

XMPP
Internet-Draft
Obsoletes: 6122 (if approved)
Intended status: Standards Track
Expires: April 21, 2014

P. Saint-Andre
Cisco Systems, Inc.
October 18, 2013

Extensible Messaging and Presence Protocol (XMPP): Address Format
draft-ietf-xmpp-6122bis-08

Abstract

This document defines the address format for the Extensible Messaging and Presence Protocol (XMPP), including support for code points outside the 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 April 21, 2014.

Copyright Notice

Copyright (c) 2013 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. Addresses	3
3.1. Fundamentals	3
3.2. Domainpart	5
3.3. Localpart	6
3.4. Resourcepart	7
4. Enforcement in JIDs and JID Parts	9
5. Internationalization Considerations	11
6. IANA Considerations	11
6.1. JIDlocalIdentifierClass	11
6.2. JIDresourceFreeformClass	12
7. Security Considerations	12
7.1. Reuse of PRECIS	12
7.2. Reuse of Unicode	12
7.3. Address Spoofing	12
7.3.1. Address Forging	13
7.3.2. Address Mimicking	13
8. Conformance Requirements	14
9. References	16
9.1. Normative References	16
9.2. Informative References	17
Appendix A. Differences from RFC 6122	20
Appendix B. Acknowledgements	21
Author's Address	21

1. Introduction

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 (see [RFC6885]). Instead, this document builds upon the internationalization framework defined by the IETF's PRECIS Working Group [I-D.ietf-precis-framework], while attempting to ensure that the characters allowed in Jabber Identifiers today under stringprep are still allowed and handled in the same way under PRECIS.

This document obsoletes RFC 6122.

2. Terminology

Many important terms used in this document are defined in [I-D.ietf-precis-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].

