        Recommendations for Secure Use of Transport Layer Security (TLS)
                and Datagram Transport Layer Security (DTLS)

Abstract

   Transport Layer Security (TLS) and Datagram Transport Layer Security
   (DTLS) are widely used to protect data exchanged over application
   protocols such as HTTP, SMTP, IMAP, POP, SIP, and XMPP. Over the
   last few years, several serious attacks on TLS have emerged,
   including attacks on its most commonly used cipher suites and their
   modes of operation. This document provides recommendations for
   improving the security of deployed services that use TLS and DTLS.
   The recommendations are applicable to the majority of use cases.

Status of This Memo

   This memo documents an Internet Best Current Practice.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   BCPs is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7525.

Copyright Notice

   Copyright (c) 2015 IETF Trust and the persons identified as the
   document authors. All rights reserved.

Table of Contents

1. Introduction

   Transport Layer Security (TLS) [RFC5246] and Datagram Transport
   Security Layer (DTLS) [RFC6347] are widely used to protect data
   exchanged over application protocols such as HTTP, SMTP, IMAP, POP,
   SIP, and XMPP. Over the last few years, several serious attacks on
   TLS have emerged, including attacks on its most commonly used cipher
   suites and their modes of operation. For instance, both the AES-CBC
   [RFC3602] and RC4 [RFC7465] encryption algorithms, which together
   have been the most widely deployed ciphers, have been attacked in the
   context of TLS. A companion document [RFC7457] provides detailed
   information about these attacks and will help the reader understand
   the rationale behind the recommendations provided here.

   Because of these attacks, those who implement and deploy TLS and DTLS
   need updated guidance on how TLS can be used securely. This document
   provides guidance for deployed services as well as for software
   implementations, assuming the implementer expects his or her code to
   be deployed in environments defined in Section 5. In fact, this
   document calls for the deployment of algorithms that are widely
   implemented but not yet widely deployed. Concerning deployment, this
   document targets a wide audience -- namely, all deployers who wish to
   add authentication (be it one-way only or mutual), confidentiality,
   and data integrity protection to their communications.

   The recommendations herein take into consideration the security of
   various mechanisms, their technical maturity and interoperability,
   and their prevalence in implementations at the time of writing.
   Unless it is explicitly called out that a recommendation applies to
   TLS alone or to DTLS alone, each recommendation applies to both TLS
   and DTLS.

   It is expected that the TLS 1.3 specification will resolve many of
   the vulnerabilities listed in this document. A system that deploys
   TLS 1.3 should have fewer vulnerabilities than TLS 1.2 or below.
   This document is likely to be updated after TLS 1.3 gets noticeable
   deployment.

   These are minimum recommendations for the use of TLS in the vast
   majority of implementation and deployment scenarios, with the
   exception of unauthenticated TLS (see Section 5). Other
   specifications that reference this document can have stricter
   requirements related to one or more aspects of the protocol, based on
   their particular circumstances (e.g., for use with a particular
   application protocol); when that is the case, implementers are
   advised to adhere to those stricter requirements. Furthermore, this

document provides a floor, not a ceiling, so stronger options are
always allowed (e.g., depending on differing evaluations of the
importance of cryptographic strength vs. computational load).

Community knowledge about the strength of various algorithms and
feasible attacks can change quickly, and experience shows that a Best
Current Practice (BCP) document about security is a point-in-time
statement. Readers are advised to seek out any errata or updates
that apply to this document.

2. Terminology

A number of security-related terms in this document are used in the
sense defined in [RFC4949].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in [RFC2119].

3. General Recommendations

This section provides general recommendations on the secure use of
TLS. Recommendations related to cipher suites are discussed in the
following section.

3.1. Protocol Versions

3.1.1. SSL/TLS Protocol Versions

It is important both to stop using old, less secure versions of SSL/
TLS and to start using modern, more secure versions; therefore, the
following are the recommendations concerning TLS/SSL protocol
versions:

o Implementations MUST NOT negotiate SSL version 2.

   Rationale: Today, SSLv2 is considered insecure [RFC6176].

o Implementations MUST NOT negotiate SSL version 3.

   Rationale: SSLv3 [RFC6101] was an improvement over SSLv2 and
   plugged some significant security holes but did not support strong
   cipher suites. SSLv3 does not support TLS extensions, some of
   which (e.g., renegotiation_info [RFC5746]) are security-critical.
   In addition, with the emergence of the POODLE attack [POODLE],
   SSLv3 is now widely recognized as fundamentally insecure. See
   [DEP-SSLv3] for further details.

   o Implementations SHOULD NOT negotiate TLS version 1.0 [RFC2246];
      the only exception is when no higher version is available in the
      negotiation.

      Rationale: TLS 1.0 (published in 1999) does not support many
      modern, strong cipher suites. In addition, TLS 1.0 lacks a per-
      record Initialization Vector (IV) for CBC-based cipher suites and
      does not warn against common padding errors.

   o Implementations SHOULD NOT negotiate TLS version 1.1 [RFC4346];
      the only exception is when no higher version is available in the
      negotiation.

      Rationale: TLS 1.1 (published in 2006) is a security improvement
      over TLS 1.0 but still does not support certain stronger cipher
      suites.

   o Implementations MUST support TLS 1.2 [RFC5246] and MUST prefer to
      negotiate TLS version 1.2 over earlier versions of TLS.

      Rationale: Several stronger cipher suites are available only with
      TLS 1.2 (published in 2008). In fact, the cipher suites
      recommended by this document (Section 4.2 below) are only
      available in TLS 1.2.

