjgwOVl1bnpfSDhBNWFicGx6clF0Rk91Vzh1VF90V1I4V25fMndfSD
    ZZNmJPb01Fb0haRlNqQ1FDTkd6bG1WcmpLYWRwN2FtdE1Dd2V6WGNXOWlJN
    jNxsVA3T2JMQUtFb2x3MVhhTE1FbTZkLW1UX3RaWndwTm9fcjJlS200V2hZ
    T1cyY1lMZm9NRT0iLCJleHAiOiJBUEF0Iiwia2lkIjoicm9tZW9AbW9udGV
    ndWUubGl0L2dhcmRlbiJ9XQ==
  </pkey>
</keyreq>
```

Then the receiving agent performs step 3 and sends the following:

```

<iq xmlns='jabber:client'
  from='romeo@montegue.lit/garden'
  id='xdJbWMA+'
  to='juliet@capulet.lit/balcony'
  type='get'>
  <keyreq xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3'
    id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <pkey>
      W3siYWxnIjoiUlNBIIwibW9kIjoiQUw3YW5vNURCZE5ZZktDaEdoM3h4R
      0R0QXRvek8zaUVtMjByemJORWpTTFdGanVobTQ2U1h6ZUpYVjhOc jZWV
      hjcko4Rk44NU9xTnYwZzlHQWg5Vm9IcmlORUN0Wkp6RXczVFVsOGVkb nR
      YT0FVOElauVBjaDdyclUxcHFRYW5ySjlSaldacWdkNDJMq1RwRlFXtUFp
      YlpBZlFEMmdiNU5JclpVYllzNHduOVJ4NTJydTV4a3JjcklZTHZQcks4W
      XYzZkt3dVUweTBnZjgwOVllbnpfSDhBNWFicGx6clF0Rk91Vzh1VF90Vl
      I4V25fMndfSDZZNmJPb01Fb0haRlNqQ1FDTkd6bG1WcmpLYWRwN2FtdE1
      Dd2V6WGNXOW1JNjNjSVA3T2JMQUtFb2x3MVhhTE1FbTZkLW1UX3RaWndw
      Tm9fcjJlS200V2hZTlcyYllmZm9NRT0iLCJleHAiOiJBUUFClIiwia2lkI
      joicm9tZW9AbW9udGVndWUubGl0L2dhcmRlbiJ9XQ==
    </pkey>
  </keyreq>
</iq>

```

If the sending agent accepts this key request, it performs steps 1 and 2 from Section 6.2 to generate the partial [JOSE-JWE] header:

```

{
  "alg": "RSA-OAEP",
  "kid": "romeo@montegue.lit/garden"
}

```

Then the sending agent performs step 3 to generate the encrypted SMK:

```

DsLfaD3ZPtE4klwnhshNsYyFi3spey8NNNMeYQ6L-sdLJrcWIwfOYNPK-Sb
oLvsSiXB7yw1MLtKhhrAiRovEmMnTgoLvcuzX8PIBcDbSP3ie0aMynXqStb
Rk-Lwi jdsY7NqD_WBltP4dhqo3OwlXf4bSWlBnUDEzGh8giF6KYeRzw181m
vMIJ6BCjwnvqTKBNlMDUX3cTfXSAFi9j8MkS3wLbc2MM4RzP4ESyiZZQJn
z_zkglenjAbJ_fpwJ-E4a14EyWhwYpCNW3Tb5IhyhhgBsiLB6kulm9RMYwc
TCGIDCUsfTo5vzDOKX7DBF6KtwSY8bF1AiOiGlsGo2SlrSg==

```

Then the sending agent performs step 4 to generate the <keyreq/> response:

```
<keyreq xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3'
  id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
  <header>
    eyJhbGciOiJSU0EtT0VBUCIsImtpZCI6InJvbWVvQG1vbnRlZ3VlLmxpc3QvZ2FyZGVuIn0=
  </header>
  <smk>
    DsLfAD3ZPtE4klwnhsHNsYyFi3spey8NNNMeYQ6L-sdLJrcWIwfOYNPK-Sb
    oLvsSiXB7yw1MLtKhhrAiRovEmMnTgoLvcuzX8PIBcDbSP3ie0aMynXqStb
    Rk-LwiJdSY7NqD_WBltP4dhqo3OwlXf4bSWlBnUDEzGh8giF6KYeRzw181m
    vMIJ6BCjwnvqTKBNlMDUX3cTfXSAFi9j8Mkks3wLbc2MM4RzP4ESyizzQJn
    z_zkglenjAbJ_fpwJ-E4a14EyWhwYpCNW3Tb5IhyhhgBsiLB6kulm9RMYwc
    TCGIDCUsfTo5vzDOKX7DBF6KtwSY8bF1AiOiGlsGo2SlrSg==
  </smk>
</keyreq>
```

Then the sending agent performs step 5 and sends the following:

