Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 197 (2008) 91–103

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Trusted Directory Services for Secure Internet Connectivity

Transport Layer Security using DNSSEC

J.F. Zandbelt[1]

SURFnet, Utrecht, The Netherlands

R.J. Hulsebosch and M.S. Bargh

Telematica Instituut, Enschede, The Netherlands

R. Arends

Nominet UK, Oxford, United Kingdom

Abstract

The Internet today is a highly dynamic environment which frequently requires secure communication between peers that do not have a direct trust relationship. Current solutions for establishing trust often require static and application-specific Public Key Infrastructures (PKIs). This paper presents trusted directory services as a key infrastructural technology for setting up secure Internet connections, providing an alternative to application-specific PKIs. The directory securely binds public keys to peers through their names in a flexible way that matches the dynamic nature of the Internet. We elaborate on this concept by showing how the Domain Name System (DNS) and its security extensions (DNSSEC) can be leveraged for establishing secure Transport Layer Security (TLS) connections in a dynamic way. A simple enhancement of the TLS protocol, called Extended TLS (E-TLS), required for this purpose, is proposed. We describe our E-TLS implementation and we conclude with an evaluation of our results.

Keywords: Public Key Infrastructures, secure DNS, Transport Layer Security, trusted directory services.

1 Introduction

During the second half of the previous century the number of sensitive business communications and financial transactions increased rapidly. These communications often involved parties from different organizations that had not communicated previously nor had the chance to exchange shared keys for symmetric encryption.

[1] Email: Hans.Zandbelt@SURFnet.nl

doi:10.1016/j.entcs.2007.12.019

In 1976 Diffie, Hellman and Merkle presented Public Key Cryptography (PKC) [1] as a solution for securing communications between two parties that did not have the chance to exchange shared encryption keys beforehand. A crucial requirement for PKC to work, not properly addressed at the time of invention, is that public keys must be associated with their users in a trusted (i.e. authenticated) manner, for instance by a trusted third-party (e.g. in the form of signed certificates) or in some kind of trusted directory. An infrastructural technology that addresses binding public keys to parties and the distribution of those keys is generally called Public Key Infrastructure (PKI). Although PKC is widely recognized and adopted as key technology for secure communications on the Internet, the key distribution and binding issue described above and other drawbacks, such as summarized in [2], have limited the success of many application or domain-specific PKIs.

This paper proposes an alternative approach for public key binding and distribution by leveraging the existing Internet Domain Name System (DNS) and its security extensions (DNSSEC) for creating an infrastructural solution for dynamic trust establishment. Our proposal is not tied to a specific application but can be reused across a variety of applications. It allows for establishment of secure communication channels between peers that have no prior relationship and thus accommodates for the distributed nature of trust management on the Internet. The research questions that our approach addresses can be summarized as: how to address the major shortcomings of traditional PKC/PKIs that use Certificate Authorities and statically configured root certificates, how to support dynamic secure connection setup between peers that do not have a prior relationship, using a trusted directory service, and how to leverage generic and/or existing Internet infrastructural services to implement a directory service that securely binds public keys to parties and that can be used for the distribution of those keys.

The remainder of this paper is organized as follows: section 2 presents an overview of related work in this area. Section 3 elaborates on the concept of using trusted directories in secure communications and section 4 describes the application of this concept using DNSSEC for establishing TLS connections. Section 5 describes our prototype implementation, followed by an evaluation of our approach in section 6. Finally we present our conclusions in section 7.

2 Related Work

Although much related work exists on key authentication and distribution issues, we focus here on approaches that apply directory services to address the problem. The ISO X.500 recommendation [3] was to be a global, distributed database of named entities also known as a global on-line telephone book or directory service. The ISO X.509 recommendation [4] was published as part of X.500, defining X.509 certificates that were used to bind public keys to X.500 path names (called Distinguished Names). The X.500 approach requires a single, global naming discipline and as of today there are too many legacy naming systems in use that can not be simply replaced or united in a single discipline. Therefore the X.500 idea of a single, globally unique name that everyone could use when referring to an entity, is not likely to occur. One notable exception to this rule is the DNS [5] for naming Internet resources as we will focus on in this paper. The X.509 certificates however have become a commodity on the Internet today, binding public keys to names.