   This BCP applies to TLS 1.2 and also to earlier versions. It is not
   safe for readers to assume that the recommendations in this BCP apply
   to any future version of TLS.

3.1.2. DTLS Protocol Versions

   DTLS, an adaptation of TLS for UDP datagrams, was introduced when TLS
   1.1 was published. The following are the recommendations with
   respect to DTLS:

   o Implementations SHOULD NOT negotiate DTLS version 1.0 [RFC4347].

      Version 1.0 of DTLS correlates to version 1.1 of TLS (see above).

   o Implementations MUST support and MUST prefer to negotiate DTLS
      version 1.2 [RFC6347].

      Version 1.2 of DTLS correlates to version 1.2 of TLS (see above).
      (There is no version 1.1 of DTLS.)

3.1.3. Fallback to Lower Versions

   Clients that "fall back" to lower versions of the protocol after the
   server rejects higher versions of the protocol MUST NOT fall back to
   SSLv3 or earlier.

   Rationale: Some client implementations revert to lower versions of
   TLS or even to SSLv3 if the server rejected higher versions of the
   protocol. This fallback can be forced by a man-in-the-middle (MITM)
   attacker. TLS 1.0 and SSLv3 are significantly less secure than TLS
   1.2, the version recommended by this document. While TLS 1.0-only
   servers are still quite common, IP scans show that SSLv3-only servers
   amount to only about 3% of the current Web server population. (At
   the time of this writing, an explicit method for preventing downgrade
   attacks has been defined recently in [RFC7507].)

3.2. Strict TLS

   The following recommendations are provided to help prevent SSL
   Stripping (an attack that is summarized in Section 2.1 of [RFC7457]):

   o In cases where an application protocol allows implementations or
      deployments a choice between strict TLS configuration and dynamic
      upgrade from unencrypted to TLS-protected traffic (such as
      STARTTLS), clients and servers SHOULD prefer strict TLS
      configuration.

   o Application protocols typically provide a way for the server to
      offer TLS during an initial protocol exchange, and sometimes also
      provide a way for the server to advertise support for TLS (e.g.,
      through a flag indicating that TLS is required); unfortunately,
      these indications are sent before the communication channel is
      encrypted. A client SHOULD attempt to negotiate TLS even if these
      indications are not communicated by the server.

   o HTTP client and server implementations MUST support the HTTP
      Strict Transport Security (HSTS) header [RFC6797], in order to
      allow Web servers to advertise that they are willing to accept
      TLS-only clients.

   o Web servers SHOULD use HSTS to indicate that they are willing to
      accept TLS-only clients, unless they are deployed in such a way
      that using HSTS would in fact weaken overall security (e.g., it
      can be problematic to use HSTS with self-signed certificates, as
      described in Section 11.3 of [RFC6797]).

   Rationale: Combining unprotected and TLS-protected communication
   opens the way to SSL Stripping and similar attacks, since an initial
   part of the communication is not integrity protected and therefore
   can be manipulated by an attacker whose goal is to keep the
   communication in the clear.

3.3. Compression

   In order to help prevent compression-related attacks (summarized in
   Section 2.6 of [RFC7457]), implementations and deployments SHOULD
   disable TLS-level compression (Section 6.2.2 of [RFC5246]), unless
   the application protocol in question has been shown not to be open to
   such attacks.

   Rationale: TLS compression has been subject to security attacks, such
   as the CRIME attack.

   Implementers should note that compression at higher protocol levels
   can allow an active attacker to extract cleartext information from
   the connection. The BREACH attack is one such case. These issues
   can only be mitigated outside of TLS and are thus outside the scope
   of this document. See Section 2.6 of [RFC7457] for further details.

3.4. TLS Session Resumption

   If TLS session resumption is used, care ought to be taken to do so
   safely. In particular, when using session tickets [RFC5077], the
   resumption information MUST be authenticated and encrypted to prevent
   modification or eavesdropping by an attacker. Further
   recommendations apply to session tickets:

   o A strong cipher suite MUST be used when encrypting the ticket (as
      least as strong as the main TLS cipher suite).

   o Ticket keys MUST be changed regularly, e.g., once every week, so
      as not to negate the benefits of forward secrecy (see Section 6.3
      for details on forward secrecy).

   o For similar reasons, session ticket validity SHOULD be limited to
      a reasonable duration (e.g., half as long as ticket key validity).

   Rationale: session resumption is another kind of TLS handshake, and
   therefore must be as secure as the initial handshake. This document
   (Section 4) recommends the use of cipher suites that provide forward
   secrecy, i.e. that prevent an attacker who gains momentary access to
   the TLS endpoint (either client or server) and its secrets from
   reading either past or future communication. The tickets must be
   managed so as not to negate this security property.

3.5. TLS Renegotiation

   Where handshake renegotiation is implemented, both clients and
   servers MUST implement the renegotiation_info extension, as defined
   in [RFC5746].

   The most secure option for countering the Triple Handshake attack is
   to refuse any change of certificates during renegotiation. In
   addition, TLS clients SHOULD apply the same validation policy for all
   certificates received over a connection. The [triple-handshake]
   document suggests several other possible countermeasures, such as
   binding the master secret to the full handshake (see [SESSION-HASH])
   and binding the abbreviated session resumption handshake to the
   original full handshake. Although the latter two techniques are
   still under development and thus do not qualify as current practices,
   those who implement and deploy TLS are advised to watch for further
   development of appropriate countermeasures.