```
<iq xmlns='jabber:client'
  from='juliet@capulet.lit/balcony'
  id='xdJbWMA+'
  to='romeo@montague.lit/garden'
  type='result'>
  <keyreq xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3'
    id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <header>
      eyJhbGciOiJSU0EtT0VBUCIsImtpZCI6InJvbWVvQG1vbnRlZ3VlLmxpc3QvZ2FyZGVuIn0=
    </header>
    <smk>
      DsLfAD3ZPtE4klwnhsHNsYyFi3spey8NNNMeYQ6L-sdLJrcWIwfOYNPK-
      SboLvsSiXB7yw1MLtKhhrAiRovEmMnTgoLvcuzX8PIBcDbSP3ie0aMynX
      qStbRk-LwiJdSY7NqD_WBltP4dhqo3OwlXf4bSWlBnUDEzGh8giF6KYeR
      zw181mvMIJ6BCjwnvqTKBNlMDUX3cTfXSAFi9j8Mkks3wLbc2MM4RzP4E
      SyizzQJnz_zkglenjAbJ_fpwJ-E4a14EyWhwYpCNW3Tb5IhyhhgBsiLB6
      kulm9RMYwcTCGIDCUsfTo5vzDOKX7DBF6KtwSY8bF1AiOiGlsGo2SlrSg
      ==
    </smk>
  </keyreq>
</iq>
```

7. Inclusion and Checking of Timestamps

Timestamps are included to help prevent replay attacks. All timestamps MUST conform to [XEP-0082] and be presented as UTC with no offset, and SHOULD include the seconds and fractions of a second to three digits. Absent a local adjustment to the sending agent's

perceived time or the underlying clock time, the sending agent MUST ensure that the timestamps it sends to the receiver increase monotonically (if necessary by incrementing the seconds fraction in the timestamp if the clock returns the same time for multiple requests). The following rules apply to the receiving agent:

- o It MUST verify that the timestamp received is within five minutes of the current time, except as described below for offline messages.
- o It SHOULD verify that the timestamp received is greater than any timestamp received in the last 10 minutes which passed the previous check.
- o If any of the foregoing checks fails, the timestamp SHOULD be presented to the receiving entity (human or application) marked as "old timestamp", "future timestamp", or "decreasing timestamp", and the receiving entity MAY return a stanza error to the sender.

The foregoing timestamp checks assume that the recipient is online when the message is received. However, if the recipient is offline then the server might store the message for delivery when the recipient is next online (offline storage does not apply to <iq/> or <presence/> stanzas, only <message/> stanzas). As described in [XEP-0160], when sending an offline message to the recipient, the server SHOULD include delayed delivery data as specified in [XEP-0203] so that the recipient knows that this is an offline message and also knows the original time of receipt at the server. In this case, the recipient SHOULD verify that the timestamp received in the encrypted message is within five minutes of the time stamped by the recipient's server in the <delay/> element.

8. Interaction with Stanza Semantics

The following limitations and caveats apply:

- o Undirected <presence/> stanzas SHOULD NOT be encrypted. Such stanzas are delivered to anyone the sender has authorized, and can generate a large volume of key requests.
- o Stanzas directed to multiplexing services (e.g. multi-user chat) SHOULD NOT be encrypted, unless the sender has established an acceptable trust relationship with the multiplexing service.

9. Mandatory-to-Implement Cryptographic Algorithms

All algorithms that MUST be implemented for [JOSE-JWE] also MUST be implemented for this specification.

10. Security Considerations

10.1. Storage of Encrypted Stanzas

The recipient's server might store any <message/> stanzas received until the recipient is next available; this duration could be anywhere from a few minutes to several months.

10.2. Re-use of Session Master Keys

A sender SHOULD NOT use the same SMK for stanzas intended for different recipients, as determined by the localpart and domainpart of the recipient's JID.

A sender MAY re-use a SMK for several stanzas to the same recipient. In this case, the SID remains the same, but the sending agent MUST generate a new CMK and IV for each encrypted stanza. The sender SHOULD periodically generate a new SMK; however, this specification does not mandate any specific algorithms or processes.

In the case of <message/> stanzas, a sending agent might generate a new SMK each time it generates a new ThreadID, as outlined in [XEP-0201].

11. IANA Considerations

11.1. XML Namespace Name for e2e Data in XMPP

A URN sub-namespace of encrypted content for the Extensible Messaging and Presence Protocol (XMPP) is defined as follows.

URI: urn:ietf:params:xml:ns:xmpp-e2e:3

Specification: RFC XXXX

Description: This is an XML namespace name of encrypted content for the Extensible Messaging and Presence Protocol as defined by RFC XXXX.

Registrant Contact: IESG, <iesg@ietf.org>

12. References

12.1. Normative References

- [E2E-REQ] Saint-Andre, P., "Requirements for End-to-End Encryption in the Extensible Messaging and Presence Protocol (XMPP)", draft-saintandre-xmpp-e2e-requirements-01 (work in progress), March 2010.
- [JOSE-JWA] Jones, M., "JSON Web Algorithms (JWA)", draft-ietf-jose-json-web-algorithms-03 (work in progress), July 2012.
- [JOSE-JWE] Jones, M., Rescola, E., and J. Hildebrand, "JSON Web Encryption (JWE)", draft-ietf-jose-json-web-encryption-03 (work in progress), July 2012.
- [JOSE-JWK] Jones, M., "JSON Web Key (JWK)", draft-ietf-jose-json-web-key-03 (work in progress), July 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4684] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.
- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.
- [RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, March 2011.
- [XEP-0030] Eatmon, R., Hildebrand, J., Millard, P., and P. Saint-Andre, "Service Discovery", XSF XEP 0030, June 2006.
- [XEP-0082] Saint-Andre, P., "XMPP Date and Time Profiles", XSF XEP 0082, May 2003.
- [XEP-0115] Hildebrand, J., Troncon, R., and P. Saint-Andre, "Entity

Capabilities", XSF XEP 0115, February 2008.

[XEP-0203]

Saint-Andre, P., "Delayed Delivery", XSF XEP 0203, September 2009.

[XEP-0297]

Wild, M. and K. Smith, "Stanza Forwarding", XSF XEP 0297, July 2012.

12.2. Informative References

[RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3923, September 2003.

[RFC3923] Saint-Andre, P., "End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)", RFC 3923, October 2004.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", RFC 4086, June 2005.

[XEP-0160]

Saint-Andre, P., "Best Practices for Handling Offline Messages", XSF XEP 0160, January 2006.

[XEP-0201]

Saint-Andre, P., Paterson, I., and K. Smith, "Best Practices for Message Threads", XSF XEP 0203, November 2010.

Appendix A. Schema for urn:ietf:params:xml:ns:xmpp-e2e:3

The following XML schema is descriptive, not normative.

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

