Abstract

In this paper, I look at the implementability of the 14th draft of the DNS-based Authentication for Named Entities (DANE) specification. To this end a tool has been made and released (called swede) to create and verify TLSA records. All permutations of TLSA records were put into DNS and successfully verified end-to-end using a TLS service. Apart from a current discussion within the DANE Working Group about the definition of “pass PKIX validation” and how it relates to usage 2, DANE is implementable.

February 9, 2012
Acknowledgments

I would like to thank the following for their contributions:

- Bert Hubert - For his insight into DNS, and supervising
- The DANE working group at the IETF - For answering my questions
- The people in #powerdns on OFTC - For their DNS knowledge and support
- The proofreaders
Introduction

The current system used for trust on the Internet depends on certificates issued by Certificate Authorities. In this system there is no way of allowing only a specific CA to sign certificates for an organization or service. Unfortunately, end-user applications trust a large number of these CAs. This makes CAs targets for crackers wanting to impersonate a secured service. Several projects have been started to combat this problem; the main problem seems to be the secure, scalable distribution of certificates and their metadata.

Although Domain Name System Extensions (DNSSEC) are currently being deployed by the Top-Level-Domain (TLD) operators around the world, a ‘killer’ application for this technology does not exist yet. However, the guarantee that data received from the DNS has not been tampered with opens the way for adding more than address records into DNS. The inclusion of cryptographic information for applications is one of them.

The DNS-based Authentication of Named Entities (DANE) working group of the IETF\(^1\) has been founded to create standards to accomplish this. As per the charter:

Specify mechanisms and techniques that allow Internet applications to establish cryptographically secured communications by using information distributed through DNSSEC for discovering and authenticating public keys which are associated with a service located at a domain name.

Motivation

As shown in the coming chapter, the current trust system used on the internet suffers from drawbacks. The DANE specification allows the creation of an out-of-band system to pin certificates to DNS names. Because the specification is still in development is makes for an interesting research topic.

In a previous OS3 Research Project, Danny Groenewegen and Pieter Lange implemented a DNSSEC validator with DANE support in a Firefox plugin [6]. This proved that it is not hard to implement this validation in the end-user browser. This project will look at the other end of DANE, namely the deployment side.

---

\(^1\)Internet Engineering Task Force, the organization creating Internet standards
Chapter 1

Trust on the Internet

The current technology used for trust management and secure connections on the Internet is mostly based on the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) communication standards, collectively called TLS in this report, combined with the use of X.509 certificates. In the last few years, problems with this infrastructure have come to the surface. The IETF is drafting specifications to combat one of these problems.

The problem with the current trust infrastructure

The trust infrastructure on the Internet is based on the Public Key Infrastructure using X.509 version 3 certificates (PKIX) and is defined in RFC5280 [1]. This infrastructure relies on Certificate Authorities (CAs) to certify that a cryptographic key-pair belongs to a certain entity (in this report this entity is limited to ‘end-machine hostname’).

When a program initiates a TLS connection, it is presented with a PKIX certificate for that service. The user’s program verifies that the certificate is issued by a CA it trusts and has the name in the Subject field of the certificate matches the name of the server it connects to. User’s browsers and other applications that are able to use TLS come with a large number of built-in trusted CA certificates [3, p. 19].

A CA can issue certificates for any domain name. This means that a certificate for mybank.example.com signed by a CA is equally valid as one signed by another CA. If an attacker could get a CA to issue a certificate stating an attacker-controlled key is for mybank.example.com, this attacker could use DNS Cache-poisoning[9] to send users to a seemingly valid, but fake bank website. This property makes CAs targets for crackers wanting to obtain false certificates for important domains (like *.google.com or update.microsoft.com). Lately, this has happened with DigiNotar¹ and Comodo².

The need for a solution

A solution to this problem is ‘pinning’[4] a public key or certificate out of band, to make sure only certain CAs can issue certificates for certain (sub-)domains or organizations.

The HTTP Strict Transport Security (HSTS) [12] specification [7] provides a way to mandate the use of a secure connection. Browser vendors complement this specification by supplying whitelists of which CA may sign certificates for certain (sub-)domains ³. This is, however, not a scalable solution.

Another proposed solution is the ‘sovereign keys’ project by the Electronic Frontier Foundation [5] (EFF). This solution that uses a “semi-centralized, verifiably append-only data structure” containing the keys and revocations. These keys can only be added when it is strongly verified that the domain belongs to the requesting party. A browser would, when connecting to an TLS service, lookup the certificate from this key-store.