3. Addresses

3.1. 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 ID ("JID"). A valid JID is a string of Unicode code points [UNICODE], 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*1023(localpoint)
               ;
               ; a "localpoint" is a UTF-8 encoded
               ; Unicode code point that conforms to
               ; the "JIDlocalIdentifierClass" profile
               ; of the PRECIS IdentifierClass
               ;
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*1023(domainpoint)
               ;
               ; a "domainpoint" is a UTF-8 encoded
               ; Unicode code point that conforms to
               ; RFC 5890
               ;
resourcepart  = 1*1023(resourcepoint)
               ;
               ; a "resourcepoint" is a UTF-8 encoded
               ; Unicode code point that conforms to
               ; the "JIDresourceFreeformClass" profile
               ; of the PRECIS FreeformClass
               ;
```

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 octets in length and MUST NOT be more than 1023 octets in length, resulting in a maximum total size (including the '@' and '/' separators) of 3071 octets.

Implementation Note: The length limits on JIDs and JID parts are based on octets (bytes), not characters. UTF-8 encoding can result in more than one octet per character.

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.

3.2. 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 purpose 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 XMPP URI or IRI [RFC5122]. In particular, the character MUST be stripped before any other canonicalization steps are taken.

In general, the content of a domainpart is an Internationalized Domain Name ("IDN") as described in the specifications for Internationalized Domain Names in Applications (commonly called "IDNA2008") [RFC5890], and a domainpart is an "IDNA-aware domain name slot". The following rules apply to a domainpart that consists of a fully-qualified domain name:

- o The domainpart MUST contain only NR-LDH labels and U-labels as defined in [RFC5890] and MUST consist only of Unicode code points that conform to the rules specified in [RFC5892].
- o The domainpart MUST NOT include A-labels as defined in [RFC5890]; each A-label MUST be converted to a U-label during preparation of a domainpart, and comparison MUST be performed using U-labels, not A-labels.
- o After conversion of A-labels to U-labels if necessary, all uppercase and titlecase code points within the domainpart MUST be mapped to their lowercase equivalents.
- o After (and in addition to) case mapping and width mapping, other mappings MAY be applied to the domainpart, such as those defined in [I-D.ietf-precis-mappings] or [RFC5895].

After any and all normalization, conversion, and mapping of code points, a domainpart MUST NOT be zero octets in length and MUST NOT be more than 1023 octets in length. (Naturally, the length limits of [RFC1034] apply, and nothing in this document is to be interpreted as overriding those more fundamental limits.)

3.3. 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 octets in length and MUST NOT be more than 1023 octets in length. This rule is to be enforced after any normalization and mapping of code points.

A localpart MUST consist only of Unicode code points that conform to the "JIDlocalIdentifierClass" profile of the "IdentifierClass" base string class defined in [I-D.ietf-precis-framework]. The JIDlocalIdentifierClass profile includes all code points allowed by the IdentifierClass base class, 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 normalization and mapping rules for the JIDlocalIdentifierClass are as follows, where the operations specified MUST be completed in the order shown:

1. Fullwidth and halfwidth characters MUST be mapped to their decomposition equivalents.
2. Additional mappings MAY be applied, such as those defined in [I-D.ietf-precis-mappings].
3. Uppercase and titlecase characters MUST be mapped to their lowercase equivalents.
4. 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).

3.4. 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 octets in length and MUST NOT be more than 1023 octets in length. This rule is to be enforced after any normalization and mapping of code points.

A resourcepart MUST consist only of Unicode code points that conform to the "JIDresourceFreeformClass" profile of the "FreeformClass" base string class defined in [I-D.ietf-precis-framework].

The normalization and mapping rules for the resourcepart of a JID are as follows, where the operations specified MUST be completed in the order shown:

1. Fullwidth and halfwidth characters MAY be mapped to their decomposition equivalents.
2. Map any instances of non-ASCII space to ASCII space (U+0020).
3. Other additional mappings MAY be applied, such as those defined in [I-D.ietf-precis-mappings].
4. Uppercase and titlecase characters MAY be mapped to their lowercase equivalents.
5. All characters MUST be mapped using Unicode Normalization Form C (NFC).
6. Leading and trailing whitespace (i.e., one or more instances of the ASCII space character at the beginning or end of a resourcepart) MUST be removed (e.g., "stpeter " is mapped to "stpeter").

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 "handle" 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) handle of <user@host>. However, chatroom services do not necessarily check such an asserted handle against the occupant's real JID.

In some contexts, it might be appropriate to apply more restrictive rules to the preparation and comparison of XMPP resourceparts. For example, in the context of XMPP Multi-User Chat [XEP-0045], it might be appropriate to apply the rules specified in [I-D.ietf-precis-nickname]. However, the application of such more restrictive rules is out of scope for resourceparts in general and is properly defined in specifications for the relevant XMPP extensions.

4. 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 this document before including them in protocol slots within XML streams (such that JIDs and parts of JIDs are in conformance), XMPP servers MUST enforce the rules wherever possible and reject stanzas and other XML elements that violate the rules (for stanzas, by returning a <jid-malformed/> error to the sender as described in Section 8.3.3.8 of [RFC6120]).

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.

A server is responsible for enforcing the address format rules when receiving protocol elements from clients where the server is expected to handle such elements directly or to use them for purposes of routing a stanza to another domain or delivering a stanza to a local entity; 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).

A server is not responsible for enforcing the rules when the protocol elements are intended for communication among other entities, typically within the payload of a stanza that the server is merely

routing to another domain or delivering to a local entity, such as a connected client or an add-on service. 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 themselves and not depend on the server to do so, and client implementers need to understand that not enforcing the rules can lead to a degraded user experience or to security vulnerabilities. However, when an add-on service (e.g., a multi-user chat service) handles a stanza directly, it ought to enforce the rules as well, as defined by the relevant specification for that type of service.

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 when they are handled directly or used for purposes of routing to another domain or delivering to a local entity:

- 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 the Blocking Command [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 <iq/>, <message/>, and <presence/> elements for the Jabber Component Protocol [XEP-0114].

Developers of XMPP clients and specialized XMPP add-on services are advised to check the appropriate specifications for JID slots, localpart slots, and resourcepart slots in XMPP protocol extensions such as Service Discovery [XEP-0030], 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].

5. Internationalization Considerations

XMPP applications MUST support IDNA2008 for domainparts as described under Section 3.2, the "JIDlocalIdentifierClass" profile for localparts as described under Section 3.3, and the "JIDresourceFreeformClass" profile for resourceparts as described under Section 3.4. This enables XMPP addresses to include a wide variety of characters outside the ASCII range. Rules for enforcement of the XMPP address format are provided in [RFC6120] and specifications for various XMPP extensions.

Interoperability Note: For backward compatibility, many existing XMPP implementations and deployments support IDNA2003 [RFC3490] for domainparts, and the stringprep [RFC3454] profiles Nodeprep and Resourceprep [RFC3920] for localparts and resourceparts.

6. IANA Considerations

The IANA shall add the following entries to the PRECIS Profiles Registry.

6.1. JIDlocalIdentifierClass

Name: JIDlocalIdentifierClass.
Applicability: Localparts of XMPP addresses.
Base Class: IdentifierClass.
Replaces: Nodeprep.
Width Mapping: Map fullwidth and halfwidth characters to their decomposition equivalents.
Additional Mappings: None required or recommended.
Case Mapping: Map uppercase and titlecase characters to lowercase.
Normalization: NFC.
Directionality: The "Bidi Rule" defined in RFC 5893 applies.
Exclusions: Eight legacy characters in the ASCII range.

Enforcement: In general, XMPP servers are responsible for enforcing the rules (although XMPP clients and components can also be responsible for doing so, depending on the JID slots, localpart slots, and resourcepart slots where JIDs or JID parts are used).
Specification: RFC XXXX. [Note to RFC Editor: please change XXXX to the number issued for this specification.]

6.2. JIDresourceFreeformClass

Profile: JIDresourceFreeformClass.
Applicability: Resourceparts of XMPP addresses.
Base Class: FreeformClass
Replaces: The Resourceprep profile of Stringprep.
Width Mapping: Optional.
Additional Mappings: Map non-ASCII space to ASCII space.
Case Mapping: Optional.
Normalization: NFC.
Directionality: The "Bidi Rule" defined in RFC 5893 applies.
Exclusions: None.
Enforcement: In general, XMPP servers are responsible for enforcing the rules (although XMPP clients and components can also be responsible for doing so, depending on the JID slots, localpart slots, and resourcepart slots where JIDs or JID parts are used).
Specification: RFC XXXX. [Note to RFC Editor: please change XXXX to the number issued for this specification.]

7. Security Considerations

7.1. Reuse of PRECIS

The security considerations described in [I-D.ietf-precis-framework] apply to the "IdentifierClass" and "FreeformClass" 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.

7.2. Reuse of Unicode

The security considerations described in [UTS39] apply to the use of Unicode characters in XMPP addresses.

7.3. Address Spoofing

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

7.3.1. 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.

7.3.2. 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], [I-D.ietf-precis-framework], [UTR36], and [UTS39], 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 private 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 [I-D.ietf-precis-framework].

8. 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 octet in length and at most 1023 octets in length, and that it conforms to the underlying length limits of the DNS.

Section: Section 3.2

Roles: Server MUST, client SHOULD.

Feature: address-domain-prep

Description: Ensure that the domainpart of an XMPP address conforms to IDNA2008, that it contains only NR-LDH labels and U-labels (not A-labels), and that all uppercase and titlecase code points are mapped to their lowercase equivalents.

Section: Section 3.2

Roles: Server MUST, client SHOULD.

Feature: address-localpart-length

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

Section: Section 3.3

Roles: Server MUST, client SHOULD.

Feature: address-localpart-prep

Description: Ensure that the localpart of an XMPP address conforms to the "JIDlocalIdentifierClass" profile.

Section: Section 3.3

Roles: Server MUST, client SHOULD.

Feature: address-resource-length

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

Section: Section 3.4

Roles: Server MUST, client SHOULD.

Feature: address-resource-prep

Description: Ensure that the resourcepart of an XMPP address conforms to the "JIDresourceFreeformClass" profile.

Section: Section 3.4

Roles: Server MUST, client SHOULD.

9. References

9.1. Normative References

- [I-D.ietf-precis-framework]
Saint-Andre, P. and M. Blanchet, "Precis Framework: Handling Internationalized Strings in Protocols", draft-ietf-precis-framework-10 (work in progress), October 2013.
- [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.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [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.
- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.
- [UNICODE] The Unicode Consortium, "The Unicode Standard, Version 6.2", 2012,
<<http://www.unicode.org/versions/Unicode6.2.0/>>.
- [UTR36] The Unicode Consortium, "Unicode Technical Report #36: Unicode Security Considerations", July 2012,
<<http://www.unicode.org/reports/tr36/>>.

9.2. Informative References

- [I-D.ietf-precis-mappings]
Yoneya, Y. and T. NEMOTO, "Mapping characters for PRECIS classes", draft-ietf-precis-mappings-04 (work in progress), October 2013.
- [I-D.ietf-precis-nickname]
Saint-Andre, P., "Preparation and Comparison of Nicknames", draft-ietf-precis-nickname-06 (work in progress), July 2013.
- [INTEROP] Masinter, L., "Formalizing IETF Interoperability Reporting", Work in Progress, October 2005.
- [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.
- [RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.
- [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.
- [RFC5894] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, August 2010.
- [RFC5895] Resnick, P. and P. Hoffman, "Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008", RFC 5895, September 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.
- [RFC6885] Blanchet, M. and A. Sullivan, "Stringprep Revision and Problem Statement for the Preparation and Comparison of Internationalized Strings (PRECIS)", RFC 6885, March 2013.
- [UTS39] The Unicode Consortium, "Unicode Technical Standard #39: Unicode Security Mechanisms", July 2012, <<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, February 2012.
- [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 2011.
- [XEP-0077] Saint-Andre, P., "In-Band Registration", XSF XEP 0077, January 2012.
- [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., "Blocking Command", XSF XEP 0191, July 2012.

[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, August 2012.

[XEP-0292]

Saint-Andre, P. and S. Mizzi, "vCard4 Over XMPP", XSF XEP 0292, October 2011.

[XML]

Maler, E., Yergeau, F., Sperberg-McQueen, C., Paoli, J., and T. Bray, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <<http://www.w3.org/TR/2008/REC-xml-20081126>>.

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 JIDlocalIdentifierClass profile of the PRECIS IdentifierClass (instead of the Nodeprep profile of Stringprep).
- o Changed resourcepart preparation to use the JIDresourceFreeformClass profile of the PRECIS FreeformClass (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 fullwidth and halfwidth must be mapped to their decomposition equivalents (previously handled through the use of NFKC).
- o Specified the use of Unicode normalization form C (instead of KC as specified in the Nodeprep and Resourceprep profiles of Stringprep).
- o Specified that servers must enforce the address formatting rules.

Appendix B. Acknowledgements

Thanks to Miguel Garcia, 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: April 23, 2014

P. Saint-Andre
M. Miller
Cisco Systems, Inc.
October 20, 2013

Domain Name Associations (DNA) in the Extensible Messaging and Presence
Protocol (XMPP)
draft-ietf-xmpp-dna-04

Abstract

This document improves the security of the Extensible Messaging and Presence Protocol (XMPP) in two ways. First, it specifies how "prooftypes" can establish a strong association between a domain name and an XML stream. Second, it describes how to securely delegate 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 April 23, 2014.

Copyright Notice

Copyright (c) 2013 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	2
2. Terminology	3
3. Flow Chart	3
4. A Simple Scenario	6
5. One-Way Authentication	7
6. Piggybacking	8
6.1. Assertion	8
6.2. Supposition	9
7. Alternative Proofypes	11
7.1. DANE	11
7.2. POSH	12
8. Secure Delegation and Multi-Tenancy	12
9. Proofype Model	13
10. IANA Considerations	13
10.1. Well-Known URI for xmpp-client Service	13
10.2. Well-Known URI for xmpp-server Service	14
11. Security Considerations	14
12. References	14
12.1. Normative References	14
12.2. Informative References	15
Authors' Addresses	16

1. Introduction

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 the Extensible Messaging and Presence Protocol (XMPP) [RFC6120]. Because XMPP servers are typically identified by DNS domain names, 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. In relation to such associations, this document does the following:

1. Generalizes the model currently in use so that additional proofypes can be defined
2. Provides a basis for modernizing some proofypes to reflect progress in underlying technologies such as DNS Security [RFC4033]

3. Describes the flow of operations for establishing a domain name association (DNA)

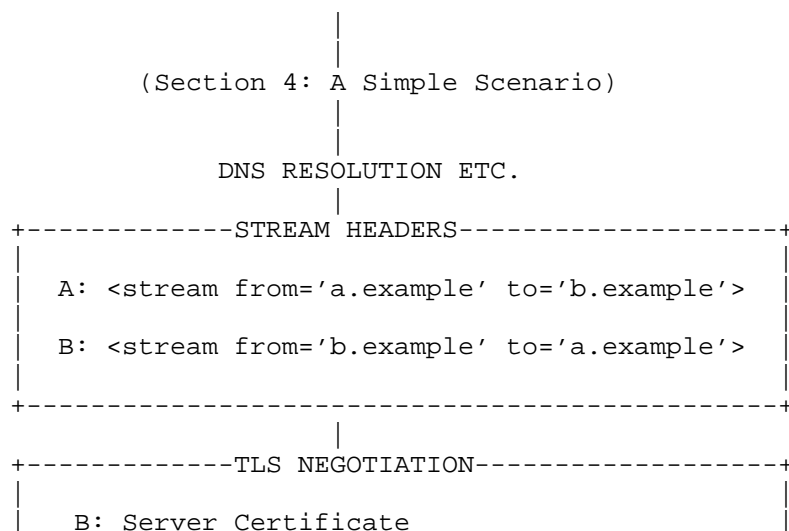
This document also provides guidelines for secure delegation. The need for secure delegation arises because the process for resolving the domain name of an XMPP service into the IP address at which an XML stream will be negotiated (defined in [RFC6120]) can involve delegation of a source domain (say, example.com) to a derived domain (say, hosting.example.net) using technologies such as DNS SRV records [RFC2782]. If such delegation is not done in a secure manner, then the domain name association cannot be authenticated.

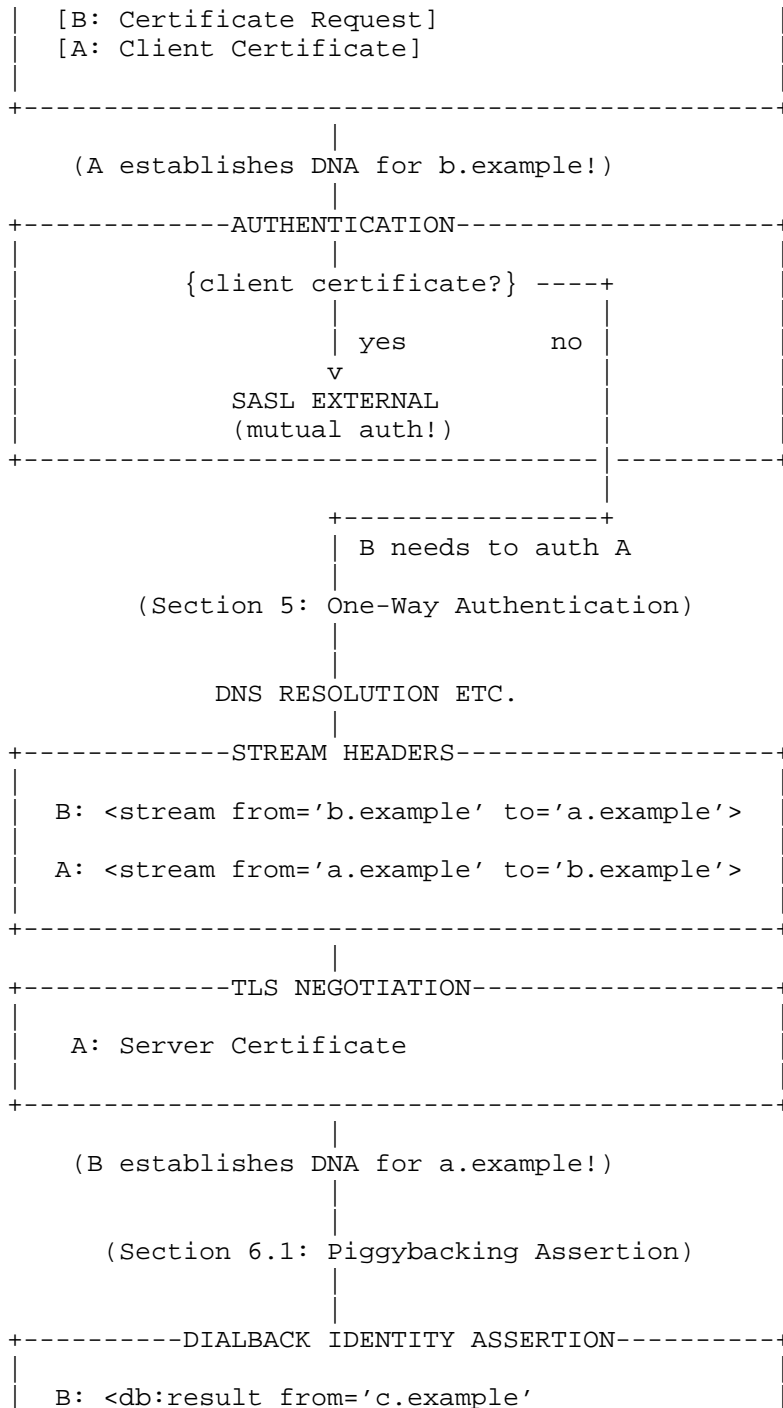
2. Terminology

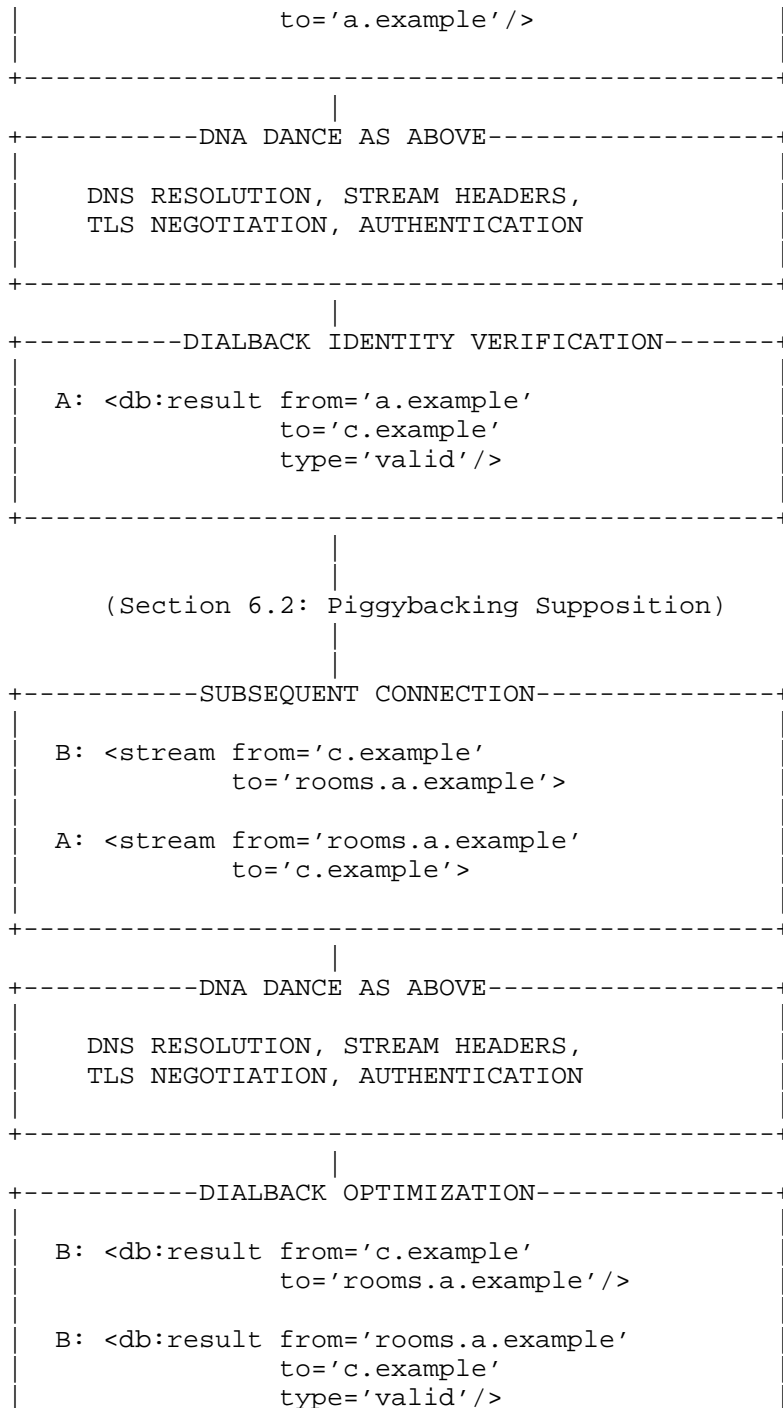
This document inherits XMPP terminology from [RFC6120] and [XEP-0220], DNS terminology from [RFC1034], [RFC1035], [RFC2782] and [RFC4033], and security terminology from [RFC4949] and [RFC5280]. The terms "source domain", "derived domain", "reference identity", and "presented identity" are used as defined 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.

3. Flow Chart

The following flow chart illustrates the protocol flow for establishing domain name associations between Server 1 and Server 2, as described in the remaining sections of this document.







|
+-----+
|

4. A Simple Scenario

To illustrate the problem, consider the simplified order of events (see [RFC6120] for details) in establishing an XML stream between Server 1 (a.example) and Server 2 (b.example):

1. Server 1 resolves the DNS domain name b.example.
2. Server 1 opens a TCP connection to the resolved IP address.
3. Server 1 sends an initial stream header to Server 2, asserting that it is a.example:

`<stream:stream from='a.example' to='b.example'>`
4. Server 2 sends a response stream header to Server 1, asserting that it is b.example:

`<stream:stream from='b.example' to='a.example'>`
5. The servers attempt TLS negotiation, during which Server 2 (acting as a TLS server) presents a PKIX certificate proving that it is b.example and Server 1 (acting as a TLS client) presents a PKIX certificate proving that it is a.example.
6. Server 1 checks the PKIX certificate that Server 2 provided and Server 2 checks the PKIX certificate that Server 1 provided; if these proofs are consistent with the XMPP profile of the matching rules from [RFC6125], each server accepts that there is a strong domain name association between its stream to the other party and the DNS domain name of the other party.

Several simplifying assumptions underlie the happy scenario just outlined:

- o Server 1 presents a PKIX certificate during TLS negotiation, which enables the parties to complete mutual authentication.

- o There are no additional domains associated with Server 1 and Server 2 (say, a subdomain rooms.a.example on Server 1 or a second domain c.example on Server 2).
- o The server administrators are able to obtain PKIX certificates in the first place.
- o The server administrators are running their own XMPP servers, rather than using hosting services.

Let's consider each of these "wrinkles" in turn.

5. One-Way Authentication

If Server 1 does not present its PKIX certificate during TLS negotiation (perhaps because it wishes to verify the identity of Server 2 before presenting its own credentials), Server 2 is unable to mutually authenticate Server 1. Therefore, Server 2 needs to negotiate and authenticate a stream to Server 1, just as Server 1 has done:

1. Server 2 resolves the DNS domain name a.example.
2. Server 2 opens a TCP connection to the resolved IP address.
3. Server 2 sends an initial stream header to Server 1, asserting that it is b.example:

`<stream:stream from='b.example' to='a.example'>`
4. Server 1 sends a response stream header to Server 2, asserting that it is a.example:

`<stream:stream from='a.example' to='b.example'>`
5. The servers attempt TLS negotiation, during which Server 1 (acting as a TLS server) presents a PKIX certificate proving that it is a.example.

6. Server 2 checks the PKIX certificate that Server 1 provided; if it is consistent with the XMPP profile [RFC6120] of the matching rules from [RFC6125], Server 2 accepts that there is a strong domain name association between its stream to Server 1 and the DNS domain name a.example.

At this point the servers are using two TCP connections instead of one, which is somewhat wasteful. However, there are ways to tie the authentication achieved on the second TCP connection to the first TCP connection; see [XEP-0288] for further discussion.

6. Piggybacking

6.1. Assertion

Consider the common scenario in which Server 2 hosts not only b.example but also a second domain c.example (a "multi-tenanted" environment). If a user of Server 2 associated with c.example wishes to communicate with a friend at a.example, Server 2 needs to send XMPP stanzas from the domain c.example rather than b.example. Although Server 2 could open a new TCP connection and negotiate new XML streams for the domain pair of c.example and a.example, that too is wasteful. Server 2 already has a connection to a.example, so how can it assert that it would like to add a new domain pair to the existing connection?

The traditional method for doing so is the Server Dialback protocol, first specified in [RFC3920] and since moved to [XEP-0220]. Here, Server 2 can send a <db:result/> element for the new domain pair over the existing stream.