```
<xs:schema
```

```
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='urn:ietf:params:xml:ns:xmpp-e2e:3'
  xmlns='urn:ietf:params:xml:ns:xmpp-e2e:3'
  elementFormDefault='qualified'>
```

```
  <xs:element name='e2e'>
```

```
    <xs:complexType>
```

```
      <xs:attribute name='id' type='xs:string' use='required' />
```

```
      <xs:sequence>
```

```
        <xs:element ref='header' minOccurs='1' maxOccurs='1' />
        <xs:element ref='cmk' minOccurs='1' maxOccurs='1' />
        <xs:element ref='data' minOccurs='1' maxOccurs='1' />
        <xs:element ref='mac' minOccurs='1' maxOccurs='1' />
    </xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name='keyreq'>
  <xs:complexType>
    <xs:attribute name='id' type='xs:string' use='required' />
    <xs:sequence>
      <xs:element ref='header' minOccurs='0' maxOccurs='1' />
      <xs:element ref='pkey' minOccurs='0' maxOccurs='1' />
      <xs:element ref='smk' minOccurs='0' maxOccurs='1' />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name='cmk'>
  <xs:complexType>
    <xs:simpleType>
      <xs:extension base='xs:string'>
      </xs:extension>
    </xs:simpleType>
  </xs:complexType>
</xs:element>

<xs:element name='data'>
  <xs:complexType>
    <xs:simpleType>
      <xs:extension base='xs:string'>
      </xs:extension>
    </xs:simpleType>
  </xs:complexType>
</xs:element>

<xs:element name='header'>
  <xs:complexType>
    <xs:simpleType>
      <xs:extension base='xs:string'>
      </xs:extension>
    </xs:simpleType>
  </xs:complexType>
</xs:element>