3.6. Server Name Indication

   TLS implementations MUST support the Server Name Indication (SNI)
   extension defined in Section 3 of [RFC6066] for those higher-level
   protocols that would benefit from it, including HTTPS. However, the
   actual use of SNI in particular circumstances is a matter of local
   policy.

   Rationale: SNI supports deployment of multiple TLS-protected virtual
   servers on a single address, and therefore enables fine-grained
   security for these virtual servers, by allowing each one to have its
   own certificate.

4. Recommendations: Cipher Suites

   TLS and its implementations provide considerable flexibility in the
   selection of cipher suites. Unfortunately, some available cipher
   suites are insecure, some do not provide the targeted security
   services, and some no longer provide enough security. Incorrectly
   configuring a server leads to no or reduced security. This section
   includes recommendations on the selection and negotiation of cipher
   suites.

4.1. General Guidelines

   Cryptographic algorithms weaken over time as cryptanalysis improves:
   algorithms that were once considered strong become weak. Such
   algorithms need to be phased out over time and replaced with more
   secure cipher suites. This helps to ensure that the desired security
   properties still hold. SSL/TLS has been in existence for almost 20

years and many of the cipher suites that have been recommended in
various versions of SSL/TLS are now considered weak or at least not
as strong as desired. Therefore, this section modernizes the
recommendations concerning cipher suite selection.

o Implementations MUST NOT negotiate the cipher suites with NULL
   encryption.

   Rationale: The NULL cipher suites do not encrypt traffic and so
   provide no confidentiality services. Any entity in the network
   with access to the connection can view the plaintext of contents
   being exchanged by the client and server. (Nevertheless, this
   document does not discourage software from implementing NULL
   cipher suites, since they can be useful for testing and
   debugging.)

o Implementations MUST NOT negotiate RC4 cipher suites.

   Rationale: The RC4 stream cipher has a variety of cryptographic
   weaknesses, as documented in [RFC7465]. Note that DTLS
   specifically forbids the use of RC4 already.

o Implementations MUST NOT negotiate cipher suites offering less
   than 112 bits of security, including so-called "export-level"
   encryption (which provide 40 or 56 bits of security).

   Rationale: Based on [RFC3766], at least 112 bits of security is
   needed. 40-bit and 56-bit security are considered insecure today.
   TLS 1.1 and 1.2 never negotiate 40-bit or 56-bit export ciphers.

o Implementations SHOULD NOT negotiate cipher suites that use
   algorithms offering less than 128 bits of security.

   Rationale: Cipher suites that offer between 112-bits and 128-bits
   of security are not considered weak at this time; however, it is
   expected that their useful lifespan is short enough to justify
   supporting stronger cipher suites at this time. 128-bit ciphers
   are expected to remain secure for at least several years, and
   256-bit ciphers until the next fundamental technology
   breakthrough. Note that, because of so-called "meet-in-the-
   middle" attacks [Multiple-Encryption], some legacy cipher suites
   (e.g., 168-bit 3DES) have an effective key length that is smaller
   than their nominal key length (112 bits in the case of 3DES).
   Such cipher suites should be evaluated according to their
   effective key length.

   o Implementations SHOULD NOT negotiate cipher suites based on RSA
      key transport, a.k.a. "static RSA".

      Rationale: These cipher suites, which have assigned values
      starting with the string "TLS_RSA_WITH_*", have several drawbacks,
      especially the fact that they do not support forward secrecy.

   o Implementations MUST support and prefer to negotiate cipher suites
      offering forward secrecy, such as those in the Ephemeral Diffie-
      Hellman and Elliptic Curve Ephemeral Diffie-Hellman ("DHE" and
      "ECDHE") families.

      Rationale: Forward secrecy (sometimes called "perfect forward
      secrecy") prevents the recovery of information that was encrypted
      with older session keys, thus limiting the amount of time during
      which attacks can be successful. See Section 6.3 for a detailed
      discussion.

4.2. Recommended Cipher Suites

   Given the foregoing considerations, implementation and deployment of
   the following cipher suites is RECOMMENDED:

   o TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

   o TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

   o TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

   o TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

   These cipher suites are supported only in TLS 1.2 because they are
   authenticated encryption (AEAD) algorithms [RFC5116].

   Typically, in order to prefer these suites, the order of suites needs
   to be explicitly configured in server software. (See [BETTERCRYPTO]
   for helpful deployment guidelines, but note that its recommendations
   differ from the current document in some details.) It would be ideal
   if server software implementations were to prefer these suites by
   default.

   Some devices have hardware support for AES-CCM but not AES-GCM, so
   they are unable to follow the foregoing recommendations regarding
   cipher suites. There are even devices that do not support public key
   cryptography at all, but they are out of scope entirely.

4.2.1. Implementation Details

   Clients SHOULD include TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as the
   first proposal to any server, unless they have prior knowledge that
   the server cannot respond to a TLS 1.2 client_hello message.

   Servers MUST prefer this cipher suite over weaker cipher suites
   whenever it is proposed, even if it is not the first proposal.

   Clients are of course free to offer stronger cipher suites, e.g.,
   using AES-256; when they do, the server SHOULD prefer the stronger
   cipher suite unless there are compelling reasons (e.g., seriously
   degraded performance) to choose otherwise.

   This document does not change the mandatory-to-implement TLS cipher
   suite(s) prescribed by TLS. To maximize interoperability, RFC 5246
   mandates implementation of the TLS_RSA_WITH_AES_128_CBC_SHA cipher
   suite, which is significantly weaker than the cipher suites
   recommended here. (The GCM mode does not suffer from the same
   weakness, caused by the order of MAC-then-Encrypt in TLS
   [Krawczyk2001], since it uses an AEAD mode of operation.)
   Implementers should consider the interoperability gain against the
   loss in security when deploying the TLS_RSA_WITH_AES_128_CBC_SHA
   cipher suite. Other application protocols specify other cipher
   suites as mandatory to implement (MTI).

