Network Working Group	R.L. Barnes
Internet-Draft	BBN Technologies
Intended status: Standards Track	J. Lindberg
Expires: September 15, 2011	Google
	March 14, 2011

Domain Name Assertions
draft-ietf-xmpp-dna-01

Abstract

The current authentication process in XMPP requires the XMPP server for a domain to present a certificate that contains that domain's name. This requirement causes several problems in scenarios where XMPP services have been delegated from one domain to another, especially when one domain provides XMPP services for many domains. This document describes an extension to the XMPP authentication process that allows domains to be securely delegated, simplifying authorization in delegation scenarios.

Requirements Language

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 RFC 2119 *[RFC2119]*.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

When connecting two XMPP services to provide inter-domain communication, it is important for a service to be able to determine the identity of a peer service to prevent traffic spoofing. The Jabber communities first approach to identity verification was the Server Dialback protocol. When the Jabber protocols were formalized by the XMPP working group of the IETF 2002-04, support for strong identity verification using TLS + SASL was added.
Server Dialback [XEP-0220] provides weak identity verification and makes it more difficult to spoof hostnames of servers XMPP network. However, it does not provide authentication between servers and is not a security mechanism. It is susceptible to DNS poisoning attacks (unless DNSSEC is used) and cannot protect against attackers capable of hijacking the IP address of a remote service.
TLS + SASL provides strong identity verification but requires a obtaining a digital certificate by a trusted CA (or the XMPP Intermediate Certification Authority) and using it in the XMPP service,

which may be hosted by a 3rd party. This solution does not allow for multiplexing traffic for multiple domain pairs over a connection, possibly requiring a large number of connections between two hosting providers.
Server Dialback can be used with TLS. When STARTTLS negotiation succeeds with a peer service but the peer's certificate cannot be used to establish the peer's identity, the remote domain may use on Server Dialback for (weak) identity verification. One use case can be an originating server that wish to use TLS for encryption, but only can present a self signed certificate.
In practice, many XMPP server deployments rely on Server Dialback and either do not support XMPP 1.0 or do not offer negotiation of TLS + SASL.
This goal of this document is to describe secure authentication using a hosting provide TLS certificate from a trusted CA, combined with a dialback mechanism providing secure delegation based on DNS record delgation verified using DNSSEC.

2. Terminology

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 RFC 2119 [RFC2119].
We will refer to four different types of domains in this document:

- *Sender domain: The domain that initially sends out an XMPP message

- *Target domain: The ultimate destination of an XMPP message

- *Originating domain: The originating domain of a particular server-to-server connection

- *Receiving domain: The receiving domain of a particular server-to-server connection

In outsourcing scenarios, the sending and receiving domains are outsourced to the originating and receiving domains, respectively.

3. Protocol Overview

Consider a scenario in which the domain sender.tld has outsourced XMPP services to originating.tld, and target.tld has outsourced to receiving.tld. The particular hosts providing services are xmpp1.originating.tld and xmpp1.receiving.tld. Users romeo@sender.tld and juliet@target.tld maintain client-to-server connections to these servers.

```
romeo@sender.tld -- xmpp1.originating.tld
                            .
                            .
                    xmpp1.receiving.tld -- juliet@target.tld
```

When Romeo wants to send a message to Juliet, Provider A's server will have to establish a server-to-server connection to Provider B's server. Since they are both acting on behalf of other domains, however, each side will have to verify that the other is authorized to act in that role.

The first step is to provision records that can be used to verify these delegations. In order for XMPP to work, when the hosting relationships are set up, sender.tld and target.tld have to provision SRV records pointing to their providers' servers. To make this delegation secure, they sign these records using DNSSEC [RFC4033]. On the XMPP servers themselves, the originating and receiving domains provision certificates that can be used to authenticate the names xmpp1.originating.tld and xmpp1.receiving.tld.