Another way of solving the problem is using ‘multi-path probing’[13] to ensure the correct certificate is offered to the end-user. When a user contacts a TLS service, it sends a request to a number of trusted ‘notaries’, these notaries also connect to that service and send (the hash) of the certificate to the end-user. The user can then validate if it

³http://dev.chromium.org/sts
is connecting to the right service. The first project to implement this behavior is the Perspectives Project [11] from the Carnegie Mellon University. Recently, security researcher Moxie Marlinspike created a browser plugin called Convergence [10] to implement this behavior in web-browser while maintaining end-user privacy from the notaries.

Yet another possible solution is adding certificate information to DNS, leveraging the existing DNSSEC trust to authenticate this data. This solution is called DNS-based Authentication of Named Entities (DANE).

The DANE working-group of the IETF is currently drafting this specification and it appears to be nearing completion. This report is focused on this specification and will not discuss other solutions.
Chapter 2

DANE

Note: this section describes the 14th draft of the DANE specification, which may have been superseded. Please consult the latest version for implementation details. See the Addendum for updates.

2.1 Use cases

Before starting work on the specification, the working group has set forth a document containing use-cases and demands for this specification [2]. This document describes 3 major use cases [2, Section 3] and their associated certificate constraints:

2.1.1 CA Constraints

This constraint should limit the number of CAs that can issue certificates for an organization. It should allow the service maintainer to express “The certificates for my services must be signed by MyTrustedCA”.

2.1.2 Service Certificate Constraints

When a CA should secretly issue new certificates for the service, there should be a way a maintainer can express “This specific certificate is the only valid one for this service” to the end-user.

2.1.3 Trust Anchor Assertion and Domain-Issued Certificates

The last use case deals with the ability to create private CAs and allow the use of self-signed certificates that will be considered valid by the application initiating a TLS connection.

2.2 DANE specification

The DANE specification [8] describes a new DNS record that contains ‘certificate association data’. A compliant implementation will, before sending any information to the service, look up this DNS record and validate the certificate from the TLS service against it. If there is a match, and the constraints set in the record are met, the connection is resumed. Otherwise it is aborted without the ability for the user to ‘click-through’ the warning.

DANE does not mandate the use of DNSSEC when deploying it, it does recommend it as it leverages the trust provided by DNSSEC as an out-of-band (i.e. not within the PKIX infrastructure), authenticated mechanism to distribute certificate association data.

The new DNS resource record is called TLSA (for ‘TLS Association’). The name belonging to the record contains the port and the protocol on which the TLS service resides. A truncated example is shown below:

_443._tcp.dane.kiev.practicum.os3.nl IN TLSA ( 1 0 1 E8EB600F620... )
2.3 **TLSA record data**

A **TLSA** record consists of 4 fields, the first three describing the fourth (the actual association data):

- 1 octet ‘usage’ field
- 1 octet ‘selector’ field
- 1 octet ‘matching type’ field
- A variable length ‘certificate for association’ field

A visual overview of the wire-format is shown in 2.1.

![Figure 2.1: A TLSA record (wire-format)](image)

### 2.3.1 Usage

The usage field describes how the association data should be treated, this is known as a ‘constraint’. There are 3 values defined for this field:

- 0 - CA constraint
- 1 - End Entity constraint
- 2 - ‘Domain-Issued’ certificate

Usage 0 and 1 mandate that the certificate presented chains to a valid CA certificate. So apart from matching the **TLSA** record, the certificate must be issued by a trusted CA.

With usage 0, the certificate in the **TLSA** record must be a CA certificate that is trusted by the end-user’s program. This allows for only one CA to sign certificate for a service. It also means that the services’ certificate can be renewed without changing the **TLSA** record.

With usage 1, the certificate in the **TLSA** record must match the one received from the TLS service and it must be issued by a trusted CA.

Usage 2 is somewhat unclear in this revision of the specification (see section 4.5 for the issues). The certificate in the record is either a CA or an End Entity certificate and it must be used as a trust-anchor when constructing the certificate chain.

### 2.3.2 Selector

The selector indicates **what** should be matched, it is either

- 0 - The full certificate
- 1 - The SubjectPublicKeyInfo of the certificate
When selector 1 is used together with usage 1, an administrator can change CAs without updating its TLSA records if he uses the same key to create a new request.

