Workgroup: DANCE
Internet-Draft:
draft-wilson-dance-architecture-01
Published: 9 November 2021
Intended Status: Informational
Expires: 13 May 2022
Authors: A. Wilson S. Huque O. Johansson
Valimail Salesforce Edvina.net

An Architecture for DNS-Bound Client and Sender Identities

Abstract

This architecture document defines terminology, interaction, and authentication patterns, related to the use of DANE DNS records for TLS client and messaging peer identity, within the context of existing object security and TLS-based protocols.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the DANE Authentication for Network Clients Everywhere Working Group mailing list (dance@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/dance/.

Source for this draft and an issue tracker can be found at https://github.com/ashdwilson/draft-dance-architecture.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction
2. Conventions and Definitions
3. Communication Patterns
  3.1. Client/Server
  3.2. Peer2peer
  3.3. Decoupled
4. Use Cases
  4.1. Mutual TLS Authentication
    4.1.1. Example 1: TLS authentication for HTTPS API interaction, DANE preauthorization
    4.1.2. IoT: Device to cloud
    4.1.3. Oauth2
    4.1.4. Edge Computing
    4.1.5. SIP and WebRTC inter-domain privacy
    4.1.6. DNS over TLS client authentication
    4.1.7. SMTP, STARTTLS
    4.1.8. SSH client
    4.1.9. Network Access
    4.1.10. LoRaWAN
  4.2. Object Security
    4.2.1. Structured data messages: JOSE/COSE
  4.3. Operational anomaly reporting
    4.3.1. MUD reporting for improper provisioning
    4.3.2. XARF for abuse reporting
  4.4. Adjacent Ecosystem Components
    4.4.1. Certification Authority
5. Security Considerations
  5.1. Confidentiality
  5.2. Integrity
  5.3. Availability
    5.3.1. DNS Scalability
    5.3.2. Change of ownership

6. IANA Considerations
7. Normative References
Acknowledgments
Authors' Addresses

1. Introduction

A digital identity, in an abstract sense, possesses at least two features: an identifier (or name), and a means of proving ownership of the identifier. One of the most resilient mechanisms for tying an identifier to a method for proving ownership of the identifier is the digital certificate, issued by a well-run Certification Authority (CA). The CA acts as a mutually trusted third party, a root of trust.

Certificate-based identities are limited in scope by the issuing CA, or by the namespace of the application responsible for issuing or validating the identity.

An example of this limitation is well-illustrated by organizational Public Key Infrastructure (PKI). Organizational PKI is very often coupled with email and LDAP systems, and can be used for associating a human or machine identity identifier with a public key. Within the organization, authentication systems already agree on the roots of trust for validating entity certificates issued by organizational PKI.

Attempting to use organizational PKI outside the organization can be challenging. In order to authenticate a certificate, the certificate's CA must be trusted. CAs have no way of controlling identifiers in certificates issued by other CAs. Consequently, trusting multiple CAs at the same time can enable entity identifier collisions. Asking an entity to trust your CA implies trust in anything that your CA signs. This is why many organizations operate a private CA, and require devices connecting to the organization's networks or applications to possess certificates issued by the organization's CA.

These limitations make the implementation and ongoing maintenance of a PKI costly, and have a chilling effect on the broader adoption of certificate-based IoT device identity. If certificate-based device identity were easier to manage, more broadly trusted, and less operationally expensive, more organizations and applications would be able to use it.

The lack of trust between PKI domains has lead to a lack of simple and globally scalable solutions for secure end-to-end inter-domain communication between entities, such as SIP phones, email and chat accounts and IoT devices belonging to different organizations.

DANCE seeks to make PKI-based IoT device identity universally discoverable, more broadly recognized, and less expensive to maintain by using DNS as the constraining namespace and lookup mechanism. DANCE builds on patterns established by the original DANE RFCs to enable client and sending entity certificate, public key, and trust anchor discovery. DANCE allows entities to possess a first-class identity, which, thanks to DNS, may be trusted by any application also trusting the DNS. A first-class identity is an application-independent identity.

2. Conventions and Definitions

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This section will be interesting to define. We have great examples of identity terminology in the https://datatracker.ietf.org/doc/html/draft-sarikaya-t2trg-sbootstrapping-06 document, but this document also admits that there is semantic drift on terms like "bootstrapping", depending on who's talking.

Identity provisioning: This refers to the set of tasks required to securely provision an asymmetric key pair for the device, sign the certificate (if the public credential is not simply a raw public key), and publish the public key or certificate in DNS. Under some circumstances, these steps are not all performed by the same party or organization. A manufacturer may instantiate the key pair, and a systems integrator may be responsible for issuing (and publishing) the device certificate in DNS. In some circumstances, a manufacturer may also publish device identity records in DNS. In this case, the system integrator needs to perform network and application access configuration, since the identity already exists in DNS.