Jones et al. proposed the Internet Key Service (IKS), which is a distributed architecture for authenticated distribution of public keys, layered on DNSSEC [6]. Clients use DNSSEC to securely discover the identities of the relevant IKS servers, and send key lookup or management requests directly to these servers using a special-purpose protocol. We believe that the suitability and scalability of the DNS itself does not justify the development of a new protocol and system. Moreover our approach also addresses the operational aspects of the system through actual applications rather than focusing on the system itself.

IPsec can use opportunistic encryption and rely on DNSSEC as a keying mechanism within the Internet Key Exchange (IKE) protocol [7]. The objective of opportunistic encryption is to allow encryption without having any pre-arrangements in place that are specific to the systems involved. Each system administrator adds public key information to DNS records to support opportunistic encryption and then enables this feature in the nodes' IPsec stack. Once this is done, any two such nodes can communicate securely by retrieving a KEY Resource Record (RR) from DNSSEC to be used for communicating with the host or identity associated with the name. This is similar to the solution that we propose in this paper, but we use DNSSEC for building opportunistic TLS channels.

Le and Guyennet propose a solution to grid applications' specific needs in data transport security in [8]. To address the key management part, the solution secures the local DNS server with the DNSSEC extensions, making it become a local certification authority. This design choice solves a dual problem: secret key distribution and scalability of the solution. The authors have strictly focused on setting up SSH sessions. They have developed an implementation of their proposal by modifying an SSH client. There are no formal specification of the protocol or architecture and no evaluation against other approaches. Thus the results cannot be generalized easily.

Schlyter et al. describe the use of the DNS as a certificate store and its implication for path validation in an X.509 PKI [9]. This proposal resembles the approach followed in this paper with the exception that DNS is used without the DNSSEC extensions. Therefore the resolved certificate must be checked against a local certificate store instead of relying on a shared trusted infrastructure provided by DNSSEC. In [10] Schlyter and Griffin present a proposal for securing SSH by using DNS lookups. As described in the previous paragraph, the main difference with our approach is that trust is verified explicitly on-request by the user instead of relying on a mandatory trusted DNSSEC infrastructure. Moreover this approach concentrates on the SSH application only. In [11] Josefsson describes how to use the DNS as a certificate directory. This approach resembles the one we follow in this paper but offers a less generic perspective. Its focus is on the machinery of storing and resolving certificates and how to use those certificates for electronic messaging.

3 Authenticated Public Key Directories

Binding authentication to the keying material is crucial to avoid MITM attacks. Providing this binding is not straightforward: the use of public keys only proves ownership of the key pair, not who really owns the keys. Usually a Trusted Third Party (TTP) somehow ensures the binding between the peers and their keys.

3.1 Public Key Authentication and Distribution

Public key authentication is the process of validating the binding of a public key to a named entity. The most widely used approaches for public key authentication are the certifying authority model using certificates, e.g., SSL/TLS [12], and the web-of-trust model, e.g., Pretty Good Privacy (PGP) [13]. Methods for obtaining a public key can generally be classified in between two following options:
1) Through an untrusted channel: The receiver's public key could be sent to the sender over an arbitrary, insecure channel. Since the message containing the recipient's public key in itself cannot be trusted because it was sent by an untrusted party, additional signing of this message by a TTP is required to authenticate the key. Verification of the signature on this message requires local availability of the signer's public key at the sender side. The X.509 certificates are commonly used for securing Internet communications today, e.g. in IPsec, SSL and TLS [12].
2) Through a trusted channel: Perhaps the most natural way to obtain the public key of an initially unknown party is by consulting a TTP through a trusted channel. The trust relation with the TTP and the assumption that the communication with the TTP is secure, guarantee the integrity of the public key and in fact establish the authentication of the unknown party. Since a TTP can probably guarantee only a small number of public keys, it would be more convenient to use a shared public trusted channel that manages and provides public keys for a potentially large number of parties. Examples of such a public channel, with a rather static nature, are phonebooks and newspapers publicizing keying information, but online directories (thus trusted directory services) are a more dynamic alternative.

