DANE
Internet-Draft
Intended status: Informational
Expires: January 29, 2012

R. Barnes
BBN Technologies

July 28, 2011

# Use Cases and Requirements for DNS-based Authentication of Named Entities (DANE)
## draft-ietf-dane-use-cases-05.txt

## Abstract

Many current applications use the certificate-based authentication features in TLS to allow clients to verify that a connected server properly represents a desired domain name. Typically, this authentication has been based on PKIX certificate chains rooted in well-known CAs, but additional information can be provided via the DNS itself. This document describes a set of use cases in which the DNS and DNSSEC could be used to make assertions that support the TLS authentication process. The main focus of this document is TLS server authentication, but it also covers TLS client authentication for applications where TLS clients are identified by domain names.

## Status of this Memo

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

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

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

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

Table of Contents

## 1. Introduction

Transport-Layer Security (TLS) is used as the basis for security features in many modern Internet application service protocols to provide secure client-server connections [RFC5246]. It underlies secure HTTP and secure email [RFC2818][RFC2595][RFC3207], and provides hop-by-hop security in real-time multimedia and instant-messaging protocols [RFC3261][RFC6120].

Application service clients typically establish TLS connections to application servers identified by DNS domain names. The process of obtaining this "source" domain is application specific [RFC6125]. The name could be entered by a user or found through an automated discovery process such as an SRV or NAPTR record. After obtaining the address of the server via an A or AAAA DNS record, the client conducts a TLS handshake with the server, during which the server presents a PKIX certificate [RFC5280]. The TLS layer performs PKIX validation of the certificate, including verification that the certificate chains to one of the client's trust anchors. If this validation is successful, then the application layer determines whether the DNS name for the application service presented in the certificate matches the source domain name [RFC6125]. Typically, if the name matches, then the client proceeds with the TLS connection.

The certificate authorities (CAs) that issue PKIX certificates are asserting bindings between domain names and the public keys they certify. Application service clients are verifying these bindings and making authorization decisions -- whether to proceed with connections -- based on them.

Clients thus rely on CAs to correctly assert bindings between public keys and domain names, in the sense that the holder of the corresponding private key should be the domain holder. Today, an attacker can successfully authenticate as a given application service domain if he can obtain a "mis-issued" ciertificate from one of the widely-used CAs -- a certificate containing the victim application service's domain name and a public key whose corresponding private key is held by the attacker. If the attacker can additionally insert himself as a man in the middle between an client and server (e.g., through DNS cache poisoning of an A or AAAA record), then the attacker can convince the client that a server of the attacker's choice legitimately represents the victim's application service.

With the advent of DNSSEC [RFC4033], it is now possible for DNS name resolution to provide its information securely, in the sense that clients can verify that DNS information was provided by the domain holder and not tampered with in transit. The goal of technologies for DNS-based Authentication of Named Entities (DANE) is to use the

DNS and DNSSEC to provide additional information about the cryptographic credentials associated with a domain, so that clients can use this information to increase the level of assurance they receive from the TLS handshake process. This document describes a set of use cases that capture specific goals for using the DNS in this way, and a set of requirements that the ultimate DANE mechanism should satisfy.

Finally, it should be noted that although this document will frequently use HTTPS as an example application service, DANE is intended to apply equally to all applications that make use of TLS to connect to application services named by domain names.

## 2. Definitions

This document also makes use of standard PKIX, DNSSEC, and TLS terminology. See RFC 5280 [RFC5280], RFC 4033 [RFC4033], and RFC 5246 [RFC5246], respectively, for these terms. In addition, terms related to TLS-protected application services and DNS names are taken from RFC 6125 [RFC6125].

Note in particular that the term "server" in this document refers to the server role in TLS, rather than to a host. Multiple servers of this type may be co-located on a single physical host, using different ports, and each of these can use different certificates.

## 3. Use Cases

In this section, we describe the major use cases that the DANE mechanism should support. This list is not intended to represent all possible ways that the DNS can be used to support TLS authentication. Rather it represents the specific cases that comprise the initial goals for DANE.

In the below use cases, we will refer to the following dramatis personae:

Alice: The operator of a TLS-protected application service on the host alice.example.com, and administrator of the corresponding DNS zone.