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

```
    <xs:simpleType>
      <xs:extension base='xs:string'>
        </xs:extension>
      </xs:simpleType>
    </xs:complexType>
  </xs:element>

  <xs:element name='pkey'>
    <xs:complexType>
      <xs:simpleType>
        <xs:extension base='xs:string'>
          </xs:extension>
        </xs:simpleType>
      </xs:complexType>
    </xs:element>

  <xs:element name='smk'>
    <xs:complexType>
      <xs:simpleType>
        <xs:extension base='xs:string'>
          </xs:extension>
        </xs:simpleType>
      </xs:complexType>
    </xs:element>

  <xs:element name='bad-timestamp' type='empty' />
  <xs:element name='decryption-failed' type='empty' />
  <xs:element name='insufficient-information' type='empty' />

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

Author's Address

Matthew Miller
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3204
Email: mamille2@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2013

M. Miller
P. Saint-Andre
Cisco Systems, Inc.
July 13, 2012

Using PKIX over Secure HTTP (POSH) as a Proofotype for XMPP Domain Name
Associations
draft-miller-xmpp-posh-proofotype-01

Abstract

This document defines a proofotype involving PKIX over Secure HTTP (POSH) for associating a domain name with an XML stream in the Extensible Messaging and Presence Protocol (XMPP). It also defines a method involving HTTPS redirects (appropriate for use with the POSH proofotype) for securely delegating a source domain to a derived domain in XMPP.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on January 14, 2013.

Copyright Notice

Copyright (c) 2012 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Proofotype	4
4. Secure Delegation	5
5. Caching Results	6
6. Examples	6
7. Security Considerations	6
8. IANA Considerations	7
8.1. The "posh._xmpp-client._tcp.cer" Well-Known URI	7
8.2. The "posh._xmpp-server._tcp.cer" Well-Known URI	7
9. References	7
9.1. Normative References	7
9.2. Informative References	8
Authors' Addresses	8

1. Introduction

The [XMPP-DNA] specification defines a framework for secure delegation and authenticated domain name associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP). This document defines a proofotype for DNA, using PKIX certificates obtained over secure HTTP ("POSH"), as well as a secure delegation method, based on HTTPS redirects, that is appropriate for use with the POSH proofotype.

The rationale for POSH is driven by current operational realities. It is effectively impossible for a hosting service to provide and maintain PKIX certificates [RFC5280] that include the appropriate [RFC6125] identifiers for each hosted domain. It is true that DNS-based technologies are emerging for secure delegation, in the form of DNS Security [RFC4033] and [DANE]); however, these technologies are not yet widely deployed and might not be deployed in the near future for domains outside the most common top-level domains (e.g., ".COM", ".NET", ".EDU"). Because the XMPP community wishes to deploy secure delegation and authenticated domain name associations as widely and as quickly as possible, this document specifies how to use secure HTTP [RFC2616] and PKIX certificates [RFC5280] to verify that a domain is delegated to a hosting provider and authenticate an association between a domain name and an XML stream.

2. Terminology

This document inherits XMPP-related terminology from [RFC6120] and security-related terminology from [RFC5280]. The terms "source domain", "derived domain", "reference identifier", and "presented identifier" are used as defined in the "CertID" specification [RFC6125].

This document is applicable to connections made from an XMPP client to an XMPP server ("_xmpp-client._tcp") or between XMPP servers ("_xmpp-server._tcp"). In both cases, the XMPP initiating entity acts as a TLS client and the XMPP receiving entity acts as a TLS server. Therefore, to simplify discussion this document uses "_xmpp-client._tcp" to describe both cases, unless otherwise indicated.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Proofotype

POSH stands for PKIX Over Secure HTTP: the verification materials consist of a PKIX certificate [RFC5280], they are obtained by retrieving the certificate over HTTPS [RFC2818] from a well-known URI [RFC5785], the certificate is checked according to the rules from [RFC6120] and [RFC6125], and secure DNS is not necessary since the HTTPS retrieval mechanism relies on the chain of trust based on the public key infrastructure.

The process for retrieving a PKIX certificate over secure HTTP is as follows.

1. The initiating entity performs an HTTPS GET at the source domain to the path `"/.well-known/posh._<service>._tcp.cer"`; where `"_<service>"` MUST be either `"_xmpp-client"` for XMPP client-to-server connections or `"_xmpp-server"` for XMPP server-to-server connections:

```
HTTP GET /.well-known/posh._xmpp-server._tcp.cer HTTP/1.1
Host: im.example.com
```

2. If the source domain HTTPS server has a certificate for the requested path, it MUST respond with a success status code, with the message body as the DER certificate (optionally encoded as base64 [RFC4684]) that the XMPP server at the source domain will present during the TLS negotiation phase of XMPP stream setup:

```
HTTP/1.1 200 OK
Content-Type: application/pkix-cert
Content-Length: 839
```

-----BEGIN CERTIFICATE-----

```
MIICPTCCAaYCCQDDVeBaBmWC/jANBgkqhkiG9w0BAQUFADBjMQswCQYDVQQGEwJV
UzERMA8GA1UECBMIQ29sb3JhZG8xDzANBgNVBACTBKRlbnZlcjEXMBUGA1UEChMO
aW0uZmV4b3R5b3R5b3R5b3R5b3R5b3R5b3R5b3R5b3R5b3R5b3R5b3R5b3R5b3R5
MTIxNTQ0NFoXDTIyMDYwOTIxNTQ0NFowYzELMAkGA1UEBhMCVVMxETAPBgNVBAGT
CENvbG9yYWRvMQ8wDQYDVQQHEwZlZW52ZXIxFzAVBgNVBAoTDmV4YW1wY29tMB4X
DTIyMDYwOTIxNTQ0NFoXDTIyMDYwOTIxNTQ0NFowYzELMAkGA1UEBhMCVVMxETAP
BgNVBAGTjQAwgYkCgYEA4hoKHi/B07eQH+1NB9gWiNFDT//AbTHQOEC0AOr4Gh/o9
PUP7kD0gklU4uv7rSAhAyCe4WaOiQ/HSzEryGfHiZmWht0BaYmj19iuPWRecZOXWq
KZji9NtAxn9l3kdon/YLJcrPGyNTGK66+ggNaqy8LkQQpI4rff60yHHZ/0XkCAwEA
ATANBgkqhkiG9w0BAQUFAAOBgQDcwu30bSMlykWyZ+tTDSlQ3wLSVB9RsR8jXmJv
Mo7y7icXwg54a9M3xipjZtrfAhYM5I5iqUTQPki6s26n9SQpRm5boneFDA3WGwrw
ma35biP9+NSBWzSaDF8AztwFNKXXl6/U6hWwG05G/NdeS11gpww9NUDraJgVoDpRK
04tg==
```

-----END CERTIFICATE-----

4. Secure Delegation

When PKIX Over Secure HTTP (POSH) is the DNA proofotype, it is possible to use HTTPS redirects in determining if a domain is securely delegated, as follows:

1. The initiating entity performs an HTTPS GET at the source domain to the path `"/.well-known/posh._<service>._tcp.cer"`; where `"_<service>"` MUST be either `"_xmpp-client"` for XMPP client-to-server connections or `"_xmpp-server"` for XMPP server-to-server connections. Here is an example:

```
GET /.well-known/posh._xmpp-server._tcp.cer HTTP/1.1
Host: im.example.com
```

2. If the source domain HTTPS server has delegated to a derived domain, it MUST respond with one of the redirect mechanisms provided by HTTP (e.g., using the 302, 303, or 307 response). The 'Location' header MUST specify an HTTPS URL, where the hostname and port is the derived domain HTTPS server, and the path MUST match the pattern `"_<service>._tcp.cer"`; where `"_<service>"` MUST be identical to the `"_<service>"` portion of the original request (line breaks added for readability):

```
HTTP/1.1 302 Found
Location: https://hosting.example.net/.well-known
        /posh._xmpp-server._tcp.cer
```

3. The initiating entity performs an HTTPS GET to the URL specified in the 'Location' header:

```
GET /.well-known/posh._xmpp-server._tcp.cer HTTP/1.1
Host: hosting.example.net
```

4. If the derived domain HTTPS server has a certificate, it MUST respond with a success status code, with the message body as the DER certificate (optionally encoded as base64 [RFC4684]) that the XMPP server at the derived domain will present during the TLS negotiation phase of XMPP stream setup:

HTTP/1.1 200 OK
Content-Type: application/pkix-cert
Content-Length: 863

-----BEGIN CERTIFICATE-----
MIICUTCCABoCCQCtNQRNu3194zANBgqhkiG9w0BAQUFADBtMQswCQYDVQQGEwJV
UzERMA8GA1UECBMIQ29sb3JhZG8xDzANBgNVBACtBkRlbnZlcjEcmBoGA1UEChMT
aG9zdGluZy5leGFtcGxlLm5ldDECMBoGA1UEAxMTaG9zdGluZy5leGFtcGxlLm5l
dDAeFw0xMjA2MTEyMTQ1MjZaFw0yMjA2MDkyMTQ1MjZaMG0xCzAJBgNVBAYTA1VT
MREwDwYDVQQIEWhDb2xvcmFkbzEPMA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKEExNo
b3N0aW5nLmV4YW1wbGUubmV0MRwwGgYDVQQDEExNob3N0aW5nLmV4YW1wbGUubmV0
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDi46kMWnCfsg0DTrlcTc6AQUCi5
Lulf2RKRWPehz8qyt/CO0N5VpxKQMLGp6TApQzFdAfxCUA3rniYFpMq4Hemw2S74
v1LROwvROKviKRzunDP3EhPXf6GbgNHRlfbX4yvZtcR1BMnkxgJtbTAJu4/wTRXY
RE5FKk3xT4IBXTIQFwIDAQABMA0GCSqGSIb3DQEBAQUAA4GBAAvRohCXsfSnHXjv
84beqmFSYKcZvhVymgxQfhB2ZLNfQvfTO3Qsp/MR0hRRXrJ25n86t49EEXicjC0r
EdmWaIhdDFhw7hva2byYziww7fJuelD0tpL9nfF5uOIMp3JYyXCbn/FKJhi9HMR1
d8avm8gJ5Iu7L96qosWzL3epHYW7
-----END CERTIFICATE-----

5. Caching Results

Ideally, the initiating entity relies on the expiration time of the certificate obtained via POSH, and not on HTTP caching mechanisms. To that end, the HTTPS servers for source and derived domains SHOULD specify a 'Cache-Control' header indicating a short duration (e.g., max-age=60) or "no-cache" to indicate the response (redirect or content) is not appropriate to cache at the HTTP level.

6. Examples

Detailed examples will be provided in a future version of this specification.

7. Security Considerations

This document supplements but does not supersede the security considerations provided in [RFC2616], [RFC2818], [RFC6120], and [RFC6125].

Specifically, communication via HTTPS depends on checking the identity of the HTTP server in accordance with [RFC2818].

8. IANA Considerations

8.1. The "posh._xmpp-client._tcp.cer" Well-Known URI

This specification registers the "posh._xmpp-client._tcp.cer" well-known URI in the Well-Known URI Registry as defined by [RFC5785].

URI suffix: posh._xmpp-client._tcp.cer

Change controller: IETF

Specification document(s): RFCXXXX.

8.2. The "posh._xmpp-server._tcp.cer" Well-Known URI

This specification registers the "posh._xmpp-server._tcp.cer" well-known URI in the Well-Known URI Registry as defined by [RFC5785].

URI suffix: posh._xmpp-server._tcp.cer

Change controller: IETF

Specification document(s): RFCXXXX.

9. References

9.1. Normative References

- [XMPP-DNA] Saint-Andre, P. and M. Miller, "Domain Name Associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP)", draft-saintandre-xmpp-dna-00 (work in progress), June 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [RFC4684] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,

Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

9.2. Informative References

[DANE] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", draft-ietf-dane-protocol-23 (work in progress), June 2012.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, May 2005.

Authors' Addresses

Matthew Miller
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Email: mamille2@cisco.com

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Email: psaintan@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2012

P. Saint-Andre
M. Miller
Cisco Systems, Inc.
June 27, 2012

Domain Name Associations (DNA) in the Extensible Messaging and Presence
Protocol (XMPP)
draft-saintandre-xmpp-dna-00

Abstract

This document defines a framework for improving the security of the Extensible Messaging and Presence Protocol (XMPP) in two respects. First, it introduces the concept of a proofotype for establishing a strong association between a domain name and an XML stream. Second, it provides guidelines for securely delegating a source domain to a derived domain, which is especially important in virtual hosting environments.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on December 29, 2012.

Copyright Notice

Copyright (c) 2012 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Problem Statement	4
4. Framework	5
5. Prooftypes	7
5.1. PKI	7
5.2. DANE	7
5.3. POSH	7
5.4. Dialback Keys	7
