UNIVERSITY OF AMSTERDAM

Faculty of Physics, Mathematics and Informatics
Graduate School of Informatics
System and Network Engineering MSc

DANE verification test suite report

Research Project 1

Hamza Boulakhrif
hamza.boulakhrif@os3.nl

Guido Kroon
guido.kroon@os3.nl

Supervisor:
Michiel Leenaars

April 12, 2015

Abstract

DANE is a protocol that allows certificates used for TLS to be coupled to DNS domain names requiring DNSSEC. This paper describes the specification of DANE (RFC 6698) and the analysis of this specification. The analysis of this specification allowed the team to build a test suite which can test current, but also future DANE implementations. On the basis of this analysis a number of test cases have been derived which are divided into good, bad and grey cases.

The test suite consists of a DNS server within the DNSSEC chain of trust, and a web server that provides certificates which DANE implementations are able to validate. Within the DNS zone, the test suite provides a number of examples, each with their own descriptions as to why these examples are good, bad or grey, following the original DANE RFC 6698. Testing DANE implementations against these examples can be insightful how the DANE implementation behaves when trying to validate these examples.

After building the test suite, the team also tested DANE implementations to see how existing tools react to the test suite. Some DANE implementations seem to DANE validate the examples better than others. The test suite is designed to be independent of existing DANE implementations, meaning that newer DANE implementations can also be tested against the test suite.

Keywords – dane, dns, dnssec, bind, ldns, ldns-dane, ssl, tls, gnutls.

Contents

Chapter 1

Introduction

Most encrypted forms of communication on the Internet nowadays use Transport Layer Security (TLS). TLS is used as a means to validate a server's certificates to which the clients connect to. In order to validate these certificates, the Internet relies on trusted third parties called Certificate Authorities (CAs) that cryptographically sign these certificates. The client then checks if the certificate, which it is presented by the server, or end entity (EE), is indeed a valid certificate of the CA it has been signed with.

DNS-Based Authentication of Named Entities (DANE) is a new standard by the Internet Engineering Task Force (IETF) [4]. It is used as an improvement to validate the aforementioned secure servers by validating a server's certificate, which is now stored as a TLSA resource record (TLSA RR) in the Domain Name System (DNS) zone of that domain. If these zones are part of the DNS Security Extensions (DNSSEC) chain of trust, the validity of these TLSA RRs can be verified as the records are signed by trusted keys within the DNSSEC chain of trust.

This project has provided a test-suite to test the current DANE-tools against the DANE specification as per Request for Comments (RFC) 6698 [4].

Currently there is no way to verify that the current DANE-tools correctly verify DANE implementations. This project aims to provide a test suite that one can use to check if these DANE-tools indeed verify DANE implementations correctly.

1.1 Research question

The overall discussion of the problem produced the following research question:

Can a test suite be devised to allow developers and implementers to validate the reliability and consistency of an implementation of DANE, and its ability to correctly handle unforeseen input or deviations from the official TLSA syntax as per RFC 6698?

1.2 Previous and related research

Few research has been done on DANE, which is likely due to the fact that very few domains even support DNSSEC [1]. In 2011/2012, Miguel Medeiros Correia and Mustafa Tok carried out a case study on DANE, as part of their Computer Security MSc study [2] at the University of Porto, which describes this new way to authenticate named entities. However, most of their information reflects the original RFC 6698 as they only describe how DANE works, together with explanations on PKI, DNS, DNSSEC and TLSA RRs.

Pieter Lexis, at the time a Dutch graduate student at the University of Amsterdam, created a DANE implementation in 2012 [5]. This implementation, which is called `swede`, can be used to test and verify TLSA RRs in a domain. As this may seem very similar to this research, it should be noted that Lexis created an implementation, whereas this research created a test suite to check such implementations.

VeriSign has written an paper about how the X.509 attack surface can be reduced by using DANE as a new means to validate certificates, stating that *"some of the fundamental problems that exist with today's CA model"*. Even though VeriSign themselves are a CA, they do recognise the problems that this current CA model poses, like that every system and application needs to keep track and decide which CA root certificates it will trust and which not to trust. There are no prescriptions as to what CAs should or should not be included in these collection, thus resulting in different collections per application.

VeriSign [8] and NIST [7] have also provided similar DANE test suites. VeriSign's test suite is a rather basic one that only tests 4 different use cases, whereas NIST's test suite is more extensive.

1.3 Scope

The scope of this project is to check the proper functioning of DANE-tools as per RFC 6698 [4] and RFC 6394 [3]. NLnet explicitly underlined the fact that they want their own test suite that is built directly after these original specifications. This as opposed to built upon the already existing test suites as mentioned in section 1.2, which have been developed after the interpretations of these specifications by their developers.