```
<db:result from='c.example' to='a.example'>
  some-dialback-key
</db:result>
```

This element functions as Server 2's assertion that it is (also) c.example, and thus is functionally equivalent to the 'from' address of an initial stream header as previously described.

In response to this assertion, Server 1 needs to obtain some kind of proof that Server 2 really is also c.example. It can do the same thing that it did before:

1. Server 1 resolves the DNS domain name c.example.

2. Server 1 opens a TCP connection to the resolved IP address (which might be the same IP address as for b.example).
3. Server 1 sends an initial stream header to Server 2, asserting that it is a.example:

```
<stream:stream from='a.example' to='c.example'>
```

4. Server 2 sends a response stream header to Server 1, asserting that it is c.example:

```
<stream:stream from='c.example' to='a.example'>
```

5. The servers attempt TLS negotiation, during which Server 2 (acting as a TLS server) presents a PKIX certificate proving that it is c.example.
6. Server 1 checks the PKIX certificate that Server 2 provided; if it is consistent with the XMPP profile [RFC6120] of the matching rules from [RFC6125], Server 1 accepts that there is a strong domain name association between its stream to Server 2 and the DNS domain name c.example.

Now that Server 1 accepts the domain name association, it informs Server 2 of that fact:

```
<db:result from='a.example' to='c.example' type='valid'/>
```

The parties can then terminate the second connection, since it was used only for Server 1 to associate a stream over the same IP:port combination with the domain name c.example (the dialback key links the original stream to the new association).

6.2. Supposition

Piggybacking can also occur in the other direction. Consider the common scenario in which Server 1 provides XMPP services not only for a.example but also for a subdomain such as a groupchat service at rooms.a.example (see [XEP-0045]). If a user from c.example at Server 2 wishes to join a room on the groupchat service, Server 2 needs to send XMPP stanzas from the domain c.example to the domain rooms.a.example rather than a.example. Therefore, Server 2 needs to negotiate and authenticate a stream to rooms.a.example:

1. Server 2 resolves the DNS domain name rooms.a.example.
2. Server 2 opens a TCP connection to the resolved IP address.
3. Server 2 sends an initial stream header to Server 1 acting as rooms.a.example, asserting that it is b.example:

```
<stream:stream from='b.example' to='rooms.a.example'>
```
4. Server 1 sends a response stream header to Server 2, asserting that it is rooms.a.example:

```
<stream:stream from='rooms.a.example' to='b.example'>
```
5. The servers attempt TLS negotiation, during which Server 1 (acting as a TLS server) presents a PKIX certificate proving that it is rooms.a.example.
6. Server 2 checks the PKIX certificate that Server 1 provided; if it is consistent with the XMPP profile [RFC6120] of the matching rules from [RFC6125], Server 2 accepts that there is a strong domain name association between its stream to Server 1 and the DNS domain name rooms.a.example.

As before, the parties now have two TCP connections open. So that they can close the now-redundant connection, Server 2 sends a dialback key to Server 1 over the new connection.