When Romeo wants to send a stanza to Juliet, he will first send it to his server, xmpp1.originating.tld. Seeing that the 'to' domain of the stanza is target.tld, the server will retrieve the SRV records for _xmpp-server._tcp.target.tld, plus any associated DNSSEC records [RFC4034].

```
_xmpp-server._tcp.target.tld. 400 IN SRV
            20 0 5269 xmpp1.receiving.tld

_xmpp-server._tcp.target.tld. 400 IN RRSIG
            SRV 5 3 400 20030322173103 (
              20030220173103 2642 _tcp.target.tld.
              oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
              PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
              B9wfuh3DTJXUAfI/M0zmO/zz8bW0Rznl8O3t
              GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG
              J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

If there are no DNSSEC records, or if the DNSSEC records do not validate, then there is nothing new to do; the server simply connects to the remote domain using normal XMPP procedures. If there is a valid DNSSEC signature on the SRV record, then the server knows that he can allow the remote server to authenticate as either target.tld or xmpp1.receiving.tld.

Once the TLS connection is established, the two sides negotiate a single bidirectional stream to run over it, using their own names:

```
I: <?xml version='1.0'?>
   <stream:stream
       from='xmpp1.originating.tld'
       to='xmpp1.receiving.tld'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'>

R: <?xml version='1.0'?>
   <stream:stream
       from='xmpp1.receiving.tld'
       id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
       to='xmpp1.originating.tld'
       version='1.0'
       xml:lang='en'
       xmlns='jabber:server'
       xmlns:stream='http://etherx.jabber.org/streams'>

R: <stream:features>
     <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
     <bidi xmlns='urn:xmpp:bidi'/>
   </stream:features>
```

When this stream is created, it can immediately carry stanzas directly between the two servers. In order to send messages to and from other domains, the servers have to authenticate and request permission. So to send Romeo's stanza to Juliet, xmpp1.originating.tld requests permission to send from sender.tld to target.tld.

The originating server uses STARTTLS to set up a TLS connection. In the ClientHello message initiating the connection, the xmpp1.originating.tld includes a Server Name Indication extension set to xmpp1.receiving.tld [RFC4366]. The remote server xmpp1.receiving.tld responds to this request with a certificate for its own name, xmpp1.receiving.tld and requests a client certificate from the originating server. The originating server presents a certificate for its own name, xmpp1.originating.tld.

At this point, the server xmpp1.originating.tld knows that xmpp1.receiving.tld is authorized to represent either xmpp1.receiving.tld (via the certificate) or target.tld (via DNSSEC). The other server, xmpp1.receiving.tld knows only that the other server repressents xmpp1.originating.tld.

Once the two servers have authenticated their own names over TLS, they can request permission to send stanzas:

```
I: <db:result from='sender.tld' to='target.tld' />
```

Since xmpp1.receiving.tld doesn't yet know whether xmpp1.originating.tld is authorized to represent sender.tld, it has to check, using an abbreviated form of dialback. Just as the Provider A server did earlier for target.tld, the Provider B server looks up the SRV records for _xmpp-server._tcp.sender.tld and any associated DNSSEC records. If there are no DNSSEC records or the signature is not valid, then the server rejects the request to send stanzas from that domain. If the record is DNSSEC-signed, then the server checks that the server name in the SRV record is one of the names authenticated for the remote side.

```
R: <db:result type='invalid' from='sender.tld' to='target.tld' />
```

On the other hand, if the DNSSEC signature is valid, then the server can accept the request to send stanzas, and the two servers can exchange stanzas for those domains.