The goal is to provide an extensible test suite for DANE implementations. This means that this research is not about creating a new DANE implementation, but actually testing the proper functioning of already existing, as well as future DANE implementations. The test suite is to verify as many as possible types of variations for TLSA RRs which are the new RRs introduced by DANE. And by making the test suite extensible allows future research to add additional use cases. According to specifications RFC 6698 [4] and RFC 6394 [3], a TLSA RR consists of the following required fields:

1. Certificate Usage Field;
2. Selector Field;
3. Matching Type Field;
4. Certificate Association Data Field.

Note that the first field of the TLSA RR is the `Certificate Usage Field`, which supports four different Certificate Usages, as described in specification RFC 6698. Due to the limited time and the high amount of variations one can generate, the project members only chose to test Certificate Usages 1, 2 and 3, thus omitting Usage 0.

However beyond the scope of this project, checking the correct working of DANE-tools should ultimately be performed by also:

1. (Re)writing of certain DANE-tools when applicable;
2. (Re)compiling of DANE-tools from source on different platforms, as this may introduce unexpected complications when, for instance, using different compiling flags, compilers and libraries.

The verification results of DANE-tools should then be compared to the expected results according to the specification, RFC 6698 [4].

1.4 Research Approach

The team analysed RFC 6698 [4] before performing further research. The team then deployed a test environment in order to build the test suite by closely following RFC 6698 [4]. The test suite is subdivided into three categories which are good, bad and grey. The good and bad test cases are as the words imply good and bad examples of DANE which should respectively validate correctly and incorrectly. The grey test cases can either validate as good or bad dependent on the interpretation of the DANE specification. A number of test cases have been devised for testing purposes of DANE implementations. The test suite itself consists of a DNS authoritative domain with all necessary RRs, and a web server that provides corresponding certificates to test. The team also roughly analysed some DANE implementations to improve the test suite.

Chapter 2

Background

This chapter introduces the DNS-based Authentication for Named Entities (DANE). It describes how DANE works, what its application is, what it relies on, what services make use of it, as well as additional background information.

The DANE protocol was developed to improve the TLS authentication [4]. DANE improves on that by enabling the administrators of domain names to specify the certificates used in TLSA RRs on the DNS servers.

2.1 TLS

TLS allows secure communication channels between client and server applications which prevents eavesdropping, tampering and forgery over the Internet. This is accomplished by a server application that provides a signed certificate to a client application. The client application then uses the (Trusted) Certification Authority's (CA) certificate (that is already present) to verify this certificate.

2.2 DANE

DANE introduces the new TLSA RR type which is used to associate a TLS server certificate with the domain name where the RR is found. The TLSA RR consists of the following fields:

	Usage	**Selector**	**Matching Type**	**Association Data**
Size	8-bit	8-bit	8-bit	Dependent on Selector and Matching Type values.

Table 2.1: TLSA RR format.

2.2.1 Certificate Usages

The Certificate Usages field is an 8-bit unsigned integer which allows a total of 256 different usages. Currently there are four Certificate Usages that can be used for different purposes in a TLSA RR:

0. CA constraint;
1. Service certificate constraint;
2. Trust anchor assertion;
3. Domain-issued certificate.

CA constraint

CA constraint (also known as "usage 0") is used to specify which CA certificate can be used to validate an end entity (EE) certificate. This means that a TLSA record can specify which CA can issue certificates for its domain.

Service certificate constraint

Service certificate constraint also known as "usage 1" is used to specify an EE certificate that must be matched with the certificate that is provided by server application. The certificate of the server must also pass PKIX certification validation.

Trust anchor assertion

Trust anchor assertion also known as "usage 2" is used as a trust anchor when validating the EE certificate. This usage allows a domain administrator to specify a new trust anchor which is specified in a TLSA RR. The certificate that is presented by the server application must then PKIX validation using the new trust anchor.

Domain-issued certificate

Domain-issued certificate also known as "usage 3" is used for a domain-issued/self-signed certificate which does not require a (third party) CA. The certificate that is presented by the server application must match the TLSA RR.

Table 2.2 and figure 2.1 below show a summary of the usages and how they DANE validate.