Security Domain: DNS-bound client identity allows the device to establish secure communications with any server with a DNS-bound identity, as long as a network path exists, the entity is configured to trust its communicating peer by name, and agreement on protocols can be achieved. The act of joining a security domain, in the past, may have involved certificate provisioning. Now, it can be as simple as using a manufacturer-provisioned identity to join the device to the network and application. [Is the security domain defined by how broadly the identity is recognized, or by the breadth of the application or network access policy?]

Client: This architecture document adopts the definition of "Client" from RFC 8446: "The endpoint initiating the TLS connection"

Server: This architecture document adopts the definition of "Server" from RFC 8446: "The endpoint that did not initiate the TLS connection"

Sending agent: Software which encodes and transmits messages. A sending agent may perform tasks related to generating cryptographic signatures and/or encrypting messages before transmission.

Receiving agent: Software which interprets and processes messages. A receiving agent may perform tasks related to the decryption of messages, and verification of message signatures.

Store-and-forward system: A message handling system in-path between the sending agent and the receiving agent.

Hardware supplier role: The entity which manufactures or assembles the physical device. In many situations, multiple hardware suppliers are involved in producing a given device. In some cases, the hardware supplier may provision an asymmetric key pair for the device and establish the device identity in DNS. In some cases, the hardware supplier may ship a device with software pre-installed.

Systems integrator: The party responsible for configuration and deployment of application components. In some cases, the systems integrator also installs the software onto the device, and may provision the device identity in DNS.

Consumer: The entity or organization which pays for the value provided by the application, and defines the success criteria for the output of the application.

3. Communication Patterns

3.1. Client/Server

Client/server communication patterns imply a direct connection between an entity which provides a service (the server), and an entity which initiates a connection to the server, called a client. A secure implementation of this pattern includes a TLS-protected session directly between the client and the server. A secure implementation may also include public key-based mutual authentication.

Extending DANE to include client identity allows the server to authenticate clients independent of the private PKI used to issue the client certificate. This reduces the complexity of managing the CA certificate collection, and mitigates the possibility of client identifier collision.

3.2. Peer2peer

The extension also allows an application to find an application identity and set up a secure communication channel directly. This pattern can be used in mesh networking, IoT and in many communication protocols for multimedia sessions, chat and messaging.

3.3. Decoupled

Decoupled architecture, frequently incorporating store-and-forward systems, provides no direct connection between the producer and consumer of information. The producer (or sending agent) and consumer (or receiving agent) are typically separated by at least one layer of messaging-oriented middleware. The Messaging-oriented middleware components may act as a server for the purpose of establishing TLS sessions for the producer and consumer. This allows the assertion of identity between the middleware and sending agent, and the middleware and receiving agent. The trust relationship between the sending agent and receiving agent is based on the presumed trustworthiness of the middleware, unless an identity can be attached to the message itself, independent of transport and middleware components.

Within many existing store-and-forward protocols, certificates may be transmitted within the signed message itself. An example of this is S/MIME. Within IoT applications, we find that networks may be more constrained. Including certificates in message payloads can present an unnecessary overhead on constrained network links. Decoupled applications benefit from an out-of-band public key discovery mechanism, which may enable the retrieval of certificates only when needed, and sometimes using a less expensive network connection.

4. Use Cases

4.1. Mutual TLS Authentication

Using DNS to convey certificate information for authenticating TLS clients gives a not-yet-authenticated client the ability to trigger a DNS lookup on the server side of the TLS connection. An opportunity for DDOS may exist when malicious clients can trigger arbitrary DNS lookups. For instance, an authoritative DNS server which has been configured to respond slowly, may cause a high concurrency of in-flight TLS authentication processes as well as open connections to upstream resolvers. This sort of attack (of type slowloris) could have a performance or availability impact on the TLS server.

4.1.1. Example 1: TLS authentication for HTTPS API interaction, DANE preauthorization

* The client initiates a TLS connection to the server.
* The TLS server compares the dane_clientid (conveyed via the DANE Client Identity extension) to a list of allowed client domains.
* If the dane_clientid is allowed, the TLS server then performs a DNS lookup for the client's TLSA record. If the dane_clientid is not allowed, authentication fails.
* If the client's TLSA record matches the presented certificate or public key, the TLS handshake completes successfully and the authenticated dane_clientid is presented to the web application in the (TBD) header field.

This pattern has the following advantages:

* This pattern translates well to TLS/TCP load balancers, by using a TCP TLV instead of an HTTP header.
* No traffic reaches the application behind the load balancer unless DANE client authentication is successful.

