Using TLS to protect data

Recommended profiles to securely configure TLS for the most common versions and scenarios, with additional guidance for managing older versions.

Introduction

This guidance is aimed at system administrators who are responsible for configuring servers or clients within their estate. It describes recommended profiles for the secure configuration of TLS covering the most common versions and scenarios, with additional guidance for the disabling of insecure features present in older versions of TLS. Additional considerations for deploying TLS to web and mail servers is included at the end of this guidance.

About TLS

Transport Layer Security (TLS) is a protocol that provides security for digital communications between two parties. When a server and client communicate, well-configured TLS ensures that no third party can eavesdrop or tamper with any message. However, configuring TLS can be difficult, and if configured incorrectly, it can lead to a false sense of security.

Deprecated protocols

The predecessor to the TLS protocol was the Secure Sockets Layer (SSL) protocol, all versions of which are formally deprecated and regarded as insecure. They must **not** be used.

Equally, TLS versions 1.1 and 1.0 have been formally deprecated by the IETF, with RFC 8996, and should not be used. The NCSC recommends that government systems do **not** support deprecated versions of the protocol, and instead are upgraded to support TLS version 1.3.

Current version

At time of writing, the most recent version of TLS is 1.3, which is designed to be more secure than previous iterations. It is the only current version that will support post-quantum cryptography (see below).

TLS 1.2 is not expected to receive any new features, but it is still considered to provide strong protection where appropriate cryptographic choices are made.

Post-quantum cryptography in TLS

There is a threat from a cryptographically relevant quantum computer to key exchange and digital signature algorithms securing TLS. The primary mitigation to this risk will be to migrate to post-quantum cryptography (PQC).

The standardisation of PQC within TLS 1.3 is underway within the Internet Engineering Taskforce (IETF), with drafts at varying maturity levels. NCSC will update this guidance to include TLS PQC profile(s) as the RFCs reach maturity and there is broad enough support for those profiles in commercial TLS implementations.

Because TLS 1.2 will not be updated to support PQC, you should be building a plan for transitioning from TLS 1.2 to TLS 1.3. You should note that the services that you rely on may remove support for TLS 1.2 sooner than others - and in a few cases, will already have done do - so your client migration plan will need to accommodate this.

General TLS configuration

The profiles in this section are recommended for use with TLS in all servers and clients that are under the control of the system administrator.

There are two recommended profiles for each of TLS 1.3 and TLS 1.2. In each case, the profiles differ only in the choice of digital signature algorithm; both the choices offered provide equivalent protection.

Summary

- You should use TLS 1.3 and/or TLS 1.2, configured with the Recommended Profiles.
- While TLS 1.2 and TLS 1.3 configured with recommended cipher suites both provide cryptographically strong protection against non-quantum attacks, TLS 1.3 removes some old encryption options and makes certain attacks much harder.
- If you currently only use TLS 1.2, you will need a plan to move to TLS 1.3 to ensure you can use post-quantum cryptography in future.
- If you need to support a wide variety of clients, you may want to support the TLS 1.2 Compatibility Profile.
- You should disable TLS 1.1 and 1.0.
- You must disable TLS features known to be insecure.
- All servers and clients should use the most up-to-date software version available. Implementation issues can introduce vulnerabilities that can be exploited, if not patched promptly.

TLS 1.3 configuration

TLS 1.3 introduced some major changes. It uses different cipher suite definitions to earlier TLS versions, and has different configuration options.

Some implementations separate TLS 1.3 configuration from previous TLS versions. Please consult your implementation documentation to determine how to configure options specific to TLS 1.3.

The TLS 1.3 specification only supports strong cryptographic algorithms. However, the NCSC recommends the following profiles:

Recommended Profiles for TLS 1.3

Key Exchange	ECDHE using P-256 curve
Authentication	ECDSA with SHA256 on P-256 curve
Encryption	AES with 128-bit key using GCM
HKDF Algorithm	SHA256
Cipher Suite	TLS_AES_128_GCM_SHA256

Key Exchange	ECDHE using P-256 curve
Authentication	RSA signatures with 2048-bit modulus and SHA256 digests
Encryption	AES with 128-bit key using GCM
HKDF Algorithm	SHA256
Cipher Suite	TLS_AES_128_GCM_SHA256

TLS 1.2 configuration

Where client and server implementations support AES-GCM, the Recommended Profiles below should be used.

If AES-GCM is not supported, then the Compatibility Profile for TLS 1.2 may be used.

