Domain Name Assertions
draft-barnes-xmpp-dna-00.txt

Abstract

Many Internet applications allow service delegation via the DNS. However, in the absence of DNSSEC, these delegations are unauthenticated, so clients have to authenticate the delegate as if he were the original service. This situation causes several operational problems. This document describes a mechanism for clients to discover and validate information that authenticates DNS-based service delegations, without relying on the global deployment of DNSSEC.

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 February 20, 2011.

Copyright Notice

Copyright (c) 2010 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 .................................................. 4
3. General Procedure ............................................ 5
4. Authenticator Discovery ....................................... 5
   4.1. DNS-Based Discovery .................................. 6
   4.2. Security-Protocol Discovery ........................... 7
   4.3. Application-Layer Discovery ........................... 7
5. Authenticator Formats and Validation ......................... 8
   5.1. Attribute Certificate .................................. 8
   5.2. DNSSEC External Trust Anchor ........................ 9
   5.3. PKIX Certificate with SRVName ......................... 10
6. Acknowledgements .............................................. 11
7. Security Considerations ..................................... 11
8. IANA Considerations ......................................... 11
9. References ..................................................... 11
   9.1. Normative References ................................. 11
   9.2. Informative References ............................... 12
Author’s Address ............................................... 12
1. Introduction

Many Internet applications use the DNS for service discovery and delegation. The most long-standing example of this usage of the DNS is of course the use of MX records to discover mail servers [11]. That pattern has been re-used with modern services, having been generalized through the introduction of SRV records and the Dynamic Delegation Discovery System (DDDS) [1][2]. For example, XMPP and SIP services are found using SRV records [12][13], while LoST and HELD services use NAPTR records (i.e., they use DDDS) [14][15].

Because these records direct a client from one domain to another (an "source" name to a "target" name), they can create challenges for server authentication. When the client ultimately connects to a remote endpoint, should the client expect it to authenticate as the source or the target? The answer to this question has important operational impact, especially in situations where the source and target domains are operated by different entities.

If the client cannot verify that the DNS records involved in the delegation are authentic, then these records cannot be trusted. Since spoofed delegation records could be used to point the client to an attacker’s server (e.g., to insert a man-in-the-middle), the client must require the server to authenticate as the source domain.

For example: Let us assume that a company called Example.com wishes to offload responsibility for its corporate instant messaging service ("im.example.com") to a hosting provider called Apps.Example.Net using XMPP. The company sets up DNS service location records that point im.example.com at apps.example.net:

```plaintext
_xmpp-client._tcp.im.example.com. 90 IN SRV 0 0 5222 apps.example.net
_xmpp-server._tcp.im.example.com. 90 IN SRV 0 0 5269 apps.example.net
```

When a user juliet@example.com attempts to log in to the IM service at im.example.com, her client discovers apps.example.net and resolves that name to an IP address and port. However, Juliet wants to be sure that the connection is encrypted using Transport Layer Security [3] so her client checks the certificate offered by the XMPP service at the resolved IP address and port.

Her client expects the server identity in the certificate to be "im.example.com" (or perhaps ".example.com"). But what if the identity is, instead, "apps.example.net" or ".example.net"? Now her client will need to prompt Juliet to accept this certificate mismatch either temporarly or permanently. Because such security warnings are unnerving to end users, the owners of the company would prefer that the IM service offer a certificate with an identity of...
"im.example.com". Unfortunately, the IM server software used by the hosting provider probably needs runtime access to the private key associated with the certificate. This makes both the security personnel at Example.com and the lawyers at Apps.Hosting.Net uncomfortable.

If the delegation records in question are authenticated, then the client can verify that the service has indeed been delegated to the target domain, and can authenticate the server as the target. This authentication can be provided with DNSSEC [4], but only if the signatures on the delegation records chain back to a key that the client accepts as a trust anchor (ideally, the root key). In the current DNSSEC deployment environment, only a few domains have full chains back to the root, and there is no general agreement on trust anchors other than the root.

This document discusses an intermediate solution for authenticating DNS delegation records in situations where DNSSEC cannot be used. We define a general process that clients can use to determine whether to use the source or target domain as the identifier that a server must authenticate, then consider some specific techniques for accomplishing this general procedure and their practical trade-offs.