4.1.1.1. Example 2: TLS authentication for HTTPS API interaction, DANE matching in web application

* The client initiates a TLS connection to the server.
* The TLS server accepts any certificate for which the client can prove possession of the corresponding private key.
* The TLS server passes the certificate to the web application in (TBD) header field.
* The HTTP request body contains the dane_clientid, and is passed to the web application.
* The web application compares the dane_clientid to a list of allowed clients or client domains.
* If the dane_clientid is allowed, the web application makes the DNS query for the TLSA records for dane_clientid
* If the presented certificate (which was authenticated by the TLS server) matches at least one TLSA record for dane_clientid, authentication succeeds.

This pattern has the following advantages:

* In a web application where a TLS-terminating load balancer sits in front of a web application, the authentication logic in the load balancer remains simple.

* The web application ultimately decides whether to make the DNS query to support DANE authentication. This allows the web application to reject clients with identifiers which are not allowed, before making a DNS query for TLSA retrieval and comparison. No need to manage an allow-list in the load balancer.

* This can be implemented with no changes to the TLS handshake.

4.1.2. IoT: Device to cloud

Direct device-to-cloud communication is common in simple IoT applications. Authentication in these applications is usually accomplished using shared credentials like API keys, or using client certificates. Client certificate authentication frequently requires the consumer to maintain a CA. The CA trust anchor certificate is installed into the cloud application, and used in the TLS authentication process.

Using DANE for device identity can allow parties other than the implementer to operate the CA. A hardware manufacturer can provide a pre-established identity, with the certificate or public key already published in DNS. This makes PKI-based identity more approachable for small organizations which currently lack the resources to operate an organizational CA.

4.1.3. Oauth2

[This can be a broad topic. Should we include, or wait until a re-chartering to update?]

4.1.4. Edge Computing

https://datatracker.ietf.org/doc/html/draft-hong-t2trg-iot-edge-computing-01 may require devices to mutually authenticate in the field. A practical example of this pattern is the edge computing in construction use case [https://datatracker.ietf.org/doc/html/draft-hong-t2trg-iot-edge-computing-01#section-6.2.1]. Using traditional certificate-based identity, the sensor and the gateway may have certificates issued by the same organizational PKI. By using DANE for client and sender identity, the sensor and the gateway may have identities represented by the equipment supplier, and still be able to mutually authenticate. Important sensor measurements forwarded by the gateway to the cloud may bear the DNS name and signature of the originating sensor, and the cloud application may authenticate the measurement independent of the gateway which forwarded the information to the application.

4.1.5. SIP and WebRTC inter-domain privacy

End to end security in SIP is currently based on a classical S/MIME model which has not received much implementation. There are also SIP standards that build upon a trust chained anchored on the HTTP trust chain (SIP identity, STIR). WebRTC has a trust model between the web browser and the servers using TLS, but no inter-domain trust infrastructure. WebRTC lacks a definition of namespace to map to DNS, where SIP is based on an email-style addressing scheme. For WebRTC the application developer needs to define the name space and mapping to DNS.

By using DNS as a shared root of trust SIP and WebRTC end points can anchor the keys used for DTLS/SRTP media channel setup. In addition, SIP devices can establish security in the SIP messaging by using DNS to find the callee's and the callers digital identity.

[https://datatracker.ietf.org/doc/html/draft-johansson-sipcore-dane-sip]

NOTE: include reference to earlier drafts for SIP + DANE

4.1.6. DNS over TLS client authentication

4.1.7. SMTP, STARTTLS

4.1.8. SSH client

4.1.9. Network Access

4.1.9.1. EAP-TLS with RADIUS

4.1.9.1.1. Terminology

Supplicant: The entity which acts as the TLS client in the EAP-TLS authentication protocol. This term is defined in IEEE 802.1x. The suppliant acts as a client in the EAPOL (EAP over LAN) protocol, which is terminated at the authenticator (defined below).

Authentication server: The entity which acts as the TLS server in the EAP-TLS protocol. RADIUS (RFC 2865) is a frequently-used authentication server protocol.

Authenticator: The authenticator is the device which acts as a server the EAPOL (EAP over LAN) protocol, and is a client of the authentication server. The authenticator is responsible for passing EAP messages between the supplicant and the authentication server, and for ensuring that only authenticated supplicants gain access to the network.

https://datatracker.ietf.org/doc/html/rfc5216 is a mature and widely-used protocol for network authentication, for IoT and IT equipment. IEEE 802.1x defines the encapsulation of EAP over LAN access technologies, like IEEE 802.11 wireless and IEEE 802.3 ethernet. RADIUS is a protocol and server technology frequently used for supporting the server side of EAP-TLS authentication. Guidance for implementing RADIUS strongly encourages the use of a single common CA for all supplicants, to mitigate the possibility of identifier collisions across PKIs. The use of DANE for client identity can allow the safe use of any number of CAs. DNS acts as a constraining namespace, which prevents two unrelated CAs from issuing valid certificates bearing the same identifier. Certificates represented in DNS are valid, and all others are un-trusted.