3.2 Naming

The index in the public directory that points to the entry holding the public key is formed by a unique name. A party must be identified using a name that is valid and unique in a name space that is shared between the public key owner and the public key requestor. Certificate identifiers often consist of person and company names as used in everyday, called "Common Names" in X.509 terminology. These are typically not good identifiers since they are not part of a well defined naming space. As such they cannot be guaranteed to be unique, at least not over different Certificate Authorities [2]. The Internet has named the whole world with domain names, e-mail names and URLs. These names are unique by assignment because of a hierarchy of naming authorities with a single root. Although the X.500 naming scheme is in use for the most widespread type (X.509) of certificates used on the Internet today, we suggest using the DNS system and its associated naming hierarchy for resolving X.509 certificates when for securing Internet connectivity.

3.3 Towards a Trusted Directory Service Solution

While several solutions to the aforementioned problems can be devised, this paper proposes a trusted online directory approach for public key and certificate lookups. This approach conveniently solves the public key distribution and identification issues needed for peer authentication in PKC. Moreover, some of the major problems found in current PKI deployment are related to static configuration, complexity and administrative overhead (see [2] for a survey). The use of online trusted directories addresses some of these problems by providing:

- Ease of public key revocation: this is achieved by simply deleting the revoked keys from the directory.
- Extensibility: any party that thinks a particular certificate can be trusted can place it easily in the directory, and this propagates the trust to its users.
- Single point of entrance: the problems with trying to handle multiple root CAs and understanding interacting policies and trust are avoided.
- Simplicity: there is less demand on the use of complex CA software since self-signed certificates can be used. This leads to lower deployment cost.

The approach we pursue is to use DNS and its secure extensions, called DNSSEC [14], for implementing public key and certificate directories. We expect that DNSSEC key/certificate repositories will be formed around so-called virtual organizations (VOs); organizations or communities that have a shared interest in exchanging credentials within the scope of that VO (see also [15]). This resembles the VO concept in the grid computing [16].

4 DNSSEC and TLS

This section elaborates on the features of DNSSEC and how we use it to create a trusted directory service and to apply it to TLS connectivity.

4.1 DNS and DNSSEC

DNS [5] is an Internet directory service. It is a well known, globally distributed, redundant and highly available database. Its main use is associating domain names (such as ”www.example.com”) with Internet protocol addresses. DNS is distributed hierarchically, where the structure of the hierarchy is embedded in the domain name. Many types of data are stored in the DNS, ranging from address (denoted by type A) and text records (type TXT) to certificate records (type CERT). This data can be resolved through its associated name which functions as an index in the DNS directory. The DNS protocol suffers from several distinct classes of threats [17]. As a protection against these threats, the DNS Security Extensions (DNSSEC) [14] were conceived. DNSSEC is a seamless addition to DNS, where trust relations are organized using the same DNS hierarchy. A DNS resolver, configured with a trust anchor for part of the hierarchy, is able to authenticate data in the hierarchy.

4.2 DNSSEC as a Trusted Directory Service

Although X.509 certificate features allow for federation and syndication (for a description see hereafter), it is fairly cumbersome and has its drawbacks [2]. DNS supports the storage of X.509 certificates in the DNS in the form of CERT records. The DNS protocol already allows for syndication, while federation can be emulated trivially. DNSSEC can thus be used to supply a layer of trust and it can be used to implement a PKI.