2. Terminology

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 [5].

This document uses the word "delegation" to encompass several different mechanisms for delegating services from one name to another using the DNS, including MX records, SRV records, and the DDDS. These mechanisms delegate services from one DNS name to another. We call the name being delegated the "source name" or "source domain" and the recipient of the delegation and the recipient of the delegation the "target" name or domain. We do not distinguish the case where the output of the delegation may be something other than a domain name (e.g., a URI), as can happen in DDDS.

We generally consider these delegations in the form of a single record; in cases where multiple delegations are chained together (as can happen in DDDS), the techniques here may be repeated.

Delegation records will be authenticated using signed data objects, which we generally refer to as "authenticators". Clients find these data objects using an "authenticator discovery" process based on the source domain whose services they are seeking.
3. General Procedure

When a client seeks to connect to a service that is located using DNS delegation records, it needs to be able to authenticate the validity of these records so that it knows whether to authenticate the located server as the source or target domain. The general process for discovering and authenticating a delegation is as follows:

1. The client retrieves the delegation record (e.g., MX, SRV, NAPTR).
2. If the delegation record is protected by DNSSEC, chaining back to one of the client’s trust anchors, then the client matches the server’s authenticated identity against the target name for the delegation.
3. The client queries for an authenticator for the delegation record.
4. If an authenticator is present, the client validates that it corresponds to the delegation record and that it is issued by a trusted authority.
5. If the authenticator is present and valid, the client matches the server’s authenticated identity against the target name for the delegation.
6. Otherwise, the client matches against the source name for the delegation.

The specific procedures for authenticator discovery and validation are discussed in Section 4 and Section 5.

Note that this procedure smoothly degrades as DNSSEC becomes more widely available. Technically speaking, DNSSEC signatures could be considered as a form of authenticator, but because of the importance of transition for this mechanism (and because of some conceptual differences from other authenticators), the procedure above accommodates with a specific "short cut".

4. Authenticator Discovery

To allow clients to authenticators delegation records, there needs to be a way for them to gain access to the signed objects or "authenticators" that attest to the delegation. There are basically three points in the connection process where an authenticator can be provided:
1. Before connection establishment, as part of the DNS

2. During connection establishment, as part of the security protocol

3. After connection establishment, as part of the application

In the below sections, we outline how authenticators can be provided in each of these cases, and discuss their relative merits.

4.1. DNS-Based Discovery

Since all of the below authenticator formats are based on digital certificates, source domain operators can publish them using DNS CERT records [6] under a well-known name related to the source name. For example, if im.example.com is delegating its XMPP services to apps.example.net, and authenticating with an X.509 certificate (to be obtained from a URL), it would provision the following records:

- `_xmpp-client._tcp.im.example.com. 90 IN SRV 0 0 5222 apps.example.net`
- `_dna._xmpp-client._tcp.im.example.com. 90 IN CERT 4 12345 5 http://example.com/im.crt`
- `_xmpp-server._tcp.im.example.com. 90 IN SRV 0 0 5269 apps.example.net`
- `_dna._xmpp-server._tcp.im.example.com. 90 IN CERT 4 12345 5 http://example.com/im.crt`

A client can then construct the name for the delegation (the source name) using any technique defined by the delegation mechanism, then append the reserved label "_dna" to find the authenticator for the delegation. (As a simplification, one might also consider simply provisioning the CERT record under the same name as the delegation record. The _dna label is useful, however, to distinguish these certificates from certificates used for other purposes.)

This discovery mechanism has the benefit of directness. Authenticators have to ultimately originate with the source domain, since that is the domain that is authoritative for the delegation. In the other two cases below, the authenticator has to be provided to the target server, who then provides it to the client, while in this case, the client can simply receive the authenticator directly from the source domain.

Likewise, this technique is very general. It is completely agnostic to the type of security protocol or application protocol being used, and requires no changes to either protocol. It also applies without modification to all the different record types that can be used for delegation.