```
<db:result from='c.example' to='rooms.a.example'>  
  some-dialback-key  
</db:result>
```

Server 1 then informs Server 2 that it accepts the domain name association:

```
<db:result from='rooms.a.example' to='c.example' type='valid'>
```

Server 2 can now close the connection over which it tested the domain name association for rooms.a.example.

7. Alternative Proofotypes

The foregoing protocol flows assumed that domain name associations were proved using the standard PKI proofotype specified in [RFC6120]: that is, the server's proof consists of a PKIX certificate that is checked according to the XMPP profile [RFC6120] of the matching rules from [RFC6125], the client's verification material is obtained out of band in the form of a trusted root, and secure DNS is not necessary.

However, sometimes XMPP server administrators are unable or unwilling to obtain valid PKIX certificates for their servers. As one example, a certificate authority (CA) might try to send email messages to authoritative mailbox names [RFC2142], but the administrator of a subsidiary service such as `im.cs.podunk.example` can't receive email sent to `mailto:hostmaster@podunk.example`. As another example, a hosting provider such as `hosting.example.net` might not want to take on the liability of holding the certificate and private key for a tenant such as `example.com` (or the tenant might not want the hosting provider to hold its certificate and private key). In these circumstances, proofotypes other than PKIX are desirable. As described below, two alternatives have been defined so far: DNS-Based Authentication of Named Entities (DANE) and and PKIX Over Secure HTTP (POSH).

7.1. DANE

In the DANE proofotype, the server's proof consists of a PKIX certificate that is compared as an exact match or a hash of either the `SubjectPublicKeyInfo` or the full certificate, and the client's verification material is obtained via secure DNS.

The DANE proofotype makes use of the DNS-Based Authentication of Named Entities [RFC6698], specifically the use of DANE with DNS SRV records [I-D.ietf-dane-srv]. For XMPP purposes, the following rules apply:

- o If there is no SRV resource record, pursue the fallback methods described in [RFC6120].
- o Use the 'to' address of the initial stream header to determine the domain name of the TLS client's reference identifier (since use of the TLS Server Name Indication is purely discretionary in XMPP, as mentioned in [RFC6120]).

7.2. POSH

In the POSH proof type, the server's proof consists of a PKIX certificate that is checked according to the rules from [RFC6120] and [RFC6125], the client's verification material is obtained by retrieving the PKIX certificate over HTTPS at a well-known URI [RFC5785], and secure DNS is not necessary since the HTTPS retrieval mechanism relies on the chain of trust from the public key infrastructure.

POSH is defined in [I-D.miller-posh]. For XMPP purposes, the well-known URIs [RFC5785] to be used are:

- o `"/.well-known/posh._xmpp-client._tcp.json"` for client-to-server connections
- o `"/.well-known/posh._xmpp-server._tcp.json"` for server-to-server connections

8. Secure Delegation and Multi-Tenancy

One common method for deploying XMPP services is multi-tenancy or virtual hosting: e.g., the XMPP service for `example.com` is actually hosted at `hosting.example.net`. Such an arrangement is relatively convenient in XMPP given the use of DNS SRV records [RFC2782], such as the following pointer from `example.com` to `hosting.example.net`:

```
_xmpp-server._tcp.example.com. 0 IN SRV 0 0 5269 hosting.example.net
```

Secure connections with multi-tenancy can work using the PKIX proof type on a small scale if the provider itself wishes to host several domains (e.g., several related domains such as `jabber-de.example` and `jabber-ch.example`). However, in practice the security of multi-tenancy has been found to be unwieldy when the provider hosts large numbers of XMPP services on behalf of multiple tenants. Typically there are two main reasons for this state of affairs: the service provider (say, `hosting.example.net`) wishes to limit its liability and therefore does not wish to hold the certificate and private key for the tenant (say, `example.com`) and the tenant wishes to improve the security of the service and therefore does not wish to share its certificate and private key with service provider. As a result, server-to-server communications to `example.com` go unencrypted or the communications are TLS-encrypted but the certificates are not checked (which is functionally equivalent to a connection using an anonymous key exchange). This is also true of client-to-server communications, forcing end users to override certificate warnings or configure their clients to accept certificates for

hosting.example.net instead of example.com. The fundamental problem here is that if DNSSEC is not used then the act of delegation via DNS SRV records is inherently insecure.

The specification for use of SRV and MX records with DANE [I-D.ietf-dane-srv] explains how to use DNSSEC for secure delegation with the DANE prooftype, and the POSH specification [I-D.miller-posh] explains how to use HTTPS redirects for secure delegation with the POSH prooftype.

9. Prooftype Model

In general, a domain name association (DNA) prooftype conforms to the following definition:

prooftype: A mechanism for proving an association between a domain name and an XML stream, where the mechanism defines (1) the nature of the server's proof, (2) the matching rules for comparing the client's verification material against the server's proof, (3) how the client obtains its verification material, and (4) whether the mechanism depends on secure DNS.

The PKI, DANE, and POSH prooftypes adhere to this model. In addition, other prooftypes are possible (examples might include PGP keys rather than PKIX certificates, or a token mechanism such as Kerberos or OAuth).

Some prooftypes depend on (or are enhanced by) secure DNS and thus also need to describe how they ensure secure delegation.

10. IANA Considerations

The POSH specification [I-D.miller-posh] provides guidelines for registering the well-known URIs [RFC5785] of protocols that make use of POSH. This specification registers two such URIs, for which the completed registration templates follow.

10.1. Well-Known URI for xmpp-client Service

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

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

Change controller: IETF

Specification document(s): [[this document]]

10.2. Well-Known URI for xmpp-server Service

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

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

Change controller: IETF

Specification document(s): [[this document]]

11. Security Considerations

This document supplements but does not supersede the security considerations of [RFC6120] and [RFC6125]. Relevant security considerations can also be found in [I-D.ietf-dane-srv] and [I-D.miller-posh].

12. References

12.1. Normative References

- [I-D.ietf-dane-srv]
Finch, T., "Using DNS-Based Authentication of Named Entities (DANE) TLSA records with SRV and MX records.", draft-ietf-dane-srv-02 (work in progress), February 2013.
- [I-D.miller-posh]
Miller, M. and P. Saint-Andre, "PKIX over Secure HTTP (POSH)", draft-miller-posh-02 (work in progress), September 2013.
- [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.
- [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.
- [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.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.
- [XEP-0220] Miller, J., Saint-Andre, P., and P. Hancke, "Server Dialback", XSF XEP 0220, September 2013.

12.2. Informative References

- [RFC2142] Crocker, D., "MAILBOX NAMES FOR COMMON SERVICES, ROLES AND FUNCTIONS", RFC 2142, May 1997.
- [RFC3920] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, October 2004.
- [XEP-0045] Saint-Andre, P., "Multi-User Chat", XSF XEP 0045, February 2012.
- [XEP-0238] Saint-Andre, P., "XMPP Protocol Flows for Inter-Domain Federation", XSF XEP 0238, March 2008.
- [XEP-0288] Hancke, P. and D. Cridland, "Bidirectional Server-to-Server Connections", XSF XEP 0288, September 2013.

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

XMPP Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 09, 2014

L. Stout, Ed.
&yet
J. Moffitt
Mozilla
E. Cestari
cstar industries
September 05, 2013

An XMPP Sub-protocol for WebSocket
draft-ietf-xmpp-websocket-00

Abstract

This document defines a binding for the XMPP protocol over a WebSocket transport layer. A WebSocket binding for XMPP provides higher performance than the current HTTP binding for 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 March 09, 2014.

Copyright Notice

Copyright (c) 2013 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	2
2. Terminology	3
3. XMPP Sub-Protocol	3
3.1. Handshake	3
3.2. Messages	4
3.3. XMPP Stream Setup	4
3.4. Stream Errors	5
3.5. Closing the Connection	5
3.6. Stanzas	6
3.7. Stream Restarts	6
3.8. Pings and Keepalives	6
3.9. Use of TLS	7
3.10. Stream Management	7
4. Discovering Connection Method	7
5. Security Considerations	8
6. IANA Considerations	8
7. Informative References	8
Authors' Addresses	9

1. Introduction

Applications using XMPP (see [RFC6120] and [RFC6121]) on the Web currently make use of BOSH (see [XEP-0124] and [XEP-0206]), an XMPP binding to HTTP. BOSH is based on the HTTP long polling technique, and it suffers from high transport overhead compared to XMPP's native binding to TCP. In addition, there are a number of other known issues with long polling [RFC6202], which have an impact on BOSH-based systems.

It would be much better in most circumstances to avoid tunneling XMPP over HTTP long polled connections and instead use the XMPP protocol directly. However, the APIs and sandbox that browsers have provided do not allow this. The WebSocket protocol [RFC6455] now exists to solve these kinds of problems. The WebSocket protocol is a bi-directional protocol that provides a simple message-based framing layer over raw sockets and allows for more robust and efficient communication in web applications.

The WebSocket protocol enables two-way communication between a client and a server, effectively emulating TCP at the application layer and therefore overcoming many of the problems with existing long-polling techniques for bidirectional HTTP. This document defines a WebSocket sub-protocol for the Extensible Messaging and Presence Protocol (XMPP).

2. Terminology

The basic unit of framing in the WebSocket protocol is called a message. In XMPP, the basic unit is the stanza, which is a subset of the first-level children of each document in an XMPP stream (see Section 9 of [RFC6120]). XMPP also has a concept of messages, which are stanzas whose top-level element name is message. In this document, the word "message" will mean a WebSocket message, not an XMPP message stanza (see Section 3.2).

The 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. XMPP Sub-Protocol

3.1. Handshake

The XMPP sub-protocol is used to transport XMPP over a WebSocket connection. The client and server agree to this protocol during the WebSocket handshake (see Section 1.3 of [RFC6455]).

During the WebSocket handshake, the client MUST include the |Sec-WebSocket-Protocol| header in its handshake, and the value |xmpp| MUST be included in the list of protocols. The reply from the server MUST also contain |xmpp| in its own |Sec-WebSocket-Protocol| header in order for an XMPP sub-protocol connection to be established.

Once the handshake is complete, WebSocket messages sent or received will conform to the protocol defined in the rest of this document.