1) Syndication: Through so-called X.509 certificate chaining, a root certificate is able to sign a certificate request that in itself is capable of signing new certificates. In this way, an organization can syndicate trust to third parties. With DNS, syndication happens by merely delegating an administrative domain to others, e.g., the organization that controls ".com" has delegated the domain "example.com" to the Example company. With DNSSEC, third parties are able to validate this delegation.

2) Federation: An X.509 root certificate is able to sign multiple certificates. With DNS, an administrative domain, such as "example.com" can administer the federated organization under, for instance, "federation.example.com", which holds all the information about the federation. With DNSSEC, third parties are able to validate this administrative domain, for an example see [18].

3) Trusted directory based PKI: Using DNS and DNSSEC for federation and syndication through hierarchical naming, it is straightforward to create a PKI. Self-signed certificates allow for a low-cost infrastructure, while access to the DNS is already present in the Internet configuration of the participating parties. DNSSEC replaces the trust infrastructure that certificate chaining provides. Moreover, the configuration of an administrative domain in DNSSEC allows for a wider range of authenticated information than X.509 can currently provide.

4) Certificate revocation: Common PKI certificate policies require that whenever one wants to use a certificate, its validity must be checked beforehand. When omitting to do so, a revoked certificate may be incorrectly accepted as valid. This means that to effectively use a PKI one must have consulted a so-called Certificate Revocation List (CRL), in which certificate authorities indicate which certificates have been revoked. By using a trusted directory to resolve certificates, the certificate revocation problem has become almost trivial. Whenever a certificate is revoked before its expiry date, its entry is simply removed from the trusted directory. Note that usually DNS entries are cached by various servers to reduce traffic. The certificate revocation process, therefore, is limited by the frequency of DNS cache updates (max one day). A VO policy could deem this sufficient. If more frequent cert revocation is required, standard CRL checking would be needed in the cache.

4.3 Transport Layer Security Protocol

TLS provides endpoint authentication and communication privacy over the Internet using cryptography based on keying material obtained from certificates [12]. Typically, only the server is authenticated, while the client remains unauthenticated, although client authentication through a client certificate can be requested as well. Although the certificates resolved from a trusted DNSSEC directory can be applied for multiple purposes we focus here on using the certificates for setting up TLS connections. Since TLS is so widely used on the Internet today, it would benefit the most from improvements in the PKI infrastructure such as the one that we propose. There are two ways for applying resolved certificates to TLS connections:

1) Server authentication: the client connects to a server by using a hostname that is a valid entry in the DNS for which a certificate can be resolved. The server itself also presents a server certificate as part of the TLS setup, and this certificate is checked against the one that is resolved from the DNSSEC trusted directory.

2) Client authentication: the client presents a client certificate to the server as part of the TLS client authentication. The server checks the client certificate against the one resolved from the DNSSEC trusted directory for that client.

In the latter case, the server needs to obtain an identifier (or: peer name) to use as an index in the DNSSEC directory service. One option is to use reverse DNS lookup to resolve the client's IP address to a hostname that can be used as a peer name. The downside of this approach is that clients are always associated with a specific hostname which does not work well if the client is located behind a Network Address Translation (NAT) gateway or when the client is a mobile host located in a visited network. We suggest a more flexible and generic approach in which the client sends a client identifier to the server in the client authentication phase, together with the client certificate. To realize this, we extend the TLS connection setup phase with an extra feature that allows the client to present its name in a standardized way to the server, independent of the IP-address of the client (fixed or mobile). We distinguish two methods to do so, both compliant with the specification for extensions of TLS protocol [19].

1) TLS name indication: a TLS extension called TLS Client Name Indication could be standardized. The client name is binding a DNS resolvable name to a certificate regarding the IP-address.

2) Client certificate URL: an alternative solution is to use the existing TLS extension for specifying so-called Client Certificate URLs. This feature exists in TLS because it may be desirable for constrained clients to send certificate URLs in place of certificates, so that they do not need to store their certificates and can therefore save memory. Complying to [20], we defined a "dns://<Client name>" URL type indicating that certificates should be resolved from DNS. As a disadvantage, existing TLS implementations may not support this non-standard URL type.