Value	Name	Description
0	CA constraint	Public CA from PKIX tree
1	Service certificate constraint	End-Entity Certificate and PKIX
2	Trust anchor assertion	Private CA from PKIX tree
3	Domain-issued certificate	End-Entity Certificate

Table 2.2: Summary DANE usages

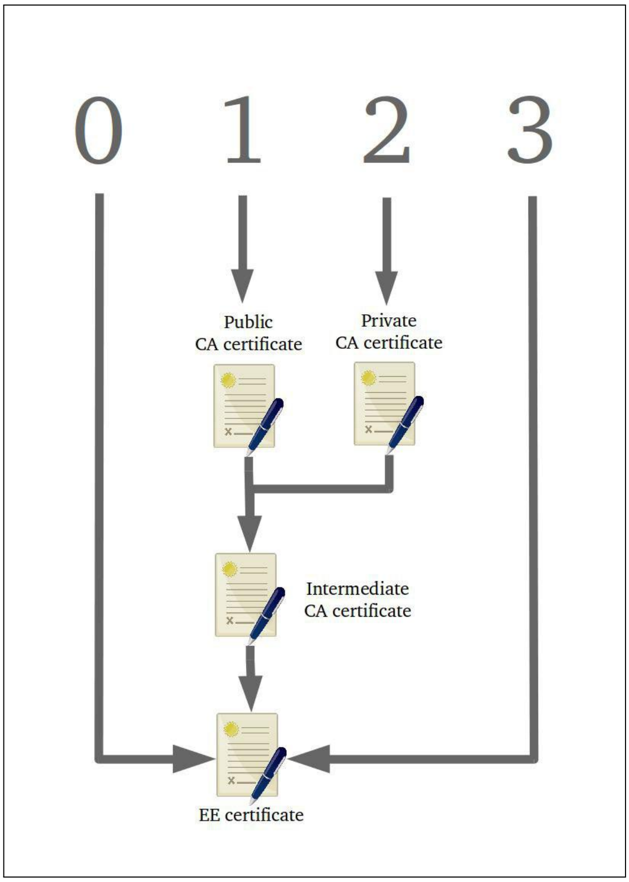

Figure 2.1: DANE Usages

2.2.2 Selectors

The usages described previously can be combined with either of two usages:

0. Full certificate;
1. SubjectPublicKeyInfo.

Full certificate means that the complete certificate in binary format is used before it is passed through the matching type in subsection 2.2.3. In case of SubjectPublicKeyInfo, the public key in binary format of the certificate is DER encoded before it is passed through the matching type in subsection 2.2.3.

2.2.3 Matching Type

As of the current RFC of DANE, three Matching types exist:

0. No hash, exact match;
1. SHA-256;
2. SHA-512.

Matching type 0 where no hash is used, uses the output that comes from the selector and uses this for the RDATA of the TLSA RR. Matching type 1 and 2 respectively put a SHA-256 and SHA-512 Digest from the output of the selector in the RDATA of the TLSA RR.

2.2.4 TLSA RR examples

In order to clarify what subsections 2.2.1, 2.2.2 and 2.2.3 describe, a couple of examples will be given.

TLSA RR example

```
_443._tcp.dane.internet.nl. IN TLSA (
    0 0 1 d2abde240d7cd3ee6b4b28c54df034b9
          7983a1d16e8a410e4561cb106618e971 )
```

The example shown above is a DNS TLSA RR for a TCP service that uses a port number of 443, which uses the domain name `dane.internet.nl`. The RDATA at the end of the TLSA RR is a SHA-256 of the complete self-signed certificate. This can be derived from the option fields behind "TLSA" as described in previous subsections. The 3, 0, 1 stand respectively for the usage, selector and matching type.

This TLSA RR can be used for, say, a web server which provides a web page over HTTPS. When a user requests the web page `dane.internet.nl`, TLS has to be set up first. The web server presents the self-signed certificate to the user. The client application which is in this case a web browser also requests the TLSA RR for `dane.internet.nl` in DNS. Depending on the TLSA RR the DANE implementation performs (a couple of) operations. In case of the TLSA RR Example a SHA-256 fingerprint of the complete certificate is made and matched against the RDATA in the TLSA RR. Only in case of a match a secure connection can be performed with the web server.

2.3 DNSSEC

As described previously, TLSA RRs are part of DNS and need to be queried in order to perform DANE validation. In order to perform DANE operations, DNSSEC needs to be in place for the authenticity of the RRs. This means that before DANE operations are performed, DNSSEC should validate the chain of trust first.

2.4 DANE Implementations

DANE is a relatively new protocol and appears to be supported by a low number of applications [10]. DANE should eventually provide support for HTTP, SMTP, SIP, XMPP, RTP, IMAP, PGP, SSH and other critical protocols that Internet users depend upon. This should enable more secure voice, video, chat, email and other communication [6].

Chapter 3

Experiments

The experiments have been performed within the test environment in a DNS name server and a web server. The DNS server has been filled with three sub domains, namely `good`, `bad` and `grey`. Within the `good` sub domain, all test cases must be validated by DANE implementations as valid. Likewise, within the `bad` sub domain, all test cases must be validated by DANE implementations as invalid. The `grey` sub domain contains test cases which could be validated both as valid and as invalid, depending on how the developer of the DANE implementation interpreted the RFC.

Underneath the bad sub domain, a special `unsigned` sub domain has been created which is not part of the DNSSEC chain of trust. All other sub domains are part of the DNSSEC chain of trust.

