Issue paper n°12

Securing End-to-End Internet communications using DANE protocol

Today, the Internet is used by nearly 2.5 billion people to communicate, provide/get information. When the communication involves sensitive information such as bank details, credit card numbers, health records etc., the communication method must be secure. The exchange of information on the Internet is not secure by default, and that leads to a variable risk of malicious attacks such as data corruption, identity theft etc.

As the Internet evolved, the need for new security mechanisms arose, either due to a new type of attack or identification of a new security hole. Solutions were proposed and deployed progressively. Such solutions include, but are not limited to Internet Protocol Security (IPSec) for securing the network layer (aka, the IP layer), Transport Layer Security (TLS) for securing communication between two Internet applications, such as a web server and a web browser, Domain Name System Security Extensions (DNSSEC) for securing the DNS resolution process, etc.

During the few recent years, some high profile attacks, targeting the X.509 Public Key Infrastructure (PKIX), used for securing Internet communication has initiated an urgent need for a technology to plug the security hole in the PKIX ecosystem. It is in this context, that the Internet Engineering Task Force (IETF) proposed the DNS-Based Authentication of Named Entities (DANE) protocol/mechanism.

This document explains the DANE protocol and also how DANE could provide the required trust in the last mile with DNSSEC. This document is intended for an audience who have some knowledge of Internet protocols in general and Domain Name System (DNS) in particular. This document introduces the DANE protocol and explains how DANE plugs the existing flaw in the Internet while securing end-to-end communication. This document is not sufficient enough to implement DANE for a domain administrator.

The following figure (Figure 1) depicts a typical Internet communication, browsing the Web.

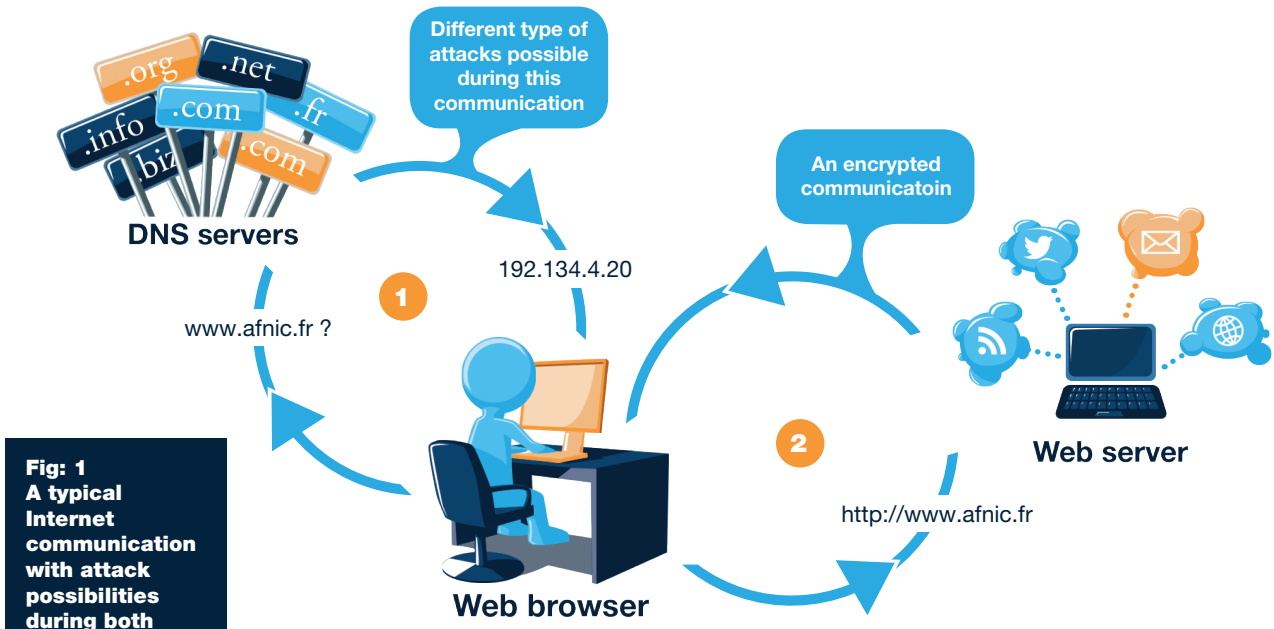


Fig: 1 A typical Internet communication with attack possibilities during both operations