5 Implementation

This section describes a draft software implementation of the proposal presented in this paper. Our implementation is realized as a custom extension on top of a standard TLS protocol implementation, with client and server authentication through certificates retrieved from a trusted DNS directory. This customized TLS protocol is referred to as Enhanced TLS (E-TLS). We show an overview of the protocol steps involved when peer A initiates a connection with peer B in the context of a VO named NVO. This VO runs trusted directory services through DNSSEC, in which member names are associated with their certificates containing member public keys. For example, the name NA for peer A, and NB for peer B are associated with their respective certificates CA and CB. Notice that A knows B's name NB (and thereby B's IP address through some other means like DNS). The steps required for setting up an E-TLS connection between A and B are (see Fig. 1):

1) A uses name NB to retrieve B's certificate CB from the trusted directory in the context of the communication, being the virtual organization VO. In our case this means that A prefixes the name NVO with name NB and resolves this composed name via DNSSEC to certificate CB by retrieval of a DNS CERT record.
2) A uses its own certificate CA as a client certificate to setup a client-authenticated TLS connection to B's IP address in which B presents its server certificate CB to A as part of the server authentication.
3) A verifies the certificate that B presented as part of the TLS server authentication against the certificate resolved in step 1.
4) B accepts the incoming client-authenticated TLS connection initially without verifying it i.e. accepting the presented client certificate by default.
5) A sends his name NA to B over the TLS connection, realized by sending a string NA followed by a carriage-return character.
6) B receives A's name and uses that to retrieve A's certificate CA from the trusted directory; similar to what A did for B's certificate CB in step 1.
7) B checks the certificate that A initially presented in the TLS connection setup in step 2/3 and compares this to the certificate retrieved in the previous step.
8) If the certificates match, B sends the string "OK" followed by a carriage return to A and "user-space" communication can start over the established TLS connection, if not B terminates the connection by closing the socket.

Additionally we created a so-called Resolver API which abstracts from the underlying resolver implementation that actually resolves names to certificates. Because of this abstraction we are able to support alternative implementations possibly binding to trusted directory services other than DNSSEC. For our purposes we developed two implementations of this API which we can use inter-changeably: *DNS Resolver* and *File Resolver*. The DNS Resolver API is the main resolver used for validating the E-TLS protocol, resolving certificates from DNSSEC as proposed in this paper. The File Resolver implementation resolves certificates through the use

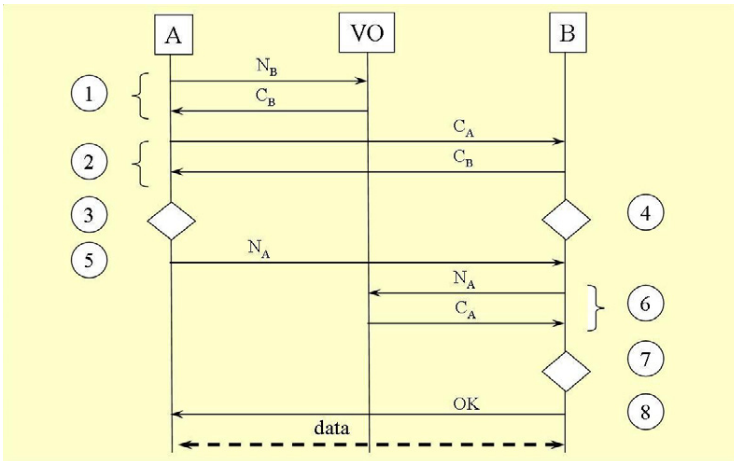

Fig. 1. E-TLS connection setup protocol steps.

of flat files, mainly useful for testing purposes and performance comparisons.

We denote the exchange of A's name and the response from B by the term handshake, formed by the extra bit of communication after establishing a legacy connection, needed to finish the E-TLS part. To support this in a generic way we designed a so-called Handshake API, allowing for applying different handshake implementations. The behavior of the E-TLS implementation is fully configurable through configuration files, including the types of socket, resolver, and handshake.