Within each sub domain, several A RRs were created, each with their matching TLSA RR containing the certificate information to validate the certificate that the web browser receives when connecting to a sub domain web page.

The test suite is devised by systematically analysing the RFC. This allowed the team to create possible test cases per element that the team came across. The test suite only tests `Usage` 1, 2 and 3, both `Selector Types` and all three `Matching Types`. Combining them, all their possible permutations were created, each with separate certificates, which DANE implementations can validate. Also some other test cases were created to validate the validity of the certificate itself. For instance, their expiration dates, or to check if they have been signed by the right CA. For a more detailed overview of the test suite, see Appendix A.

After the test suite had been designed, the team tested some existing DANE implementations (`GnuTLS`, `ldns-dane` and the `DNSSEC/TLSA Validator`) to see if these tools also correctly validated all of the test cases.

Chapter 4

Results

The test suite is designed to test DANE implementations on a number of pitfalls, by placing deliberately placed good, bad and grey examples. Each example has a description, which explains why the example is good, bad or grey, giving the user some insight in why and how their DANE implementation behaves in a certain way when tested.

For the test suite, the team created three pitfall categories:

1. Bad;
2. Good;
3. Grey.

The good and bad categories contain examples that should either be validated as good or bad, depending on checks like matching good Certificate Usage, Selector and Matching Type fields. The grey category provides some test examples that verify or fail validation, depending on how the DANE implementation handles certain discrepancies that are not clear in the DANE specification (like valid TLSA RRs but bad certificates).

4.1 DANE test case analysis

It requires in depth knowledge of DANE in order to create test cases that can support developers and implementers of DANE implementations. To achieve an optimal and objective test suite, the aforementioned specification is mainly used as the reference. For more clarity of matters RFC 6394 [3] is also used which contains use cases of DANE.

The results of analysing RFC 6698 [4] resulted in the following test cases:

- (Non-)existing Usages;
- (Non-)existing Selectors;
- (Non-)existing Matching types;
- Combination of Selector and Matching type incorrect;

- (In)correct hash (type);
- Expired certificates;
- Unsigned DNSSEC chain;
- Wildcard usage in RR;
- Incorrect signed certificates.

All these test cases are published online[1]. This website can be visited for a detailed view of the test cases.

4.2 Analysis of DANE implementations

This section is a rough analysis of DANE implementations which have been used to test against the test suite. The team decided to analyse the following implementations:

1. GnuTLS (v3.3.11);
2. ldns-dane (v1.6.17);
3. DNSSEC/TLSA Validator (v2.2.0.1).

4.2.1 GnuTLS

For this project the team decided to use the most recent version of GnuTLS, version 3.3.11.[2] Unfortunately there were no packages for the latest version on FREEBSD, nor Ubuntu or Debian. The team then compiled the GnuTLS binaries themselves, with DANE support compiled in, on a Debian x64 machine, as well as on an Ubuntu 14.04 x64 machine.

GnuTLS' DANE can be used in two ways, namely by making use of `gnutls-cli --dane`, and by making use of the `danetool`. Neither of them however check PKIX validation when testing a TLSA RR with Usage 0, 1 or 2 which require this validation; they only check if the certificate matches the TLSA RR. The following example shows this when testing `falsecacert.bad.dane.internet.nl`, which should fail if GnuTLS also does PKIX validation.

danetool without PKIX validation

```
# danetool --check falsecacert.bad.dane.internet.nl
Resolving 'falsecacert.bad.dane.internet.nl'...
Obtaining certificate from '185.49.141.29:443'...
Querying DNS for falsecacert.bad.dane.internet.nl (tcp:443)...
_443._tcp.falsecacert.bad.dane.internet.nl. IN TLSA ( 01 00 01 e
f2bc46a93cc5f17a054ac9a06e0b1b98061896f0f288d1826e8634834e3d1ca
)
Certificate usage: End-entity (01)
Certificate type:  X.509 (00)
Contents:          SHA2-256 hash (01)
```

[1] dane.internet.nl
[2] `http://gnutls.org/news.html`

```
Data:              ef2bc46a93cc5f17a054ac9a06e0b1b98061896f0f288
d1826e8634834e3d1ca

Verification: Certificate matches.
```

It should be noted that the developer of GnuTLS intentionally chose to omit the PKIX validationGnuTLS) [6] and therefore also fails to recognise an expired certificate. The developer of GnuTLS did not mention the reason for this omission.

4.2.2 ldns-dane

The `ldns-dane` tool is part of the `ldns` tools, created by NLnet Labs, which can also be used to test DANE implementations. When using `ldns-dane` to test Usage 0 and 1, the tool relies on the user to manually add the certificate of the Certificate Authority which signed the certificate of the EE. If this is not manually specified, the tool correctly fails to validate. For example, when trying to validate `falsecacert.bad.dane.internet.nl`, it fails because it could not PKIX validate (desired result).