This figure shows two operations:

1. Since the human brain is not capable of remembering many numbers (IP addresses) at a time, but is well equipped to remember names, normally domain names are used while querying for a service in the Internet. But, since Internet applications need IP addresses to communicate with each other, the DNS is used as a "Name directory" to typically obtain the IP address associated to a given server identified with its name.
2. The obtained IP address is then used by the application (i.e. the web browser in Figure 1) to engage in an Internet communication with the remote web server.

Both the two operations mentioned previously are not implicitly secure. By default, information transmitted during either DNS resolution or during accessing the server for data exchange, no authentication or encryption services are applied. Thus, there are numerous possibilities during those operations wherein an attacker could provide false information, such as rogue IP address during DNS resolution and thereby redirect the user to a fraudulent server.

For the first operation (DNS resolution), securing the communication could be provided by DNSSEC. How DNSSEC provides security will be describer later in this article? For the second operation (connection between the browser and the web server), the TLS protocol comes to the rescue, wherein it allows the client and the server to authenticate each other, and to negotiate an encryption algorithm and cryptographic keys before the data is exchanged. TLS makes sure that data cannot be read or tampered by a third-party during transit, since the data is encrypted.

Encrypting and decrypting the data in the TLS protocol is done by a matching pair of cryptographic keys: public and private key. The Data encrypted by a public key can be decrypted only by the corresponding private key, and vice versa. This makes it possible to have secure communication with unknown users. For example, a Bank publishes its public key for anyone to download. An account holder in the bank, Alice encrypts a message using the public key, and sends it to the bank. Only the bank can decrypt the message using its private key. Thus Alice is sure that her message is not read by anyone else.

In a TLS connection, the browser asks the web server to send its public key. The public key sent by the web server to the browser is in the form of X.509 certificate, which is further explained in section-II.

1 Problem Statement

Public-Key Infrastructure X.509 (PKIX)

On the other hand, there is a possibility that an impersonator publishes his/her public key posing as Alice's bank. Alice will encrypt the message using the impersonator's public key and send it to her bank, where the impersonator does a "man-in-the-middle" and copies the message. As he/she is the owner of public key, he/she also has the private key which enables him/her to decrypt and read the message. Looking at an analogy for web browsing, anyone can create a public-key for accessing any domain name. In security terms, this is a disaster, since any impersonator can create a public key for domains such as www.example.com, and fool the user to access a fraudulent server.

Hence there is a necessity of a binding between the identity (e.g. the domain name) and the public key. The X.509 standard proposed by the ITU and ISO provides mechanism to bind a particular public key to a particular identity. This binding can be autonomously done by the domain holder and in that case it is called self-signed certificate. If the self-signed certificate is obtained from a trusted source by the application using the certificate for authentication, then it is accepted, otherwise there is no guarantee of the certificate's authenticity.