We developed our prototype software in the Java programming language using the Sun Java Development Kit version 1.5.0 (JDK 1.5) [21]. We used the DNSjava [22] library version 2.0.3 for providing DNS and DNSSEC functionality, both at the client and the server side, the latter meaning that we used the Java DNS server that comes with the DNSjava package. The TLS implementation that we extended on is included in the Java Secure Sockets Extension (JSSE) implementation that comes with JDK 1.5 [23]. We developed Java classes for use at the client and server side, supporting creation of E-TLS client and server socket objects which are specializations of "plain" TCP and SSL Java socket classes.

6 Evaluation

We evaluated our approach against alternative solutions, among which setting up TLS connections using statically configured certificates.

6.1 Performance

We have compared the performance of E-TLS connection setup using our implementation against the setup of legacy TCP and TLS connections. We measured by creating 1000 connections for each connection type and calculating the average. The tests were performed on a host with a Pentium IV 3.4 GHz processor running Ubuntu Linux 6.10. We differentiated between setting up connections with and without a handshake. Although in a deployment scenario it is unrealistic to

Connection Type	setup time without handshake	setup time with handshake
TCP	1.30 ms	4.45 ms
TLS	46.88 ms	49.81 ms
E-TLS	90.71 ms	95.51 ms
E-TLS (file)	-	91.63 ms

Table 1
performance comparison of connection setup time.

accept connections without performing the handshake (i.e. accepting a client certificate without checking it), it is useful in tests and measurements. This is because it shows the overhead due to the handshake itself, apart from the overhead of resolving certificates. The results are shown below in Table 1. The last entry, E-TLS (file), shows the results when resolving certificates from a local, cached file instead of remote DNS. The table shows that the overhead of E-TLS compared to standard TLS can be expressed in two components: the overhead of the handshake for sending the client name the result and the overhead of resolving certificates at connection setup at both endpoints (expressed in the difference between E-TLS and TLS). These overheads are about 3.6 and 45 milliseconds per connection, respectively.

6.2 Administrative and Runtime Overheads

The administrative (offline) overhead involved in deploying the proposal presented in this paper concerns both DNSSEC deployment and the configuration of a naming space related to the certificates in a domain. For elaborations on generic DNSSEC deployment, the reader is referred to [25]. Although not widely deployed yet, we consider DNSSEC to be a necessary and thus future part of a common, shared secure Internet infrastructure. When comparing the administrative overhead with application specific PKI we see that the burden on the client in traditional PKI solutions, where certificates are stored and managed at the client side, is shifted to the owner/manager of the DNS domain that holds the certificate, and thus becomes a centralized effort. In our view this is an appropriate shift if only the task is performed by a dedicated party, i.e., the VO.

The runtime (online) overhead associated with this proposal when setting up TLS connections is considerable because of the encryption and verification of DNSSEC traffic, the DNSSEC operations needed for certificate retrieval at both peer ends and the communication overhead of sending a DNS name to the remote peer and a response back to the initiating peer. Although the overhead of E-TLS over standard TLS is apparently large one must realize that the comparison is slightly inappropriate. In the E-TLS case, certificate exchange is done real-time in the actual connection setup phase, whereas in standard TLS most of that work has been done off-line and the setup can be considered pre-configured. Caching of certificates in E-TLS would be a viable solution to overcome performance drawbacks in particular cases and it would yield a better comparison against standard TLS. Of course the cache would still have to respect any CRLs (see Subsection 4.2).