When selector 1 used in combination with usage 0, the CA can issue a new certificate for itself (using a stronger hashing algorithm or another name) without the domain’s administrator having to change the TLSA record.

There are security considerations with the use of selector 1, these are documented in Appendix A.1.2 of the specification [8].

2.3.3 Matching Type

The matching type indicates how the association data should be matched to the certificate from the TLS service:

- 0 - Byte-by-byte comparison
- 1 - SHA-256 match of the certificate
- 2 - SHA-512 match of the certificate

2.3.4 Certificate for Association

This contains the bytes to-be matched, the name is slightly incorrect and will be known as ‘Certificate Association Data’ in the next draft as it can contain a hash or SubjectPublicKeyInfo instead of a full certificate.
3

Research

The research is split into two parts, the first part was focused on implementing the DANE specification into a tool that can create and verify TLSA records and verify the validity of them by checking the record against the certificate(-chain) offered by the TLS service. The second part was focused on testing this validation in a real-world, end-to-end situation: on an HTTP server offering TLS services.

3.1 Research Question

The question that this project tried to answer is “Is DANE in its current form implementable and does it achieve its goal of securely binding DNS names to TLS certificates on end-hosts?”. It was expected that this is the case.

A side-effect of the research and in order to help the specification move forward, the results might be used as test vectors for the specification. And the server used for testing could remain operational for a time as a test bed to help implementers test their DANE implementations.

3.2 Tooling

Tooling had to be created to create and verify TLSA records. The only tool available was the dane script from the sshfp [14] package. Unfortunately, this only supported an older draft of the DANE specification and only could create 1 type of record (usage 1, selector 0, matching type 1). The tooling required needed at least the following features:

- Create all 18 permutations of TLSA records, with the ability to:
  - load certificates from disk and from the TLS service
  - create draft (TYPE65468) and RFC (TLSA) records
- Verify a TLSA record
  - Securely receive the TLSA record and address record from DNS (with the option to do this insecurely)
  - Compare the record with the certificate retrieved from the TLS session

By writing the tooling, it was possible to check if the specification was well-written and contains no ambiguities or oversights. This became an important part of the research.

3.3 End-to-end testing

In order to test whether DANE could be deployed in the real world, all 18 permutations of record would have to be created, added to DNS and verified using TLS connections. This needed 2 services, a nameserver and a TLS enabled server (a webserva)
Table 3.1: Values for the fields in the TLSA record and the certificate used for every port

<table>
<thead>
<tr>
<th>Port number</th>
<th>Usage</th>
<th>Selector</th>
<th>Matching Type</th>
<th>Certificate used</th>
<th>Other information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>Self-signed</td>
<td></td>
</tr>
<tr>
<td>1501</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>Self-signed</td>
<td></td>
</tr>
<tr>
<td>1502</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>Self-signed</td>
<td></td>
</tr>
<tr>
<td>1503</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Self-signed</td>
<td></td>
</tr>
<tr>
<td>1504</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Self-signed</td>
<td></td>
</tr>
<tr>
<td>1505</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>Self-signed</td>
<td></td>
</tr>
<tr>
<td>1506</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1507</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1508</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1509</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1510</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1511</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1512</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1513</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1514</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1515</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1516</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1517</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>Comodo PositiveSSL</td>
<td></td>
</tr>
<tr>
<td>1518(1)</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>Comodo PositiveSSL</td>
<td>Two valid TLSA records for the same hostname</td>
</tr>
<tr>
<td>1518(2)</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>Comodo PositiveSSL</td>
<td>idem</td>
</tr>
<tr>
<td>1519</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>Self-signed</td>
<td>CNAME the hostname</td>
</tr>
<tr>
<td>1520</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>Self-signed</td>
<td>idem and CNAME the TLSA record name</td>
</tr>
<tr>
<td>1521</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>None</td>
<td>Deliberately invalid TLSA record</td>
</tr>
</tbody>
</table>

a cname1.dane.kiev.practicum.os3.nl → dane.kiev.practicum.os3.nl
b cname2.dane.kiev.practicum.os3.nl → dane.kiev.practicum.os3.nl
c _1520._tcp.cname2.dane.kiev.practicum.os3.nl → _1520._tcp.dane.kiev.practicum.os3.nl

3.3.1 Setup

All experiments were performed on Debian 6 (“Squeeze”) using, where available, packages from the Debian repositories.