One challenge for this technique, as for the application-layer...
discovery discussed below, is the question of how the source domain can present a certificate chain to help the client validate the authenticator. It may be possible to address this challenge by including multiple CERT records under the same domain name. These certificate can either be considered as an unordered list, leaving the client to assemble them together into a certificate chain, or as an sequence ordered by the preference values in the records.

4.2. Security-Protocol Discovery

Security protocols already have syntax for endpoints to provide authentication credentials of enough different types to cover the different authenticator formats discussed below. However, providing these authenticators as part of the establishment of a secure channel is generally not practical.

For example, at the time of session establishment, a server typically has to provide authentication credentials before it knows what identity the client is expecting, so a server that hosts many domains would need to have a single certificate that covers all possible delegations pointing to it. In addition to creating very large credentials, such credentials would need to be re-issued whenever a delegation changed. A server would thus need effectively the same credential as if it were simply authenticating the source domain of the delegation, with the same operational issues, negating any benefit to authenticating the delegation.

4.3. Application-Layer Discovery

Several applications include mechanisms for protocol endpoints to challenge one another for authentication credentials. In XMPP, for example, one endpoint can issue a <challenge> stanza indicating what sort of proof is desired, and the other endpoint can reply with a <response> stanza containing the required authenticator. HTTP uses the WWW-Authenticate and Authorization headers in a similar way.

In order to use this mechanisms to authenticate a delegated service, the client would need to provisionally accept the credentials presented by the server in the security protocol. It would also have to make sure that no other protocol interactions occur before the authenticator has been received and validated.

The main benefit of handling delegation authentication at the application layer is that protocol interactions can be very rich. Applications that incorporate the SASL framework (such as XMPP) can benefit from the security semantics that it provides [7]. The obvious challenge, however, is that each application that wishes to benefit from authenticated delegations will have to have its own
extension to carry authenticators. In addition, the requirement to provisionally accept a secure connection could impose additional complexity and resource requirements on the client.

5. Authenticator Formats and Validation

An authenticator is a signed object that attests to the validity of a given delegation by presenting a version of it that is signed by the source domain. In this section we discuss three possible forms that such an authenticator can take, and their relative merits.

In evaluating these authenticator formats, it’s important to keep in mind a few requirements. An authenticator must have two basic parts, a representation of the delegation that is being authenticated, and signature over that representation that can be verified using a private key bound to the source domain. It is important that the signature attesting to the validity be from the source domain, since the source domain is the entity making the delegation. If another entity were to attest the delegation by presenting an authenticator under its signature, then there would be a need to verify that it was authorized to do so by the source domain.

In addition to these structural requirements, there is a practical requirement that the issuance and revocation of authenticators should not be difficult using current certificate management software and practices. Some particular usability questions that will come up are whether a single authenticator can be used for multiple services, and whether an authenticator can be applied to all the different kinds of delegation records.

The basic security assumption for all of the below formats is that the source domain operator has a certificate that binds the source domain name to a public key, and that this certificate chains up to a trust anchor recognized by the client. We will refer to this certificate as the source domain’s "authentication certificate" below. In general, when we mention a name in a certificate, we allow for this name to be provided either as a Common Name or a dNSName Subject Alternative Name (the latter being preferred, but the former included for backward compatibility).

5.1. Attribute Certificate

Format: An attribute certificate using the format specified in [draft-ietf-xmpp-dna]. The delegation is encoded in the relationship between the subject and the issuer, and in the service field of the certificate. The issuer is the source, the subject is the target, and the OID in the service field specifies the service being
delegated. The signature over the delegation is the signature on the certificate.

Validation: The client validates the attribute certificate using the public key in the source domain’s authentication certificate, then validates the authentication certificate.

Revocation: No direct revocation, only expiration of the attribute cert or revocation of the authentication cert.

Evaluation: This mechanism is fairly straightforward to implement with current commercially-available domain certificates, since attribute certs are issued by end-entity certificates from a PKI. With current libraries, however, support for attribute certificates is limited relative to public-key certificates and DNSSEC signatures (the other two cryptographic techniques used in this document); both the source domain and the client would need to be able to process attribute certificates.

