Engineer Platform

Security

40 achievements

Connect the parts of your system

The work collected under “Security” shows one part of a working system. We can help connect that part to the rest of your operation. Somebody responsible for the thing the whole company rests on: it is up, it is backed up, and it is not quietly rotting.

Explore our services Get in touch →

Find a comparable challenge

Follow an example that resembles your situation. Its review describes the decisions behind the result, beyond the technology label. Ran 10 PostgreSQL and MS SQL servers on Ubuntu — 10 database servers deployed and kept healthy

Ran 10 PostgreSQL and MS SQL servers on Ubuntu Get in touch →

Ran 10 PostgreSQL and MS SQL servers on Ubuntu

Designed, deployed, and maintained 10 PostgreSQL and MS SQL servers on Ubuntu Linux VPS, ensuring optimal server performance and reliability.

Wrote 1,000 RBAC rules to secure Linux servers

Enhanced data security by implementing 1,000 RBAC rules for developers, application instances, PostgreSQL, MS SQL, and other Linux servers, preventing unauthorized access; documented with Ansible automation.

Managed 30 Ubuntu Linux VPS instances

Managed 30 Ubuntu Linux VPS instances, implementing disaster recovery strategies and ensuring optimal network configurations.

Led access management across M365 and SSO

Prevented security breaches by leading access management initiatives, utilizing M365, 1Password, Red Hat SSO, and OKTA SSO.

Managed 8 WireGuard and IPsec VPN connections

Managed and troubleshot 8 WireGuard VPN and IPsec VPN connections, ensuring secure communication across Google Cloud and Linux systems.

Improved team communication and collaboration

Improved team communication and collaboration by implementing Slack, Mattermost, 1Password, and Jira, saving 8,000 hours of labor.

Administered network infrastructure

Administered network infrastructure for over 1,000 servers, ensuring optimal system deployment, security, and troubleshooting.

Automated SSL/TLS certs for 100 Docker apps

Automated SSL/TLS certificate creation for 100 Docker applications, ensuring secure connections across Ubuntu Linux hosts.

Configured and deployed 1,000 Wi-Fi routers

Configured and deployed 1,000 Wi-Fi routers, improving network accessibility and performance for clients.

Built organization context switching for users

Designed an organization context-switching system with client localStorage and server-side cookie mirroring, letting users act as managed organizations while enforcing least-privilege authorization.

Built email with failover and delivery logging

Built email as a platform capability — three providers with failover, delivery webhooks, send and delivery logging, templating and campaigns — behind a startup check that will not boot without one.

Provisioned Azure infrastructure as code

Provisioned Azure infrastructure as code with Bicep — Container Apps, PostgreSQL Flexible Server, Front Door/WAF and networking — across the development, staging and production environments.

Built GitHub Actions CI/CD pipelines

Built GitHub Actions CI/CD pipelines with a distroless production frontend image and multi-environment promotion.

Owned end-to-end platform deployments to Azure

Owned end-to-end deployments of the platform to Azure, managing releases across development, staging and production environments.

Hardened the application with nonce-based CSP

Hardened the application with nonce-based CSP, HSTS, SameSite cookies, least-privilege database roles and server-side entitlement re-checks.

Built the entire core infrastructure from scratch

As one of the first hires, designed and built the entire core infrastructure and supporting processes from scratch for a green-energy SaaS startup, laying the foundation for rapid growth.

Automated team collaboration

Automated team collaboration, password management, task and time management, and built a semi-automatic project-showcase system, raising team productivity.

Integrated a company-wide password manager

Integrated a company-wide password-management system, strengthening security and streamlining access control.

Built Stripe and in-app purchase entitlements

Built the payments and entitlements layer — Stripe alongside Apple and Google in-app purchase — gating the directory, search and export through an 11-table access model checked on the server.

Built first-party error monitoring and tracing

Built first-party error monitoring and OpenTelemetry tracing rather than buying them — payload sanitising, spike and regression detection, symbolication and a synthetic heartbeat — behind 11 operator views.

Built fail-closed rate limiting and anti-abuse

Built fail-closed abuse controls — 22 Redis-backed rate limiters, Cloudflare Turnstile, request idempotency and an origin lock — so the platform sheds bots and floods instead of trusting its callers.

Built company ownership claims and adjudication

Built company ownership claims end to end — a user claims a company, an administrator adjudicates, and an approval rewrites the authorization graph that decides who is allowed to edit what.