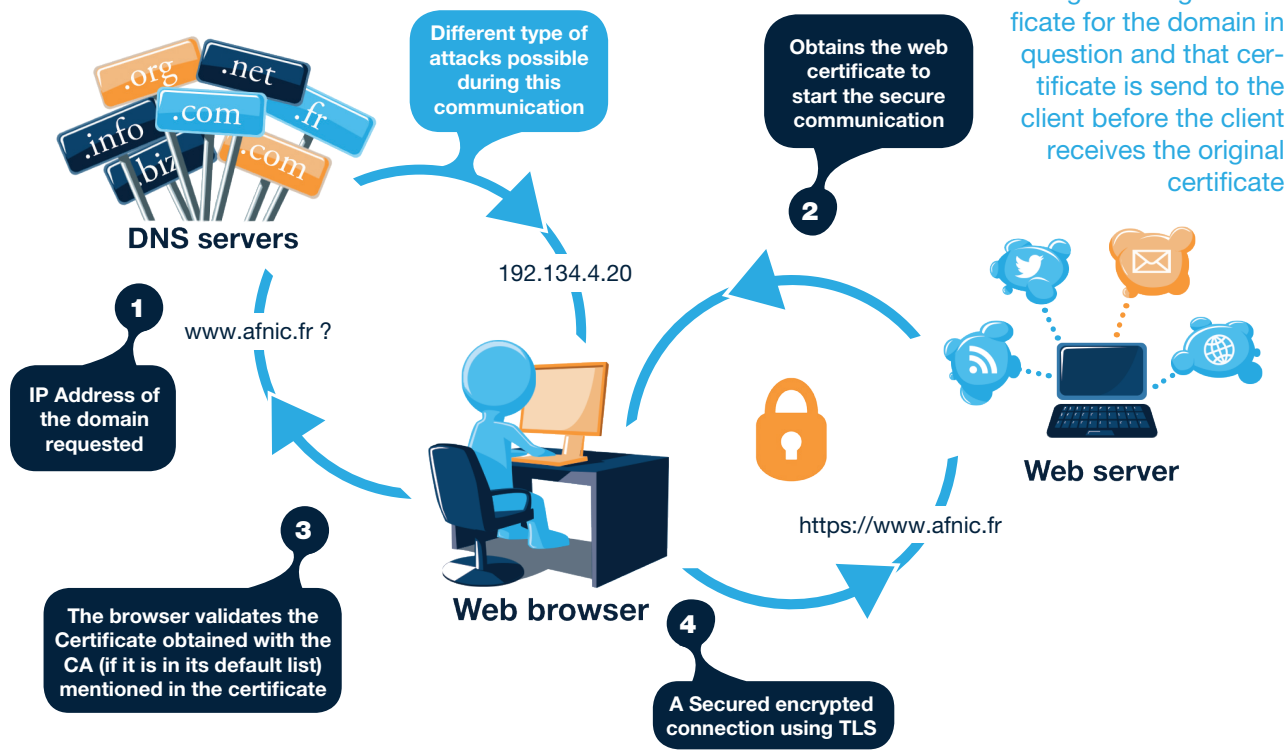
The Certification Authorities (CAs) role

This is where the need for a trusted third party arises. It is just similar to the passport case, where the trusted third party is the concerned government which has issued the passport. In a passport, the government attests that the person in the photo is identified by a particular name, surname and other credentials.

In the web browsing use case, the certificate issued, is like the passport. In the PKIX ecosystem, the role of the government is played by organizations called CAs. A certificate issued by a given CA, binds the given domain name with information such as who has assigned the certificate, the entity which has requested the certificate, its validity period etc.

Similar to wherein, a passport attested by one government, is accepted by other governments, as a validated document for authenticating a person, browser vendors such as Firefox, Chrome, Internet explorer, Safari etc., accept digital certificates created only by certain CAs. The browser vendors authorize an organization to be a CA, only after understanding that they are trustworthy, and they follow well defined principles and procedures to provide certificates only for correct domain holders. Once the browser vendors authorize an organization to be a CA, the latter is added to the list of trusted CAs in the browser library. Thus, once a client using a browser accesses a domain name which has a digital certificate generated by one of the CAs among its pre-installed list, the certificate is implicitly trusted as shown in Figure 2.

Fig: 2 Communication between the browser and the web server is secured using the PKIX ecosystem and TLS, but still there are attack possibilities



The problem of many

At a glance, if we look at the size of the list of CAs accepted by popular browsers such as Chrome, Firefox, Internet explorer, etc., it varies, but is in the range of hundreds. For example, a browser such as Firefox trusts 1,482 CA Certificates (as per EFF SSL observatory[1]) provided by 651 organizations. Complementing the issue is that in the CA ecosystem there is a practice of a CA providing authorization to other organizations, or its branches to create certificates on its behalf. They are called subordinate CAs. A browser will trust the digital certificate created by the subordinate CA also.

Even if only CA among the list of CAs, or its subordinates are compromised, it can generate a certificate for any domain name which could be then authenticated by a browser such as Firefox, and thereby compromising a secure web communication of an end-user who is using Firefox. For instance, two different CAs (where one is a compromised CA) can issue two different certificates for the same domain and both of them will be trusted by the browser. The fault here is that the owner of a domain had up to now, no way of telling the world, which CA or certificate should be used to authenticate to connect to the server of the particular domain.

