Domain Name Associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP)
draft-ietf-xmpp-dna-06

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 multi-tenanted environments.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 23, 2014.

Copyright Notice

Copyright (c) 2014 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. Client-to-Server (C2S) DNA ................................. 3
   3.1. C2S Flow ............................................. 3
   3.2. C2S Description ..................................... 4
4. Server-to-Server (S2S) DNA ................................. 5
   4.1. S2S Flow Chart ....................................... 5
   4.2. A Simple S2S Scenario ............................... 7
   4.3. One-Way Authentication ............................... 8
   4.4. Piggybacking ......................................... 9
      4.4.1. Assertion ...................................... 9
      4.4.2. Supposition .................................... 10
5. Alternative Proofotypes ...................................... 12
   5.1. DANE .................................................... 12
   5.2. POSH ................................................... 12
7. Proofotype Model ............................................. 14
8. IANA Considerations .......................................... 14
   8.1. Well-Known URI for xmpp-client Service .............. 14
   8.2. Well-Known URI for xmpp-server Service .............. 14
9. Security Considerations ....................................... 15
10. References .................................................. 15
   10.1. Normative References ................................ 15
   10.2. Informative References ............................... 16
Appendix A. Acknowledgements .................................... 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 prooftypes can be defined

2. Provides a basis for modernizing some prooftypes 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 (see [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].

3. Client-to-Server (C2S) DNA

The client-to-server case is much simpler than the server-to-server case (the client does not assert a domain name, only the server’s domain name needs to be verified, etc.). Therefore we describe it first to help the reader understand domain name associations in XMPP.

3.1. C2S Flow

The following flow chart illustrates the protocol flow for establishing a domain name association for an XML stream from a client to a server.
3.2. C2S Description

The simplified order of events (see [RFC6120] for details) in establishing an XML stream from a client (user@a.example) to a server (a.example) is as follows:

1. The client resolves the DNS domain name a.example.
2. The client opens a TCP connection to the resolved IP address.
3. The client sends an initial stream header to the server.
   <stream:stream from='user@a.example' to='a.example'>
4. The server sends a response stream header to the client, asserting that it is a.example:
   <stream:stream from='a.example' to='user@a.example'>
5. The parties attempt TLS negotiation, during which the XMPP server (acting as a TLS server) presents a PKIX certificate proving that it is a.example.
6. The client checks the PKIX certificate that the server provided; if the proof is consistent with the XMPP profile of the matching rules from [RFC6125], the client accepts that there is a strong domain name association between its stream to the server and the DNS domain name of the server.
4. Server-to-Server (S2S) DNA

The server-to-server case is much more complex than the client-to-server case, and involves checking of domain name associations in both directions along with other "wrinkles" described in the following sections.

4.1. S2S 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.

(Section 4.2: A Simple Scenario)

DNS RESOLUTION ETC.

+-----------------------------+
| STREAM HEADERS              |
| A: <stream from='a.example' to='b.example'> |
| B: <stream from='b.example' to='a.example'> |
+-----------------------------+

+-----------------------------+
| TLS NEGOTIATION             |
| B: Server Certificate       |
| [B: Certificate Request]    |
| [A: Client Certificate]     |
+-----------------------------+

(A establishes DNA for b.example)

+-----------------------------+
| AUTHENTICATION              |
| (client certificate?) ----+
| yes no 
| v
| SASL EXTERNAL               |
| (mutual auth!)              |
+-----------------------------+

| B needs to auth A |
(Section 4.3: One-Way Authentication)

DNS RESOLUTION ETC.

+-------------STREAM HEADERS--------------------+
|                                               |
|  B: <stream from='b.example' to='a.example'>  |
|  A: <stream from='a.example' to='b.example'>  |
+-----------------------------------------------+

+------------TLS NEGOTIATION---------------------+
|                                               |
|  A: Server Certificate                          |
+-----------------------------------------------+

(B establishes DNA for a.example)

(Section 4.4.1: Piggybacking Assertion)

+---------------DIALBACK IDENTITY ASSERTION--------+
|                                               |
|  B: <db:result from='c.example' to='a.example'/>|
+-----------------------------------------------+

+------------DNA DANCE AS ABOVE------------------+
| DNS RESOLUTION, STREAM HEADERS, |
| TLS NEGOTIATION, AUTHENTICATION |
+-----------------------------------------------+

+---------------DIALBACK IDENTITY VERIFICATION-----+
|                                               |
|  A: <db:result from='a.example' to='c.example' |
|     type='valid'/>                             |
+-----------------------------------------------+
(Section 4.4.2: Piggybacking Supposition)

| +-----------SUBSEQUENT CONNECTION-----------------+
| B: <stream from='c.example' to='rooms.a.example'> |
| A: <stream from='rooms.a.example' to='c.example'> |
| +-----------------------------------------------+
| +-----------DNA DANCE AS ABOVE-------------------+
| DNS RESOLUTION, STREAM HEADERS, TLS NEGOTIATION, AUTHENTICATION |
| +-----------------------------------------------+
| +-----------DIALBACK OPTIMIZATION-----------------+
| B: <db:result from='c.example' to='rooms.a.example'/> |
| B: <db:result from='rooms.a.example' to='c.example' type='valid'/> |
| +-----------------------------------------------+

4.2. A Simple S2S 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:
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:

- Server 1 presents a PKIX certificate during TLS negotiation, which enables the parties to complete mutual authentication.
- 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).
- The server administrators are able to obtain PKIX certificates in the first place.
- The server administrators are running their own XMPP servers, rather than using hosting services.

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

### 4.3. 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:
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.

4.4. Piggybacking

4.4.1. 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 an 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).

4.4.2. 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.
5. Alternative Prooftypes

The foregoing protocol flows assumed that domain name associations were proved using the standard PKI prooftype 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, prooftypes other than PKIX are desirable. As described below, two alternatives have been defined so far: DNS-Based Authentication of Named Entities (DANE) and PKIX Over Secure HTTP (POSH).

5.1. DANE

In the DANE prooftype, 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 prooftype 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:

- If there is no SRV resource record, pursue the fallback methods described in [RFC6120].

- 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]).

5.2. POSH

In the POSH prooftype, 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 PKIK 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.ietf-xmpp-posh]. For XMPP purposes, the well-known URIs [RFC5785] to be used are:

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

6. Secure Delegation and Multi-Tenancy

One common method for deploying XMPP services is multi-tenancy: 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 prooftype 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.ietf-xmpp-posh] explains how to use HTTPS redirects for secure
delegation with the POSH prooftype.

7. 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.

8. IANA Considerations

The POSH specification [I-D.ietf-xmpp-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.

8.1. 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 ]]

8.2. 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].
9. 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.ietf-xmpp-posh].

10. References

10.1. Normative References

[I-D.ietf-dane-srv]

[I-D.ietf-xmpp-posh]


10.2. Informative References


Appendix A. Acknowledgements

Thanks to Philipp Hancke for his feedback.
Authors’ Addresses

Peter Saint-Andre  
&yet  
P.O. Box 787  
Parker, CO  80134  
USA  

Email: peter@andyet.com

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

Email: mamille2@cisco.com