Ran a continuous security scanning programme

Established a continuous security programme — code scanning, DAST, dependency and vulnerability checks, SBOM generation, secret scanning and SHA-pinned actions — alongside 21 written security audits.

Three locks on the door, none of them locked

Found and closed three SSH brute-force protections that had never worked: a ban jail watching port 22 while the daemon listened on 1986, a rate limit shadowed by a broader rule above it, and a ban action whose binary never resolved, so no ban had ever applied.

Hardened SSH to 24 validated directives

Hardened SSH to 24 asserted directives with three-stage validation — the candidate file, the assembled config, then the daemon's own read-back — after the read-back caught the running server silently overriding two of the twenty-four.

Proved the intrusion ban works every run

Proved the intrusion-banning path end to end on every hardening run by banning a reserved test address, reading the resulting kernel rule and unbanning in a guaranteed cleanup block, so a jail that stops working fails a run instead of reporting healthy.

Verified firewall rules by position

Verified firewall rules by position rather than presence, reading the numbered rule list and the live packet-filter chain, because a rule that exists is not a rule any packet reaches.

Backup automation with an explicit restore method

Implemented restic backup automation with retention, integrity checks and a monthly restore-test workflow; documented storage activation and live-restore verification as separate operational steps.

Added a one-second preflight for secrets

Added a preflight play that runs the same code as the converge against operator-local secrets in about a second, after a half-applied production run died on its ninth task with the swap settings already written to the live host.

Reconciled a 20-record DNS zone as code

Reconciled a 20-record DNS zone declaratively against the Cloudflare API with separate audit and BIND-export entry points, and turned the CDN proxy back off on privacy grounds after building it.

Cut systemd sandbox exposure on every unit

Cut systemd sandbox exposure across every unit this platform installs — a dead-man's-switch service from 9.6 UNSAFE to 1.5, an internet-facing git forge from 8.3 EXPOSED to 1.5 — and added a converge-time parser check after finding a misspelled directive silently ignored in three unit templates.

Ran containers on Podman and Quadlet

Deployed a container plane on Podman and Quadlet under systemd rather than Docker, because Docker's default bridge-port publishing can bypass UFW rules — and gave deploys an unprivileged user with one fixed command instead of root.

Provisioned a second server from code

Automated the provisioning of a second server on a second cloud provider, creating the firewall before the machine so it is born behind one, with both providers' firewalls written directly against their REST APIs to avoid a third-party collection.

Wrote a scope rule after a credential leak

Wrote a scope rule into the repository after a restructure carried another company's inventory, firewall allowances and prose into it — and kept the quarantined residue under the secret scanner rather than excluding it.

Split every operational secret per host

Split the four per-host operational secrets after establishing that two hosts sharing one dead-man's switch alert less than two switches, not more, and that a shared backup passphrase can expose both hosts' backups, even in separate repositories, if someone obtains it and can read both repositories' encrypted data and key files.

Wrote a parser verifying every FFI call

Wrote a parser that reads the real 7,308-line C header and verifies every call site, every enum constant and that every pointer-owning class is final, after a hand-written placeholder header let calls to three removed functions compile, link and crash.

Reached the unused half of the core

Reached the half of the messaging core the application had never used — backup transfer, disappearing messages, message editing and resending, verified invitations, proxies and encryption policy — driving every test against the real library with no mocks.

Know the licence of everything you ship

Audited 644 Rust crates for licence compatibility on every build, and proved the check fires by rewriting one crate's licence and by moving the pinned core revision without regenerating.

Measured 13 addresses with no tracking

Measured thirteen addresses from the web server's own access log alone — no cookie, no tracker and no third-party service — masking the client network as each line is written, and folding the result into a permanent archive of 21 daily counts and 20 monthly facets.

Published a readable Tor onion mirror

Published the site as a Tor onion mirror on a readable address beginning engineer, with the key generated off the host so it never reached this repository or a laptop, and left it the one front door of four that is never counted.

Focus the engineering effort

Bring a representative dataset, the questions it must answer and the current bottleneck. We can agree how to measure quality, throughput and recovery.

See what it costs Get in touch →

Put the subject in your context

Tell us how “Security” appears in your own system. We can help choose the relevant service and define the work around it.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://platform.engineer.company/categories/security/