To serve the created TLSA records, a patched PowerDNS 3.1-pre\(^1\) was used. In order to offer the certificates inside an SSL session, the Apache Webserver was used.

3.3.2 Method

After creating the TLSA records, putting them into DNS and signing the zone, a total of 21 TCP ports were used to offer an SSL service. There were 2 certificates used, the first one was self-signed and the other one was signed by Comodo PositiveSSL. For an overview of the used ports and their TLSA field values, see Table 3.1.

The self-signed certificate was offered on the ports used for all usage 2 records, whereas the Comodo signed certificate and the corresponding certificate chain was offered for usage 0 and 1.

The TLSA records for the first 18 ports were all permutations of the usage, selector and matching type fields. One port was used to offer both a usage 0 and a usage 1 TLSA record. Two ports were used to test the tool’s ability to handle CNAME redirection, as described in paragraph A.2.1.1 of the DANE specification\[^8\].

A final record was added with illegal values of the fields, to check the record validation code in the tool.

After setting up the DNS and webserver, all records were verified using the tool. As all records were valid, all records with their subsequent retrieved certificates should verify successful. The DNS entries used during the experiment can be found in Appendix A.

\(^1\)[http://www.powerdns.com](http://www.powerdns.com)
Chapter 4

Results

This chapter describes the results of the tests performed and the issues that arose during the tests or the implementation of DANE.

4.1 swede

The first outcome of this research is swede, a tool that can create and verify TLSA records. It has the features listed in section 3.2. It is free software and can be obtained from https://github.com/pieterlexis/swede.

Swede is written in python and uses the python-bindings for libunbound\(^1\) for secure lookups, M2Crypto.SSL and M2Crypto.X509\(^2\) for SSL and certificate related work and IPAddr\(^3\) for verification of A and AAAA records.

Swede will be developed further, supporting newer drafts of the DANE specification when they are published. Paul Wouters\(^4\) wants to include swede in a new package named ‘secdns’ that will contain tools to create DNS records that include certificates or other cryptographic information.

4.2 Real-world tests

During the real-world tests all 18 permutations of records validated, as shown in Appendix B.2. The verification of the other records can be found there as well.

To verify that swede also works on records not created by itself, several verifications were done on hostnames posted to the DANE mailing list\(^5\) before the release of swede, these verifications were also successful as seen in Appendix B.3.

This shows that DNS based association of certificates to services is becoming a viable option to solve a part of the issues with trust management on the internet.

4.3 Patches for PowerDNS

During the course of this research, PowerDNS 3.1-pre was selected as the nameserver to use because of its preliminary support for TLSA records. This support was however incomplete, as it treated the certificate for association field data as base64 (in accordance with an older draft) instead of hexadecimal. This was fixed in two separate commits\(^6,7\).

\(^{1}\)http://unbound.net/documentation/pyunbound/index.html
\(^{2}\)http://chandlerproject.org/bin/view/Projects/MeTooCrypto
\(^{3}\)https://code.google.com/p/ipaddr-py/
\(^{4}\)maintainer of the sshfp package and creator of first TLSA creation tool
\(^{5}\)http://www.ietf.org/mail-archive/web/dane/current/msg04114.html
\(^{6}\)http://wiki.powerdns.com/trac/changeset/2347
\(^{7}\)http://wiki.powerdns.com/trac/changeset/2358
4.4 Test bed and test vectors

The server at dane.kiev.practicum.os3.nl will remain online as a test bed. There are TLSA records served and ports opened for TCP ports 1500 through 1521. See table 3.1 for the records.

The created records and certificates used will be offered to the DANE working group for inclusion in the specification as test vectors.

4.5 Implementation issues

During the course of this research only one real issue has come to the surface, this issue stems from the term “pass PKIX validation” in combination with usage 2 and is still being discussed within the DANE working group.

4.5.1 Usage 2

In paragraph 2.1.1 of the DANE specification, usage 2 is defined as

“The target certificate MUST pass PKIX validation, with any certificate matching the TLSA record considered to be a trust anchor for this validation”

Later on, in paragraph 4.3 it is explained as

“Certificate usage 2 is used to specify a certificate, or the public key of such a certificate, that must be used as a trust anchor when validating the end entity certificate given by the server in TLS. This usage is sometimes referred to as "domain-issued certificate" because it allows for a domain name administrator to issue certificates for a domain without involving a third-party CA”

This appears to mean 'Create a record from a certificate in the chain (even of length 1) and bypass regular certificate validation', this is however never mentioned explicitly. This could make usage 2 an under-used option in the specification, where it has the potential to give complete control to the domain-owners for issuing certificates. When asking what exactly usage 2 is for, the answer was “[..] usage 2 lets you specify an end-entity certificate that is used as a trust anchor.”