`ldns-dane with proper PKIX validation (fail)`

```
$ ldns-dane verif falsecacert.bad.dane.internet.nl 443
185.49.141.29 did not dane-validate, because: Could not PKIX val
idate
2a04:b900:0:100::29 did not dane-validate, because: Could not PK
IX validate
```

Even when specifying the `dane.internet.nl` CA certificate file, because `falsecacert.bad.dane.internet.nl` is signed with a false CA certificate (desired result):

`ldns-dane with proper PKIX validation (fail)`

```
$ ldns-dane verif falsecacert.bad.dane.internet.nl 443 -f Deskt\
op/dane.internet.nl.crt
185.49.141.29 did not dane-validate, because: Could not PKIX val
idate
2a04:b900:0:100::29 did not dane-validate, because: Could not PK
IX validate
```

It also fails to validate `100.good.dane.internet.nl`, when the CA certificate is not specified (desired result):

`ldns-dane with proper PKIX validation (fail)`

```
$ ldns-dane verif 100.good.dane.internet.nl 443
185.49.141.29 did not dane-validate, because: Could not PKIX val
idate
2a04:b900:0:100::29 did not dane-validate, because: Could not PK
IX validate
```

However, it does correctly validate when you manually specify the CA certificate (desired result):

ldns-dane with proper PKIX validation (valid)

```
$ ldns-dane verif 100.good.dane.internet.nl 443 -f dane.interne\
t.nl.crt
185.49.141.29 dane-validated successfully
2a04:b900:0:100::29 dane-validated successfully
```

Testing an expired certificate with `expired.bad.dane.internet.nl` also seems invalid. This is a desired result, but it does not indicate that the certificate was expired; only that it could not PKIX validate.

4.2.3 DNSSEC/TLSA Validator

The DNSSEC/TLSA Validator tools are add-ons/extensions to the Firefox and Chromium browsers which enables the browser to DANE validate a website. While it does check if a website's certificate matches the TLSA RR, the web browser still shows a warning whenever a web site's certificate is not signed by a trusted Certificate Authority, which is not necessary with Usage 1, 2 and 3. It would be nice to see if the browser would not show this warning, as long as the website DANE validates with the correct Certificate Usage.

It also claims to PKIX validate, but instead only checks if the TLSA RR matches the certificate. Figure 4.1 shows that a certificate is being DANE-validated with its corresponding TLSA RR, but not with its corresponding CA certificate to validate PKIX which is necessary for Certificate Usage 1 (referred to as "type 1" by the add-on).

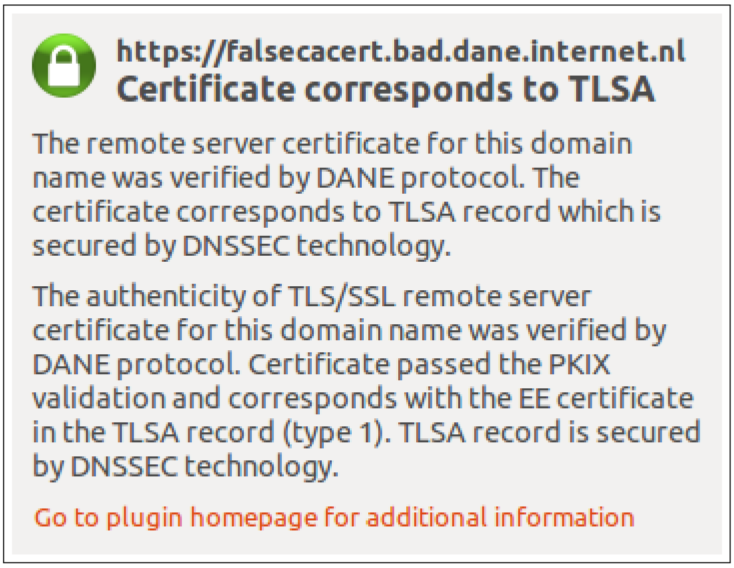

Figure 4.1: DNSSEC/TLSA Validator without proper PKIX validation.

Testing an expired certificate with `expired.bad.dane.internet.nl` also seems valid according to the DNSSEC/TLSA Validator.

As described in appendix A the test suite is deployed in a BIND authoritative name server. It is also good to note that BIND also performs checks and corrections on zone files which limited the research group in some cases.

Chapter 5

Conclusion

This chapter describes the conclusion based on the research performed.

5.1 DANE test suite

RFC 6998 about DANE is analysed in order to build the test suite. It is completely dependent on the RFC what good and bad test cases are for the test suite. It has become clear that the RFC leaves room to be interpreted in different ways which makes place for another category, namely grey. The latter case makes building a test suite more difficult.

