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 applies to any TLS service.

We have included 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, regarded as insecure, and 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 using TLS version 1.1 or 1.0 should be upgraded to TLS version 1.3 or 1.2.

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.

We recommend that only TLS versions 1.3 and 1.2 be deployed.

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. The profiles differ only in the choice of digital signature algorithm. Both provide equivalent protection, so either or both may be configured.

Summary

- You should use TLS 1.3 and/or TLS 1.2, configured with the Recommended Profiles.
- When configured correctly, both TLS 1.3 and TLS 1.2 provide strong protection for data sent between client and server. TLS 1.3 removes some outdated cryptography and makes certain attacks much harder, but support for TLS 1.3 may not always be possible (e.g. for some enterprise setups).
- 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 for use-cases where TLS versions 1.1 and 1.0 cannot 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.

A policy which enables legacy TLS versions and cipher suites should only be used when absolutely necessary, and only for the service(s) that require it, not for all services.

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 SSL 2, as it can be used to attack stronger connections.

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. When used insecurely, this mode permits replay attacks. Depending on the application protocol that is using TLS, this may result in security vulnerabilities.

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 these risks are mitigated by the application protocols and software you are using.

Choosing a Certificate Authority

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 issued by a Certificate Authority (CA) it trusts, then secure communications should be established. It is therefore important to choose a CA that is widely trusted by those likely to connect to your services.

If selecting a less common CA, it is possible that the certificates presented by your service may not be verifiable by the connecting party, and the other party may choose not to continue the session.

In order to verify the certificate presented by your server, the client may use its own store of trusted root CAs, or it may refer to the CAs trusted by the operating system. As there is no universal list of trusted CAs, you will need to research support for your chosen CA amongst the clients expected to connect to your service.

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

Many cloud providers offer CA services to customers, where customers can obtain certificates for use with TLS that are issued by a public CA, under the control of the cloud provider. If you are using cloud services, check the relevant documentation for instructions on how to obtain certificates from these services.

If you are managing your own TLS services and wish to have these services used and trusted by unknown clients, your service must use a certificate from a public CA. Many CAs exist, with a variety of acceptance by client trust stores, technical support, and methods of issuance. You will need to research which CA best meets your needs.

Limiting issuance

When you have chosen a CA, you should publish DNS Certificate Authority Authorization (CAA) records to restrict issuance of certificates for your domain to your chosen CA.

This can help reduce the risk of unauthorised certificates being issued by other CAs. Details on how to populate these records should be provided by your CA.

Revoking certificates

You should ensure you have a means of revoking compromised certificates for your services.

Your chosen CA should provide a method for requesting revocation, along with a suitable mechanism for distributing notification of revoked certificates, such as Certificate Revocation Lists (CRLs) or Online Certificate Status Protocol (OCSP).

This will allow for validating parties which check CRLs, or use OCSP to be notified of revoked certificates, and no longer consider them trusted.

Requesting a certificate

In order to have an X.509v3 certificate signed by a CA, you would normally generate your own private key, and send a Certificate Signing Request (CSR) to the CA to be signed, thus keeping your private key secret. This can be either a manual process, or automated by various tooling.

There are several parameters you can choose for your private key and signing request. As per the profiles above, we recommend using the following parameters for generation of elliptic curve and RSA keys:

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

You should investigate whether storage of the private key within a Hardware Security Module (HSM) is appropriate for your deployment.

Renewing certificates

At the time of writing, certificates from public CAs are issued with a maximum validity period of 13 months, allowing for one year of use, with a one month rollover period. It is highly recommended that system administrative processes account for this and renew the certificate at least one month before expiry.

Some CAs support the use of various tools that use certificate management protocols, which permit automated renewal of certificates without administrator intervention. These include, ACME, CMPv2, SCEP or Windows Client Certificate Enrolment Protocol. These tools should be used where possible, to avoid accidental expiry of certificates, which can lead to downtime for TLS services.

Where a CA issues short lived certificates, automated renewal is recommended to ensure that the TLS service has a valid certificate at all times. You should periodically review logs from the automated renewal software to ensure that the renewal process is operating correctly. You should also keep up-to-date with developments at your chosen CA, to ensure there are no upcoming changes that may prevent the automated renewal process from being able to renew certificates.

Monitoring

You may wish to consider monitoring Certificate Transparency (CT) logs. This will allow you to find certificates that have been issued to your domains, including sub-domains, and thus detect mis-issuance of certificates by CAs, or unexpected certificates that have been issued for your domains.

You can monitor CT logs manually by using publicly available services, or you may subscribe to commercial offerings who perform the monitoring and alerting on your behalf.

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.

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 emerging 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 automated scans.

The NCSC operates the Web Check and Mail Check services for owners of websites and mail servers in the public sector. This will check for a range of potential security issues including TLS configuration. For the private sector, there are various free or commercial tools available to help you test the TLS configuration of your web or mail server.

These scans will identify most common issues and configuration problems. They 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

21 July 2021

VERSION

1.0

WRITTEN FOR

Large organisations

Public sector

Cyber security professionals

Small & medium sized organisations