Genesys Security Deployment Guide

What is TLS

4/7/2025

Contents

What is TLS

For TCP connections, the industry standard mechanism is Transport Layer Security (TLS), the modern version of the old Secure Sockets Layer (SSL) protocol. TLS is also the underlying mechanism for many higher level protocols, such as HTTPS, SIPS, LDAPS, and so on.

TLS provides encryption of data and connection peer authentication. TLS protocol has been evolving with security in mind. The most recent version of TLS is v1.2, currently considered to be the most secure.

For more information, refer to Transport Layer Security (TLS).

Benefits of TLS

There are three benefits of using TLS, as follows:

- Authentication: Certificates are used to exchange information that validates the two parties.
- Confidentiality: Encryption is used to keep the contents of the transmission private.
- Integrity: Verification that data was not manipulated in transit.

Follow the links to read more about each benefit.

How TLS Works

TLS provides encryption of data and authentication of connecting peers. It utilizes a set of helper cryptographic algorithms, called ciphers, chosen from a vast number of supported cryptographic algorithms. For each session, a defined set of algorithms, called a cipher suite, are used to provide the hashing, key exchange, encryption, message verification, and other parts of the protocol. The algorithms chosen depend on the protocol version, implementation version and configuration, and are selected during the setup phase of the protocol, known as the handshake.

For more information, refer to Transport Layer Security (TLS).

Ciphers and Cipher Suites

A cipher suite represents a combination of encryption ciphers to achieve each of the three benefits of using TLS during handshaking, integrity checks and data exchange. TLS implementations support a variety of cipher suites. Generally, the set of available ciphers can be configured to the preferences of the user. See Tuning Available Cipher Lists for information about how to customize cipher lists.

The Handshake

The handshake is the way in which the two parties determine a mutually agreed cipher suite to use

for communication. Either party can shorten the list of acceptable cipher suites.

Until the secure connection is established, all exchanged data is not encrypted and therefore available to any party that might intercept handshake traffic. Asymmetric (public key) cryptography is used in the handshake phase to allow secure operations over a clear text connection. Well-defined key exchange algorithms (such as RSA and Diffle-Hellman provide the means to agree on an encryption key that will be used for secure communication, without providing a possible third-party intercepting party any real means of identifying that key. See Asymmetric cryptography explained for additional information about how that is possible.

At any stage of the handshake, if any party identifies any problem with the data, protocol versions, or keys provided or requested (for example, there is an encryption validation failure in steps t and u of the handshake procedure, below), the party drops the TCP connection indicating that the handshake cannot be continued. A new attempt to secure the connection can only be made by restarting the process with a new connection and a new handshake.

Important

During the handshake, the term protocol *version* has two meanings:

- The format of messages are tied to the agreed-upon TLS version, but can be upgraded. For example, an SSLv2 message can ask for TLS1.2 communication. If this is the case, communication will fall back to the agreed version
- Available cipher suites are tied to actual communication protocol version agreed-upon during the handshake.

In the simplest case, the client requests the highest version of protocol that it supports. The server responds with a protocol version equal to or lower than that requested; that is, the version that the server is willing to use for this session. If the client is not happy with the server choice (for example, the server chooses a very low version, like SSL 2), the client silently disconnects the TCP connection, and the connection is not made.

In the more complex case, specific protocol versions are specified by the client and server at the start of the negotiation. See Protocol Versions Compatibility for more information about how Genesys implements this.

Tip

99% of all errors in TLS communication occur during the handshake phase.

The handshake procedure consists of these steps. Note that this is the handshake as carried out by Genesys components; third-party components may perform the handshake differently.

