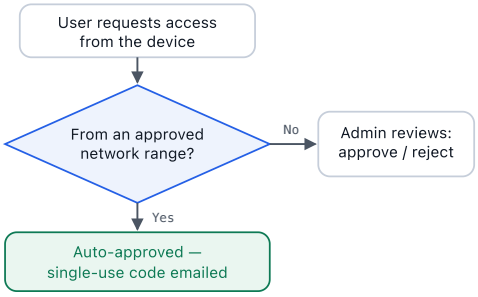
The device that enrolled is the device that connects.

PacketProx gates every connection on a device certificate whose private key is created inside the device's security chip and can never leave it. This sheet walks the enrollment cycle step by step, and what each step locks down.

First, what is mTLS? Ordinary TLS (the browser padlock) makes the *server* prove its identity. **Mutual TLS** adds the reverse: the *client* must present its own certificate *and* prove — by signing the live handshake — that it holds the matching private key. No valid certificate + key, no conversation: the PacketProx relay refuses the connection before a single byte of application data flows.

1 It starts with a request



Auto-approve is **per-domain opt-in**. Admin approval emails or texts the **enrollment token**.

`work email only` `auto or admin approval` `network fence` `single-use code`

2 The person proves who they are — once

Redeeming the code requires a **passkey** — Face ID in the app, Windows Hello in the browser — so a certificate can only ever be issued to a session **a verified human opened**. No passwords anywhere, and passkeys are **phishing-resistant by design**. This is the only moment a user credential appears: once enrolled, the device authenticates by hardware key alone (steps 4–6), never by passkey.

`WebAuthn passkey` `enrollment-time only` `phishing-resistant`

3 The device proves what it is

Before any certificate exists, the device must **attest** — manufacturer-backed proof of what it is:

iPhone / iPad — App Attest proves the genuine PacketProx app on genuine Apple hardware.

Mac — the same App Attest ceremony, with the key sealed in the Secure Enclave.

Windows 11 — the TPM's manufacturer-issued certificate plus a challenge ceremony prove a real TPM controls this key.

Android — hardware key attestation chains to Google roots (key in StrongBox/TEE, verified boot) plus a Play Integrity verdict.

Emulators, clones, and tampered apps fail here. The server **derives** the device's trust level from what it verified — the client's own claims are ignored.

`manufacturer-rooted attestation` `server-derived trust` `anti-clone`

4 A key that cannot leave the device

The device now generates its key pair **inside its security hardware** (Apple Secure Enclave / Windows TPM). The private key is **non-exportable by silicon design** — no file, no backup, no copy. The deployment's **own private certificate authority** — its signing keys sealed in a **FIPS 140-3 validated hardware security module** — then signs a **short-lived** certificate that names exactly this device (tenant + device ID are baked into it). Certificates exist only by completing this ceremony; no public CA is trusted for device identity.

`hardware keystore` `FIPS 140-3 HSM-backed CA` `unique device identity` `days, not years`

5 Every connection re-proves it

To open a TCP connection, the client presents its certificate and **signs the mTLS handshake with the hardware key** — possession is re-proven on every connection. **No user credential is involved here**: the handshake is device identity only. The relay then authorizes **each application flow individually** (is the device active? is this app assigned to it?) and writes an audit row with the true source IP and a per-flow ID.

`mTLS handshake` `per-flow authorization` `full audit trail`

6 Staying enrolled is earned

Certificates **renew at two-thirds of their short lifetime**, over mTLS, using the **same hardware key** — a device that can no longer pass checks quietly ages out. Revocation is instant: one admin click **severs live TCP connections in about a second** and every reconnect is refused from that moment.

`auto-renewal` `same key only` `instant revocation`

Why this is a guarantee, not a promise

- **A copied certificate is useless.** The certificate is only half the identity — without the private key the handshake fails, and the key physically cannot be extracted from the Secure Enclave or TPM.
- **Credentials can't be phished.** There is no password to type and no code to give away: user identity is a passkey, device identity is a chip-held key.
- **A lost device is one click from harmless.** Revoke it: live TCP connections are cut in about a second, new connections are refused, and the short certificate lifetime is the backstop.
- **There is no side door.** The relay trusts exactly one issuer — the deployment's own CA, its signing keys sealed in a FIPS 140-3 validated HSM — and that CA signs only at the end of the request → passkey → attestation ceremony above.