   Note that some profiles of TLS 1.2 use different cipher suites. For
   example, [RFC6460] defines a profile that uses the
   TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 and
   TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 cipher suites.

   [RFC4492] allows clients and servers to negotiate ECDH parameters
   (curves). Both clients and servers SHOULD include the "Supported
   Elliptic Curves" extension [RFC4492]. For interoperability, clients
   and servers SHOULD support the NIST P-256 (secp256r1) curve
   [RFC4492]. In addition, clients SHOULD send an ec_point_formats
   extension with a single element, "uncompressed".

4.3. Public Key Length

   When using the cipher suites recommended in this document, two public
   keys are normally used in the TLS handshake: one for the Diffie-
   Hellman key agreement and one for server authentication. Where a
   client certificate is used, a third public key is added.

   With a key exchange based on modular exponential (MODP) Diffie-
   Hellman groups ("DHE" cipher suites), DH key lengths of at least 2048
   bits are RECOMMENDED.

Rationale: For various reasons, in practice, DH keys are typically
generated in lengths that are powers of two (e.g., 2^10 = 1024 bits,
2^11 = 2048 bits, 2^12 = 4096 bits). Because a DH key of 1228 bits
would be roughly equivalent to only an 80-bit symmetric key
[RFC3766], it is better to use keys longer than that for the "DHE"
family of cipher suites. A DH key of 1926 bits would be roughly
equivalent to a 100-bit symmetric key [RFC3766] and a DH key of 2048
bits might be sufficient for at least the next 10 years
[NIST.SP.800-56A]. See Section 4.4 for additional information on the
use of MODP Diffie-Hellman in TLS.

As noted in [RFC3766], correcting for the emergence of a TWIRL
machine would imply that 1024-bit DH keys yield about 65 bits of
equivalent strength and that a 2048-bit DH key would yield about 92
bits of equivalent strength.

With regard to ECDH keys, the IANA "EC Named Curve Registry" (within
the "Transport Layer Security (TLS) Parameters" registry [IANA-TLS])
contains 160-bit elliptic curves that are considered to be roughly
equivalent to only an 80-bit symmetric key [ECRYPT-II]. Curves of
less than 192 bits SHOULD NOT be used.

When using RSA, servers SHOULD authenticate using certificates with
at least a 2048-bit modulus for the public key. In addition, the use
of the SHA-256 hash algorithm is RECOMMENDED (see [CAB-Baseline] for
more details). Clients SHOULD indicate to servers that they request
SHA-256, by using the "Signature Algorithms" extension defined in
TLS 1.2.

4.4. Modular Exponential vs. Elliptic Curve DH Cipher Suites

   Not all TLS implementations support both modular exponential (MODP)
   and elliptic curve (EC) Diffie-Hellman groups, as required by
   Section 4.2. Some implementations are severely limited in the length
   of DH values. When such implementations need to be accommodated, the
   following are RECOMMENDED (in priority order):

   1. Elliptic Curve DHE with appropriately negotiated parameters
       (e.g., the curve to be used) and a Message Authentication Code
       (MAC) algorithm stronger than HMAC-SHA1 [RFC5289]

   2. TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 [RFC5288], with 2048-bit
       Diffie-Hellman parameters

   3. TLS_DHE_RSA_WITH_AES_128_GCM_SHA256, with 1024-bit parameters

Rationale: Although Elliptic Curve Cryptography is widely deployed,
there are some communities where its adoption has been limited for
several reasons, including its complexity compared to modular
arithmetic and longstanding perceptions of IPR concerns (which, for
the most part, have now been resolved [RFC6090]). Note that ECDHE
cipher suites exist for both RSA and ECDSA certificates, so moving to
ECDHE cipher suites does not require moving away from RSA-based
certificates. On the other hand, there are two related issues
hindering effective use of MODP Diffie-Hellman cipher suites in TLS:

o There are no standardized, widely implemented protocol mechanisms
   to negotiate the DH groups or parameter lengths supported by
   client and server.

o Many servers choose DH parameters of 1024 bits or fewer.

o There are widely deployed client implementations that reject
   received DH parameters if they are longer than 1024 bits. In
   addition, several implementations do not perform appropriate
   validation of group parameters and are vulnerable to attacks
   referenced in Section 2.9 of [RFC7457].

Note that with DHE and ECDHE cipher suites, the TLS master key only
depends on the Diffie-Hellman parameters and not on the strength of
the RSA certificate; moreover, 1024 bit MODP DH parameters are
generally considered insufficient at this time.

With MODP ephemeral DH, deployers ought to carefully evaluate
interoperability vs. security considerations when configuring their
TLS endpoints.

4.5. Truncated HMAC

   Implementations MUST NOT use the Truncated HMAC extension, defined in
   Section 7 of [RFC6066].

   Rationale: the extension does not apply to the AEAD cipher suites
   recommended above. However it does apply to most other TLS cipher
   suites. Its use has been shown to be insecure in [PatersonRS11].