6. Assertion Mechanisms	8
6.1. TLS	8
6.2. SASL	8
6.3. <db:result>	8
6.4. A Note about Stream Attributes	8
7. Delegation Methods	8
8. Security Considerations	9
9. IANA Considerations	9
10. References	9
10.1. Normative References	9
10.2. Informative References	10
Authors' Addresses	10

1. Introduction

This document defines a framework for improving the security of the Extensible Messaging and Presence Protocol (XMPP) in two respects. First, it introduces the concept of a prooftype for establishing a strong association between a domain name and an XML stream (i.e., a domain name association or "DNA"). Second, it provides guidelines for securely delegating a source domain to a derived domain, which is especially important in virtual hosting environments.

The need to establish a strong association between a domain name and an XML stream arises in both client-to-server and server-to-server communication using XMPP, because XMPP servers are typically identified by domain names. However, a client or peer server needs to verify the identity of a server to which it connects. To date, such verification has been established based on information obtained from the Domain Name System (DNS), the Public Key Infrastructure (PKI), or similar sources. This document generalizes the model currently in use so that additional prooftypes can be defined, and also provides a basis for modernizing some prooftypes (e.g., Server Dialback [XEP-0220]) to reflect progress in several underlying technologies, especially DNS Security [RFC4033].

The process for resolving the domain name of an XMPP service into the IP address at which an XML stream will be negotiated (defined in [RFC6120]) can involve delegation of a source domain (say, `im.example.com`) to a derived domain (say, `hosting.example.net`). If such delegation is not done in a secure manner, then the domain name association cannot be authenticated. Therefore, this document also provides guidelines for defining secure delegation methods.

This document does not define any DNA prooftypes or secure delegation methods; such technologies are defined in companion documents.

2. Terminology

This document inherits XMPP-related terminology from [RFC6120] and [XEP-0220], DNS-related terminology from [RFC1034], [RFC1035], [RFC2782] and [RFC4033], and security-related terminology from [RFC4949] and [RFC5280]. The terms "source domain", "derived domain", "reference identity", and "presented identity" are used as defined in the "CertID" specification [RFC6125]. The terms "permissive federation", "verified federation", and "encrypted federation" are derived from [XEP-0238], although we substitute the term "authenticated federation" for the term "trusted federation" from that document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Problem Statement

In XMPP, each party to a stream expects the other party to provide some proof of its identity. For example, in client-to-server streams the server expects the client to present some credentials (such as a username and password or a client certificate), and ideally the client also expects the server to provide a certificate that identifies the domain(s) of the server. Similar considerations hold true for server-to-server streams, also called "interdomain federation".

When the Jabber.org open-source community developed the precursor to XMPP in 1999, it defined methods for interdomain federation but no mechanisms for authenticating or checking the identity of peer servers. We could describe this as "permissive federation", which is clearly sub-optimal given the strong potential for domain spoofing. In the year 2000, the community filled the gap to some extent by defining a technology called Server Dialback (first documented in [RFC3920] and since moved to [XEP-0220]). Although Server Dialback does not provide a strong mechanism for identity checking without the use of DNSSEC, it does provide DNS-based verification and thus has effectively prevented most instances of domain spoofing on the XMPP network since late 2000. Also, because Server Dialback typically does not involve the use of server certificates, it does not result in an encrypted stream; thus we refer to it as a technology for "verified federation".

In 2002-2004, the IETF's XMPP Working Group hardened the original Jabber.org protocols by adding Transport Layer Security (TLS) and Simple Authentication and Security Layer (SASL), thus making it possible for two servers to engage in "authenticated federation" (i.e., when two peer servers present PKIX certificates anchored to trusted roots during negotiation of a server-to-server stream) or "encrypted federation" (i.e., when two peer servers present PKIX certificates that are self-signed or not anchored to trusted roots during negotiation of a server-to-server stream).

Unfortunately, authenticated federation has not been widely deployed on the XMPP network (indeed, even encrypted federation is not widely deployed because verified federation is perceived as "good enough"); one of the primary reasons is that it is feasible (although not always easy) for single-domain servers to obtain the proper