Recommended Profiles for TLS 1.2

Key Exchange	ECDHE using P-256 curve
Authentication	ECDSA with SHA256 on P-256 curve
Encryption	AES with 128-bit key using GCM
PRF	SHA256
Cipher Suite	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

Key Exchange	ECDHE using P-256 curve
Authentication	RSA signatures with 2048-bit modulus and SHA256 digests
Encryption	AES with 128-bit key using GCM
PRF	SHA256
Cipher Suite	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

Compatibility Profile for TLS 1.2

Key Exchange	ECDHE using P-256 curve
Authentication	RSA signatures with 2048-bit modulus and SHA256 digests
Encryption	AES with 128-bit key using CBC
PRF	SHA256
Cipher Suite	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

Legacy versions

Legacy Profiles for TLS 1.1 and 1.0

TLS versions 1.1 and 1.0 lack support for current authenticated encryption cipher suites and have other known vulnerabilities. These old versions have been formally deprecated by the IETF (RFC 8996) and should no longer be used.

A time-bounded migration plan should be put in place for any government systems using deprecated TLS versions, or cipher suites. Where this is not possible, system owners should approach the NCSC for advice.

The following legacy profiles are defined only for use-cases where TLS versions 1.1 and 1.0 cannot be disabled. In all other circumstances, TLS 1.2 or 1.3 recommended profiles should be used and support for TLS 1.0 and 1.1 must be disabled.

Key Exchange	ECDHE using P-256 curve
Authentication	RSA signatures with 2048-bit modulus and SHA256 digests
Encryption	AES with 128-bit key using CBC
MAC	SHA1
Cipher Suite	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

Key Exchange	DH Group 14 (2048-bit modulus)
Authentication	RSA signatures with 2048-bit modulus and SHA256 digests
Encryption	AES with 128-bit key using CBC
MAC	SHA1
Cipher Suite	TLS_DHE_RSA_WITH_AES_128_CBC_SHA

Third-Party services

When a TLS service is hosted on the infrastructure of a third-party provider, such as a Content Delivery Network (CDN), the TLS configuration may not be under the direct control of the data owner. Where this is the case, the service provider will normally allow the data owner to choose a policy that specifies which TLS versions and cipher suites are allowable.

The data owner should choose the most restrictive of the available policies. This policy should enable TLS 1.3 and/or TLS 1.2 as required, and should include the above recommended profiles for TLS 1.3 and/or TLS 1.2.

Avoiding known insecurities

Old TLS versions (TLS 1.2 and below) include some features and cryptographic components that provide weak security. In most cases, these are only negotiated in a connection if they are supported by both client and server.

Using up-to-date software will reduce the chance of insecure features being used. To avoid some of the most common vulnerabilities, you must ensure that the following features and components are disabled in the TLS implementations you use:

- Disable TLS insecure renegotiation (by either disabling renegotiation, or enabling the Renegotiation Indication Extension, as described in RFC 5746).
- Disable TLS insecure protocol downgrade (by supporting the Fallback Signaling Cipher Suite Value, as described in RFC 7507). *Note that this is only necessary if you support TLS 1.1 or older.*
- Disable TLS record compression (as described in RFC 5246, section 6.2.2).
- Disable export key generation by ensuring EXPORT ciphers are disabled (as described in RFC 2246, section 6.3.1).
- Disable support for any deprecated SSL/TLS versions that are not required; for example, SSLv2 can be used to attack sessions created using modern TLS versions.

TLS 1.3 provides an optional zero round trip mode, known as 0-RTT, which allows clients to send data early in the TLS session, before the full TLS handshake is complete. Unlike most data exchanged in a TLS session, early data can be replayed, which can result in security vulnerabilities if use of 0-RTT isn't managed carefully.

> **Note:**
> Some application protocols built on TLS 1.3 include mitigations for attempted replay attacks, and can safely use 0-RTT mode. You should not enable 0-RTT mode unless you are confident that the application protocols and the software you are using mitigate this risk.

Using Certificates with TLS

TLS uses X.509v3 certificates to cryptographically verify identities presented by one or both communicating parties. If the connecting party can verify that the certificate presented to it is valid and has been issued by a Certificate Authority (CA) it trusts, then secure communications can be established.

In many cases a TLS service will need to be used and trusted by unknown clients (e.g. members of the public), in which case the service must use a certificate from a public CA in the Web PKI. The NCSC provides guidance on Provisioning and Managing Certificates in the Web PKI.

Many CAs exist, with a variety of acceptance by client trust stores, technical support, and methods of issuance/management. You will need to research which CA best meets your needs and is included in the trust stores of the clients you expect to connect to your service.

If you're hosting your service in the cloud, then many cloud providers offer CA services to customers and host their own CAs. Using these cloud native CA services can simplify integration and certificate management automation, as long as the cloud's CA offers sufficient security properties for your purposes. Following the NCSC's Cloud Security Guidance will help you make this assessment.

Where an organisation manages their own internal CA, and the TLS service is hosted internally with only managed clients which are configured to trust the internal CA, it may be appropriate to use certificates issued from this internal CA. The NCSC provides guidance for the operation of privately-hosted CAs.