The (generic) test suite that is developed (see appendix A) during this research, tests a number of cases that can be used by developers and implementers of DANE. It is also a test suite that is extensible which means that more test cases can be added over time.

The great majority of the cases in the test suite are clearly good or bad cases. As of this research there were just a few grey cases which could be interpreted in such a way that they could be both good and bad. Altogether the test suite provides developers and implementers a good and reliable way to test their DANE implementations for a number of test cases that are described in chapter 4.

As also mentioned in chapter 4, BIND limits the test suite because of its checking and correcting behaviour. Although a great number of test cases are covered, this didn't allow the researchers to create RR test cases that exceed BIND's limitations.

5.2 DANE Implementations

5.2.1 GnuTLS

GnuTLS is thus far the only [9] TLS implementation supporting DANE but the creator of GnuTLS intentionally [6] chose to omit PKIX validation. However, the validation of PKIX is crucial for DANE validating with Certificate Usage 0, 1 and 2. It would therefore be a good idea to implement PKIX validation for proper DANE support. It would also be good to see some kind of warning message, stating that it explicitly does not check PKIX validation for the time being.

5.2.2 ldns-dane

The `ldns-dane` tool from NLnet Labs performs DANE validation rather well, also with PKIX validation. However, the error messages are not always clear what went wrong if PKIX validation did not succeed. It would be a good idea to receive some more verbosity in the output when validating to see how the implementation comes to the conclusion that it did or did not validate a named entity successfully.

5.2.3 DNSSEC/TLSA Validator

DNSSEC/TLSA Validator is perhaps disappointing in the way it does its DANE validation. A web browser add-on would be a great addition if it truly enhances the user experience when browsing DANE-enabled websites, without the current warnings whenever a certificate is not signed by a trusted CA. However, the browser still shows these warning, even with the add-on installed. Perhaps even more disappointing is the fact that it claims to PKIX validate a certificate, while in fact it simply fails to do so. It would be nice to see this add-on being further developed to increase the user experience, as well as properly PKIX validating certificates. Perhaps a more elegant solution would be to add DANE support natively in the web browsers, because this avoids that users have to install the validator manually.

5.3 Future work

As previously scoped, the test suite does not cover all bases in order to fully test various DANE implementations. For example, only Certificate Usages 1, 2 and 3 are covered in the test suite, thus omitting Certificate Usage 0. The test suite should therefore be expanded to also include test examples to check TLSA RRs and certificates with Certificate Usage 0.

Also when applicable, all of the DANE implementations should ultimately be checked when compiling from source on different platforms, using different compiling flags or libraries in order to verify for proper functioning of the implementation.

As previously described, BIND has a checking and correcting behaviour. In some cases it does not allow bad TLSA RRs or corrects them automatically. As this test suite is also built extensibility in mind, a solution is recommended to go beyond the limitations that BIND poses.

Furthermore, beside the test suite there are also suggestions for future work concerning the DANE implementations. Analysis of source code of these tools is recommended in order to improve these implementations even more. But also provide these tools with full support of DANE as described in [4] without the omission of mandatory options and features.

Chapter 6

Discussion

It has come to the team's conclusion that Certificate Usage 2, as described in RFC 6698, can be interpreted in a few different ways. Asserting a new Trust Anchor without expecting the client to have the CAs certificate in its collection of trust anchors, may introduce a challenge for the client to perform PKIX validation, which is required for Certificate Usage 2. As the client initiates a new connection to a server, it receives the EE certificate of the server, but should also receive all other "higher" CA certificates in order to do PKIX validation. Depending on the DANE implementation it should add these new root CA certificates to its collection of trust anchors, as long as these certificates DANE validate. The EE certificate can then be PKIX validated as well, therefore eliminating the need to change the TLSA record again whenever a server's certificate is changed.

The team feels that this lack of clarity of Certificate Usage 2 may have to do with how Usage 2 is currently specified inRFC 6698. There is also errata for this RFC which emphasises this problem.[1] The errata points out that whenever a client initiates a connection, it should receive and accept all necessary certificates in order to PKIX validate the connection. The reason for this is that some services do not provide the user much interaction to accept these certificates manually (like SMTP) and should therefore be imported in the client's collection automatically.

that use Certificate Usage 1 rely on the Root CA's certificate that is needed in order to PKIX validate. But as long as the client does not have this root CA certificate in its collection, it cannot PKIX validate, meaning that these connections cannot be DANE validated. Some implementations adhere to this more strictly than others. For instance, the `ldns-dane` implementation correctly fails DANE validation of end-entities with Certificate Usage 0, 1 and 2 if it does not have the root CA certificate in order to PKIX validate. Other implementations either intentionally choose to omit the PKIX validation (`GnuTLS` [6]), while others (`DNSSEC/TLSA Validator`) seem to claim that it does PKIX validate, while it in fact does not.