```
C: GET /xmpp-websocket HTTP/1.1
   Host: example.com
   Upgrade: websocket
   Connection: Upgrade
   Sec-WebSocket-Key: dGh1IHhnbXBsZSBub25jZQ==
   Origin: http://example.com
   ...
   Sec-WebSocket-Protocol: xmpp
   Sec-WebSocket-Version: 13

S: HTTP/1.1 101 Switching Protocols
   Upgrade: websocket
   Connection: Upgrade
   ...
   Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
   Sec-WebSocket-Protocol: xmpp
```

[WebSocket connection established]

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

3.2. Messages

Data frame messages in the XMPP sub-protocol MUST be of the text type and contain UTF-8 encoded data. The close control frame's contents are specified in Section 3.5. Control frames other than close are not restricted.

Unless noted in text, the word "message" will mean a WebSocket message composed of text data frames.

3.3. XMPP Stream Setup

The first message sent after the handshake is complete MUST be an XMPP opening stream tag as defined in XMPP [RFC6120] or an XML text declaration (see Section 4.3.1 of [W3C.REC-xml-20081126]) followed by an XMPP opening stream tag. The stream tag MUST NOT be closed (i.e. the closing `</stream:stream>` tag should not appear in the message) as it is the start of the client's outgoing XML. The '`<`' character of the tag or text declaration MUST be the first character of the text payload.

The server MUST respond with a message containing an error (see Section 3.4), its own opening stream tag, or an XML text declaration followed by an opening stream tag.

Except in the case of certain stream errors (see Section 3.4), the opening stream tag, <stream:stream>, MUST appear in a message by itself.

3.4. Stream Errors

Stream level errors in XMPP are terminal. Should such an error occur, the server MUST send the stream error as a complete element in a message to the client.

If the error occurs during the opening of a stream, the stream error message MUST start with an opening stream tag (see Section 4.7.1 of [RFC6120]) and end with a closing stream tag.

After the stream error and closing stream tag have been sent, the server MUST close the connection as in Section 3.5.

3.5. Closing the Connection

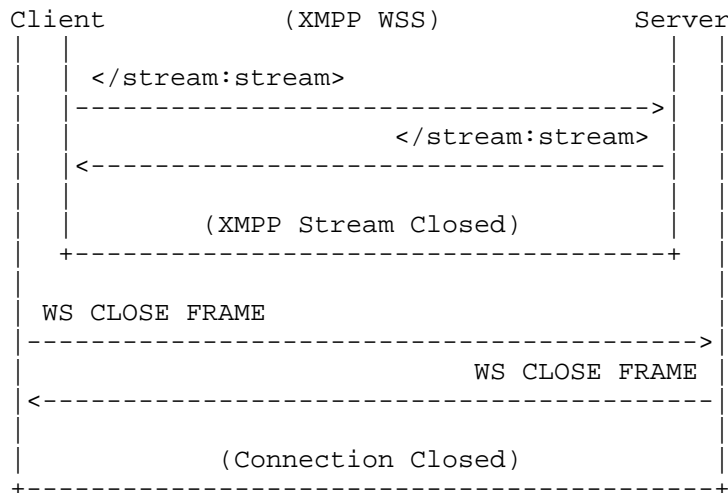
Either the server or the client may close the connection at any time. Before closing the connection, the closing party SHOULD close the XMPP stream, if it has been established, by sending a message with the closing </stream:stream> tag. The XMPP stream is considered closed when a corresponding </stream:stream> tag is received from the other party.

If a client closes the WebSocket connection without closing the XMPP stream after having enabled stream management (see Section 3.10), the server SHOULD keep the XMPP session alive for a period of time based on server policy, as specified in [XEP-0198].

To initiate closing the WebSocket connection, the closing party MUST send a normal WebSocket close message with an empty body. The connection is considered closed when a matching close message is received (see Section 1.4 of [RFC6455]).

Except in the case of certain stream errors (see Section 3.4), the closing stream tag, </stream:stream>, MUST appear in a message by itself.

An example of ending an XMPP over WebSocket session by first closing the XMPP stream layer and then the WebSocket connection layer:



3.6. Stanzas

Each XMPP stanza MUST be sent in its own message. A stanza MUST NOT be split over multiple messages. All first level children of the `<stream:stream>` element MUST be treated the same as stanzas (e.g. `<stream:features>` and `<stream:error>`).

3.7. Stream Restarts

After successful SASL authentication, an XMPP stream needs to be restarted. In these cases, as soon as the message is sent (or received) containing the success indication, both the server and client streams are implicitly closed, and new streams need to be opened. The client MUST open a new stream as in Section 3.3 and MUST NOT send a closing stream tag.

S: `<success xmlns="urn:ietf:params:xml:ns:xmpp-sasl" />`

[Streams implicitly closed]

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

3.8. Pings and Keepalives

XMPP servers send whitespace pings as keepalives between stanzas, and XMPP clients can do the same as these extra whitespace characters are not significant in the protocol. Servers and clients SHOULD use WebSocket ping control frames instead for this purpose.

In some cases, the WebSocket connection might be served by an intermediary connection manager and not the XMPP server. In these situations, the use of WebSocket ping messages are insufficient to test that the XMPP stream is still alive. Both the XMPP Ping extension [XEP-0199] and the XMPP Stream Management extension [XEP-0198] provide mechanisms to ping the XMPP server, and either extension (or both) MAY be used to determine the state of the connection.

3.9. Use of TLS

TLS cannot be used at the XMPP sub-protocol layer because the sub-protocol does not allow for raw binary data to be sent. Instead, enabling TLS SHOULD be done at the WebSocket layer using secure WebSocket connections via the `wss` URI scheme. (See Section 10.6 of [RFC6455]).

Because TLS is to be provided outside of the XMPP sub-protocol layer, a server MUST NOT advertise TLS as a stream feature (see Section 4.6 of [RFC6120]), and a client MUST ignore any advertised TLS stream feature, when using the XMPP sub-protocol.

3.10. Stream Management

In order to alleviate the problems of temporary disconnections, the XMPP Stream Management extension [XEP-0198] MAY be used to confirm when stanzas have been received by the server.

In particular, the use of session resumption in [XEP-0198] MAY be used to allow for recreating the same stream session state after a temporary network unavailability or after navigating to a new URL in a browser.

4. Discovering Connection Method

The XMPP extension Discovering Alternate XMPP Connection Methods [XEP-0156] provides a mechanism to discover the additional information needed to connect to an XMPP server outside of the procedure defined in in Section 3 of [RFC6120].

For the XMPP over Websocket connection type, the connection method name `"_xmpp-client-websocket"` is used to specify a URI for the server's WebSocket connection endpoint.

An example entry advertising that the URI "wss://example.com/xmpp" is an XMPP over WebSocket endpoint, using a DNS TXT record as specified in [XEP-0156]:

```
_xmppconnect IN TXT "_xmpp-client-websocket=wss://example.com/xmpp"
```

Implementation Note: A server is able to expose both BOSH [XEP-0206] and WebSocket endpoints over the registered port 5280, using the URI path and connection upgrade headers to determine which transport to serve.

5. Security Considerations

Since application level TLS cannot be used (see Section 3.9), applications which need to protect the privacy of the XMPP traffic need to do so at the WebSocket or other appropriate layer.

The Security Considerations for both WebSocket (See Section 10 of [RFC6455] and XMPP (See Section 13 of [RFC6120]) apply to the WebSocket XMPP sub-protocol.

6. IANA Considerations

This specification requests IANA to register the WebSocket XMPP sub-protocol under the "WebSocket Subprotocol Name" Registry with the following data:

Subprotocol Identifier: xmpp

Subprotocol Common Name: WebSocket Transport for the Extensible Messaging and Presence Protocol (XMPP)

Subprotocol Definition: RFC XXXX

[[NOTE TO RFC EDITOR: Please change XXXX to the number assigned to this document upon publication.]]

7. Informative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [RFC6202] Loreto, S., Saint-Andre, P., Salsano, S., and G. Wilkins, "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP", RFC 6202, April 2011.
- [RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.
- [W3C.REC-xml-20081126]
Sperberg-McQueen, C., Yergeau, F., Paoli, J., Bray, T., and E. Maler, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008,
<<http://www.w3.org/TR/2008/REC-xml-20081126>>.
- [XEP-0124] Paterson, I., Smith, D., Saint-Andre, P., and J. Moffitt, "Bidirectional-streams Over Synchronous HTTP (BOSH)", XSF XEP 0124, July 2010.
- [XEP-0156] Hildebrand, J. and P. Saint-Andre, "Discovering Alternative XMPP Connection Methods", XSF XEP 0156, June 2007.
- [XEP-0198] Karneges, J., Saint-Andre, P., Hildebrand, J., Forno, F., Cridland, D., and M. Wild, "Stream Management", XSF XEP 0198, June 2011.
- [XEP-0199] Saint-Andre, P., "XMPP Ping", XSF XEP 0199, June 2009.
- [XEP-0206] Paterson, I. and P. Saint-Andre, "XMPP Over BOSH", XSF XEP 0206, July 2010.

Authors' Addresses

Lance Stout (editor)
&yet

Email: lance@andyet.net

Jack Moffitt
Mozilla

Email: jack@metajack.im

Eric Cestari
cstar industries

Email: eric@cestari.info

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 31, 2014

M. Miller
P. Saint-Andre
Cisco Systems, Inc.
September 27, 2013

PKIX over Secure HTTP (POSH)
draft-miller-posh-02

Abstract

Experience has shown that it is extremely difficult to deploy proper PKIX certificates for TLS in multi-tenanted environments, since certification authorities will not issue certificates for hosted domains to hosting services, hosted domains do not want hosting services to hold their private keys, and hosting services wish to avoid liability for holding those keys. As a result, domains hosted in multi-tenanted environments often deploy non-HTTP applications such as email and instant messaging using certificates that identify the hosting service, not the hosted domain. Such deployments force end users and peer services to accept a certificate with an improper identifier, resulting in obvious security implications. This document defines two methods that make it easier to deploy certificates for proper server identity checking in non-HTTP application protocols. The first method enables the TLS client associated with a user agent or peer application server to obtain the end-entity certificate of a hosted domain over secure HTTP as an alternative to standard PKIX techniques. The second method enables a hosted domain to securely delegate a non-HTTP application to a hosting service using redirects provided by HTTPS itself or by a pointer in a file served over HTTPS at the hosted domain.

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 March 31, 2014.

Copyright Notice

Copyright (c) 2013 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. Discussion Venue	4
3. Terminology	4
4. Obtaining Verification Materials	4
4.1. Source Domain Possesses PKIX Certificate	6
4.2. Source Domain References PKIX Certificate	7
4.3. Performing Verification	8
5. Secure Delegation	9
6. Order of Operations	9
7. Caching Results	10
8. Alternates and Roll-over	10
9. IANA Considerations	11
10. Security Considerations	12
11. References	13
11.1. Normative References	13
11.2. Informative References	14
Appendix A. Acknowledgements	15
Authors' Addresses	15

1. Introduction

We start with a thought experiment.

Imagine that you work on the operations team of a hosting company that provides the "foo" service (or email or instant messaging or social networking service) for ten thousand different customer organizations. Each customer wants their service to be identified by the customer's domain name (e.g., foo.example.com), not the hosting company's domain name (e.g., hosting.example.net).

In order to properly secure each customer's "foo" service via Transport Layer Security (TLS) [RFC5246], you need to obtain PKIX certificates [RFC5280] containing identifiers such as foo.example.com, as explained in the "CertID" specification [RFC6125]. Unfortunately, you can't obtain such certificates because:

- o Certification authorities won't issue such certificates to you because you work for the hosting company, not the customer organization.
- o Customers won't obtain such certificates and then give them (plus the associated private keys) to you because their legal department is worried about liability.
- o You don't want to install such certificates (plus the associated private keys) on your servers anyway because your legal department is worried about liability, too.

Given your inability to deploy public keys / certificates containing the right identifiers, your back-up approach was always to use a certificate containing hosting.example.net as the identifier. However, more and more customers and end users are complaining about warning messages in user agents and the inherent security issues involved with taking a "leap of faith" to accept the identity mismatch between the source domain (foo.example.com) and the delegated domain (hosting.example.net).

This situation is both insecure and unsustainable. You have investigated the possibility of using DNS Security [RFC4033] and DNS-Based Authentication of Named Entities (DANE) [RFC6698] to solve the problem. However, your customers and your operations team have told you that they will not be able to deploy DNSSEC and DANE for several years at least. The product managers in your company are pushing you to find a method that can be deployed more quickly to overcome the lack of proper server identity checking for your hosted customers.

One possible approach is to ask each customer to provide the public key / certificate for the "foo" service at a special HTTPS URI on their website ("https://foo.example.com/.well-known/posh.foo.json" is one possibility). This could be a public key that you generate for the customer, but because the customer hosts it via HTTPS, any user agent can find that public key and check it against the public key you provide during TLS negotiation for the "foo" service (as one added benefit, the customer never needs to hand you a private key). Alternatively, the customer can redirect requests for that special HTTPS URI to an HTTPS URI at your own website, thus making it explicit that they have delegated the "foo" service to you.

The approach sketched out above, called POSH ("PKIX Over Secure HTTP"), is explained in the remainder of this document.

2. Discussion Venue

The discussion venue for this document is the posh@ietf.org mailing list; visit <https://www.ietf.org/mailman/listinfo/posh> for subscription information and discussion archives.

3. Terminology

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

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].

4. Obtaining Verification Materials

Server identity checking (see [RFC6125]) involves three different aspects:

1. A proof of the TLS server's identity (in PKIX, this takes the form of a PKIX certificate [RFC5280]).
2. Rules for checking the certificate (which vary by application protocol, although [RFC6125] attempts to harmonize those rules).

3. The materials that a TLS client uses to verify the TLS server's identity or check the TLS server's proof (in PKIX, this takes the form of chaining the end-entity certificate back to a trusted root and performing all validity checks as described in [RFC5280], [RFC6125], and the relevant application protocol specification).

When POSH is used, the first two aspects remain the same: the TLS server proves its identity by presenting a PKIX certificate [RFC5280] and the certificate is checked according to the rules defined in the appropriate application protocol specification (such as [RFC6120] for XMPP). However, the TLS client obtains the materials it will use to verify the server's proof by retrieving a JSON Web Key (JWK) set [JOSE-JWK] over HTTPS ([RFC2616] and [RFC2818]) from a well-known URI [RFC5785].

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

1. The TLS client performs an HTTPS GET at the source domain to the path `"/.well-known/posh.{servicedesc}.json"`. The value of `"{servicedesc}"` is application-specific; see Section 9 of this document for more details. For example, if the application protocol is some hypothetical "Foo" service, then `"{servicedesc}"` could be `"foo"`; thus if a Foo client were to use POSH to verify a Foo server for the domain `"foo.example.com"`, the HTTPS GET request would be as follows:

```
GET /.well-known/posh.foo.json HTTP/1.1
Host: foo.example.com
```

2. The source domain HTTPS server responds in one of three ways:
 - * If it possesses a PKIX certificate for the requested path, it responds as detailed in Section 4.1.
 - * If it has a reference to where the PKIX certificate can be obtained, it responds as detailed in Section 4.2.
 - * If it does not have any PKIX certificate for the requested path, it responds with a client error status code (e.g., 404).

4.1. Source Domain Possesses PKIX Certificate

If the source domain HTTPS server possesses the certificate information, it responds to the HTTPS GET with a success status code and the message body set to a JSON Web Key (JWK) set [JOSE-JWK]. The JWK set MUST contain at least one JWK object, and MUST contain an "expires" field whose value is the number of seconds after which the TLS client ought to consider the key information to be stale (further explained under Section 7).

Each included JWK object MUST possess the following information:

- o The "kty" field set to the appropriate key type used for TLS connections (e.g., "RSA" for a certificate using an RSA key).
- o The required public parameters for the key type (e.g., "n" and "e" for a certificate using an RSA key).
- o The "x5t" field set to the certificate thumbprint, as described in section 3.6 of [JOSE-JWK].

Each JWK object MUST NOT possess the private parameters for the key type (e.g., "d", "p", "q" for a certificate using an RSA key).

Each JWK object MAY possess other parameters as desired by application servers (e.g., the "x5c" field containing the entire X.509 certificate chain, as per section 3.7 of [JOSE-JWK]).

The following example illustrates the usage described above.

Example Content Response

```
HTTP/1.1 200 OK
Content-Type: application/jwk-set+json
Content-Length: 2785
```

```
{
  "keys": [
    {
      "kty": "RSA",
      "kid": "c8fb8b80-1193-11e3-b2b1-835742119fe8",
      "n": "ANxwssdcU3LbODErec3owrwUhlzjtuskAn8rAcBMRPImn5xA
        JRX-1T5g2D7MTozWWFk4TlpgzAR5slvM0tc35qAI9I0Cqk4Z
        LChQrYsWuY7a1TrnNXdusHUYc6Eq89DZaH2knTcp57wAXzJP
        IG_tpBi5F7ck9LVRvRjybix0HJ7i4YrL-GeLuSgrj04-GDcX
        Ip8oV0FMKZH-NoMfUITlWYl_JcXlD0WUAiuAnyWtD4Kh_qMJ
        U6FZuupZGHqPdc3vrXtp27LWgxzxjFa9qnOU6y53vCCJXLLI
        5sy2fCwEDzLJqh2T6UIItIzjrSUZMIsK8r2pXkroI0uYuNn3W
        y-jAzK8",
      "e": "AQAB",
      "x5t": "UpjRI_A3afKE8_AIEtZ5oIdECTY"
    }
  ],
  "expires": 604800
}
```

The "expires" value is a hint regarding the expiration of the keying materials. If no "expires" field is included, a TLS client SHOULD consider these verification materials invalid. See Section 7 for how to reconcile this "expires" field with the reference's "expires" field.

4.2. Source Domain References PKIX Certificate

If the source domain HTTPS server has a reference to the certificate information, it responds to the HTTPS GET with a JSON document. The document MUST contain a "url" field whose value is the HTTPS URL where TLS clients can obtain the actual JWK set, and MUST contain an "expires" field whose value is the number of seconds after which the TLS client ought to consider the delegation to be stale (further explained under Section 7).

Example Reference Response

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Length: 78
```

```
{
  "url": "https://hosting.example.net/.well-known/posh.foo.json",
  "expires": 86400
}
```

The client performs an HTTPS GET for the URL specified in the "url" field value. The HTTPS server for the URI to which the client has been redirected responds to the request with a JWK set. The content retrieved from the "url" location MUST NOT itself be a reference (i.e., containing a "url" fields instead of a "keys" field), in order to prevent circular delegations.

Note: The JSON document returned by the source domain HTTPS server MUST contain either a reference or a JWK-set, but MUST NOT contain both.

Note: See Section 10 for discussion about HTTPS redirects.

The "expires" value is a hint regarding the expiration of the source domain's delegation of service to the delegated domain. If no "expires" field is included, a TLS client SHOULD consider the delegation invalid. See Section 7 for guidelines about reconciling this "expires" field with the JWK-set's "expires" field.

4.3. Performing Verification

The TLS client compares the PKIX information obtained from the TLS server against each JWK object in the POSH results, until a match is found or the collection of POSH verification materials is exhausted. If none of the JWK objects match the TLS server PKIX information, the TLS client SHOULD reject the connection (the TLS client might still accept the connection if other verification schemes are successful).

The TLS client SHOULD compare the fingerprint of the PKIX certificate from the TLS server against the "x5t" field of the JWK object (note the "x5t" field is the base64url encoding of the fingerprint).

The TLS client MAY verify the certificate chain provided in the "x5c" field of the JWK object (if present), but it MUST NOT implicitly consider the final certificate in the "x5c" field to be a trust anchor itself; the TLS client only uses the end entity certificate information for verification.

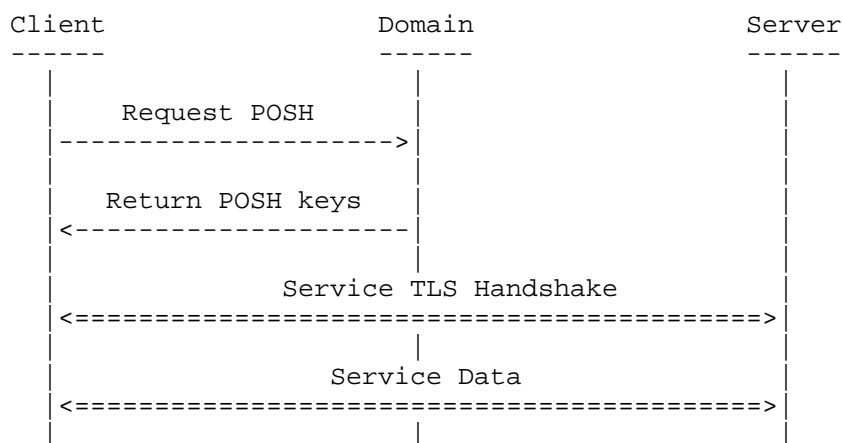
5. Secure Delegation

The delegation from the source domain to the delegated domain can be considered secure if the certificate offered by the TLS server matches the POSH certificate, regardless of how the POSH certificates are obtained.