Need for a solution

The use of PKIX for securing web communication has been there for a while. Browser vendors, users, Standards Development Organizations (SDOs) have been all aware of the issue - «Problem of many». There were some attempts (Perspectives, Trust on First Use (ToFU), Channel Id, Certificate Transparency, etc.) to limit the issue. But these attempts became more focused after the two high profile attacks on the CAs - Comodo and DigiNotar.

The sequence of the events was as follows. Comodo found that on March 15 2011, one of its affiliate Registration Authority[2] (RA) was compromised and the attacker created a user account with the affiliate RA. Using this account, the attacker created 9 Certificate Signing Requests for high valued web sites such as *login.live.com*, *mail.google.com*, *login.yahoo.com* etc. and it is believed that the attacker got at least one X.509 certificate issued out of their 9 requests.

Not going into the politics of this attack, an attacker who has created the fraudulent certificates as in the case of Comodo, could do a Man-In-the-Middle, and redirect the user to a spoofed server which resembles that of the original site (phishing). The certificates provided by the spoofed server will be accepted by the browser, since it is generated by the CA which is trusted by the browser. Everything the user reads or writes (such as user name, password, email etc.) can be seen and copied by the spoofing server.

[1] https://www.eff.org/observatory

[2] RAs collect and verify identity information from Direct Subscribers using certain procedures that implement the identity validation policies. The RA creates the Certificate Signing Requests for submission to a CA. The CA signs the Certificate Signing Requests and issue public X.509 certificates to direct subscribers.

As per reports, the same hacker who intruded into Comodo was also responsible for intruding into DigiNotar systems. Even though the attack came into notice publicly in end of August 2011, investigations reveal that the intruder gained access to DigiNotar system as early as June 17, 2011. Similar to Comodo attack, the intruder has created fraudulent digital certificates for high profile web sites. Investigations also reveal that, the generated certificates were used to redirect users to spoofed servers and obtain user credentials.

Both DigiNotar and Comodo were not ordinary companies. They were high profile security companies trusted by number of organizations including governments and millions of users. Subsequent attacks on such high profile companies demonstrated that it was not just enough in increasing the security of the infrastructure of the CAs, but emphasized the need for reducing the attack scope in the PKIX model.

Limiting the attack surface, what are the options?

As mentioned earlier, the issue is not the security of the PKIX technology, but with such a big list of the CAs accepted by the browsers by default, there is a higher probability of being compromised while establishing the TLS connection, using the PKIX; rather than resolving the IP addresses, using the DNS. As mentioned earlier, with the current PKIX model, a domain owner does not have the possibility of telling the browser that any user's connection to his/her domain should get validated by a certificate provided by a particular CA.

Different techniques were proposed to reduce the attack probability in the PKIX model such as Trust on First Use (ToFU), Perspectives[3], Certificate Transparency[4] (CT), Certificate Authentication and Authorization (CAA)[5] and DANE

Of the different technologies proposed to limit the attack surface, ToFU is the easiest to implement because it needs only for a browser to install the ToFU compatible browser add-on. Perspectives and CT are based on a system of Notary service which does not completely coexist with the current PKIX model and needs additional services acting as notary services. CAA is like a hack which does not need any modifications, and in the short term looks a better option in limiting the attack surface. But looking at security from an end-to-end perspective and at providing more options to the users (such as self-signed certificates), DANE happens to rank high.

[3] http://perspectives-project.org/

[4] http://www.certificate-transparency.org/

[5] http://tools.ietf.org/html/rfc6844

2 A solution which permits to have end-to-end security: DANE

DANE - Augmenting the security in PKIX

This section will further elaborate on how DANE reduces the scope of an attack in the PKIX ecosystem and based on a secure DNS infrastructure thanks to DNS Security Extensions (DNSSEC[6]). Using DANE protocol, a domain owner will sign the certificate provided by the web server based on different options (explained below) and publish it in the domain's DNS zone (signed with DNSSEC), thus, enabling the domain owner, an option of informing the application (e.g. browser) on how to validate the certificate obtained from the web server. For example, if the CA for the domain www.example.com is «X», with DANE mechanism, the browser will only accept a certificate from CA «X» for authenticating the server, thus, reducing the attack probability.