certificates, but much more difficult (or practically impossible) for large XMPP hosting providers to do so. The primary challenge here is operational: it is highly unlikely that an organization (say, example.com) wishing to delegate its XMPP service (say, im.example.com) to a hosting provider (say, hosting.example.net) will hand over its private key to the hosting provider. Even if that were feasible, further operational challenges (e.g., maintaining large numbers of certificates for hosted domains, and configuring XMPP software to present the correct certificate based on the 'to' address of the initial stream header) have also discouraged deployment of authenticated federation in virtual hosting environments, which happen to be a common deployment scenario.

Furthermore, the prevalence of delegation to hosting providers leads to one additional shortcoming, caused by the use of DNS SRV records [RFC2782] in XMPP: if DNSSEC is not used, the act of delegation is inherently insecure. Unfortunately, no existing documentation explains how to use DNSSEC for secure delegation, with the result that clients and servers often take a "leap of faith" if using an SRV record to determine that when communicating with, say, im.example.com they actually need to connect to, say, hosting.example.net.

In order to meet the requirements for strong security [RFC3365], both authenticated federation and secure delegation are needed so that the association between a domain name and an XML stream can be trusted by XMPP entities. Unfortunately, authenticated federation is uncommon and secure delegation is unheard of on the XMPP network today. Because the current situation is clearly sub-optimal, this document defines a framework for both authenticated federation and secure delegation in XMPP.

4. Framework

In essence, we need to establish an association between a domain and an XML stream: is the XMPP server to which a client or peer server connects "allowed" to accept stanzas for or send stanzas from a given domain? If so, we say that there is a domain name association ("DNA") for the stream.

For TLS in general, the TLS client has some expectations about the identity of the TLS server (in the language of the "CertID" specification [RFC6125], the TLS client has a "reference identity"), and then checks some material presented by the TLS server (the "presented identity" within the server certificate) to verify that its expectations have been met. In XMPP, Server Dialback follows a similar model, except that the verification material takes the form of a token instead of a certificate. The DNS-Based Authentication of

Named Entities protocol [DANE], at least in some of its modes, adds another kind of verification material: not the presented identity within a PKIX certificate, but a complete certificate or hash thereof. And other kinds of verification material could be envisioned (e.g., OpenPGP keys, Kerberos tickets, OAuth tokens), although they are not considered here.

No matter what kind of verification material is used, an XMPP client or peer server that wishes to verify a domain name association needs a way to obtain the verification material it will refer to when establishing the association. For instance, when a server presents a PKIX certificate during TLS negotiation, the connecting client or peer server has traditionally obtained its verification material out of band or via configuration from a certification authority (i.e., in the form of a root certificate contained in a certificate bundle). In the Server Dialback protocol, the verification material is a token that is obtained over XMPP itself. In DANE, the verification material is obtained from the Domain Name System. In the PKIX Over Secure HTTP ("POSH") method described in an accompanying specification, the verification material is obtained over secure HTTP. And other methods for obtaining verification material could be envisioned (e.g., IPsec), although they are not considered here.

Furthermore, the matching rules for checking the verification material will depend on the nature of that material; for example, [RFC6120] defines a profile of the rules from the "CertID" specification [RFC6125], Server Dialback [XEP-0220] typically performs a character-for-character comparison of tokens, DANE might compare the SubjectPublicKeyInfo data or the full certificate, and so on.

Finally, given the relationship between XMPP and the DNS (XMPP services are usually identified by domain name, not IP address), it is important to make it clear whether a given verification method can (or must) be used only with secure DNS or also with insecure DNS.

Putting these pieces together, we define a "DNA prooftype" as follows.

prooftype: A mechanism for proving an association between a domain name and an XML stream, where the mechanism defines (1) the verification material to be used, (2) the matching rules for comparing the reference version and presented version of the material, (3) how the verification material is obtained, and (4) whether the mechanism depends on secure DNS.

The following sections outline several prooftypes that are used, or could be used, in XMPP; detailed definitions are provided in separate

specifications.

Note: So far, our definition of a prooftype does not include the exact protocol mechanism that is used to assert a domain name; this is explained further under Section 6.