Bob: A client connecting to alice.example.com

Charlie: A well-known CA that issues certificates with domain names as identifiers

Oscar: An outsourcing provider that operates TLS-protected application services on behalf of customers

Trent: A CA that issues certificates with domain names as identifiers, but is not generally well-known.

These use cases are framed in terms of adding verification steps to TLS server identity checking on the part of application service clients. In application services where the clients are also identified by domain names (e.g., XMPP server-to-server connections), the same considerations and use cases are applicable to the application server's checking of identities in TLS client certificates.

### 3.1. CA Constraints

Alice runs a website on alice.example.com and has obtained a certificate from the well-known CA Charlie. She is concerned that other well-known CAs might issue certificates for alice.example.com without her authorization, which clients would accept. Alice would like to provide a mechanism for visitors to her site to know that they should expect alice.example.com to use a certificate issued under the CA that she uses (Charlie) and not another CA. That is, Alice is recommending that the client verify that there is a valid certificate chain from the server certificate to Charlie before accepting the server certificate. (For example, in the TLS handshake, the server might include Charlie's certificate in the server Certificate message's certificate_list structure [RFC5246]).

When Bob connects to alice.example.com, he uses this mechanism to verify that that the certificate presented by the server was issued under the proper CA, Charlie. Bob also performs the normal PKIX validation procedure for this certificate, in particular verifying that the certificate chains to a trust anchor (possibly Charlie's CA, if Bob accepts Charlie's CA as a trust anchor).

Alice may wish to provide similar information to an external CA operator Charlie. Prior to issuing a certificate for alice.example.com to someone claiming to be Alice, Charlie needs to verify that Alice is actually requesting a certificate. Alice could indicate her preferred CA using DANE to CAs as well as relying parties. Charlie could then check to see whether Alice said that her certificates should be issued by Charlie or another CA. Note that this check does not guarantee that the precise entity requesting a certification from Charlie actually represents Alice, only that Alice

has authorized Charlie to issue certificates for her domain to properly authorized individuals.

In principle, DANE information expressing CA constraints can be presented with or without DNSSEC protection. Presenting DANE information without DNSSEC protection does not introduce any new vulnerabilities, but neither does it add much assurance. Deletion of records removes the protection provided by this constraint, but the client is still protected by CA practices (as now). Injected or modified false records are not useful unless the attacker can also obtain a certificate for the target domain. Thus, In the worst case, tampering with these constraints increases the risk of false authentication to the level that is now standard.

Using DANE information for CA constraints without DNSSEC provides a very small incremental security feature. Many common attacks against TLS connections already require the attacker to inject false A or AAAA records in order to steer the victim client to the attacker's server. An attacker that can already inject false DNS records can also provide fake DANE information (without DNSSEC) by simply spoofing the additional records required to carry the DANE information.

Injected or modified false DANE information of this type can be used for denial of service, even if the attacker does not have a certificate for the target domain. If an attacker can modify DNS responses that a target host receives, however, there are already much simpler ways of denying service, such as providing a false A or AAAA record. In this case, DNSSEC is not helpful, since an attacker could still case a denial of service by blocking all DNS responses for the target domain.

Continuing to require PKIX validation also limits the degree to which DNS operators (as distinct from the holders of domains) can interfere with TLS authentication through this mechanism. As above, even if a DNS operator falsifies DANE records, it cannot masquerade as the target server unless it can also obtain a certificate for the target domain.

## 3.2. Service Certificate Constraints

Alice runs a website on alice.example.com and has obtained a certificate from the well-known CA Charlie. She is concerned about additional, unauthorized certificates being issued by Charlie as well as by other CAs. She would like to provide a way for visitors to her site to know that they should expect alice.example.com to present a specific certificate. In TLS terms, Alice is letting Bob know that this specific certificate must be the first certificate in the server

Certificate message's certificate_list structure [RFC5246].

When Bob connects to alice.example.com, he uses this mechanism to verify that that the certificate presented by the server is the correct certificate. Bob also performs the normal PKIX validation procedure for this certificate, in particular verifying that the certificate chains to a trust anchor.