6.3 *PKI Management*

Regarding our proposed solution one may conclude that the PKI key management problem is now merely shifted to DNS providers and DNSSEC deployment and configuration. We believe, however, that the approach addresses the PKI problem at an appropriate, shared infrastructural level. Moreover it involves fewer parties and these parties are typically better suited towards management of infrastructural assets such as keys and certificates. In the end this shift addresses the dynamic requirement of trusted connection establishment as expressed in the objectives of this paper. The mentioned characteristics imply also that this proposal is not geared towards end-users: having to create and manage per-client certificates and to maintain a domain entry in a DNS tree are likely to be too much to ask from end-users; this has to be accomplished on a higher-than-individual, organizational level. DNS data tends to change slowly and is controlled by domain administrators. Allowing users some level of direct control over their keys would violate the existing model. Supporting dynamic DNS update for DNSSEC is difficult in general [26].

6.4 *DNS Suitability*

As stated in [27], DNS was designed to identify network resources. Inherent support for resolving credentials is provided though the KEY and CERT records [28]. Using DNS as a trusted directory service for application keys and certificates has been proposed a number of times as shown in section 2. Still it is a controversial topic in the DNS community as can be observed from multiple discussions on the Internet. There are yet two concerns about DNSSEC:
1) Scalability: Proposals to house per-user information in DNS did not anticipate that the growth in Internet user population would far surpass the growth in DNS-registered host systems. Adding DNSSEC signature records to a zone increases the size of the zone data by a factor of 8 or 9 [30], and adding per-user keys and their signatures would further increase the size of the zone data. Moreover the application of our approach would increase the average number of requests per second on the DNS servers and resolvers. This load would have to be accommodated for by applying scalability techniques such as server pool dimensioning, load balancing and providing additional network bandwidth.
2) Performance: The DNS has been designed and optimized for very small message exchanges, about 300 bytes in size. Returning key and certificate data (1.2 KB) complemented by the associated signatures in DNS responses would significantly increase network load. Moreover the UDP maximum packet length may not be appropriate for large certificates, in which case a fallback to TCP has to occur, at the cost of more time and resources. However, storing and resolving full-blown certificates from the DNS is a convenience solution in our proposal, useful to be able to apply the resolved certificates directly to TLS connections. We take into account that the use of CERT records could be replaced by using a newly defined application specific KEY RR containing only a public key, as proposed in [31].

6.5 Security

Resolving and checking of certificates in our approach now depend on the availability of the DNS service. As such the DNS service is a single point of attack and a single point of failure. We feel however that these issues have to be addressed for DNS services anyway, and can not be avoided for any shared trusted directory solution. The presented E-TLS protocol is vulnerable to DoS attacks because the presented certificates have to be processed. This vulnerability also exists for TLS [24].

7 Conclusions

We have presented an approach and an implementation for setting up secure (TLS) connections using certificates resolved from a trusted directory service (DNSSEC). Our proposal reuses existing Internet infrastructure that already operates in a hierarchical global naming scheme. Application of credentials resolved through a trusted directory service is not limited to a specific protocol (e.g. TLS) but can be applied to any protocol that requires credentials obtained through a peer identifier mapping (e.g. IPsec and encrypted UDP traffic). We have shown how TLS can be extended to use certificates resolved from a trusted directory, in this case DNSSEC.

8 Acknowledgment

The work described in this paper was conducted in the GigaPort Next Generation program, sponsored by the Dutch government and coordinated by SURFnet.

References

[1] Diffie, W., and M. E. Hellman, "New directions in cryptography," IEEE Transactions on Information Theory, vol. IT-22, Nov. 1976, pp. 644-654.

[2] Ellison, C., and B. Schneier, "Ten risks of PKI: What you're not being told about Public Key Infrastructure," Computer Security Journal, vol. 16, no. 1, 2000, pp. 1-7.

[3] CCITT, Recommendation X.500, "The directory: Overview of concepts, models and service," Switzerland, 1988.

[4] CCITT, Recommendation X.509, "The directory: Authentication framework," Switzerland, 1988.

[5] Mockapetris, P., "Domain names - concepts and facilities," IETF, RFC 1034, Nov. 1987.