This basically comes down to that Certificate Usage 0 and 2 are both very similar. The only difference is that Usage 0 only works when the CA certificate is already part of the client's CA collection in order to PKIX validate. Usage 2

[1] `www.rfc-editor.org/errata_search.php?rfc=6698`

adds to this that the server should also hand the client all other CA certificates, as part of the TLS handshake. However, a server could be configured to hand all these certificates either way, regardless of what Certificate Usage has been configured in the TLSA RR. If that is the case, then there's no real distinction between Certificate Usage 0 and 2. In the team's opinion, there is only a distinction between Certificate Usage 0 and 2 when, for Usage 0, a client only accepts the EE certificate, and not accepting all others it may receive in order to PKIX validate. However, this is nowhere specified in RFC 6698, and it is therefore open for interpretation.

Bibliography

[1] Frederic Cambus. StatDNS - DNS and Domain Name statistics. http://www.statdns.com/, January 2015.

[2] M.M. Correia and M. Tok. DNS-based Authentication of Named Entities (DANE). http://web.fe.up.pt/~jmcruz/ssi/ssi.1112/trabs-als/final/G7T12-digit.cert.altern-final.pdf, January 2015.

[3] Internet Engineering Task Force. Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE). https://tools.ietf.org/html/rfc6394, October 2011.

[4] Internet Engineering Task Force. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. https://tools.ietf.org/html/rfc6698, August 2012.

[5] Pieter Lexis and Bert Hubert. Implementing a dane validator. Technical report, Technical report, University of Amsterdam (February 2012), http://staff. science. uva. nl/˜ delaat/rp/2011-2012/p29/report. pdf, 2012.

[6] Nikos Mavrogiannopoulos. [gnutls-devel] DANE validation. http://lists.gnutls.org/pipermail/gnutls-devel/2013-February/006145.html, January 2013.

[7] National Institute of Standards and Technology. NIST High Assurance Domain (HAD) Project. https://www.had-pilot.com/tlsa-test.html, February 2014.

[8] VeriSign. Verisign Labs DANE/TLSA Demonstration. http://dane.verisignlabs.com/, January 2014.

[9] Wikipedia. Comparison of TLS implementations. https://en.wikipedia.org/wiki/Comparison_of_TLS_implementations#Certificate_verification_methods, January 2015.

[10] Wikipedia. DNS-based Authentication of Named Entities. https://en.wikipedia.org/wiki/DNS-based_Authentication_of_Named_Entities#Support, January 2015.

Appendix A

Environment

This section describes the environment during this project. The team has been assigned a FREEBSD 9.3 jail to build the test suite in. Eventually the test suite was built using one system, using the following environment (see table A.1):

Operating system	FreeBSD 9.3 (jail)
Name server	BIND 9.9.5 (ESV[1])
Web server	Apache 2.4.10

Table A.1: Environment

See table A.2 for the IP-addresses that were assigned to the test suite.

IPv4	185.49.141.29
IPv6	2a04:b900:0:100::29

Table A.2: IP-addresses

A.1 Test suite configuration

For the name server, the team has been assigned `dane.internet.nl` as the domain. The team created a zone for this domain to which the server could respond authoritative data for. Then the team setup DNSSEC and sent the DNSKEY to NLnet Labs so that they could add `dane.internet.nl` to the DNSSEC chain of trust.

See figure A.1 for a graph tree representation of the `dane.internet.nl` domain.

The `bad`, `good` and `grey` sub domains are all part of the DNSSEC chain of trust (except for the `unsigned.bad` sub domain). Within each sub domain, the team generated separate TLSA RRs which could then be tested individually.

For the web server, the team compiled the latest stable Apache web server, version 2.4.10, with the SSL module.[2] Then for each sub domain, a new wildcard TLS certificate was generated, which will match every underlying TLSA RRs.

[1]Extended Support Version

[2]`https://httpd.apache.org/`

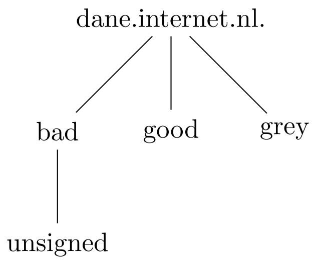

Figure A.1: DNS domains.

A wildcard TLS certificate saves generating separate TLS certificates for every different TLSA RR. The web server is serving several name-based virtual hosts, each with a different TLS-certificate to match to each TLSA RR.