DANE was conceived and standardized at the IETF. Two RFCs relating to DANE have been published by the IETF:

1. DANE Use case RFC 6394
2. DANE Protocol RFC 6698

RFC 6698 focuses on standardizing and usage of the TLSA resource record. The basic role of this record is to be published in a DNS zone, and to indicate the certificate information that corresponds to a specific service on a specific port of a name in that zone.

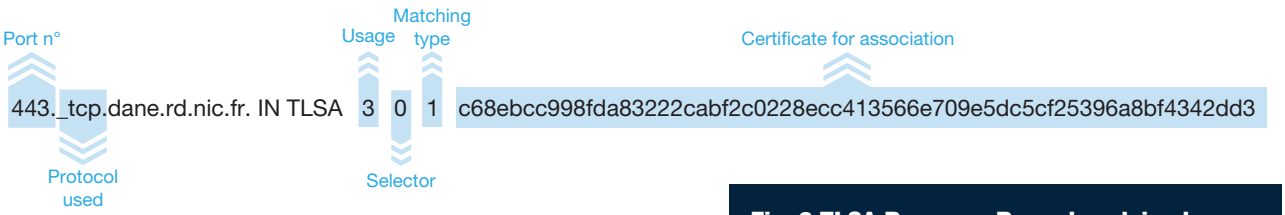


Fig: 3 TLSA Resource Record explained

As shown in Figure 3, the TLSA resource record consists of four fields: the "certificate Usage", "a Selector", "a Matching type" and the "Certificate for association" data. The application must match the 'certificate for association data field' in the TLSA RR with the target certificate (i.e. the certificate obtained from the domain's web server) based on the other values (certificate usage, selector and matching type) in the TLSA resource record.

The 'certificate Usage field' is briefly summarized here. For further details on other parameters of the TLSA RR please refer to RFC 6698.

- '0' - During validation, the browser should use only the specific CA mentioned in the "Certificate for association» field of the TLSA RR for validating the target certificate.
- '1' - The browser should validate the target certificate only with the certificate mentioned in the «Certificate for association» field of the TLSA RR.

There are other two values ('2' and '3') in the "certificate Usage" field which will be explained later.