```
R: <db:result type='valid' from'sender.tld' to='target.tld' />

I: <!-- stanza -->
```

Now that the two servers have established this connection, they can re-used it for other stanzas and other domains. If either server finds another domain that is delegated to the other server, it can send a <db:result> requesting permission to send stanzas for that domain, and the other server will grant or deny permission after checking the delegation.
The following figure summarizes the overal process:

```
Originating                   DNS                     Receiving
  Server                    Server                     Server
-----------               ---------                   --------
    |                          |                          |
    |  Lookup _xmpp-server     |                          |
    |  DNS SRV record for      |                          |
    |  target.tld to find      |                          |
    |  delegation of service   |                          |
    |  to Receiving Server.    |                          |
    |  Verify zone signature   |                          |
    | -----------------------> |                          |
    |                          |                          |
    |  'Receiving Server'      |                          |
    | <----------------------- |                          |
    |                          |                          |
    |                                                     |
    |                                                     |
    |  <stream from='originating.tld' to='receiving.tld'> |
    | --------------------------------------------------> |
    |                                                     |
    |  <stream from='receiving.tld' to='originating.tld'> |
    | <-------------------------------------------------- |
    |                                                     |
    |  <features><starttls></features>                    |
    | <-------------------------------------------------- |
    |                                                     |
    |  <starttls/>                                        |
    | --------------------------------------------------> |
    |                                                     |
    |  <proceed/>                                         |
    | <-------------------------------------------------- |
    |                                                     |
    |                                                     |
    | <====================== TLS ======================> |
    |                                                     |
    |                                                     |
    |  <stream from='originating.tld' to='receiving.tld'> |
    | --------------------------------------------------> |
    |                                                     |
    |  <stream from='receiving.tld' to='originating.tld'> |
    | <-------------------------------------------------- |
    |                                                     |
    |  <features><bidi></features>                        |
    | <-------------------------------------------------- |
    |                                                     |
    |  <db:result from='sender.tld' to='target.tld'/>     |
    | --------------------------------------------------> |
    |                                                     |
    | ...                                                 |
```

4. Connection Model

The core challenge for managing inter-server connections is the multiplexing of stanzas for multiple domains onto a single transport-layer connection. There are two key pieces of state associated with this multiplexing: A list of domain names that have been authenticated for use on a connection, and a table binding pairs of domains that are authorized for a connection.
First table that a server maintains is a connection table. Each entry in this table contains a connection and a set of domain names. The domain names represent the set of names for which the remote server has been authenticated, according to the procedures described in Section Section 5. This set of domain names constrains the set of domain pairs that can be bound to this channel; the remote server cannot ask to transmit stanzas for an unauthenticated domain name.

Connection	Server Domain Names	Delegated Domain Names
XXX	xmpp1.provider.com	capulet.example
YYY	xmpp2.provider.com	capulet.example
AAA	paris.example	paris.example

To determine how to handle incoming and outgoing stanzas, each server maintains a channel binding table. Each row in the binding table contains a "local" domain name, a "remote" domain name, and an ordered list of connections. The identifier for a connection is the stream ID for the single XMPP stream that it carries.

Local	Remote	Connections
montague.example	capulet.example	XXX, YYY
laurence.example	capulet.example	AAA
laurence.example	paris.example	YYY, AAA

The binding table acts as a routing table for outgoing stanzas and a filter for incoming stanzas. When the server wishes to send a stanza, it looks in the binding table for a row that has the 'from' domain as the local domain and the 'to' domain as the remote domain. If there is such a in the binding table, then the server MUST transmit the on the first connection in the connection list. Thus, in the above example, a stanza from montague.example to capulet.example would be routed on channel XXX.