4.1.9.2. RADSEC

The RADIUS protocol has a few recognized security problems. https://datatracker.ietf.org/doc/html/rfc6614 addresses the challenges related to the weakness of MD5-based authentication and confidentiality over untrusted networks by establishing a TLS session between the RADIUS protocol client and the RADIUS protocol server. RADIUS datagrams are then transmitted between the authenticator and authentication server within the TLS session. Updating the RADSEC standard to include the use of DANE for client and server identity would allow a RADIUS server and client to mutually authenticate, independent of the client's and server's issuing CAs. The benefit for this use case is that a hosted RADIUS service may mutually authenticate any client device, like a WiFi access point or ethernet switch, via RADSEC, without requiring the distribution of CA certificates.

4.1.10. LoRaWAN

We should ask S. if he wants to contribute to this section

4.2. Object Security

4.2.1. Structured data messages: JOSE/COSE

JOSE and COSE provide formats for exchanging authenticated and encrypted structured data. JOSE defines the x5u field in https://datatracker.ietf.org/doc/html/rfc7515#section-4.1.5, and COSE defines a field of the same name in https://datatracker.ietf.org/doc/html/draft-ietf-cose-x509-08#section-2 which can be used for out-of-band x.509 certificate discovery. By adopting DANE for out-of-band certificate discovery, CBOR and JSON data may be authenticated, even if the originating sending agent not have IP connectivity, provided that the sending agent's certificate is discoverable in DNS and the receiving agent has access to DNS.

4.3. Operational anomaly reporting

4.3.1. MUD reporting for improper provisioning

4.3.2. XARF for abuse reporting

4.4. Adjacent Ecosystem Components

4.4.1. Certification Authority

5. Security Considerations

5.1. Confidentiality

DNS clients should use DNS over TLS with trusted DNS resolvers to protect the identity of authenticating peers.

5.2. Integrity

The integrity of public keys represented in DNS is most important. An altered public key can enable device impersonation, and the denial of existence for a valid identity can cause devices to become un-trusted by the network or the application. DNS records should be validated by the DNS stub resolver, using the DNSSEC protocol.

Compartmentalizing failure domains within an application is a well-known architectural best practice. Within the context of protecting DNS-based identities, this compartmentalization may manifest by hosting an identity zone on a DNS server which only supports the resource record types essential for representing device identities. This can prevent a compromised identity zone DNS server from presenting records essential for impersonating web sites under the organization's domain name.

The naming pattern suggested in https://datatracker.ietf.org/doc/html/draft-huque-dane-client-cert includes an underscore label (_device) which also prevents the issuance of Web PKI-validating certificates in the event a DNS server hosting a client identity zone, which is capable of presenting A and AAAA records, is compromised.

5.3. Availability

5.3.1. DNS Scalability

In the use case for IoT an implementation must be scalable to a large amount of devices. In many cases, identities may also be very short lived as revocation is performed by simply removing a DNS record. A zone will have to manage a large amount of changes as devices are constantly added and de-activated.

In these cases it is important to consider the architecture of the DNS zone and when possible use a tree-like structure with many subdomain parts, much like reverse DNS records or how telephone numbers are represented in the ENUM standard (RFC 6116).

If an authoritative resolver were configured to respond quite slowly (think slow loris), is it possible to cause a DoS on the TLS server via complete exhaustion of TCP connections?

The availability of a client identity zone is essential to permitting clients to authenticate. If the DNS infrastructure hosting client identities becomes unavailable, then the clients represented by that zone cannot be authenticated.

OEJ: We may want to have a discussion with the IETF DNS directorate. The scalability section above is from a discussion with one of the members...

5.3.2. Change of ownership

What happens if an organization owning the client identity goes out of business? What's the best way to transfer an identifier to another zone? <note: there may be an opportunity here to take advantage of EST, or another protocol supporting certificate renewal, to allow client devices to rotate to another zone>

6. IANA Considerations

This document has no IANA actions.

7. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Ash Wilson
Valimail

Email: ash.d.wilson@gmail.com

Shumon Huque
Salesforce

Email: shuque@gmail.com

Olle Johansson
Edvina.net

Email: oej@edvina.net