The security implications for this case are the same as for the "CA Constraints" case above.

### 3.3. Trust Anchor Assertion and Domain-Issued Certificates

Alice would like to be able to generate and use certificates for her website on alice.example.com without involving an external CA at all. Alice can generate her own certificates today, making self-signed certificates and possibly certificates subordinate to those certificates. When Bob receives such a certificate in a TLS handshake, however, he doesn't automatically have a way to verify that the issuer of the certificate is actually Alice, because he doesn't necessarily possess Alice's corresponding trust anchor. This concerns him because an attacker could present a different certificate and perform a man in the middle attack. Bob would like to protect against this.

Alice would thus like to publish information so that visitors to her site can know that the certificates presented by her application services are legitimately hers. When Bob connects to alice.example.com, he uses this information to verify that the certificate presented by the server has been issued by Alice. Since Bob can bind certificates to Alice in this way, he can use Alice's CA as a trust anchor for purposes of validating certificates for alice.example.com. Alice can additionally recommend that clients accept only her certificates using the CA constraints described above.

As in Section Section 3.1 above, Alice may wish to represent this information to potential third-party CAs (Charlie) as well as to relying parties (Bob). Since publishing a certificate in a DANE record of this form authorizes the holder of the corresponding private key to represent alice.example.com, a CA that has received a request to issue a certificate from alice.example.com could use the DANE information to verify the requestor's authorization to receive a certificate for that domain. For example, a CA might choose to issue a certificate for a given domain name and public key only when the holder of the domain name has provisioned DANE information with a certificate containing the public key.

Note that this use case is functionally equivalent to the case where Alice doesn't issue her own certificates, but uses Trent's CA, which is not well-known. In this case, Alice would be advising Bob that he should treat Trent as a trust anchor for purposes of validating Alice's certificates, rather than a CA operated by Alice herself. Bob would thus need a way to securely obtain Trent's trust anchor information, namely through DANE information.

Alice's advertising of trust anchor material in this way does not guarantee that Bob will accept the advertised trust anchor. For example, Bob might have out-of-band information (such as a pre-existing local policy) that indicates that the CA advertised by Alice (Trent's CA) is not trustworthy, which would lead him to decide not to accept Trent as a TA, and thus to reject Alice's certificate if it is issued under Trent's CA.

Providing trust anchor material in this way clearly requires DNSSEC, since corrupted or injected records could be used by an attacker to cause clients to trust an attacker's certificate (assuming that the attacker's certificate is not rejected by some other local policy). Deleted records will only result in connection failure and denial of service, although this could result in clients re-connecting without TLS (a downgrade attack), depending on the application. Therefore, in order for this use case to be safe, applications must forbid clients from falling back to unsecured channels when records appear to have been deleted (e.g., when a missing record has no NSEC or NSEC3 record).

By the same token, this use case puts the most power in the hands of DNS operators. Since the operator of the appropriate DNS zone has de facto control over the content and signing of the zone, he can create false DANE records that bind a malicious party's certificate to a domain. This risk is especially important to keep in mind in cases where the operator of a DNS zone is a different entity than the holder of the domain, as in DNS hosting/outsourcing arrangements, since in these cases the DNS operator might be able to make changes to a domain that are not authorized by the holder of the domain.

It should be noted that DNS operators already have the ability to obtain certificates for domains under their control, under certain CA policies. In the current system, CAs need to verify that an entity requesting a certificate for a domain is actually the legitimate holder of that domain. Typically this is done using information published about that domain, such as WHOIS email addresses or special records inserted into a domain. By manipulating these values, it is possible for DNS operators to obtain certificates from some well-known certificate authorities today without authorization from the true domain holder.

### 3.4. Delegated Services

In addition to guarding against CA mis-issue, CA constraints and certificate constraints can also be used to constrain the set of certificates that can be used by an outsourcing provider. Suppose that Oscar operates alice.example.com on behalf of Alice. In particular, Oscar then has de facto control over what certificates to present in TLS handshakes for alice.example.com. In such cases, there are few ways that DNS-based information about TLS certificates could be configured, for example:

1. Alice has the A/AAAA records in her DNS and can sign them along with the DANE record, but Oscar and Alice now need to have tight coordination if the addresses and/or the certificates change.