4.5.2 PKIX validation

The term 'PKIX validation' is not clearly defined by the IETF or the X.509 specification. According to a post on the mailing list, it means “The certificate chain must be traversed successfully”. In other emails, it is said to mean that a program must implement the algorithm described in section 6 of RFC5280 [1].

4.5.3 swede’s interpretation

While creating swede, I used the following definition of usage 2 and PKIX:

“Any certificate in the valid certificate-chain offered by the SSL/TLS service MUST match the TLSA record.”

There is discussion in the working group about the definition, when there is consensus swede will be updated to reflect this new definition.

Apart from a better definition of usage 2, the working group is planning to include a new usage (3) to the next draft that is defined as “The target certificate MUST match the TLSA record.”. This means that in the future it will be easier to deploy self-signed certificates on the Internet.

---

8There currently is discussion on the mailing list whether or not a chain of this length is a valid PKIX chain
9http://www.ietf.org/mail-archive/web/dane/current/msg04099.html
10http://www.ietf.org/mail-archive/web/dane/current/msg04096.html
11http://www.ietf.org/mail-archive/web/dane/current/msg04260.html
Chapter 5

Conclusion

Based on the observations and experiences during the course of this research, the following can be said about DANE.

5.1 Coverage of use cases

The DANE specification covers the three constraints set out in the Use cases section. The usage field covers these cases.

5.2 Implementation

Apart from the issues arising from the lack of clarity of the phrase “pass PKIX validation” combined with usage 2, DANE is a specification that can be implemented and could be the killer application DNSSEC needs for wide-spread deployment.

5.3 Future work

The most pressing matter are the issues described in section 4.5 and are currently being discussed by the working group.

After the specification has become an RFC, SSL/TLS libraries need to implement the secure look up of TLSA records and organizations need to deploy TLSA records, perhaps along with a revision of their certificate practices. Especially high-profile targets for certificate forgery like banks and software update services can benefit from the added security of DANE.

5.4 Discussion

This project focused purely on the implementation of the DANE specification itself. The following topics were not considered:

- The issues with the PKI system in general
- Possible weaknesses in the specification
- Any of the security issues already mentioned in the specification
Bibliography


Chapter 6

Addendum

On the fourth of February 2012\(^1\), the working group released the 15\(^{th}\) draft of the DANE specification. This chapter discusses the changes introduced compared to the previous draft (which are discussed in this report).

6.1 Usage 3

A new usage (3) has been added, and is defined as:

“Certificate usage 3 is used to specify a certificate, or the public key of such a certificate, that must match the end entity certificate given by the server in TLS. This usage is sometimes referred to as “domain-issued certificate” because it allows for a domain name administrator to issue certificates for a domain without involving a third-party CA.”

This allows the administrator to issue a self-signed certificate for a service that will be accepted by the TLS client as valid.

6.2 Updated usage 2 description

The description of usage 2 has been updated to better describe how it should be used:

“Certificate usage 2 is used to specify a certificate, or the public key of such a certificate, that must be used as a trust anchor when validating the end entity certificate given by the server in TLS. This usage allows a domain name administrator to specify a new trust anchor, such as if the domain issues its own certificates under its own CA that is not expected to be in the end users collection of trust anchors.”

6.3 Pass PKIX Validation

The term “pass PKIX validation” is still not well defined in the latest draft. A reference to the PKI Path algorithm in RFC 5280 could help. But the additional information provided with the usage 2 description clears up some confusion on the term. At this moment an amendmend to the draft is discussed to include such a phrase.

6.4 Updates

A few hours after the release of draft 15, \textit{swede} was updated with support for this draft. This allowed the test bed to be updated as well, adding the records and ports mentioned in Table 6.1.