When requesting or renewing a certificate, there are several parameters you can choose for your private key and certificate signing request. Whether you're using a public or a private CA, as per the profiles above, we recommend using the following parameters for generation of ECC and RSA keys:

- ECDSA-256 with SHA256 on the P-256 curve,
- 2048-bit RSA with SHA256.

It should be noted that both ECC and RSA keys are vulnerable to attack from future quantum computers. Therefore when obtaining new certificates and deciding on appropriate certificate lifetimes you should take into account your PQC migration timelines to ensure that certificates using ECC or RSA keys don't have lifetimes extending beyond the date you anticipate for ending reliance on traditional, non-quantum safe cryptography. It would also be beneficial to collate information on any certificates you already hold whose validity period extends beyond this date so that these can be upgraded appropriately.

Deploying TLS for web servers

Web servers should use TLS as described in the sections above. However, there are some HTTPS-specific features you should use to improve the security of your web service or website:

- Publish services only using HTTPS.
- Redirect unencrypted HTTP requests to the HTTPS version.
- Use HTTP Strict Transport Security (HSTS) to force all connections to your web server to use HTTPS instead of unencrypted HTTP and preload your HSTS policy (see hstspreload.org for more details).
- Use the upgrade-insecure-requests directive in your Content Security Policy, to force all content on your site (including third party content) to be loaded using HTTPS instead of unencrypted HTTP.
- Ensure that cookies are marked as “secure” so they are only transmitted within HTTP requests that are performed over TLS.

Remember that TLS offers a specific set of security properties, and that use of TLS is a significant, but not the only, part of making your web application secure.

Deploying TLS for mail servers

TLS in this setting can be more complicated than other scenarios, so extra care should be taken to prevent problems, such as mail being sent in plaintext, or mail failing (silently) to be delivered.

For handling connections using SMTP STARTTLS on port 25, with other Mail Transfer Agents (MTAs), mail servers should use TLS as described in the sections above.

Note that a failure to negotiate a TLS session may result in a fall-back to the email being sent in plain text. To reduce the likelihood of this, you should try to maximise compatibility with other MTAs by using the Recommended Profiles for TLS 1.3 and 1.2 and the Compatibility Profile for TLS 1.2. You should inspect your mail server logs to ensure that your TLS configuration is not causing connections to fall-back to no TLS, due to a strict set of TLS profiles.

For handling connections to end users via Mail User Agents (MUAs) using protocols such as IMAP, POP3 and SMTP Submission, mail servers should use TLS as described in the sections above with the Recommended Profiles for TLS 1.3 and 1.2.

As per RFC 8314, STARTTLS for communications between end users and mail services is no longer recommended. Instead, "Implicit TLS" should be used, where the TLS session starts immediately after connection establishment.

For example, with IMAP, this means using the dedicated IMAPS TCP/993 port, instead of STARTTLS on the IMAP TCP/143 port.

In addition to this TLS guidance, you should consult the NCSC guidance on Email security and anti-spoofing.

Optionally, you may wish to consider supporting the following standards that make TLS for email more visible and robust:

- Deploy Mail Transfer Agent Strict Transport Security (MTA-STS) to force supporting MTAs to only deliver mail using TLS. You should start by deploying in *testing* mode, before moving to *enforce* mode, to ensure no loss of mail delivery occurs in the event that configuration errors are present.
- Deploy SMTP TLS Reporting (TLS-RPT), as described in RFC 8460, to allow supporting MTAs to report TLS issues to mail server administrators. This is strongly recommended if deploying MTA-STS, as it can alert to issues that prevent mail delivery.

It is important to note that the verification of mail server certificates will be stricter with MTAs that implement these standards. Once MTA-STS is activated, the mail server certificate is expected to match against the host name listed in the MX record that corresponds to the mail server, and the certificate must chain to a CA trusted by the sender.

Testing web and mail servers

Given the wide range of configuration options available for TLS, we recommend that you regularly test the configuration of your web servers and mail servers by running External Attack Surface Management (EASM) products. These will identify most common issues and configuration problems; further advice is available in the NCSC's External attack surface management (EASM) buyers guide. EASM products are not a replacement for skilled penetration testing of your services, but if you have already used tools such as these to help identify and mitigate common issues, then penetration testers will have more time to spend ensuring there are not more subtle or unique flaws in your service.

Note that, whilst it is possible for others to test your ability to receive email securely, it is not possible for others, without your cooperation, to test the ability of your services to send email securely. You must send an email to a testing service.

PUBLISHED

21 July 2021

REVIEWED

10 December 2025

VERSION

1.1

WRITTEN FOR

Large organisations

Small & medium sized organisations

Public sector