2. Alice refers to Oscar's DNS by delegating a sub-domain name to Oscar, and has no control over the A/AAAA, DANE or any other pieces under Oscar's control.

3. Alice can put DANE records into her DNS server, but delegate the address records to Oscar's DNS server. This means that Alice can control the usage of certificates but Oscar is free to move the servers around as needed. The only coordination needed is when the certificates change, and then it would depend on how the DANE record is set up (i.e. a CA or an end entity certificate pointer).

Which of these deployment patterns is used in a given deployment will determine what sort of constraints can be expressed by which actors. In cases where Alice controls DANE records (1 and 3), she can use CA and certificate constraints to control what certificates Oscar presents for Alice's application services. For instance, Alice might require Oscar to use certificates under a given set of CAs. This control, however, requires that Alice update DANE records when Oscar needs to change certificates. Cases where Oscar controls DANE records allow Oscar to maintain more autonomy from Alice, but by the same token, Alice cannot enforce any requirements on the certificates that Oscar presents in TLS handshakes.

## 4. Other Requirements

In addition to supporting the above use cases, the DANE mechanism must satisfy several lower-level operational and protocol requirements and goals.

Multiple Ports: DANE should be able to support multiple application services with different credentials on the same named host, distinguished by port number.

No Downgrade: An attacker who can tamper with DNS responses must not be able to make a DANE-compliant client treat a site that has deployed DANE and DNSSEC like a site that has deployed neither.

Encapsulation: If there is DANE information for the name alice.example.com, it must only affect application services hosted at alice.example.com.

Predictability: Client behavior in response to DANE information must be defined in the DANE specification as precisely as possible, especially for cases where DANE information might conflict with PKIX information.

Opportunistic Security The DANE mechanism must allow a client to determine whether DANE information is available for a site, so that a client can provide the highest level of security possible for a given application service. Clients that do not support DANE should continue to work as specified, regardless of whether DANE information is present or not.

Combination: The DANE mechanism must allow multiple DANE statements of the above forms to be combined. For example, a domain holder should be able to specify that clients should accept a particular certificate (Section Section 3.2) as well as any certificate issued by its own CA (Section Section 3.3). The precise types of combination allowed will be defined by the DANE protocol.

Roll-over: The DANE mechanism must allow a site to transition from using one DANE mechanism to another. For example, a domain holder should be able to migrate from using DANE to assert a domain issued certificate (Section Section 3.3) to using DANE to require an external CA (Section Section 3.1), or vice versa. The DANE mechanism must also allow roll-over between records of the same-type, e.g., when changing CAs.

Simple Key Management: DANE should have a mode in which the domain holder only needs to maintain a single long-lived public/private key pair.

Minimal Dependencies: It should be possible for a site to deploy DANE without also deploying anything else, except DNSSEC.

Minimal Options: Ideally, DANE should have only one operating mode. Practically, DANE should have as few operating modes as possible.

Wild Cards: The mechanism for distributing DANE information should allow the use of DNS wild card labels (*) for setting DANE information for all names within a wild card expansion.

Redirection: The mechanism for distributing DANE information should work when the application service name is the result of following a DNS redirection chain (e.g., via CNAME or DNAME).

## 5. Acknowledgements

Thanks to Eric Rescorla for the initial formulation of the use cases, Zack Weinberg and Phillip Hallam-Baker for contributing other requirements, and the whole DANE working group for helpful comments on the mailing list.

## 6. IANA Considerations

This document makes no request of IANA.

## 7. Security Considerations

The primary focus of this document is the enhancement of TLS authentication procedures using the DNS. The general effect of such mechanisms is to increase the role of DNS operators in authentication processes, either in place of or in addition to traditional third-party actors such as commercial certificate authorities. The specific security implications of the respective use cases are discussed in their respective sections above.

## 8. References

### 8.1. Normative References

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

[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.

[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.

## 8.2. Informative References

[RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, February 2002.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

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

## Author's Address

Richard Barnes
BBN Technologies
9861 Broken Land Parkway
Columbia, MD 21046
US

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