\footnote{\url{http://www.ietf.org/mail-archive/web/dane/current/msg04268.html}}
Table 6.1: Values for the fields in the TLSA record and the certificate used for the corresponding port

<table>
<thead>
<tr>
<th>Port number</th>
<th>Usage</th>
<th>Selector</th>
<th>Matching Type</th>
<th>Certificate used</th>
<th>Other information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1522</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>Private CA</td>
<td></td>
</tr>
<tr>
<td>1523</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>Self-signed</td>
<td>Usage 3</td>
</tr>
</tbody>
</table>
Appendix B

swede outputs

B.1 A sample of full verification output

```
$ ./swede verify -p 1513 dane.kiev.practicum.os3.nl
Received the following record for name _1513._tcp.dane.kiev.practicum.os3.nl.:  
Usage: 0 (CA Constraint)  
Selector: 0 (Certificate)  
Matching Type: 1 (SHA-256)  
Certificate for Association: e8eb600f62046e372db8180ba38fd9f23f83a8690b910ad18458b2255c8d8c52  
This record is valid (well-formed).
Attempting to verify the record with the TLS service...
Got the following IP: 145.100.105.165
SUCCESS (Usage 0): A certificate in the certificate chain offered by the server matches the one mentioned in the TLSA record and is a CA certificate
The matched certificate has Subject: /C=US/ST=UT/L=Salt Lake City/O=The USERTRUST Network/OO=http://www.usertrust.com/CN=UTN-USERFirst-Hardware
```

B.2 Verification of records using swede

B.2.1 All 18 permutations

```
$ for x in {1500..1517}; do  
    echo '\n\n======> $x'; ./swede verify -p $x -q dane.kiev.practicum.os3.nl
done
```

SUCCESS (Usage 0): A certificate in the certificate chain offered by the server matches the one mentioned in the TLSA record and is a CA certificate
The matched certificate has Subject: /C=US/ST=UT/L=Salt Lake City/O=The USERTRUST Network/OO=http://www.usertrust.com/CN=UTN-USERFirst-Hardware
SUCCESS (usage 2): A certificate in the certificate chain (including the end-entity certificate) offered by the server matches the TLSA record
======> 1506
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
======> 1507
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
======> 1508
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
======> 1509
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
======> 1510
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
======> 1511
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
======> 1512
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
======> 1513
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
======> 1514
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
======> 1515
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
======> 1516
SUCCESS (Usage 0): A certificate in the certificate chain offered by the server matches the one mentioned in the TLSA record and is a CA certificate
======> 1517
SUCCESS (Usage 0): A certificate in the certificate chain offered by the server matches the one mentioned in the TLSA record and is a CA certificate
======> 1518

B.2.2 Validation of two record for the same name

$ ./swede verify -p 1518 dane.kiev.practicum.os3.nl

Received the following record for name _1518._tcp.dane.kiev.practicum.os3.nl.:
Usage: 0 (CA Constraint)
Selector: 0 (Certificate)
Matching Type: 2 (SHA-512)
Certificate for Association: b1bb6103a83a9579a0723978b631d1b956eab759a4d93e744370090525f5fcb4027c05cb06c8e8ba5022f509bb2a51e1a10b9c6d4fc67d2899ce417b3
This record is valid (well-formed).
Attempting to verify the record with the TLS service... Got the following IP: 145.100.105.165
SUCCESS (Usage 0): A certificate in the certificate chain offered by the server matches the one mentioned in the TLSA record and is a CA certificate
The matched certificate has Subject: /C=US/ST=UT/L=Salt Lake City/O=The USERTRUST Network/OO=http://www.usertrust.com/CN=UTN-USERFirst-Hardware

Received the following record for name _1518._tcp.dane.kiev.practicum.os3.nl.:
Usage: 1 (End-Entity Constraint)
Selector: 0 (Certificate)
Matching Type: 0 (Certificate)
Certificate for Association: 629d023aeb206b736197aa4e47930115ce0f217cb4e8f055e5d64598647bb9220a759f0f3dc28f286c45d076eeaf71c7f9813ee697c85a80f20374848c7655
This record is valid (well-formed).
Attempting to verify the record with the TLS service... Got the following IP: 145.100.105.165
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
The matched certificate has Subject: /OU=Domain Control Validated/OO=PositiveSSL/CN=dane.kiev.practicum.os3.nl