5. Prooftypes

5.1. PKI

The PKI prooftype is a DNA proof that follows the rules from [RFC6120]: that is, the verification materials consist of a PKIX certificate that is checked according to a profile of the matching rules from [RFC6125], the client's verification materials are obtained out of band in the form of a trusted root, and secure DNS is not necessary.

5.2. DANE

In the DANE prooftype, the verification materials consist of a PKIX certificate that is compared as an exact match or a hash of either the SubjectPublicKeyInfo or the full certificate, and the verification materials are obtained via secure DNS. See the accompanying [XMPP-DANE] spec for complete discussion and examples.

5.3. POSH

POSH stands for PKIX Over Secure HTTP: the verification materials consist of a PKIX certificate, it is obtained by retrieving it over HTTPS at a well-known URI [RFC5785], the certificate is checked according to the rules from [RFC6120] and [RFC6125], and secure DNS is not necessary since the HTTPS retrieval mechanism relies on the chain of trust from the public key infrastructure. See the accompanying [XMPP-POSH] spec for complete discussion and examples.

5.4. Dialback Keys

The Dialback Keys prooftype formalizes the existing Server Dialback protocol: the verification materials consist of a token obtained over XMPP, the token is checked by the authoritative server for a given domain using implementation-specific methods such as character-by-character comparison, and secure DNS is needed in order to place significant trust in such tokens, although it is known that at the time of this writing many domains use Dialback Keys even in the absence of secure DNS.

6. Assertion Mechanisms

An assertion is a server's statement that an XML stream is to be associated with the asserted domain.

6.1. TLS

During TLS negotiation, an XMPP server acting as a TLS server sends its certificate to the connecting client or peer server acting as a TLS client. This certificate is interpreted as an assertion of the server's identity.

6.2. SASL

During SASL negotiation after TLS negotiation, an XMPP server acting as a TLS server can include an authorization identity; such an authzid is an assertion of the server's identity.

6.3. <db:result>

When two servers use the Server Dialback protocol [XEP-0220], the originating server asserts its identity by sending a <db:result/> element to the receiving server, where the 'from' attribute specifies the domain name being asserted by the originating server.

Note: Although historically the <db:result/> element has contained a dialback key as XML character data, the <db:result/> element can also be used without dialback keys as a mere assertion; this usage is sometimes colloquially referred to as "dialback without dialback".

6.4. A Note about Stream Attributes

XML streams include 'to' and 'from' attributes. However, these are not assertions of identity, and are merely early indications of the identity that a client or server will later assert during TLS negotiation, SASL negotiation, or Server Dialback negotiation.

7. Delegation Methods

Although domain name associations are closely tied to delegation in some scenarios, delegation is irrelevant when the source domain is exactly the same as the hostname of the XMPP service, as is often the case with single-domain services. There are two methods for secure delegation: DNSSEC (see the [XMPP-DANE] spec) and HTTPS Redirect (see the [XMPP-POSH] spec).

8. Security Considerations

This document supplements but does not supersede the security considerations provided in [RFC6120] and [RFC6125].

9. IANA Considerations

This document has no actions for the IANA.

10. References

10.1. Normative References

- [DANE] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", draft-ietf-dane-protocol-23 (work in progress), June 2012.
- [XMPP-DANE] Miller, M. and P. Saint-Andre, "Using DNS Security Extensions (DNSSEC) and DNS-based Authentication of Named Entities (DANE) as a Prooftype for XMPP Domain Name Associations", draft-miller-xmpp-dnssec-prooftype-02 (work in progress), June 2012.
- [XMPP-POSH] Miller, M. and P. Saint-Andre, "Using PKIX over Secure HTTP (POSH) as a Prooftype for XMPP Domain Name Associations", draft-miller-xmpp-posh-prooftype-00 (work in progress), June 2012.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.
- [RFC3365] Schiller, J., "Strong Security Requirements for Internet Engineering Task Force Standard Protocols", BCP 61,

RFC 3365, August 2002.

- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, May 2005.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.
- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.
- [XEP-0220] Miller, J., Saint-Andre, P., and P. Hancke, "Server Dialback", XSF XEP 0220, August 2011.

10.2. Informative References

- [RFC3920] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, October 2004.
- [XEP-0238] Saint-Andre, P., "XMPP Protocol Flows for Inter-Domain Federation", XSF XEP 0238, March 2008.

Authors' Addresses

Peter Saint-Andre
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Email: psaintan@cisco.com

Matthew Miller
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Email: mamille2@cisco.com