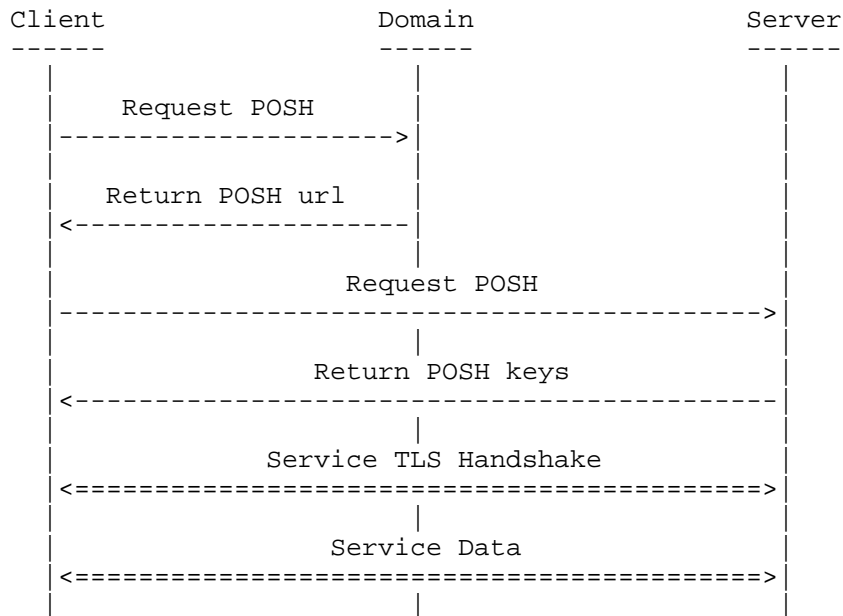
6. Order of Operations

In order for the TLS client to perform verification of reference identifiers without potentially compromising data, POSH processes MUST be complete before any application-level data is exchanged for the source domain. The TLS client SHOULD perform all POSH retrievals before opening any socket connections to the application protocol server. For application protocols that use DNS SRV, the POSH processes ideally ought to be done in parallel with resolving the SRV records and the addresses of any targets, similar to the "happy eyeballs" approach for IPv4 and IPv6 [RFC6555].

The following diagram illustrates the possession flow:



While the following diagram illustrates the reference flow:



7. Caching Results

The TLS client MUST NOT cache results (reference or JWK-set) indefinitely. If the source domain returns a reference, the TLS client MUST use the lower of the two "expires" values when determining how long to cache results (i.e., if the reference "expires" value is lower than the JWK-set "expires" value, honor the reference "expires" value). Once the TLS client considers the results stale, it SHOULD perform the entire POSH process again starting with the HTTPS GET to the source domain. The TLS client MAY use a lower value than any provided in the "expires" field(s), or not cache results at all.

The TLS client SHOULD NOT rely on HTTP caching mechanisms, instead using the expiration hints provided in the POSH reference or JWK-set documents. To that end, the HTTPS servers for source and derived domains SHOULD specify a 'Cache-Control' header indicating a very short duration (e.g., max-age=60) or "no-cache" to indicate that the response (redirect, reference, or content) is not appropriate to cache at the HTTP level.

8. Alternates and Roll-over

To indicate alternate PKIX certificates (such as when an existing

certificate will soon expire), the returned JWK set MAY contain multiple JWK objects. The JWK set SHOULD be ordered with the most relevant certificate first as determined by the application service operator (e.g., the renewed certificate), followed by the next most relevant certificate (e.g., the certificate soonest to expire). Here is an example:

```
{
  "keys":[
    {
      "kty": "RSA",
      "kid": "cfc0ca70-1193-11e3-b2b1-835742119fe8",
      "n": "AM-ktWkQ8btj_HEdAA6kOpzJGgoHNZsJmxjh_PifpgAUfQeq
MO_YBR100IdJZRzJfULyhRwn9bikCq87WToxgPWond3sH3qT
YiAcIR5S6tBbsyp6WYmwMlyuC0vLCo6SoDzdK1SvkQKM3QWk
0GFNU4l4qXYAMxaSw83i6yv5DBVbST7E92vS6Gq_4pgI26l1
0JhybZuTEVPRUCG6pTKAXQpLxmjJ5oG9M9lRP17nsuQeE7Ng
0Ap4BBn5hocojkfthwgbX4lqBMecpBAnky5jn6slmzS_rL-L
w-_8hUldaTPD9MHlHPrvcsRV5uw8wK5MB6Qyfs6wF4b0Kj2T
vYceN1E",
      "e": "AQAB",
      "x5t": "Ae0sLVtm78VT-mQXJQop-ENOM6o"
    },
    {
      "kty": "RSA",
      "kid": "dbc28570-1193-11e3-b2b1-835742119fe8",
      "n": "AM-ktWkQ8btj_HEdAA6kOpzJGgoHNZsJmxjh_PifpgAUfQeq
MO_YBR100IdJZRzJfULyhRwn9bikCq87WToxgPWond3sH3qT
YiAcIR5S6tBbsyp6WYmwMlyuC0vLCo6SoDzdK1SvkQKM3QWk
0GFNU4l4qXYAMxaSw83i6yv5DBVbST7E92vS6Gq_4pgI26l1
0JhybZuTEVPRUCG6pTKAXQpLxmjJ5oG9M9lRP17nsuQeE7Ng
0Ap4BBn5hocojkfthwgbX4lqBMecpBAnky5jn6slmzS_rL-L
w-_8hUldaTPD9MHlHPrvcsRV5uw8wK5MB6Qyfs6wF4b0Kj2T
vYceN1E",
      "e": "AQAB",
      "x5t": "lYZC2n9TBpOaUsBclEIacQTKToA"
    }
  ]
}
```

9. IANA Considerations

This document registers a well-known URI [RFC5785] for protocols that use POSH. The completed template follows.

URI suffix: posh.
Change controller: IETF
Specification document: [[this document]]
Related information: Because the "posh." string is merely a prefix, protocols that use POSH need to register particular URIs that are prefixed with the "posh." string.

Note that the registered URI is "posh." (with a trailing dot). This is merely a prefix to be placed at the front of well-known URIs [RFC5785] registered by protocols that use POSH, which themselves are responsible for the relevant registrations with the IANA. The URIs registered by such protocols SHOULD match the URI template [RFC6570] path `"/.well-known/posh.{servicedesc}.json"`; that is, begin with "posh." and end with ".json" (indicating a media type of application/json [RFC4627] or application/jwk-set+json [JOSE-JWK]).

For POSH-using protocols that rely on DNS SRV records [RFC2782], the `"{servicedesc}"` part of the well-known URI SHOULD be `"{service}.{proto}"`, where the `"{service}"` is the DNS SRV "Service" prepended by the underscore character "_" and the `"{proto}"` is the DNS SRV "Proto" also prepended by the underscore character "_". As an example, the well-known URI for XMPP server-to-server connections would be `"posh._xmpp-server._tcp.json"` since XMPP [RFC6120] registers a service name of `"xmpp-server"` and uses TCP as the underlying transport protocol.

For other POSH-using protocols, the `"{servicedesc}"` part of the well-known URI can be any unique string or identifier for the protocol, which might be a service name registered with the IANA in accordance with [RFC6335] or which might be an unregistered name. As an example, the well-known URI for the mythical "Foo" service could be `"posh.foo.json"`.

Note: As explained in [RFC5785], the IANA registration policy [RFC5226] for well-known URIs is Specification Required.

10. Security Considerations

This document supplements but does not supersede the security considerations provided in specifications for application protocols that decide to use POSH (e.g., [RFC6120] and [RFC6125] for XMPP). Specifically, the security of requests and responses sent via HTTPS depends on checking the identity of the HTTP server in accordance with [RFC2818]. Additionally, the security of POSH can benefit from other HTTP hardening protocols, such as HSTS [RFC6797] and key pinning [KEYPIN], especially if the TLS client shares some

information with a common HTTPS implementation (e.g., platform-default web browser).

Note well that POSH is used by a TLS client to obtain the public key of a TLS server to which it might connect for a particular application protocol such as IMAP or XMPP. POSH does not enable a hosted domain to transfer private keys to a hosting service via HTTPS. POSH also does not enable a TLS server to engage in certificate enrollment with a certification authority via HTTPS, as is done in Enrollment over Secure Transport [EST].

A web server at the source domain might redirect an HTTPS request to another URL. The location provided in the redirect response MUST specify an HTTPS URL. Source domains SHOULD use only temporary redirect mechanisms, such as HTTP status codes 302 (Found) and 307 (Temporary Redirect). Clients MAY treat any redirect as temporary, ignoring the specific semantics for 301 (Moved Permanently) and 308 (Permanent Redirect) [HTTP-STATUS-308]. To protect against circular references, clients MUST NOT follow an infinite number of redirects. It is RECOMMENDED that clients follow no more than 10 redirects, although applications or implementations can require that fewer redirects be followed.

11. References

11.1. Normative References

[JOSE-JWK]

Jones, M., "JSON Web Key (JWK)",
draft-ietf-jose-json-web-key-16 (work in progress),
September 2013.

[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.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

- [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.
- [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.

11.2. Informative References

- [EST] Pritikin, M., Yee, P., and D. Harkins, "Enrollment over Secure Transport", draft-ietf-pkix-est-09 (work in progress), August 2013.
- [HTTP-STATUS-308] Reschke, J., "The Hypertext Transfer Protocol (HTTP) Status Code 308 (Permanent Redirect)", draft-reschke-http-status-308-07 (work in progress), March 2012.
- [KEYPIN] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", draft-ietf-websec-key-pinning-08 (work in progress), July 2013.
- [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.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA)

Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

- [RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, April 2012.
- [RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, March 2012.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.
- [RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTPS Strict Transport Security (HSTS)", RFC 6797, November 2012.

Appendix A. Acknowledgements

Many thanks to Philipp Hancke, Joe Hildebrand, and Tobias Markmann for their implementation feedback. Thanks also to Dave Cridland, Max Pritikin, and Joe Salowey for their input on the specification.

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
Updates: 6120 (if approved)
Intended status: Standards Track
Expires: April 22, 2014

P. Saint-Andre
Cisco Systems, Inc.
October 19, 2013

Use of Transport Layer Security (TLS) in the Extensible Messaging and
Presence Protocol (XMPP)
draft-saintandre-xmpp-tls-02

Abstract

This document provides recommendations for the use of Transport Layer Security (TLS) in the Extensible Messaging and Presence Protocol (XMPP). This document updates RFC 6120.

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 April 22, 2014.

