

Keeping Renewal Boring

Short-lived certificates are only a good idea if renewal is reliably automated. The price of short lifetimes is that expiry must be a non-event, and the only way to get there is automation.

Certificates in this setup presented live 90 days at most, usually less. Across a dozen hosts, each with a server certificate and one or more client certificates, manual renewal is tedious, and risks that something will expire unnoticed, and take a service down. The goal is a system where certificates are renewed before expiry, and the only time a human is involved is when renewal fails.

Per-host renewal

Each host runs the step-ca renewal service that watches its own certificates and renews them as they approach expiry. As a systemd unit:

```
[Unit]
Description=Auto-renew internal host certificate
After=network.target

[Service]
ExecStart=/usr/bin/step ca renew /etc/ssl/certs/service.crt /etc/ssl/private/service.key --
daemon --exec "systemctl reload nginx"
Restart=always
RestartSec=10
User=root
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=step-renew

[Install]
WantedBy=multi-user.target
```

The important pieces:

- **--daemon**: step runs continuously, waking on its own schedule to check whether the certificate is close enough to expiry to warrant renewal. It renews in place, so the certificate and key files are refreshed without intervention.

- **Restart=always**: if the renewal daemon dies, systemd brings it back. A renewal daemon that has quietly stopped is the silent failure this whole system exists to avoid.
- **SyslogIdentifier=step-renew**: renewal events are tagged in syslog. That tag is what makes renewal auditable so the logs can be forwarded, reviewed, and have alerts triggered on renewal failure.
- **--exec "systemctl reload nginx"**: reloads the nginx service after certificate renewal to refresh the certificates cached in memory.

The WAF's client certificates

The WAF holds a client certificate for every backend it talks to (**waf-to-service**, one per host). These are renewed on a weekly timer which executes a script to check and renew the certificates approaching expiry:

The **timer** runs it weekly. This is enabled in systemd, not the service file. The timer is the scheduler; it holds no logic beyond when.

```
[Unit]
Description=Run WAF mTLS renewal weekly

[Timer]
# Run every Monday at 02:00
OnCalendar=Mon *-*- * 02:00:00
Persistent=true
AccuracySec=1h
Unit=waf-mtls-renew.service

[Install]
WantedBy=timers.target
```

- **OnCalendar**: fires it every Monday at 02:00.
- **Persistent=true**: runs a missed occurrence at next boot if the machine was down at the scheduled time.
- **AccuracySec=1h**: lets systemd batch the wake-up for efficiency rather than firing on the exact second.

The **service** file performs the renewal pass, reissuing any **waf-to-<host>** certificate that expires within the next 21 days. The service is a oneshot: it runs the renewal script once and exits, which is the correct shape for timer-driven work rather than a long-running daemon.

```

[Unit]
Description=Renew WAF mTLS client certificates (waf-to-*.crt)
After=network-online.target
Wants=network-online.target

[Service]
Type=oneshot
ExecStart=/usr/local/bin/renew-waf-mtls.sh
User=root
PrivateTmp=yes
ProtectSystem=full
ProtectHome=read-only
NoNewPrivileges=yes

StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=waf-mtls-renew

```

- **The hardening directives** `ProtectSystem`, `ProtectHome`, `NoNewPrivileges`, and `PrivateTmp` constrain what the script can touch since it runs as root.
- `SyslogIdentifier=waf-mtls-renew`: tags its output so renewals are greppable in the logs and can feed alerting.

The script walks every waf-to-*.crt in the certificate directory, inspects each one's expiry with step certificate inspect, and renews only those inside the threshold window. Certificates with time to spare are logged as healthy and skipped, so a run that renews nothing still records the full certificate inventory and its expiry dates. When any certificate is renewed, nginx is reloaded once at the end so the WAF picks up the new certificates. When nothing is renewed, the reload is skipped to avoid needless churn.

```

#!/bin/bash
set -euo pipefail

STEP_BIN="${STEP_BIN:-/usr/bin/step}"
CERT_DIR="${CERT_DIR:-/etc/ssl/certs}"
KEY_DIR="${KEY_DIR:-/etc/ssl/private}"
CA_URL="${CA_URL:-https://10.x.x.2:9000}"
RELOAD_CMD="${RELOAD_CMD:-systemctl reload nginx}"
RENEW_THRESHOLD_HOURS=504 # 21 days

log() {

```

```

logger -p local5.info -t waf-mtls-renew "$1"
echo "$(date -Is) [waf-mtls-renew] $1"
}

RENEWED_ANY=0
SKIPPED=()
NOW_EPOCH=$(date +%s)
THRESHOLD_SECONDS=$((RENEW_THRESHOLD_HOURS * 3600))

for CRT_PATH in "$CERT_DIR"/waf-to-*.crt; do
    [[ -e "$CRT_PATH" ]] || continue

    BASENAME=$(basename "$CRT_PATH" .crt)
    KEY_PATH="$KEY_DIR/${BASENAME}.key"

    EXPIRES_TS=$("$STEP_BIN" certificate inspect --format json "$CRT_PATH" \
        | jq -r '.validity.end')
    EXPIRES_EPOCH=$(date -d "$EXPIRES_TS" +%s)
    EXPIRES_HUMAN=$(date -d "$EXPIRES_TS" '+%Y-%m-%d %H:%M %Z')
    DAYS_LEFT=$(( (EXPIRES_EPOCH - NOW_EPOCH) / 86400 ))

    if (( EXPIRES_EPOCH - NOW_EPOCH < THRESHOLD_SECONDS )); then
        log "Renewing $BASENAME (expires $EXPIRES_HUMAN, ${DAYS_LEFT}d remaining)"

        "$STEP_BIN" ca renew \
            --ca-url "$CA_URL" \
            --force \
            "$CRT_PATH" "$KEY_PATH"

        log "Renewed $BASENAME"
        RENEWED_ANY=1
    else
        SKIPPED+=("$BASENAME|$EXPIRES_HUMAN|${DAYS_LEFT}d")
    fi
done

if [[ $RENEWED_ANY -eq 1 ]]; then
    log "Reloading nginx"
    $RELOAD_CMD
else

```

```
log "No renewals needed. Current certificate status:"
for ENTRY in "${SKIPPED[@]}"; do
    IFS='|' read -r NAME EXPIRY DAYS <<< "$ENTRY"
    log "  OK  $NAME – expires $EXPIRY ($DAYS remaining)"
done
fi
```

Three pieces divide the work cleanly. The timer decides when (weekly, with catch-up for missed runs), the service defines how it runs (once, sandboxed, logged), and the script does the actual work (inspect every client certificate, renew what's near expiry, reload nginx only if something changed). The design goal behind all three is that certificate renewal is observable and self-correcting. It logs what it did and what it skipped, it recovers from a missed run, and it renews with enough margin that no single failure reaches an outage. Expiry becomes a routine log entry rather than an incident.

Revision #12

Created 2026-07-20 19:36:54 UTC by Chris Landis

Updated 2026-07-27 15:01:21 UTC by Chris Landis