How mTLS Works

TLS, in its normal form, is good for identifying the server you are connecting to. However, in some cases, the server also needs to be able to identify the user or machine that is talking to it. For this, we use Mutual TLS, aka mTLS. This is also known as a Client Certificate in some cases, and is used extensively in Virtual Private Networks (VPN's), amongst other things. For example, one option for creating an OpenVPN connection with the opnSense router firmware is to set up a Client Certificate.

With mTLS, two certificates are created; one for the server, and one for the user (or machine). See Create an mTLS Cert Package for instructions on how to create an internal mTLS pair.

The control flow for mutual TLS (mTLS) communication involves several key steps that ensure both the client and server authenticate each other before establishing a secure connection. Here's a structured overview of the mTLS communication process:

1. Initial Connection Request

- **Client Initiates Connection:** The client sends a connection request to the server, typically using the TLS protocol.

2. Server Responds with Certificate Request

- **Server Sends Certificate:** The server responds with its own digital certificate, which includes its public key and is signed by a trusted Certificate Authority (CA).
- **Server Requests Client Certificate:** The server also requests a certificate from the client to authenticate it.

3. Client Responds with Certificate

- **Client Sends Certificate:** The client sends its own digital certificate to the server, which includes its public key and is also signed by a trusted CA.

4. Certificate Verification

- **Server Verifies Client Certificate:**
  - The server checks the validity of the client's certificate against its trusted CA.
  - It verifies that the certificate is not expired, revoked, and that it is signed by a trusted CA.
- **Client Verifies Server Certificate:**
  - Similarly, the client verifies the server's certificate against its trusted CA.

5. Key Exchange

- **Generate Session Keys:** Both the client and server generate session keys for encrypting the data that will be transmitted during the session.

- **Key Exchange Protocol:** They may use a key exchange protocol (like Diffie-Hellman) to securely exchange keys.

6. Secure Connection Established

- **Finished Messages:** Both parties send a “Finished” message to indicate that the handshake is complete and that they are ready to start secure communication.
- **Encrypted Communication:** From this point onward, all data transmitted between the client and server is encrypted using the session keys.

7. Data Transmission

- **Mutual Authentication:** Throughout the session, both the client and server can authenticate each other using their respective certificates.
- **Secure Data Exchange:** The client and server exchange data securely, ensuring confidentiality and integrity.

8. Session Termination

- **Close Connection:** Either party can initiate the termination of the session.
- **Session Closure:** The connection is closed, and any session keys are discarded.

Attributions

- **TLS Protocol:** The mTLS process is built on the Transport Layer Security (TLS) protocol, which provides a secure channel over an insecure network.
- **Certificate Authorities (CAs):** CAs play a crucial role in the mTLS process by issuing and validating digital certificates, ensuring trust between the client and server.
- **Public Key Infrastructure (PKI):** mTLS relies on PKI for managing digital certificates and public-private key pairs, enabling secure communication.

Summary

The mTLS control flow emphasizes mutual authentication, where both the client and server present their certificates to each other. This process enhances security by ensuring that both parties are who they claim to be, thus preventing man-in-the-middle attacks and ensuring secure communication.

From:
https://kb.unixservertech.com/ - **Unix Server Tech Knowledge Base**

Permanent link:
https://kb.unixservertech.com/software/tls/mtls

Last update: **2025/07/23 07:10**