In the same way, when a server receives a stanza over a connection from a remote server, it looks up the relevant entry in the binding table, this time using the 'to' domain as the local domain and the 'from' domain as the remote domain. If the server finds a binding table entry and the connection over which the stanza arrived is listed in the entry, then it accepts the stanza. Otherwise, it MUST discard the stanza and return a stanza error <invalid-connection/>. In the above example, a stanza from capulet.example to escalus.example would be accepted on connections AAA and BBB, but no others.
When a connection is opened (and at some points thereafter), entries in the name table are established using the processes in Section Section 5. Once a connection is open, binding table entries are added or removed using the processes in Section Section 6. When a connection is closed, both servers MUST delete its entry in the name table and remove it from all entries in the binding table.

5. Channel Establishment and Authentication

When a server wants to send a stanza and doesn't have an entry in the connection table for the destination domain, it sets one up. The first step is to establish a connection to a server for the destination domain, and validate that the server is authorized to represent the destination domain.
The originating server MUST take the following steps to establish a secure connection to the server for example.com:

1. Retrieve SRV records for XMPP services for example.com [I-D.ietf-xmpp-3920bis].

2. Verify that the SRV records have been signed using DNSSEC [RFC4033]. The originating server may either retrieve DNSSEC records directly or rely on a validating resolver. If the SRV records are not secured with DNSSEC, then the connection fails.

3. If there is already a connection in the connection table that has the target of any SRV record in its "server names" list, then this process terminates and the server attempts to use that connection (See Section Section 6)

4. If there is no existing connection that matches, establish a TCP connection to any of the servers listed in an SRV record and negotiate an XMPP stream with the following parameters:

   *'from' domain: The originating server's name

   *'to' domain: The receiving server's name from the SRV record

   *[[TODO: Add a stream feature to indicate support for this extension]]

5. Upgrade the connection to TLS using STARTTLS, using a cipher suite that requires the server to present an X.509 certificate.

6. Verify that the certificate is valid and chains to a local trust anchor. If the certificate is invalid, the connection fails.

7. Construct a list of all names that the certificate presents [I-D.saintandre-tls-server-id-check].

8. Verify that the target name in the SRV record is one of the names in the certificate. If the target name is not found in the list of names from the certificate, then the connection fails.

A server receiving such a connection MUST perform the following steps:

1. Accept the TCP connection from the remote side and accept the stream negotiation using server names.

2. In the TLS negotiation, require a client certificate from the remote side.

3. Verify that the remote server name in the stream matches the client certificate [I-D.saintandre-tls-server-id-check]. If the certificate does not match, the TLS negotiation fails, and the server MAY terminate the TCP connection.

If this process establishes a new connection, then the originating server knows that it has established a connection to a server that legitimately represents example.com. It should thus initialize a row in the connection table for this connection:

*Server names: The list of names in the server's certificate

*Delegated names: example.com

If the process terminated at Step 3, then the server simply updates the connection table entry to add example.com to the list of delegated names. In either case, the row for a connection is removed from the connection table when the connection is closed.
In order for this process to work, the domain owner and the hosting provider need to publish information that other XMPP entities can use to verify the delegation. XMPP services are delegated via SRV records (see Section 3.2.1 of [I-D.ietf-xmpp-3920bis]), so in order for the delegation to be secure, the domain owner MUST sign these records with DNSSEC. In other words, if the delegated domain is example.com, then the zone _xmpp-server._tcp.example.com MUST be signed. Each server that acts for a domain MUST be provisioned with a certificate that contains the target name used by SRV records.

The server on the receiving end of the TLS connection MUST request a client certificate from the originating server during the TLS handshake, and the originating server MUST provide a client certificate. The receiving server can then also initialize an entry in its connection table to which delegated names can be added later:

*Server names: The list of names from the client certificate (from the originating server), if present. Otherwise, empty.

*Delgated names: Empty.

Once the two servers have established a TLS connection, they MUST set up an XMPP stream that will be used for domains that they represent. This process follows the normal stream initiation procedure [I-D.ietf-xmpp-3920bis], except that the 'to' and 'from' domains MUST be set to the names of the servers themselves: The originating server sends a <stream> stanza with the 'from' domain set to a name for itself that is contained in its client certificate, and the 'to' domain set to the server name used in the SRV record for this connection. If stream negotiation fails, then the connection fails. If it succeeds, then both sides MUST set the connection identifier in the connection table to be the stream ID for the negotiated stream.
Since server-to-server connections are by default directional, it is RECOMMENDED that servers also request the <bidi> stream feature to enable bidirectional flows on this connection [XEP-0288].