Copyright Notice

Copyright (c) 2013 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. Discussion Venue	3
4. Recommendations	3
4.1. Support for TLS	3
4.2. Protocol Versions	4
4.3. Ciphersuites	4
4.4. Public Key Length	6
4.5. Certificate Validation	6
4.6. Unauthenticated Connections	6
4.7. Server Name Indication	7
4.8. Session Resumption	7
4.9. Compression	7
4.10. Human Factors	7
5. Implementation Notes	8
6. IANA Considerations	8
7. Security Considerations	8
8. References	8
8.1. Normative References	8
8.2. Informative References	9
Appendix A. Acknowledgements	10
Author's Address	10

1. Introduction

The Extensible Messaging and Presence Protocol (XMPP) [RFC6120] (along with its precursor, the so-called "Jabber protocol") has used Transport Layer Security (TLS) [RFC5246] (along with its precursor, Secure Sockets Layer or SSL) since 1999. Both [RFC6120] and its predecessor [RFC3920] provided recommendations regarding the use of TLS in XMPP. Given the evolving threat model on the Internet today (see, for example, [I-D.trammell-perpass-ppa]), it is necessary to provide stronger recommendations (see also [I-D.sheffer-tls-bcp]). This document updates [RFC6120].

2. Terminology

Various security-related terms are to be understood in the sense defined in [RFC4949].

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. Discussion Venue

The discussion venue for this document is the mailing list of the XMPP Working Group, for which archives and subscription information can be found at <<https://www.ietf.org/mailman/listinfo/xmpp>>.

4. Recommendations

4.1. Support for TLS

Support for TLS (specifically, the XMPP profile of STARTTLS) is mandatory for XMPP implementations. If the server to which an XMPP client or peer server connects does not offer a stream feature of <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/> as described in [RFC6120], the initiating entity MUST NOT proceed with the stream negotiation and MUST instead abort the connection attempt. Although XMPP servers SHOULD include the <required/> child element to indicate that negotiation of TLS is mandatory, clients and peer servers MUST NOT depend on receiving the <required/> flag.

4.2. Protocol Versions

It is important both to stop using old, insecure versions of SSL/TLS and to start using modern, more secure versions. Therefore:

- o XMPP implementations MUST NOT negotiate SSL version 2.

Rationale: SSLv2 has serious security vulnerabilities [RFC6176].

- o XMPP implementations MUST NOT negotiate SSL version 3.

Rationale: SSLv3 [RFC6101] was an improvement over SSLv2, but did not support strong ciphersuites.

- o XMPP implementations SHOULD NOT negotiate TLS version 1.0 [RFC2246].

Rationale: TLS 1.0 (published in 1999) includes a way to downgrade the connection to SSLv3 and does not support more modern, strong ciphersuites.

- o XMPP implementations MAY negotiate TLS version 1.1 [RFC4346].

Rationale: TLS 1.1 (published in 2006) prevents downgrade attacks to SSL, but does not support certain stronger ciphersuites.

- o XMPP implementations MUST support, and prefer to negotiate, TLS version 1.2 [RFC5246].

Rationale: Several stronger ciphersuites are available only with TLS 1.2 (published in 2008).

As of the date of this writing, the latest version of TLS is 1.2. When TLS is updated to a newer version, this document will be updated to recommend support for the latest version. If this document is not updated in a timely manner, it can be assumed that support for the latest version of TLS is recommended.

4.3. Ciphersuites

It is important both to stop using old, insecure ciphersuites and to start using modern, more secure ciphersuites. Therefore:

- o XMPP implementations MUST NOT negotiate the NULL ciphersuites.

Rationale: The NULL ciphersuites offer no encryption whatsoever and thus are completely insecure.

- o XMPP implementations MUST NOT negotiate RC4 ciphersuites

Rationale: The RC4 stream cipher has a variety of cryptographic weaknesses, documented in [I-D.popov-tls-prohibiting-rc4].

- o XMPP implementations MUST NOT negotiate ciphersuites that use so-called "export-level" encryption (including algorithms with 40 bits or 56 bits of security).

Rationale: These ciphersuites are deliberately "dumbed down" and are very easy to break.

- o XMPP implementations MUST NOT negotiate ciphersuites that use algorithms that offer less than 128 bits of security (even if they advertise more bits, such as the 168-bit 3DES ciphersuites).

Rationale: Although these ciphersuites are not actively subject to breakage, their useful life is short enough that stronger ciphersuites are desirable.

- o XMPP implementations SHOULD prefer ciphersuites that use algorithms with at least 256 bits of security.

Rationale: The useful life of such ciphersuites is probably at least 3-5 years.

- o XMPP implementations MUST support, and SHOULD prefer to negotiate, ciphersuites that offer authentication, such as the "AES-GCM" family.

Rationale: Authenticated connections are better than unauthenticated connections (although, as explained under Section 4.6, unauthenticated connections are better than nothing).

- o XMPP implementations MUST support, and SHOULD prefer to negotiate, ciphersuites that offer forward secrecy, such as those in the "EDH", "DHE", and "ECDHE" families.

Rationale: Forward secrecy (sometimes called "perfect forward secrecy") prevents the recovery of information that was encrypted with older keys, thus limiting the amount of time during which attack can be successful.

Given the foregoing considerations, implementation of the following ciphersuites is RECOMMENDED:

- o TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- o TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- o TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- o TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

Unfortunately, those ciphersuites are supported only in TLS 1.2. A future version of this document might recommend ciphersuites for earlier versions of TLS.

4.4. Public Key Length

Because Diffie-Hellman keys of 1024 bits are estimated to be roughly equivalent to 80-bit symmetric keys, it is better to use longer keys for the "DH" family of ciphersuites. Unfortunately, some existing software cannot handle (or cannot easily handle) key lengths greater than 1024 bits. The most common workaround for these systems is to prefer the "ECDHE" family of ciphersuites instead of the "DH" family, then use longer keys. Key lengths of at least 2048 bits are RECOMMENDED, since they are estimated to be roughly equivalent to 112-bit symmetric keys and might be sufficient for at least the next 10 years.

Note: The foregoing recommendations are preliminary and will likely be corrected and enhanced in a future version of this document.

4.5. Certificate Validation

Both the core XMPP specification [RFC6120] and the "CertID" specification [RFC6125] provide recommendations and requirements for certificate checking. This document does not supersede those specifications.

4.6. Unauthenticated Connections

The core XMPP specification [RFC6120] states a preference for the use of TLS for encryption along with SASL [RFC4422] for authentication. In general, it is preferable for a connection to be authenticated, including proper identity checking as defined by the "CertID" specification [RFC6125]. However, given the pervasiveness of passive eavesdropping, even an unauthenticated connection might be better than an unencrypted connection (this is similar to the "better than nothing security" approach for IPsec [RFC5386]). In particular, given current deployment challenges for authenticated connections between XMPP servers [I-D.ietf-xmpp-dna], it might be reasonable for XMPP server implementations to accept unauthenticated connections when the Server Dialback protocol [XEP-0220] is used for weak identity verification; this will at least enable encryption of server-to-server connections. Unauthenticated connections include

connections negotiated using anonymous Diffie-Hellman algorithms or using self-signed certificates, among other scenarios.

4.7. Server Name Indication

Although there is no harm in supporting the TLS Server Name Indication (SNI) extension [RFC6066], this is not necessary since the same function is served in XMPP by the 'to' address of the initial stream header as explained in Section 4.7.2 of [RFC6120].

4.8. Session Resumption

If TLS session resumption is used (e.g., in concert with the XMPP Stream Management extension [XEP-0198]), care ought to be taken to do so safely. In particular, the resumption information (either session IDs [RFC5246] or session tickets [RFC5077]) needs to be authenticated and encrypted to prevent modification or eavesdropping by an attacker. Use of session IDs [RFC5246] is RECOMMENDED instead of session tickets [RFC5077], since session tickets require use of a relatively small key size and a relatively weak ciphersuite (AES_128_CBC_SHA256) that does not support forward secrecy.

4.9. Compression

XMPP is not generally subject to attacks based on TLS-layer compression (e.g., the "CRIME" attack), since it is not typically used to communicate static strings of the kind communicated over HTTP, such as "cookies" [RFC6265]. However, because XMPP also supports an application-layer compression technology [XEP-0138], implementers might wish to prefer XMPP compression over TLS compression.

4.10. Human Factors

It is RECOMMENDED that XMPP clients provide ways for end users (and that XMPP servers provide ways for administrators) to complete the following tasks:

- o Determine if a client-to-server or server-to-server connection is encrypted and authenticated.
- o Determine the version of TLS used for a client-to-server or server-to-server connection.
- o Inspect the certificate offered by an XMPP server.
- o Determine the ciphersuite used to encrypt a connection.
- o Be warned if the certificate changes for a given server.

5. Implementation Notes

Some governments enforce legislation prohibiting the export of strong cryptographic technologies. Nothing in this document ought to be taken as advice to violate such prohibitions.

6. IANA Considerations

This document requests no actions of the IANA.

7. Security Considerations

This entire document discusses security.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.
- [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [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.
- [RFC6176] Turner, S. and T. Polk, "Prohibiting Secure Sockets Layer (SSL) Version 2.0", RFC 6176, March 2011.

8.2. Informative References

- [I-D.ietf-xmpp-dna]
Saint-Andre, P. and M. Miller, "Domain Name Associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP)", draft-ietf-xmpp-dna-03 (work in progress), September 2013.
- [I-D.popov-tls-prohibiting-rc4]
Popov, A., "Prohibiting RC4 Cipher Suites", draft-popov-tls-prohibiting-rc4-01 (work in progress), October 2013.
- [I-D.sheffer-tls-bcp]
Sheffer, Y., "Recommendations for Secure Use of TLS and DTLs", draft-sheffer-tls-bcp-01 (work in progress), September 2013.
- [I-D.trammell-perpass-ppa]
Trammell, B., "The Perfect Passive Adversary: A Threat Model for the Evaluation of Protocols under Pervasive Surveillance", draft-trammell-perpass-ppa-00 (work in progress), September 2013.
- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.
- [RFC3920] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, October 2004.
- [RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.
- [RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.
- [RFC5386] Williams, N. and M. Richardson, "Better-Than-Nothing Security: An Unauthenticated Mode of IPsec", RFC 5386, November 2008.
- [RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.
- [RFC6101] Freier, A., Karlton, P., and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0", RFC 6101, August 2011.
- [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265,

April 2011.

[XEP-0138]

Hildebrand, J. and P. Saint-Andre, "Stream Compression",
XSF XEP 0138, May 2009.

[XEP-0198]

Karneges, J., Saint-Andre, P., Hildebrand, J., Forno, F.,
Cridland, D., and M. Wild, "Stream Management", XSF
XEP 0198, June 2011.

[XEP-0220]

Miller, J., Saint-Andre, P., and P. Hancke, "Server
Dialback", XSF XEP 0220, September 2013.

Appendix A. Acknowledgements

Thanks to the following individuals for their input: Thijs Alkemade,
Dave Cridland, Philipp Hancke, Olle Johansson, Steve Kille, Tobias
Markmann, Matt Miller, and Rene Treffer.

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