[+] Show steps

1. Client establishes TCP connection to server.

2. Client requests TLS to be used.
3. Client provides acceptable protocol version and cipher lists for server to choose from. It determines its acceptable protocol version as follows:
   - If the default protocol version is configured on the client side, the client requests its highest available protocol version.
   - If a specific protocol version is configured on the client side and is available, the client requests the configured protocol version.
4. Client sends other cryptographic information (key exchange algorithm data).
5. Client indicates that server response is expected.
6. Server confirms TLS protocol is to be used.
7. If the version requested by the client is lower than any version supported by the server; the server drops the connection. For example, if the server is configured with TLSv12 and the client requests TLSv11, the server drops the connection. Otherwise, the server responds with the protocol version and cipher list that is to be used for the session. It determines the protocol version as follows:
   - If the default protocol version is configured on the server side, the server responds with the highest available protocol version equal to or lower than the version requested by the client. For example, if the client requests TLSv12, but the server has TLSv12 disabled and TLSv11 enabled, the server responds with TLSv11.
   - If a specific protocol version is configured on the server side and is available, the server responds with this version if it is equal to or lower than the version requested by the client. For example, if the server is configured with SSLv3 and the client requests TLSv11, the server responds with SSLv3.
8. The client receives the server's response with the negotiated version. If one or both of the following conditions are false, the client drops the connection silently and the connection is not made. If both conditions are true, the handshake continues.
   - The client has this version available; that is, the version is not disabled in the client's registry, and where applicable, is allowed by the value of the **sec-protocol** configuration option.
   - The client is configured to accept this version; for example, as specified by the value of the **sec-protocol** option.
9. Server sends its certificate for verification.
10. Server sends other cryptographic information (to complete key exchange).
11. If Mutual TLS is being used; the server explicitly requests client certificate for verification. Server sends a list of its own trusted CAs for client to select appropriate certificate for presentation.
12. Server indicates that client response is expected.
13. Client validates server certificate.
14. If server requested client certificate, client sends it.
15. Client completes key exchange calculations.
16. Client uses the resulting negotiated key and cipher list to encrypt and sign a test message (consisting of all data that was exchanged from the start of the session) and sends it to the server for verification.
17. If requested, server validates client certificate.
18. Server completes key exchange calculations.
19. Server uses the resulting negotiated key and cipher list to encrypt and sign a test message (consisting of all data that was exchanged from the start of the session) and sends it to the client for verification.

20. Server verifies the test message received from client.
21. Client verifies the test message received from server.
22. TLS session is established.

The handshake is finally verified at steps 16, 19, 20, and 21, in which both parties use the derived keys and agreed ciphers to encrypt and sign a test message consisting of all data that was previously exchanged during the handshake, and then exchange the test messages. To verify the result, the opposite party uses its own derived key data and ciphers to validate and decrypt the received message, and compares the decrypted text with known data that was exchanged during the handshake. As a result, both parties know that they may communicate in a safe, authenticated, and encrypted way.

After the handshake is complete, data exchange starts. Data is encrypted using the (symmetric) key that was derived and verified during the handshake. Messages are encrypted and signed for authentication using the cipher list agreed on in the Handshake. Since symmetric encryption is used, communication can occur much faster than the handshake process itself.

For more information about the TLS handshake, see SSL handshake explained.

Authentication

TLS authentication uses TLS certificates in X.509 format to tell the receiving party who the sending party is (in the **Subject** or **Subject Alternate Name** field), and includes the public key. The X.509 format is standard in the industry, but there are different storage formats, such as RSA, PKSC#12, PEM, DER, and so on. The certificate must be issued and signed by a Certificate Authority (CA) that both parties trust, and therefore the information in the certificate can also be trusted. The certificates themselves are stored in a keystore that the sender application can access.

Modes of Operation

TLS operates in one of three different modes:

1. **Encryption Only:** Neither party does any validation of the other's certificate, essentially removing the authentication part. This mode can be selected by explicitly specifying anonymous, non-authenticating ciphers. Genesys strongly recommends that you use this mode only for purposes of debugging.
2. **Simple TLS:** Only the server provides a certificate to the client, which the client validates for authentication. The client does not provide a certificate for server verification. This mode is used most often for HTTP(s) websites where consumers (the clients) want to know they are communicating with the real website (the server), but typically do not have a certificate to pass to the server.
3. **Mutual TLS (also known as Mutual Certificate Exchange):** Both the client and server pass certificates to each other and each validates (authenticates) the other party, establishing a two-way trust. See steps j, m, and p in the handshake procedure.

TLS Certificates

A TLS certificate is a stored and cryptographically signed entity that authenticates the presenter of the certificate. It is signed by the issuer certificate authority (CA). Certificate signature can be easily verified by any party having only the issuer and issued certificate. CAs can have a nested structure, with the issuing party chain leading to the root level, to a so-called *root CA*, which signs its own certificate. Such root CA certificates must be known to, and trusted by, the local system to establish all the chain of trust, from the leaf certificate presented for verification to the trusted root CA certificate.

For more information about CAs and certificate chains, refer to Certificate Authority.