[6] See mechanism description later

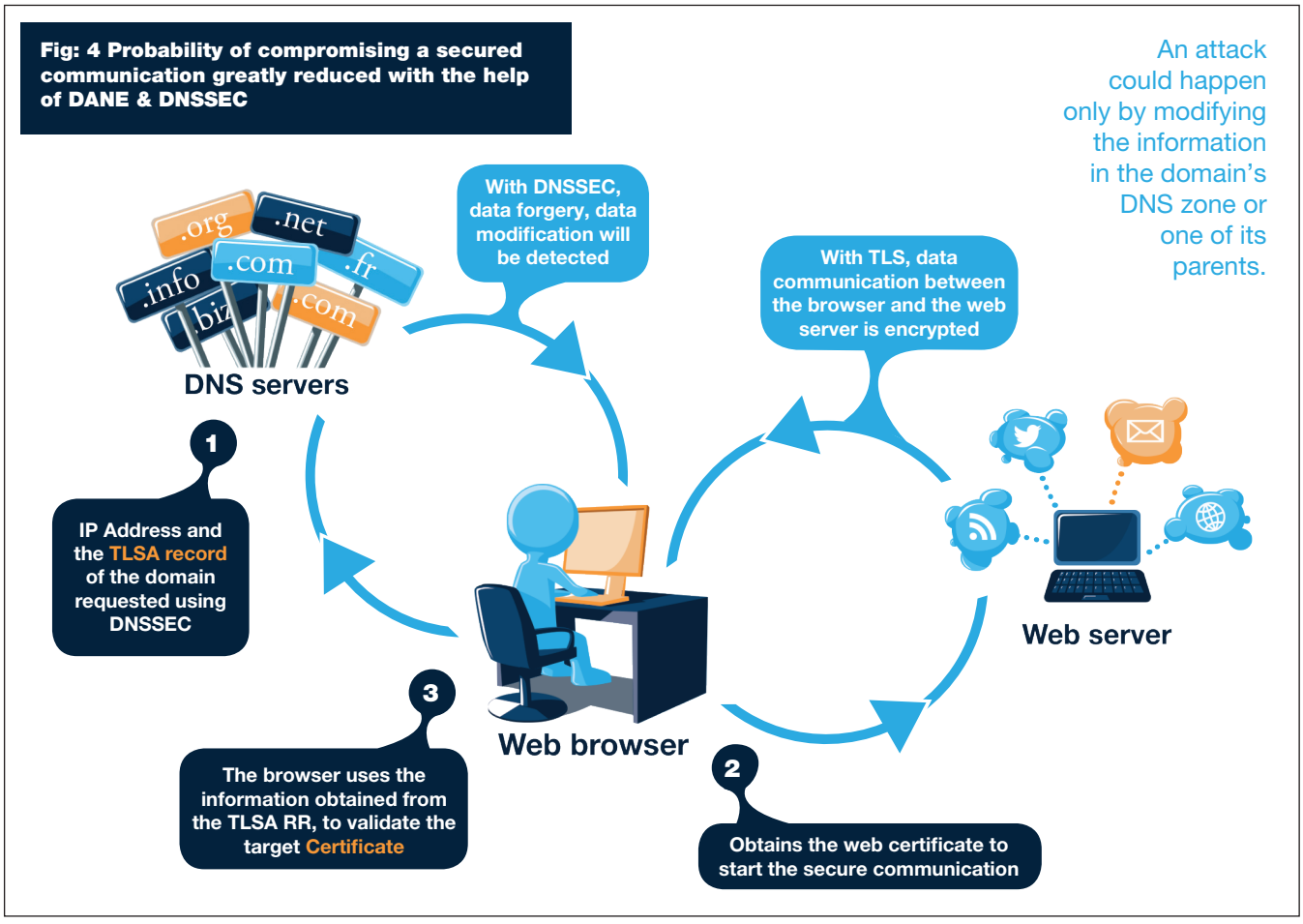
Why DNSSEC is vital for DANE?

The “certificate usage 0/1”, demonstrates how the attack surface can be reduced in the PKIX ecosystem while validating the target certificate provided by the server. Supposing the attacker has done a Man-in- the-Middle” attack during DNS resolution (i.e. the “first operation in Figure 4), and provided a fraudulent IP address for the domain requested, the browser will use the IP address obtained, to access the server. With the attacker creating a digital certificate for the spoofed server from an authorized CA, the attacker can convince the browser that a server of the attacker’s choice legitimately represents the victim’s service.

DNSSEC[7] makes it possible for a user to verify, based on a cryptographic chain of trust that the information resulting from a DNS resolution query originates from the legitimate DNS zone corresponding to the queried domain name. In other words, when used in the end-to-end DNS resolution process, DNSSEC extensions prevent data from being tampered while transiting down to the requesting user. Hence, in order for DANE to augment the security in the existing PKIX model, the information obtained during DNS resolution should be validated using DNSSEC. This is the reason why RFC 6698 (DANE protocol), states that the DNS zone which has a TLSA RR must be signed by DNSSEC and the applications which query the domain for TLSA RR validation should use a DNSSEC aware resolver. To simplify, DANE is not effective if it does not rely on a DNSSEC infrastructure.

Thus DANE with DNSSEC provides an end-to-end security for an Internet communication (as shown in Figure 4) at both stages: first during the preliminary DNS resolution, then at the connection set up with the domain’s server.

Fig: 4 Probability of compromising a secured communication greatly reduced with the help of DANE & DNSSEC



[7] http://tools.ietf.org/html/rfc6698/

DANE - Using DNSSEC as an alternative PKI

Until now, the concentration was on a PKI based on digital certificates i.e. the PKIX model. The DNS, leveraged by DNSSEC has become a *de facto* PKI. Like in the case of the PKIX model, wherein the CA key is the trust anchor, in the case of the DNSSEC PKI, the trust anchor is the DNS root the key.