B.2.3 Validation of redirected name

$ ./swede verify -p 1519 cname1.dane.kiev.practicum.os3.nl

Received the following record for name _1519._tcp.cname1.dane.kiev.practicum.os3.nl.:
Usage: 0 (CA Constraint)
Selector: 0 (Certificate)
Matching Type: 2 (SHA-512)
Certificate for Association: 872e38490eb7ba690926f25c64cd449b615a0b01be8253570440be1c4fd298d
This record is valid (well-formed).
Attempting to verify the record with the TLS service...
Got the following IP: 145.100.105.165
SUCCESS (usage 2): A certificate in the certificate chain (including the end-entity certificate) offered by the server matches the TLSA record
The matched certificate has Subject: /C=NL/ST=Noord-Holland/L=Amsterdam/O=OS3/CN=cname1.dane.kiev.practicum.os3.nl

B.2.4 Validation of a redirected name and TLSA record

$ ./swede verify -p 1520 cname2.dane.kiev.practicum.os3.nl
Received the following record for name _1520._tcp.cname2.dane.kiev.practicum.os3.nl.: 
Usage: 2 (Trust Anchor)
Selector: 0 (Certificate)
Matching Type: 1 (SHA-256)
Certificate for Association: 445bd841d18d65084e9fe1e38f8478b71da2c27a9f6e2bd55c149356007156
This record is valid (well-formed).
Attempting to verify the record with the TLS service...
Got the following IP: 145.100.105.165
SUCCESS (usage 2): A certificate in the certificate chain (including the end-entity certificate) offered by the server matches the TLSA record
The matched certificate has Subject: /C=NL/ST=Noord-Holland/L=Amsterdam/O=OS3/CN=cname2.kiev.practicum.os3.nl

B.2.5 Validation of an invalid record

$ ./swede verify -p 1521 dane.kiev.practicum.os3.nl
Received the following record for name _1521._tcp.dane.kiev.practicum.os3.nl.: 
Usage: 9 (INVALID)
Selector: 6 (INVALID)
Matching Type: 3 (INVALID)
Certificate for Association: 629d023aeb206b347b99315ec0f217cb4a8f055e5d645986d7bb9022ad759ff0f3d282f86c45d076eeaf71c79f813e6697c85a80f0d74884c76
Error: The TLSA record is invalid.
Usage: invalid (9 is not one of 0, 1 or 2)
Selector: invalid (6 is not one of 0 or 1)
Matching Type: invalid (3 is not one of 0, 1 or 2)

B.3 Validation of TLSA records on the DANE mailing list

$ ./swede verify www.ulthar.us
Received the following record for name _443._tcp.www.ulthar.us.: 
Usage: 1 (End-Entity Constraint)
Selector: 1 (SubjectPublicKeyInfo)
Matching Type: 1 (SHA-256)
Certificate for Association: 62d5414cd1c557e3d30ea56d01366e92306e725413c616a51cab4b852c6c70a1c
This record is valid (well-formed).
Attempting to verify the record with the TLS service...
Got the following IP: 68.33.77.0
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
The matched certificate has Subject: /description=465261-v5Y4F7oXeAQ7D/CR=www.ulthar.us/EmailAddress=i.grok@comcast.net

$ ./swede verify lp0.eu
Received the following record for name _443._tcp.lp0.eu.: 
Usage: 1 (End-Entity Constraint)
Selector: 1 (SubjectPublicKeyInfo)
Matching Type: 2 (SHA-512)
Certificate for Association: 490d884c778e9031d8f1bfdb4b6a7673418bad66bcb8115a36ced91laa612b688ae7cc1909bf23391574e41865e41a51a03e0bc18fa6125a5a14c7d2ba5e09f3
This record is valid (well-formed).
Attempting to verify the record with the TLS service...
Got the following IP: 81.2.80.65
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate
The matched certificate has Subject: /CN=proxima.lp0.eu

$ ./swede verify grepular.com
Received the following record for name _443._tcp.grepular.com.:  
Usage: 1 (End-Entity Constraint)  
Selector: 0 (Certificate)  
Matching Type: 1 (SHA-256)  
Certificate for Association: fda20e60d267270d6e009c196b323c33d3eb7bfc8af0e8fe4cf7bf563cb5a55d  
This record is valid (well-formed).
Attempts to verify the record with the TLS service...
Got the following IP: 178.79.145.246  
SUCCESS (Usage 1): Certificate offered by the server matches the one mentioned in the TLSA record and chains to a valid CA certificate.
The matched certificate has Subject: /description=R6zg0XWBOSk1CTp3/C=GB/CN=secure.grepular.com/emailAddress=postmaster@grepular.com