Underneath each sub domain, the following RRs are created:

bad	**good**	**grey**
falsecacert	100	expired
falseeecert	101	
hash-md5	102	
hash-sha1	110	
hash-sha256	111	
hash-sha256-2	112	
m300	300	
m301	301	
m30255	302	
m303	310	
s312	311	
s322	312	
s32552	wildcard	
u002		
u102		
u202		
u25502		
u402		
unsigned		

Table A.3: Virtual hosts, each with their own certificate and TLSA RR.

Glossary

BIND Berkely Internet Name Domain (BIND) is open source software that implements the Domain Name System (DNS) protocols for the Internet. 13, 14, 15, 19

CA Certificate Authority or Certification Authority (CA) is an entity that issues digital certificates. A digital certificate certifies the ownership of a public key by the named subject of the certificate. This allows others to rely upon signatures or on assertions made by the private key that corresponds to the certified public key. In this model of trust relationships, a CA is a trusted third party who is trusted both by the owner of the certificate and by the party relying upon the certificate. 2, 3, 5, 6, 9, 12, 13, 15, 16

Debian Debian is a GNU/Linux distribution used for servers, desktops and embedded platforms. 11

DER DER is a restricted variant of BER for producing unequivocal transfer syntax for data structures described by ASN.1.. 7

Digest A hash, or digest, is a cryptographic on-way function used to uniquely identify data. By rehashing the same data using the same hash function, one can compare the new has to the old hash, which must be identical to verify the data's integrity. Hashes therefore function a lot like fingerprints.. 7

DNS Domain Name System (DNS) is an Internet service that translates domain names into IP addresses. The functionality of DNS is extended over the years and also performs IP addresses to domain names translation and many other functions. 2, 4, 5, 8, 9

DNSSEC The Domain Name System Security Extensions (DNSSEC) is for securing certain kinds of information provided by the Domain Name System (DNS). 2, 8, 9, 11, 13, 19

EE End Entity (EE) is a certificate which is not used to validate signatures on other certificates. It is rather a certificate that appears at the end of the certificate path. An entity participates in the Public Key Infrastructure. Usually a Server, Service, Router, or a Person.. 2, 6, 12, 16

FreeBSD FreeBSD is a Unix-like operating system used for servers, desktops and embedded platforms. 11, 19

HTTP HyperText Transport Protocol (HTTP) is an application protocol which is the foundation of data communication on the Internet.. 8

HTTPS HyperText Transport Protocol Secure (HTTPS) is the result of securing the HyperText Transport Protocol with an additional layer which is the Transport Layer Security. 8

IETF Internet Engineering Task Force (IETF) develops and promotes voluntary Internet standards. It is an open standards organisation, with no formal membership or membership requirements. 2

IMAP Internet Message Access Protocol (IMAP) is a protocol for e-mail retrieval and storage. It allows an e-mail client to access e-mail on a remote mail server. 8

NIST National Institute of Standards and Technology (NIST) is the federal technology agency that works with industry to develop and apply technology, measurements, and standards.. 3

PGP Pretty Good Privacy (PGP) is a data encryption and decryption computer program that provides cryptographic privacy and authentication for data communication. 8

PKI Public Key Infrastructure (PKI) is a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates. 2

PKIX Public Key Infrastructure for X.509 (PKIX) is a standard for Public Key Infrastructure. 6, 11, 12, 13, 14, 15, 16

RDATA RDATA is the field that contains the actual data for a resource record entry in the Domain Name System (DNS). 7, 8

RFC A Request for Comment (RFC) is a publication of the Internet Engineering Task Force (IETF) and the Internet Society. 2, 3, 4, 7, 9, 10, 14, 16

RR A Resource Record (RR) is a data element that defines the structure and content of the domain name space. 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 16, 19, 20

RTP Real-time Transmission Protocol (RTP) is a network protocol for delivering audio and video over IP networks. 8

SIP Session Initiation Protocol (SIP) is a communications protocol for signalling and controlling multimedia communication sessions. 8

SMTP Simple Mail Transport Protocol (SMTP) is for e-mail transmission. 8, 16

SSH Secure SHell (SSH) is a cryptographic network protocol for initiating text-based shell sessions on remote machines in a secure way. 8

TCP Transmission Control Protocol (TCP) is a connection oriented, byte stream, transport protocol which is used in applications for reliable transmission of data. 8

TLS Transport Layer Security (TLS) is a cryptographic protocol to provide communications security over a (public) network. 2, 5, 8, 14, 16, 19

TLSA TLSA does not stand for anything, it is the name of the RRtype in the Domain Name System. 2, 3, 5, 6, 7, 8, 9, 10, 11, 13, 15, 16, 19

Ubuntu Ubuntu is a GNU/Linux distribution used for servers, desktops and embedded platforms. 11

XMPP Extensible Messaging and Presence Protocol (XMPP) is a communications protocol for messaging based on XML (Extensible Markup Language).. 8