5. Applicability Statement

   The recommendations of this document primarily apply to the
   implementation and deployment of application protocols that are most
   commonly used with TLS and DTLS on the Internet today. Examples
   include, but are not limited to:

   o Web software and services that wish to protect HTTP traffic with
      TLS.

   o Email software and services that wish to protect IMAP, POP3, or
      SMTP traffic with TLS.

   o Instant-messaging software and services that wish to protect
      Extensible Messaging and Presence Protocol (XMPP) or Internet
      Relay Chat (IRC) traffic with TLS.

   o Realtime media software and services that wish to protect Secure
      Realtime Transport Protocol (SRTP) traffic with DTLS.

   This document does not modify the implementation and deployment
   recommendations (e.g., mandatory-to-implement cipher suites)
   prescribed by existing application protocols that employ TLS or DTLS.
   If the community that uses such an application protocol wishes to
   modernize its usage of TLS or DTLS to be consistent with the best
   practices recommended here, it needs to explicitly update the
   existing application protocol definition (one example is [TLS-XMPP],
   which updates [RFC6120]).

   Designers of new application protocols developed through the Internet
   Standards Process [RFC2026] are expected at minimum to conform to the
   best practices recommended here, unless they provide documentation of
   compelling reasons that would prevent such conformance (e.g.,
   widespread deployment on constrained devices that lack support for
   the necessary algorithms).

5.1. Security Services

   This document provides recommendations for an audience that wishes to
   secure their communication with TLS to achieve the following:

   o Confidentiality: all application-layer communication is encrypted
      with the goal that no party should be able to decrypt it except
      the intended receiver.

   o Data integrity: any changes made to the communication in transit
      are detectable by the receiver.

o Authentication: an endpoint of the TLS communication is
   authenticated as the intended entity to communicate with.

With regard to authentication, TLS enables authentication of one or
both endpoints in the communication. In the context of opportunistic
security [RFC7435], TLS is sometimes used without authentication. As
discussed in Section 5.2, considerations for opportunistic security
are not in scope for this document.

If deployers deviate from the recommendations given in this document,
they need to be aware that they might lose access to one of the
foregoing security services.

This document applies only to environments where confidentiality is
required. It recommends algorithms and configuration options that
enforce secrecy of the data in transit.

This document also assumes that data integrity protection is always
one of the goals of a deployment. In cases where integrity is not
required, it does not make sense to employ TLS in the first place.
There are attacks against confidentiality-only protection that
utilize the lack of integrity to also break confidentiality (see, for
instance, [DegabrieleP07] in the context of IPsec).

This document addresses itself to application protocols that are most
commonly used on the Internet with TLS and DTLS. Typically, all
communication between TLS clients and TLS servers requires all three
of the above security services. This is particularly true where TLS
clients are user agents like Web browsers or email software.

This document does not address the rarer deployment scenarios where
one of the above three properties is not desired, such as the use
case described in Section 5.2 below. As another scenario where
confidentiality is not needed, consider a monitored network where the
authorities in charge of the respective traffic domain require full
access to unencrypted (plaintext) traffic, and where users
collaborate and send their traffic in the clear.

5.2. Opportunistic Security

There are several important scenarios in which the use of TLS is
optional, i.e., the client decides dynamically ("opportunistically")
whether to use TLS with a particular server or to connect in the
clear. This practice, often called "opportunistic security", is
described at length in [RFC7435] and is often motivated by a desire
for backward compatibility with legacy deployments.

   In these scenarios, some of the recommendations in this document
   might be too strict, since adhering to them could cause fallback to
   cleartext, a worse outcome than using TLS with an outdated protocol
   version or cipher suite.

   This document specifies best practices for TLS in general. A
   separate document containing recommendations for the use of TLS with
   opportunistic security is to be completed in the future.

6. Security Considerations

   This entire document discusses the security practices directly
   affecting applications using the TLS protocol. This section contains
   broader security considerations related to technologies used in
   conjunction with or by TLS.

6.1. Host Name Validation

   Application authors should take note that some TLS implementations do
   not validate host names. If the TLS implementation they are using
   does not validate host names, authors might need to write their own
   validation code or consider using a different TLS implementation.

   It is noted that the requirements regarding host name validation
   (and, in general, binding between the TLS layer and the protocol that
   runs above it) vary between different protocols. For HTTPS, these
   requirements are defined by Section 3 of [RFC2818].

   Readers are referred to [RFC6125] for further details regarding
   generic host name validation in the TLS context. In addition, that
   RFC contains a long list of example protocols, some of which
   implement a policy very different from HTTPS.

   If the host name is discovered indirectly and in an insecure manner
   (e.g., by an insecure DNS query for an MX or SRV record), it SHOULD
   NOT be used as a reference identifier [RFC6125] even when it matches
   the presented certificate. This proviso does not apply if the host
   name is discovered securely (for further discussion, see [DANE-SRV]
   and [DANE-SMTP]).

   Host name validation typically applies only to the leaf "end entity"
   certificate. Naturally, in order to ensure proper authentication in
   the context of the PKI, application clients need to verify the entire
   certification path in accordance with [RFC5280] (see also [RFC6125]).

6.2. AES-GCM

   Section 4.2 above recommends the use of the AES-GCM authenticated
   encryption algorithm. Please refer to Section 11 of [RFC5246] for
   general security considerations when using TLS 1.2, and to Section 6
   of [RFC5288] for security considerations that apply specifically to
   AES-GCM when used with TLS.

