Time-based one-time passwords (TOTP)

Shared secrets, time windows, factors, threats and recovery

What TOTP actually does

TOTP is defined by RFC 6238 and derives a short code from a shared secret and the current time step. The authenticator application and verifier independently compute a keyed HMAC value, truncate it and display or accept a short decimal code, commonly six digits, refreshed about every 30 seconds. The phone usually needs no network connection to generate a code.

The QR code used during enrolment carries the secret and configuration information, often in an otpauth URI. Anyone obtaining that secret can generate the same future codes. The QR code and recovery codes must therefore be treated as credentials, not decorative setup material.

```
otpauth://totp/Example:alice?secret=BASE32SECRET&issuer=Example
```

Factors versus authentication steps

Two authentication steps are not automatically two different factors. A password plus TOTP normally combines knowledge and possession. An SSH private key plus TOTP commonly combines two possession-based credentials; it adds an independent challenge but is not necessarily two-factor under strict factor-category definitions.

TOTP can be phished or relayed by an attacker in real time. Hardware-backed, origin-bound FIDO2/WebAuthn methods provide stronger phishing resistance in suitable deployments. The choice depends on the threat model, service support and operational requirements.

Clock synchronisation and validation

Both parties must agree closely enough on time to calculate the same code. Verifiers often accept a small window around the current time step to tolerate drift, but a wider window increases the number of valid codes. Rate limiting and one-time-use enforcement reduce online guessing and replay risks.

Synchronise the Linux host using a trusted time service. A device clock that is substantially wrong will cause apparently valid codes to fail. Investigate clock drift before resetting credentials.

```
timedatectl status
systemctl status systemd-timesyncd
```

Enrolment and recovery

Each account should have its own secret. Enrolling the same secret in multiple compatible apps is technically possible, but every additional copy expands the compromise surface. Google Authenticator and Microsoft Authenticator can both generate standard TOTP codes; the server does not contact those vendors during verification.

Keep recovery codes securely separate from the primary authenticator. Plan a documented recovery path for a lost phone, broken clock or damaged PAM configuration. Protect backups of authenticator secret files as carefully as passwords or SSH private keys.

Limits and deployment decisions

TOTP does not defend against a compromised server, stolen shared secret, malicious endpoint or successful real-time phishing relay. It can create operational problems for unattended jobs and emergency access. Before enabling it, decide which accounts and actions require the challenge, how administrators recover access, and how changes will be validated.

For Linux services, PAM is often the integration point. Different PAM service files can have different authentication stacks; the service name used by sudo -i, sudo, sshd or su should be confirmed on the target distribution rather than assumed.

```
man pam.d
man pam_google_authenticator
```

Reference scope: commands and defaults vary by distribution and software version. Consult the locally installed manual pages for exact syntax and behaviour.