SSL/TLS server configuration

SURFconext policy mandates the use of HTTPS URLs on all protocol endpoints. This means the use of TLS for protecting the communication between clients (a user's browser) and servers (a web server hosting IdP or SP software). This document contains a checklist for the correct configuration of such servers. Web server administrators are urged to use this checklist to assess the security of their TLS configuration, as insecure configurations can lead to problems like phishing attacks and disclosure of privacy-sensitive user information.

If you would like to test your server for configuration issues, you can use the Qualys SSL labs server test at: https://www.ssllabs.com/ssltest/. As a rule of thumb, it should have at least a "B" rating on SSLLabs.com (but preferably better, of course). For practical documentation on using openssl-based software, see https://www.feistyduck.com/books/openssl-cookbook/ or the page outlining SSL and TLS Deployment Best Practices . For other software, you can refer to the book https://www.feistyduck.com/books/bulletproof-ssl-tls-and-pki/. Practical configuration examples can be found in a separate blog post.

There are many Certification Authorities (CA's) available. If you are a customer of SURF (or another European NREN), you can use the GÉANT Trusted Certificate Service for obtaining server certificates at a flat fee.

Please note that

- this document may become out-of-date whenever new vulnerabilities in the TLS protocol or one of its associated technologies are discovered. If you spot such info, we appreciate a message.
- we may periodically scan hosts connected to SURFconext for known vulnerabilities in their TLS implementation or configuration.
- this page deals with TLS configuration of your HTTP server only, similar recommendations hold for SAML Security. See also the OWASP SAML Security Cheat Sheet.

The checklist is divided into four sections and uses terminology from RFC 2119 to indicate requirement levels:

Keys

When generating keys for your certificate, you MUST make sure that:

- your key is generated using a good random number generator.
- your key is at least 2048 bits long (when using RSA as the public key algorithm)

Certificates

When obtaining an X.509 certificate for your server, you **MUST** make sure that:

- The certificate's subject matches the FQDN of the server as listed in its SAML 2.0 metadata.
- For IdP's, the server's domain name is owned by the SURFconext member organization, and the certificate lists the organization name in its subject organization component (O). This means the certificate is obtained through Organization Validation (OV) or, preferably, Extended Validation (EV).
- The certificate is valid. Certificates that have expired cannot be trusted, for example because they are removed from certificate revocation lists. Install procedures to make sure certificates are renewed in time.
- All hostnames that point to your server are contained in the certificate, either as CN attribute in the certificate's subject DN, or as subjectAlternativeName in the certificates X.509v3 extension.
- The certificate is trusted by all common browsers by using a well-known Certification Authority (CA), i.e. one that has its root certificate embedded in such browsers.
- The certificate chain is installed correctly, i.e. all intermediate CA certificates are sent by your server in the TLS handshake, and in the right order.

Furthermore, you **SHOULD**:

- prefer a CA that uses a modern signature hash algorithm, preferably from the SHA2 family, instead of the soon obsolete SHA-1 hash algorithm.

Use Secure Protocols

Currently there are five protocols in the SSL/TLS family: SSL v2, SSL v3, TLS v1.0, TLS v1.1, and TLS v1.2. When configuring your server with a list of acceptable SSL/TLS protocol versions, you **MUST**:

- **Upgrade your server if it does not support TLS v1.2. Upgrade your platform as soon as possible if this is the case. TLS v1.2 or TLS v1.3 should be your main protocol because these versions offer modern authenticated encryption (also known as AEAD). If you don't support TLS v1.2 or TLS v1.3 today, your security is lacking.**
- **disable** the insecure SSLv2 protocol
  - SSL v2 is insecure and must not be used. This protocol version is so bad that it can be used to attack RSA keys and sites with the same name even if they are on an entirely different servers (the DROWN attack).
- **disable** the insecure SSLv3 protocol (only a problem for clients running IE6 on Windows XP - which is EOL)
  - SSL v3 is insecure when used with HTTP (the POODLE attack) and weak when used with other protocols. It's also obsolete and shouldn't be used.
- **disable** TLS v1.0
  - TLS v1.0 is a legacy protocol that shouldn't be used. Its major weakness (BEAST) has been mitigated in modern browsers, but other problems remain. Browser support for TLS v1.0 and TLS v1.1 will be dropped January 2020. Stop supporting older clients as you need to continue to support TLS v1.0 and TLS v1.1 for this. **Retire TLS v1.0.**
- **disable** TLS v1.1. See the remarks about TLS 1.0 above. Browser support for TLS v1.1 will be dropped dropped January 2020. **TLS 1.1 should also be retired.**
- **disable compression**.
  - The 2012 CRIME attack showed that TLS compression can't be implemented securely. The only solution was to disable TLS compression altogether.
- **disable CBC Ciphers**
  - Vulnerabilities like Zombie POODLE, GOLDENDOODLE, 0-Length OpenSSL and Sleeping POODLE were published for websites that use CBC (Cipher Block Chaining) block cipher modes. These vulnerabilities are applicable only if the server uses TLS 1.2 or TLS 1.1 or TLS 1.0 with CBC cipher modes.
  - For clients on Windows 7, upgrade to browsers that support recent ciphers. If you still have Windows 7 in use, take note that support will be dropped by Microsoft January 14, 2020**.**