6.3. Forward Secrecy

   Forward secrecy (also called "perfect forward secrecy" or "PFS" and
   defined in [RFC4949]) is a defense against an attacker who records
   encrypted conversations where the session keys are only encrypted
   with the communicating parties' long-term keys. Should the attacker
   be able to obtain these long-term keys at some point later in time,
   the session keys and thus the entire conversation could be decrypted.
   In the context of TLS and DTLS, such compromise of long-term keys is
   not entirely implausible. It can happen, for example, due to:

   o A client or server being attacked by some other attack vector, and
      the private key retrieved.

   o A long-term key retrieved from a device that has been sold or
      otherwise decommissioned without prior wiping.

   o A long-term key used on a device as a default key [Heninger2012].

   o A key generated by a trusted third party like a CA, and later
      retrieved from it either by extortion or compromise
      [Soghoian2011].

   o A cryptographic break-through, or the use of asymmetric keys with
      insufficient length [Kleinjung2010].

   o Social engineering attacks against system administrators.

   o Collection of private keys from inadequately protected backups.

   Forward secrecy ensures in such cases that it is not feasible for an
   attacker to determine the session keys even if the attacker has
   obtained the long-term keys some time after the conversation. It
   also protects against an attacker who is in possession of the long-
   term keys but remains passive during the conversation.

   Forward secrecy is generally achieved by using the Diffie-Hellman
   scheme to derive session keys. The Diffie-Hellman scheme has both
   parties maintain private secrets and send parameters over the network
   as modular powers over certain cyclic groups. The properties of the

so-called Discrete Logarithm Problem (DLP) allow the parties to
derive the session keys without an eavesdropper being able to do so.
There is currently no known attack against DLP if sufficiently large
parameters are chosen. A variant of the Diffie-Hellman scheme uses
Elliptic Curves instead of the originally proposed modular
arithmetics.

Unfortunately, many TLS/DTLS cipher suites were defined that do not
feature forward secrecy, e.g., TLS_RSA_WITH_AES_256_CBC_SHA256. This
document therefore advocates strict use of forward-secrecy-only
ciphers.

6.4. Diffie-Hellman Exponent Reuse

   For performance reasons, many TLS implementations reuse Diffie-
   Hellman and Elliptic Curve Diffie-Hellman exponents across multiple
   connections. Such reuse can result in major security issues:

   o If exponents are reused for too long (e.g., even more than a few
      hours), an attacker who gains access to the host can decrypt
      previous connections. In other words, exponent reuse negates the
      effects of forward secrecy.

   o TLS implementations that reuse exponents should test the DH public
      key they receive for group membership, in order to avoid some
      known attacks. These tests are not standardized in TLS at the
      time of writing. See [RFC6989] for recipient tests required of
      IKEv2 implementations that reuse DH exponents.

6.5. Certificate Revocation

   The following considerations and recommendations represent the
   current state of the art regarding certificate revocation, even
   though no complete and efficient solution exists for the problem of
   checking the revocation status of common public key certificates
   [RFC5280]:

   o Although Certificate Revocation Lists (CRLs) are the most widely
      supported mechanism for distributing revocation information, they
      have known scaling challenges that limit their usefulness (despite
      workarounds such as partitioned CRLs and delta CRLs).

   o Proprietary mechanisms that embed revocation lists in the Web
      browser's configuration database cannot scale beyond a small
      number of the most heavily used Web servers.

o The On-Line Certification Status Protocol (OCSP) [RFC6960]
   presents both scaling and privacy issues. In addition, clients
   typically "soft-fail", meaning that they do not abort the TLS
   connection if the OCSP server does not respond. (However, this
   might be a workaround to avoid denial-of-service attacks if an
   OCSP responder is taken offline.)

o The TLS Certificate Status Request extension (Section 8 of
   [RFC6066]), commonly called "OCSP stapling", resolves the
   operational issues with OCSP. However, it is still ineffective in
   the presence of a MITM attacker because the attacker can simply
   ignore the client's request for a stapled OCSP response.

o OCSP stapling as defined in [RFC6066] does not extend to
   intermediate certificates used in a certificate chain. Although
   the Multiple Certificate Status extension [RFC6961] addresses this
   shortcoming, it is a recent addition without much deployment.

o Both CRLs and OCSP depend on relatively reliable connectivity to
   the Internet, which might not be available to certain kinds of
   nodes (such as newly provisioned devices that need to establish a
   secure connection in order to boot up for the first time).

With regard to common public key certificates, servers SHOULD support
the following as a best practice given the current state of the art
and as a foundation for a possible future solution:

1. OCSP [RFC6960]

2. Both the status_request extension defined in [RFC6066] and the
    status_request_v2 extension defined in [RFC6961] (This might
    enable interoperability with the widest range of clients.)

3. The OCSP stapling extension defined in [RFC6961]

The considerations in this section do not apply to scenarios where
the DANE-TLSA resource record [RFC6698] is used to signal to a client
which certificate a server considers valid and good to use for TLS
connections.

7. References

7.1. Normative References

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

   [RFC2818]   Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000,
               <http://www.rfc-editor.org/info/rfc2818>.

   [RFC3766]   Orman, H. and P. Hoffman, "Determining Strengths For
               Public Keys Used For Exchanging Symmetric Keys", BCP 86,
               RFC 3766, April 2004,
               <http://www.rfc-editor.org/info/rfc3766>.

   [RFC4492]   Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B.
               Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites
               for Transport Layer Security (TLS)", RFC 4492, May 2006,
               <http://www.rfc-editor.org/info/rfc4492>.

   [RFC4949]   Shirey, R., "Internet Security Glossary, Version 2", FYI
               36, RFC 4949, August 2007,
               <http://www.rfc-editor.org/info/rfc4949>.

   [RFC5246]   Dierks, T. and E. Rescorla, "The Transport Layer Security
               (TLS) Protocol Version 1.2", RFC 5246, August 2008,
               <http://www.rfc-editor.org/info/rfc5246>.