Certificate Signature Hash Function

An important aspect of a TLS certificate is the cryptographic hashing function used to create the certificate signature. Certificates signed by a weak hash function can be fraudulently duplicated, resulting in a major security breach. A number of cryptographic hash functions exist in widespread usage, including the following:

- MD4
- MD5
- SHA-1
- SHA-2 (a family of algorithms)

Currently, MD4 and MD5 hashing functions are considered totally flawed and compromised. SHA-1 is currently being phased out, due to a theoretical flaw found. Major software vendors, including Mozilla and Microsoft, are working on replacing certificates signed by SHA-1, and have announced that these certificates will not be accepted starting from January, 2017. SHA-2, a family of cryptographic hash functions, is now the industry standard, including the most widely used SHA-256.

> **Important**
>
> Genesys strongly recommends that all generated certificates are signed with SHA-256.

Private and Public Keys

Another important part of the entity certificate is the cryptographic public key stored within the certificate. Asymmetric cryptography principles allow public keys to be shared universally. The public key is used to encrypt data so it can only be decrypted by using the private key of the same key pair. CA certificates contain CA public keys, which are used in the issued certificate signature verification.

To successfully decrypt the data sent by the peer, a TLS entity must use the private key generated along with the public key that is stored in the certificate. Private keys must be carefully protected, because it gives access to all cryptographic communications performed using the certificate. A compromised private key is one of the reasons a certificate can be revoked.

Certificate Exchange and Validation

Certificates are exchanged during the first stages of the TLS handshake. The purpose of this exchange is to provide the certificate to the remote party for validation.

The receiver of the certificate performs various checks before accepting the certificate as valid, such as:

- **Expiration**: Is the certificate expired?
- **Certificate Authority**: Is the certificate signed by a CA I trust? Note that this might be via a certificate chain where Intermediate Authorities signed the certificate and their certificates are signed by the root CA.
- **Revocation**: Has the certificate been revoked by being added to a Certificate Revocation List?
- **Purpose**: Is the certificate created for the purpose for which it is being used? This might be one or both of the following:
  - Is the subject of the certificate appropriate? Refer to Target Hostname Check below.
  - Is the certificate type, as provided in the `Certificate Usage` field in the certificate `Extensions`, appropriate? For example:

```
Extensions:
    Identifier: Netscape Certificate Type - 2.16.840.1.113730.1.1
      Critical: no
      Certificate Usage:
        SSL CA
        Secure Email CA
        ObjectSigning CA
```

Target Hostname Check

The client side of a TLS connection can also check that the server certificate has been issued to the same server to which it is connecting. The client side of the connection connects to the server's FQDN, and therefore might check that the target FQDN matches the FQDN listed in the server certificate. The server side of the connection is unable to perform this check, even if mutual TLS is used, because that server might not (and usually does not) have access to rDNS mechanisms required to identify the client's FQDN. The client may be located behind NATs, including server internal NATs, which do not allow the tracing of the client connection back to the originating network node. Even with a direct connection, running an rDNS check during a TLS handshake leads to additional delays in an otherwise very lengthy process.

The Target hostname check is performed against the Subject Alternative Name (SAN) in the server certificate presented for authentication. If SAN is not present, the certificate subject Common Name (CN) is used for verification. The certificate must list the server FQDN, as designed, to be accessed by client connections.

> **Warning**
> Some TLS implementations allow wildcard symbols to be used in the configured FQDN

> to match multiple (sub)domains with one certificate, but this can be very dangerous and is not recommended by Genesys. Refer to the implementation details to determine if this possible with your implementation.

For information about how Genesys enables the target hostname check, refer to Check for Certificate Host Matching.

Certificate Revocation Lists

A Certificate Revocation List (CRL) is a CA-generated list of certificates issued by the CA that are no longer to be trusted because they have been compromised in some way, or are otherwise deemed not to be trusted. Strictly speaking, a check against a valid CRL is a required step in any certificate validation. In practice, this check might be omitted if the CA is controlled by the enterprise, and certificate revocation is tracked by other means.

To set up a CRL, refer to Configuring a Certificate Revocation List.

Confidentiality

To ensure confidentiality, TLS uses a session key (derived from the public key) that is used by both parties to encrypt and decrypt messages and data. The session key is used for one session only, and then discarded, so it cannot be used in later sessions to "listen in".

For confidentiality, TLS uses symmetric encryption, which is faster than other encryption methods.

Integrity

To assure data integrity, TLS uses Message Authentication Code. The hashing algorithm in the agreed-upon cipher suite generates a checksum for the message contents and stores it in the message. If the checksum generated at the receiver's end does not match that checksum in the message, the message has been manipulated during transit, most likely as the victim of a Man-in-the-middle attack.