[6] Jones, J. P., D. F. Berger, and C. V. Ravishankar, "Layering public key distribution over secure DNS using authenticated delegation," In Proceedings of the 21st Annual Computer Security Applications Conference, 2005, pp. 409-418.

[7] Richardson, M., and D. H. Redelmeier, "Opportunistic encryption using the Internet Key Exchange (IKE)," IETF, RFC 4322, Dec. 2005.

[8] Le, V., and H. Guyennet, " IPSec and DNSSEC to support GRID application security," in 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002, p. 458.

[9] Schlyter, J., L. Johansson, "DNS as X.509 PKIX certificate storage," draft-schlyter-pkix-dns-02, IETF, Internet Draft, June 2002.

[10] Schlyter, J., and W. Griffin, "Using DNS to securely publish Secure Shell (SSH) key fingerprints," IETF, RFC 4255, Jan. 2006.

[11] Josefsson, S., "Network application security using the Domain Name System", M.Sc. thesis, Royal Institute of Technology, Dept. of Numerical Analysis and Computer Science, 2001.

[12] Dierks, T., and C. Allen, "The TLS protocol version 1.0," IETF, RFC 2246, Jan. 1999.

[13] Atkins, D., W. Stallings, and P. Zimmermann, "PGP message exchange formats," IETF, RFC 1991, Aug. 1996.

[14] Arends, R., R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," IETF, RFC 4033, March 2005.

[15] Hulsebosch, R.J, M.S. Bargh, P.H. Fennema, J.F. Zandbelt, M. Snijders, and E.H. Eertink, "Using identity management and secure DNS for effective and trusted user controlled light-path establishment," in International Conference on Networking and Services, Santa Clara, USA, July 16-19, 2006.

[16] Foster, I., C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," in Internat. Journal of Supercomputer Applications, vol. 15, no.3, 2001, pp.200-222.

[17] Atkins, D. and R. Austein, "Threat analysis of the Domain Name System (DNS)," IETF, RFC 3833, Aug. 2004.

[18] Eertink, H., A. Peddemors, R. Arends, and K. Wierenga, "Combining RADIUS with secure DNS for dynamic trust establishment between domains," presented at TERENA Networking Conference, Poznan, Poland, 2005.

[19] Blake-Wilson, S., M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright, "Transport Layer Security (TLS) extensions," IETF, RFC 3546, June 2003.

[20] Josefsson, S., "Domain Name System Uniform Resource Identifiers", IETF, RFC 4501, May 2006.

[21] Sun Microsystems, "Java Development Kit version 5," URL: http://java.sun.com/javase/downloads/index_jdk5.jsp.

[22] Wellington, B., DNSjava, URL: http://www.dnsjava.org/.

[23] Sun Microsystems, Java Secure Socket Extension, URL: http://java.sun.com/products/jsse/.

[24] Vlker, L. and M. Schller, "Secure TLS: Preventing DoS Attacks with Lower Layer Authentication," Kommunikation in Verteilten Systemen (KiVS), Springer Berlin Heidelberg, pp. 237-248, 2007.

[25] DNSSEC.net, "DNSSEC deployment, how to deploy DNSSEC," URL: http://www.dnssec.net/practical-documents.

[26] Wellington, B., "Secure Domain Name System (DNS) dynamic update," IETF, RFC 3007, Nov. 2000.

[27] Klensin, J., "Review of the DNS and its role as designed," IETF, RFC 3467, Feb. 2003.

[28] Eastlake D., and O. Gudmundsson, "Storing certificates in the Domain Name System (DNS)," IETF, RFC 2538, March 1999.

[29] Massey, D., and S. Rose, "Limiting the scope of the KEY Resource Record (RR)," IETF, RFC 3445, Dec. 2002.

[30] Gieben, R., "DNSSEC in NL," URL: http://miek.nl/publications/dnssecnl/secreg-report.pdf Jan. 2004.

[31] Schlyter, J., "Storing application public keys in the DNS," draft-schlyter-appkey-02.txt, IETF, Internet Draft, Feb. 2002.