Furthermore, you **SHOULD**:

- **enable** the TLS protocol v1.2 and TLS v1.3. Currently these are both without known security issues.
  - Although relativly new TLS v1.3 has some benefits. The performance is improved as is the latency. It has an improved security and obsolete/insecure features like cipher suites, compression etc are removed.
- **enable** Forward secrecy
  - Forward secrecy (sometimes also called perfect forward secrecy) is a protocol feature that enables secure conversations that are not dependent on the server's private key. With cipher suites that do not provide forward secrecy, someone who can recover a server's private key can decrypt *all* earlier recorded encrypted conversations. You need to support and prefer ECDHE suites in order to enable forward secrecy with modern web browsers. To support a wider range of clients, you should also use DHE suites as fallback after ECDHE. Avoid the RSA key exchange unless absolutely necessary.
- **Deploy HTTP Strict Transport Security**.
  - HTTP Strict Transport Security (HSTS) is a safety net for TLS. It was designed to ensure that security remains intact even in the case of configuration problems and implementation errors.
- **Encrypt Everything**

Ciphers

Use Secure Cipher Suites. When configuring your server with a list of acceptable ciphers, you MUST:

- exclusively use ciphers with a key length **at least 128bits**
- **exclude** cipher suites that contain MD5 as its hash algorithm
- set the cipher **suite order** on the server if TLS 1.0 is enabled (which you should not).

and understand

- anonymous Diffie-Hellman (ADH) suites do not provide authentication.
- NULL cipher suites provide no encryption.
- export cipher suites are insecure when negotiated in a connection, but they can also be used against a server that prefers stronger suites (the FREAK attack).
- Suites with weak ciphers (112 bits or less) use encryption that can easily be broken are insecure.
- RC4 is insecure.
- 64-bit block cipher (3DES / DES / RC2 / IDEA) are weak.
- Cipher suites with RSA key exchange are weak i.e. TLS_RSA

Futhermore:

- Prefer to use AEAD (Authenticated Encryption with Associated Data) cipher suites – CHACHA20_POLY1305, GCM and CCM
- As well as PFS (Perfect Forward Secrecy) ciphers – ECDHE_RSA, ECDHE_ECDSA, DHE_RSA, DHE_DSS, CECPQ1 and all TLS 1.3 ciphers
- Use only cipher suites that do not contain RC4 (See also RFC 7465).
- prefer ciphers in Galois/Counter Mode (GCM) by listing them first

Use strong Key Exchange

For the key exchange, public sites can typically choose between the classic ephemeral Diffie-Hellman key exchange (DHE) and its elliptic curve variant, ECDHE. There are other key exchange algorithms, but they're generally insecure in one way or another. The RSA key exchange is still very popular, but it doesn't provide forward secrecy.

In 2015, a group of researchers published new attacks against DHE; their work is known as the Logjam attack. The researchers discovered that lower-strength DH key exchanges (e.g., 768 bits) can easily be broken and that some well-known 1,024-bit DH groups can be broken by state agencies. To be on the safe side, if deploying DHE, configure it with at least 2,048 bits of security. Some older clients (e.g., Java 6) might not support this level of strength. For performance reasons, most servers should prefer ECDHE, which is both stronger and faster. The `secp256r1` named curve (also known as `P-256`) is a good choice in this case.

Cookies

You should ensure that all valuable cookies your site uses, have the Secure flag set so they cannot leak to plain http. Also where appropriate, make sure the HttpOnly flag is set.

In Shibboleth, you need to enable this explicitly, like this: `<Sessions .. handlerSSL="true" cookieProps="https">`. In simpleSAMLphp, see Securing your simpleSAMLphp setup.

For your own application, it depends on the programming language and framework you use.

Practical examples of server configuration

- https://communities.surf.nl/artikel/ssllabs-rating-pimp-yours
- https://wiki.mozilla.org/Security/Server_Side_TLS#Recommended_Server_Configurations

See also

- Algorithms, key size and parameters report 2014, an ENISA report with guidelines for the selection of ciphers.
- Factsheet HTTPS could be a lot more secure, an NCSC fact sheet on the use of SHA2, PFS, and HSTS.
- ICT-beveiligingsrichtlijnen voor Transport Layer Security of the National Cyber Security Centre.
- Getting an A+ on the Qualys SSL Test - Windows Edition
- Improving SSL Security for Forefront Threat Management Gateway (TMG) 2010 Published Web Sites