Certificate usage ('2'/'3') of the TLSA resource record explains how end-to-end security of web browsing could be done without involving the CA ecosystem. That is, a domain holder creates a self-signed certificate, but can still be authenticated by the browser vendors.

- '2' - In a use case, wherein an organization has planned to create its own CA and each department in the organization creates its own certificates with the created CA as trust anchor for their respective department web sites. During validation, the browsers will not normally trust the organization department web site, since it does not have the organization CA in its list of trusted CA. But, when it receives the TLSA RR as part of the response after DNSSEC validation, it is sure that the TLSA payload is not forged unless and otherwise someone has access to the domain's DNS zone. To validate the certificate, the browser has to make sure that the CA of the target certificate is the same as that of the «Certificate for association» field in the TLSA RR.
- '3' - In a use case, wherein the domain administrator issues the self-signed certificate which is stored as target certificate in the web server, and a fingerprint of the certificate is added in the domain's DNS zone as the "Certificate for association" field in the TLSA RR. To validate the certificate, the browser has to make sure that the target certificate matches the «Certificate for association» field in the TLSA RR.

Thus DANE technology not only reinforces the security of web browsing at the last mile using the existing PKIX model, but also provides an alternative option i.e. only using DNS leveraged by DNSSSEC, hence completely bypassing the mechanism of providing and managing X.509 certificates via PKIX.

Implementing and securing using DANE

The first step towards setting DANE for a domain name is to create a TLSA resource record for the domain by the domain administrator. There are several tools available to generate a TLSA record. One of them is SWEDE[8]. The generated TLSA resource record is provisioned in the DNS zone of the domain by the domain administrator and the zone is signed using DNSSEC.

During DNS resolution, the TLSA resource record should also be queried as shown in the Figure 4. If the 'certificate Usage' field in the TLSA resource record has values '0' or '1', then the application must validate the target certificate using the PKIX infrastructure (Refer III.1). If the certificate Usage field in the TLSA resource record has values '2' or '3', then validation is done using DNSSEC (Refer III.3).

[8] https://github.com/pieterlexis/swede

3 Conclusion: DANE - the missing piece in End-to-End Internet security

DANE is not only for web browsing

DANE was conceived to solve the issues relating to web browsing. Now, there have been efforts in the DANE Working Group[9] (WG) at the IETF, to extend its usage to securing other application such as mail (s/MIME), instant messaging (XMPP) etc. All these works are ongoing process and if they are adopted by the IETF and published as RFCs, there will be implementations. DNSSEC is a common prerequisite infrastructure for all these implementations.

Role of DANE in accelerating DNSSEC deployment

As explained in the beginning of this document, a typical Internet communication involves the DNS ecosystem to resolve the address of a particular domain name. DNSSEC makes sure that the data obtained through DNS resolution is from the legitimate zone for the domain name (i.e. data origin authentication) and the data is not tampered in transit (i.e. data integrity). These security extensions make DNSSEC as a vital component for Internet communications requiring a high-level of trust in the DNS infrastructure.

As in many of the important technologies (such as IPv6), the chicken and egg problem for DNSSEC exists. Many service and network infrastructure providers are in a “wait and see” approach to join the DNSSEC bandwagon. The reasons vary from complexity in implementing DNSSEC to unwarranted breakdowns and commercial incentives. Many of them are ready to wait until there is a scenario that will force them to deploy DNSSEC in their network infrastructure.

Many of the discussions for slow DNSSEC adoption has been attributed to the lack of a “Killer app” using DNSSEC as the security foundation. The commercial opportunity that such an application creates may generate the consumer pressure for DNSSEC adoption. Even though applications built on DNSSEC and DANE protocol may not be the “Killer app” for DNSSEC, but if implemented seamlessly, it will provide security to millions of users using Internet for secured communication. Inherent usage of these applications by millions of users which require the DNSSEC-Secured network infrastructure will force the stakeholders to deploy DNSSEC. Thus, DANE could be a catalyst in accelerating DNSSEC adoption.

Read all of our issues papers:
http://www.afnic.fr/en/resources/publications/issue-papers-6.html

[9] https://datatracker.ietf.org/wg/dane/charter/