Using attribute certificates also imposes some limits on the re-use of certificates. Because the service being delegated is encoded in the certificate, each service delegated requires a different certificate. In addition, while this approach is well-suited for SRV records, it is not clear how it would work for other types of delegation.

5.2. DNSSEC External Trust Anchor

Format: The original domain’s authentication certificate and an RRSIG record over the delegation record or an RR set containing it, using the private key corresponding to the public key in the authentication certificate. The delegation is encoded in the delegation record, and the signature over the delegation is the RRSIG record.

Validation: The client validates that the RRSIG has a valid DNSSEC signature over the delegation record or RR set using the public key in the source domain’s authentication certificate, then validates the authentication certificate.

Revocation: Removal of the RRSIG record (or removal of the delegation record from the RR set), or revocation of the authentication certificate.

Evaluation: As with the DNS-based discovery approach described above, the use of external trust anchors has the benefit of generality. Because the delegation is not re-encoded in any way (e.g., by being transcribed into a certificate), the signature can be applied to any type of record, and no semantics are added or lost. The same key and
certificate can be used for several different RRSIGs or delegations. Issuing authenticators of this type only requires the source domain to have an end-entity certificate, not a CA certificate.

The major challenge for this approach is figuring out how it integrates with operational practices, in particular with regard to certificate management and DNS operations. Using certificates as DNSSEC external trust anchors requires that the key pair used to construct the RRSIG also be included in the source domain’s public key certificat. This should not be a problem, since the subject of a certificate can choose the key that is included. Note that this requirement doesn’t necessarily mean that the key pair needs to be used for anything other than DNSSEC; the source domain could obtain separate certificates for other purposes (e.g., HTTPS).

The other operational concern arises with regard to the requirement that the client be able to validate an RRSIG record, effectively managing the set of DNSSEC trust anchors. In particular, because of this need for local trust anchor management, the client cannot make use of any DNSSEC support in the DNS infrastructure, e.g., validating resolvers. As long as the client can retrieve the proper RRSIG record, however, the process should work. (That request can of course be routed through the normal DNS system, resolvers and all.)

5.3. PKIX Certificate with SRVName

Format of the authenticator: X.509 certificate containing an SRVName Subject Alternative Name [8][9], issued by the source domain. The delegation is encoded in the issuer and SRVName in the certificate. The source domain is encoded in both the issuer’s name and in the SRVName (which thus must both have the same name), and the target domain is encoded in the subject’s name. The SRVName also contains an indication of the service being delegated.

Validation: The client verifies that the service and names in the certificate match the service and names in the delegation, then validates the certificate following the normal X.509 validation algorithm.

Revocation: Normal X.509 revocation.

Evaluation: Using SRVName as a mechanism for authenticating delegations leads to several deployment challenges. Because the certificate needs to be issued by the source domain, the source domain will need to have a CA certificate; CA certificates are commonly much more costly than end-entity certificates. Certificates can be re-used, with a different SRVName for each service being delegated, but they clearly cannot be used for any delegation method.
that uses a record type other than SRV.

6. Acknowledgements

We would like to thank Joe Hildebrand and Sean Turner for first articulating the problem of authenticating delegated services, in the context of XMPP, and Peter Saint-Andre for helping generalize that discussion.

7. Security Considerations

This document defines a mechanism for authenticating DNS-based delegations in support of authentication based on domain names. This functionality can be provided using DNSSEC, but that requires that all the parents of the delegated domain support DNSSEC as well as the delegating domain itself. The mechanisms discussed in this document provide a transitional step that allows the authenticity of DNS records to be rooted in an alternative hierarchy, namely a hierarchy of X.509 certificates. Since this mechanism is intended to be transitional, it includes a specific provision that prevents its use when DNSSEC is available.

8. IANA Considerations

This document currently makes no request of IANA. If DNS_based discovery is used, then this document will register the label "_dna" to be used for discovering certificates.

9. References

9.1. Normative References


March 2005.


9.2. Informative References


Author’s Address

Richard Barnes
BBN Technologies
9861 Broken Land Pkwy, Suite 400
Columbia, MD  21046
USA

Phone: +1 410 290 6169
Email: rbarnes@bbn.com