CASE STUDY

Automated, and Still Expired.

A SaaS platform automated every certificate renewal, then learned that automation needs a backstop.

DOCUMENT	Case Study
INDUSTRY	SaaS
PRODUCT	Certificate Monitoring
PROFILE	Real customer deployment, anonymized

About this case study: based on a real Generator Labs customer deployment, published with identifying details removed.

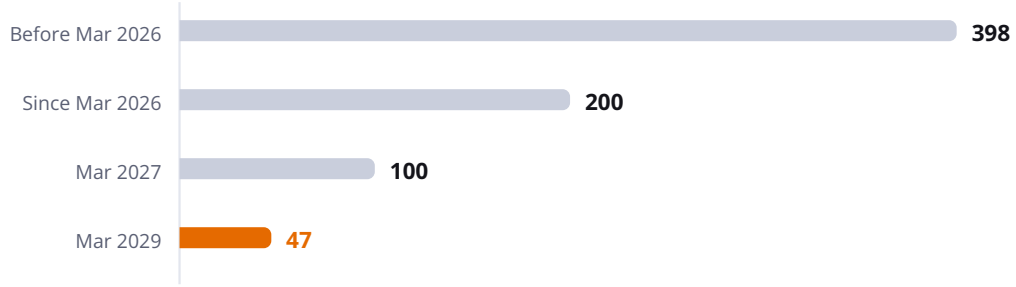
The migration was the responsible move. A small-to-mid-size SaaS company moved every TLS certificate from manual yearly renewals to automated issuance with Let's Encrypt, ahead of the industry's shrinking renewal windows. Forty-five days later, a certificate expired anyway. Five days after that, a second one did. The automation was not the problem. Everything around it was.

01

CA/Browser Forum ballot SC-081v3 phases maximum certificate lifetimes down to 47 days by March 2029.

The Migration

The company had been renewing certificates the way most teams still do: by hand, once a year, from a list. What pushed it to change was the calendar ahead, not the one behind. With maximum certificate lifetimes already down to 200 days and heading for 47, yearly habits were about to become a treadmill nobody could walk manually.



Maximum TLS certificate validity, days · CA/Browser Forum SC-081v3

So the team did the right thing. Every certificate moved to Let's Encrypt, renewed automatically by certbot on a short cycle. On paper, the problem was solved: renewals now ran on a timer instead of a calendar.

02

Both failures were environmental. Neither was a bug in certbot or Let's Encrypt.

Two Expiries in the First Fifty Days

day 45 **First Expiry**

A certificate expired in production. The renewal had been silently failing: a broken Python dependency in certbot's environment. Worse, the failure notifications ran through the same broken package, so the alarm died with the renewal.

day 50	**Second Expiry**	A different certificate expired for a different reason: an incorrect AWS EC2 instance role denied the permissions the renewal needed. A policy problem, invisible to the renewal tooling itself.
day 51	**The Backstop**	Generator Labs Certificate Monitoring went in the next day, not to replace certbot, but to independently confirm that every renewal certbot was supposed to make actually happened.

Automation Moves the Failure, It Does Not Remove It

Neither outage was a renewal-tool bug. A dependency broke; an IAM role was wrong. Automated renewal replaced the calendar, but it also failed silently, and in the first incident the failure notifications ran on the very dependency that had broken. When the alarm shares a failure mode with the thing it watches, it is not an alarm.

> *Certbot did not break. Its environment did, twice in fifty days, and nothing was watching the outcome.*

03

Monitoring checks the certificate a real client receives, independent of the renewal pipeline.

Verify the Outcome, Not the Process

The monitoring sits outside the renewal pipeline entirely and checks the only thing that matters: the certificate a connecting client actually receives from each endpoint. If a renewal fails for any reason, a dependency, a permission, a deploy that never shipped the new cert, the countdown to expiry becomes an alert instead of an outage.

Early Expiry Warnings	Escalating alerts ahead of every expiry, long before a failed renewal becomes an incident.
Chain and SAN Validation	Every check validates the full chain and hostnames, not just the expiry date on the leaf.
Internal Endpoints Included	Service-to-service APIs and internal hosts get the same watch as the public edge.

Alerts Where the Team Works	Warnings land in team chat via webhook; critical alerts page the on-call engineer.

04

Results as reported by the customer.

Since Day 51

2	2	0
EXPIRIES IN 50 DAYS	ISSUES CAUGHT SINCE	OUTAGES SINCE DAY 51

Since the backstop went in, monitoring has identified two additional misconfiguration issues, and both were fixed before they became outages. Certbot still does every renewal; monitoring confirms each one worked. The division of labor is the point: automation handles the process, and an independent check verifies the result.

Why the Backstop Earns Its Keep

The failure modes that took the platform down twice, a broken dependency and a wrong IAM role, are exactly the kind that renewal tooling cannot see from the inside. An outside observer checking the served certificate catches all of them, regardless of cause.

KEY TAKEAWAYS

- Automating renewals is the right response to shrinking certificate lifetimes. It is not the last step.
- Renewal automation fails through its environment, dependencies, permissions, and roles, and it fails silently.
- Independent monitoring verifies the outcome a client sees, not the process that was supposed to produce it.
- As a backstop, monitoring caught two further misconfigurations before they became outages.

This case study is based on a real Generator Labs customer deployment and is published with identifying details removed. The CA/Browser Forum certificate-lifetime schedule is documented at cabforum.org (Ballot SC-081v3, April 2025).

Take Expiry Off Your Incident List

Generator Labs Certificate Monitoring independently verifies every renewal your automation makes, and alerts you with days to spare instead of minutes. Start free.

portal.generatorlabs.com/signup