BAILEY FITCHETT
Senior DevOps / Platform Engineer

Verify the build. Rehearse recovery.

INDEPENDENT CASE STUDY / DELIVERY AND RELIABILITY

2 controls	7 cases	12 requests
signed package accepted	invalid cases rejected	healthy after recovery

The problem and my contribution

A release needs a verifiable origin and an inspectable recovery procedure. I built a Python package-verification and local-promotion reference, then a separate HTTP recovery lab with retained request samples.

Key decisions and a real failure

Verify origin before state changes. Use GitHub attestation verification with an explicit repository, workflow, issuer, runner and commit policy. A checksum alone does not establish trusted origin.

Test the real dependency. The first release passed mutually exclusive GitHub CLI flags; mocks missed the defect. I removed the redundant selector, retained the exact identity policy, and added real-verifier coverage with bounded diagnostics.

Recorded results

Signed delivery, v1.0.1: 29 tests passed. The real GitHub CLI accepted two trusted controls and rejected seven constructed invalid cases. The three negative cases that attempted promotion left local state unchanged.

Separate unsigned recovery lab: an injected HTTP 503 fault produced four failed requests. The harness runbook restored the previous package; twelve retained recovery requests were healthy.

Scope and limits

These are independent local demonstrations. The unsigned loopback lab uses checksum-only helpers and does not exercise the attestation gate. Its synthetic traffic does not establish a production SLO or deployment result. Release JSON records observations transcribed from the job log; it is not a signature bundle.

Read the full story and sources

Full case study: baileyfitchett.com/secure-delivery/

Signed-release observations (GitHub, source pinned in link)

Recovery record (portfolio JSON)