   [RFC5288]   Salowey, J., Choudhury, A., and D. McGrew, "AES Galois
               Counter Mode (GCM) Cipher Suites for TLS", RFC 5288,
               August 2008, <http://www.rfc-editor.org/info/rfc5288>.

   [RFC5289]   Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-
               256/384 and AES Galois Counter Mode (GCM)", RFC 5289,
               August 2008, <http://www.rfc-editor.org/info/rfc5289>.

   [RFC5746]   Rescorla, E., Ray, M., Dispensa, S., and N. Oskov,
               "Transport Layer Security (TLS) Renegotiation Indication
               Extension", RFC 5746, February 2010,
               <http://www.rfc-editor.org/info/rfc5746>.

   [RFC6066]   Eastlake 3rd, D., "Transport Layer Security (TLS)
               Extensions: Extension Definitions", RFC 6066, January
               2011, <http://www.rfc-editor.org/info/rfc6066>.

   [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and
              Verification of Domain-Based Application Service Identity
              within Internet Public Key Infrastructure Using X.509
              (PKIX) Certificates in the Context of Transport Layer
              Security (TLS)", RFC 6125, March 2011,
              <http://www.rfc-editor.org/info/rfc6125>.

   [RFC6176] Turner, S. and T. Polk, "Prohibiting Secure Sockets Layer
              (SSL) Version 2.0", RFC 6176, March 2011,
              <http://www.rfc-editor.org/info/rfc6176>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, January 2012,
              <http://www.rfc-editor.org/info/rfc6347>.

   [RFC7465] Popov, A., "Prohibiting RC4 Cipher Suites", RFC 7465,
              February 2015, <http://www.rfc-editor.org/info/rfc7465>.

7.2. Informative References

   [BETTERCRYPTO]
              bettercrypto.org, "Applied Crypto Hardening", April 2015,
              <https://bettercrypto.org/static/
              applied-crypto-hardening.pdf>.

   [CAB-Baseline]
              CA/Browser Forum, "Baseline Requirements for the Issuance
              and Management of Publicly-Trusted Certificates Version
              1.1.6", 2013, <https://www.cabforum.org/documents.html>.

   [DANE-SMTP]
              Dukhovni, V. and W. Hardaker, "SMTP security via
              opportunistic DANE TLS", Work in Progress, draft-ietf-
              dane-smtp-with-dane-16, April 2015.

   [DANE-SRV] Finch, T., Miller, M., and P. Saint-Andre, "Using DNS-
              Based Authentication of Named Entities (DANE) TLSA Records
              with SRV Records", Work in Progress,
              draft-ietf-dane-srv-14, April 2015.

   [DEP-SSLv3]
              Barnes, R., Thomson, M., Pironti, A., and A. Langley,
              "Deprecating Secure Sockets Layer Version 3.0", Work in
              Progress, draft-ietf-tls-sslv3-diediedie-03, April 2015.

   [DegabrieleP07]
              Degabriele, J. and K. Paterson, "Attacking the IPsec
              Standards in Encryption-only Configurations", IEEE
              Symposium on Security and Privacy (SP '07), 2007,
              <http://dx.doi.org/10.1109/SP.2007.8>.

   [ECRYPT-II]
              Smart, N., "ECRYPT II Yearly Report on Algorithms and
              Keysizes (2011-2012)", 2012,
              <http://www.ecrypt.eu.org/ecrypt2/>.

   [Heninger2012]
              Heninger, N., Durumeric, Z., Wustrow, E., and J.
              Halderman, "Mining Your Ps and Qs: Detection of Widespread
              Weak Keys in Network Devices", Usenix Security Symposium
              2012, 2012.

   [IANA-TLS] IANA, "Transport Layer Security (TLS) Parameters",
              <http://www.iana.org/assignments/tls-parameters>.

   [Kleinjung2010]
              Kleinjung, T., "Factorization of a 768-Bit RSA modulus",
              CRYPTO 10, 2010, <https://eprint.iacr.org/2010/006.pdf>.

   [Krawczyk2001]
              Krawczyk, H., "The Order of Encryption and Authentication
              for Protecting Communications (Or: How Secure is SSL?)",
              CRYPTO 01, 2001,
              <https://www.iacr.org/archive/crypto2001/21390309.pdf>.

   [Multiple-Encryption]
              Merkle, R. and M. Hellman, "On the security of multiple
              encryption", Communications of the ACM, Vol. 24, 1981,
              <http://dl.acm.org/citation.cfm?id=358718>.

   [NIST.SP.800-56A]
              Barker, E., Chen, L., Roginsky, A., and M. Smid,
              "Recommendation for Pair-Wise Key Establishment Schemes
              Using Discrete Logarithm Cryptography", NIST Special
              Publication 800-56A, 2013,
              <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/
              NIST.SP.800-56Ar2.pdf>.

   [POODLE]   US-CERT, "SSL 3.0 Protocol Vulnerability and POODLE
              Attack", Alert TA14-290A, October 2014,
              <https://www.us-cert.gov/ncas/alerts/TA14-290A>.

[PatersonRS11]
          Paterson, K., Ristenpart, T., and T. Shrimpton, "Tag size
          does matter: attacks and proofs for the TLS record
          protocol", 2011,
          <http://dx.doi.org/10.1007/978-3-642-25385-0_20>.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision
          3", BCP 9, RFC 2026, October 1996,
          <http://www.rfc-editor.org/info/rfc2026>.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0",
          RFC 2246, January 1999,
          <http://www.rfc-editor.org/info/rfc2246>.

[RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher
          Algorithm and Its Use with IPsec", RFC 3602, September
          2003, <http://www.rfc-editor.org/info/rfc3602>.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security
          (TLS) Protocol Version 1.1", RFC 4346, April 2006,
          <http://www.rfc-editor.org/info/rfc4346>.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
          Security", RFC 4347, April 2006,
          <http://www.rfc-editor.org/info/rfc4347>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig,
          "Transport Layer Security (TLS) Session Resumption without
          Server-Side State", RFC 5077, January 2008,
          <http://www.rfc-editor.org/info/rfc5077>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated
          Encryption", RFC 5116, January 2008,
          <http://www.rfc-editor.org/info/rfc5116>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
          Housley, R., and W. Polk, "Internet X.509 Public Key
          Infrastructure Certificate and Certificate Revocation List
          (CRL) Profile", RFC 5280, May 2008,
          <http://www.rfc-editor.org/info/rfc5280>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic
          Curve Cryptography Algorithms", RFC 6090, February 2011,
          <http://www.rfc-editor.org/info/rfc6090>.

[RFC6101] Freier, A., Karlton, P., and P. Kocher, "The Secure
          Sockets Layer (SSL) Protocol Version 3.0", RFC 6101,
          August 2011, <http://www.rfc-editor.org/info/rfc6101>.

   [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence
              Protocol (XMPP): Core", RFC 6120, March 2011,
              <http://www.rfc-editor.org/info/rfc6120>.

   [RFC6460] Salter, M. and R. Housley, "Suite B Profile for Transport
              Layer Security (TLS)", RFC 6460, January 2012,
              <http://www.rfc-editor.org/info/rfc6460>.

   [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication
              of Named Entities (DANE) Transport Layer Security (TLS)
              Protocol: TLSA", RFC 6698, August 2012,
              <http://www.rfc-editor.org/info/rfc6698>.

   [RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict
              Transport Security (HSTS)", RFC 6797, November 2012,
              <http://www.rfc-editor.org/info/rfc6797>.

   [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A.,
              Galperin, S., and C. Adams, "X.509 Internet Public Key
              Infrastructure Online Certificate Status Protocol - OCSP",
              RFC 6960, June 2013,
              <http://www.rfc-editor.org/info/rfc6960>.

   [RFC6961] Pettersen, Y., "The Transport Layer Security (TLS)
              Multiple Certificate Status Request Extension", RFC 6961,
              June 2013, <http://www.rfc-editor.org/info/rfc6961>.

   [RFC6989] Sheffer, Y. and S. Fluhrer, "Additional Diffie-Hellman
              Tests for the Internet Key Exchange Protocol Version 2
              (IKEv2)", RFC 6989, July 2013,
              <http://www.rfc-editor.org/info/rfc6989>.

   [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection
              Most of the Time", RFC 7435, December 2014,
              <http://www.rfc-editor.org/info/rfc7435>.

   [RFC7457] Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing
              Known Attacks on Transport Layer Security (TLS) and
              Datagram TLS (DTLS)", RFC 7457, February 2015,
              <http://www.rfc-editor.org/info/rfc7457>.

   [RFC7507] Moeller, B. and A. Langley, "TLS Fallback Signaling Cipher
              Suite Value (SCSV) for Preventing Protocol Downgrade
              Attacks", RFC 7507, April 2015.

   [SESSION-HASH]
             Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A.,
             Langley, A., and M. Ray, "Transport Layer Security (TLS)
             Session Hash and Extended Master Secret Extension", Work
             in Progress, draft-ietf-tls-session-hash-05, April 2015.

   [Smith2013]
             Smith, B., "Proposal to Change the Default TLS
             Ciphersuites Offered by Browsers.", 2013,
             <https://briansmith.org/browser-ciphersuites-01.html>.

   [Soghoian2011]
             Soghoian, C. and S. Stamm, "Certified lies: Detecting and
             defeating government interception attacks against SSL",
             Proc. 15th Int. Conf. Financial Cryptography and Data
             Security, 2011.

   [TLS-XMPP] Saint-Andre, P. and a. alkemade, "Use of Transport Layer
             Security (TLS) in the Extensible Messaging and Presence
             Protocol (XMPP)", Work in Progress,
             draft-ietf-uta-xmpp-07, April 2015.

   [triple-handshake]
             Delignat-Lavaud, A., Bhargavan, K., and A. Pironti,
             "Triple Handshakes Considered Harmful: Breaking and Fixing
             Authentication over TLS", 2014,
             <https://secure-resumption.com/>.

During IESG review, Richard Barnes, Alissa Cooper, Spencer Dawkins,
Stephen Farrell, Barry Leiba, Kathleen Moriarty, and Pete Resnick
provided comments that led to further improvements.

Ralph Holz gratefully acknowledges the support by Technische
Universitaet Muenchen. The authors gratefully acknowledge the
assistance of Leif Johansson and Orit Levin as the working group
chairs and Pete Resnick as the sponsoring Area Director.

Authors' Addresses

   Yaron Sheffer
   Intuit
   4 HaHarash St.
   Hod HaSharon 4524075
   Israel

   EMail: yaronf.ietf@gmail.com

   Ralph Holz
   NICTA
   13 Garden St.
   Eveleigh 2015 NSW
   Australia

   EMail: ralph.ietf@gmail.com

   Peter Saint-Andre
   &yet

   EMail: peter@andyet.com
   URI: https://andyet.com/