```
Originating                   DNS                     Receiving
  Server                    Server                     Server
-----------               ---------                   --------
    |                          |                          |
    |  Lookup _xmpp-server     |                          |
    |  DNS SRV record for      |                          |
    |  target.tld to find      |                          |
    |  delegation of service   |                          |
    |  to Receiving Server.    |                          |
    |  Verify zone signature   |                          |
    | -----------------------> |                          |
    |                          |                          |
    |  'Receiving Server'      |                          |
    | <----------------------- |                          |
    |                          |                          |
    |                                                     |
    |                                                     |
    |  <stream from='originating.tld' to='receiving.tld'> |
    | --------------------------------------------------> |
    |                                                     |
    |  <stream from='receiving.tld' to='originating.tld'> |
    | <-------------------------------------------------- |
    |                                                     |
    |  <features><starttls></features>                    |
    | <-------------------------------------------------- |
    |                                                     |
    |  <starttls/>                                        |
    | --------------------------------------------------> |
    |                                                     |
    |  <proceed/>                                         |
    | <-------------------------------------------------- |
    |                                                     |
    |                                                     |
    | <====================== TLS ======================> |
    |                                                     |
    |                                                     |
    |  <stream from='originating.tld' to='receiving.tld'> |
    | --------------------------------------------------> |
    |                                                     |
    |  <stream from='receiving.tld' to='originating.tld'> |
    | <-------------------------------------------------- |
    |                                                     |
    |  <features><bidi></features>                        |
    | <-------------------------------------------------- |
```

6. Authorizing XMPP Stanzas

Before sending traffic from a Sender Domain to a Target Domain using an established connection, the originating server MUST request permission to do so, and wait until it has received authorization from the remote service. A service receiving traffic MUST verify that the Sender and Target domain pair has been authorized on the connection being used. An originating server MUST go through the following steps to reqeust authorization to send traffic from a Sender Domain to a Target Domain:

1. Send a <db:result/> [XEP-0220] element with Sender Domain as 'from' and Target Domain as 'to'. The server may also include a Dialback Key as part of the element's character data, to support legacy deployments.

2. Wait for remote service to respond with a <db:result> with Target Domain as 'from', Sender Domain as 'to' and 'type' attribute that is either 'valid' or 'invalid'. In case of 'invalid', the originating server SHOULD examine the error cause and take appropriate action and MAY retry requesting authorization on the same connection in the future.

3. If response 'type' was 'valid', the originating server updates its binding table to indicate that Sender Domain (Local) and Target Domain (Remote) is authorized in the sending direction for the connection used.

4. Originating server proceeds with sending traffic from Sender Domain to Target Domain.

Upon receiving a <db:result/> stanza, the receiving server MUST take following steps:

1. Verify that the receiving direction is supported for this connection. If not, fail by disconnecting the stream. (By default, connections are one-way)

2. Verify that domain in to-attribute is hosted by the service. If not, fail and respond with an <item-not-found/> error.

3. Verify that domain in from-attribute delegates hosting of their XMPP to the remote Server Domain Name by looking up SRV and verifying that the zone is signed. If not, fail with a <not-authorized/> error. Note: a service MAY accept a less secure delegation mechanism such a SRV records in a non signed zone, subject to local policy.

4. Once secure delegation from Sending Domain to remote Server Domain name has been verified, service adds Sending Domain to list of Delegated Domain Names in the Connection Table, and updates the Binding Table indicating that the Sending Domain (remote) is allowed to send traffic to Target Domain (local) on the connection.

5. Respond to remote service with a <db:result/> stanza with 'type' set to 'valid'.

A service may revoke authorization for a domain pair at any time by sending a <db:result> with 'type' set to invalid. Once authorization has been revoked, the remote side MUST re-aquire authorization before sending any futher traffic for the domain pair.
If a server receives a stanza for a to/from pair that it does not consider authorized, then it MUST return a <not-authorized/> error and MAY terminate the TCP connection.

```
Originating               Receiving                     DNS
  Server                    Server                     Server
-----------               ---------                   --------
    |                          |                          |
    |  <db:result              |                          |
    |     from='sender.tld'    |                          |
    |     to='target.tld'/>    |                          |
    | -----------------------> |                          |
    |                          |  Lookup _xmpp-server     |
    |                          |  DNS SRV record for      |
    |                          |  sender.tld to verify    |
    |                          |  signed delegation of    |
    |                          |  delegation of service   |
    |                          |  to Originating Server   |
    |                          | -----------------------> |
    |                          |                          |
    |                          |  Result                  |
    |                          | <----------------------- |
    |                          |
    |  <db:result              |
    |    from='target.tld'     |
    |    to='sender.tld'       |
    |    type='valid'/>        |
    | <----------------------- |
    |                          |
    |  (Traffic authorized     |
    |   from sender.tld to     |
    |   target.tld, in one     |
    |   direction.)            |
    |                          |
    |  <message                |
    |    from='r@sender.tld'   |
    |    to='j@target.tld'>    |
    |    <body>hi</body>       |
    |  </message>              |
    | -----------------------> |
```

7. Backward Compatibility

Using Server Domain Names as to/from attributes in <stream> stanzas is incompatible with XMPP services that do not support this protocol, because it was previously assumed that when receiving a connection the stream to attibute will contains an XMPP domain hosted by the receiving service. It is RECOMMENDED that if the connection fails, the service tries again using the Remote Domain as stream to-attribute.
Presenting a certificate for the Server Domain Name is incompatible with XMPP services that do not support this protocol, because those will expect the Remote Domain in the certificate. It is RECOMMENDED that if the authorization fails, the service tries again presenting the certificate for the Remote Domain. A service may also choose to fall back on a weaker identification mechanism such as Server Dialback, subject to local policy.

8. Operational Considerations

[[What names to put in certs for servers in a cluster, i.e., all of them.]]
[[Do TLS clients support multiple names in certs?]]
[[How DNSSEC validation is done can vary depending on deployment scenario.]]
[[Since SNI is used to signal support for this extension, recommended not to serve end users on the same domain as hosting services.]]
[[Load balancing thoughts, since each connection will handle a lot more traffic?]]

9. IANA Considerations

[[Register XML schema for assertions, if necessary]]
[[Define invalid-connection error element]]

10. Security Considerations

[[This document simplifies authentication and authorization of XMPP servers in certain scenarios. When used together with DNSSEC-protected delegations, it does not introduce any new security risks.]]
[[If a provider chooses to omit DNSSEC checks or]]

11. Acknowledgements

Thanks to Joe Hildebrand and Sean Turner for prompting the original work on this problem, and to Stephen Farrell for his work on initial versions of this draft.

12. References

[RFC2119]

	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC4033]	Arends, R., Austein, R., Larson, M., Massey, D. and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
[RFC4034]	Arends, R., Austein, R., Larson, M., Massey, D. and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.
[RFC4366]	Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J. and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 4366, April 2006.
[I-D.ietf-xmpp-3920bis]	Saint-Andre, P, "Extensible Messaging and Presence Protocol (XMPP): Core", Internet-Draft draft-ietf-xmpp-3920bis-22, December 2010.
[I-D.saintandre-tls-server-id-check]	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)", Internet-Draft draft-saintandre-tls-server-id-check-14, January 2011.
[XEP-0288]	Hancke, P. and D. Cridland, "Bidirectional Server-to-Server Connections", XSF XEP 0288, October 2010.
[XEP-0220]	Miller, J., Saint-Andre, P. and P. Hancke, "Server Dialback", XSF XEP 0220, March 2010.

Authors' Addresses

Richard L. Barnes Barnes BBN Technologies EMail: rbarnes@bbn.com

Jonas Lindberg Lindberg Google EMail: jonasl@google.com