David Londoño

+57 317 2939347 | dlondon.dev@gmail.com | linkedin.com/in/davidldv | github.com/davidldv | davidlondon.dev

Software Engineer, Application Security. I ship production systems in TypeScript, Next.js and PostgreSQL, and I write the tools that attack them. **authzscan** is an IDOR/BOLA scanner for Next.js App Router repos, scored against a benchmark of **16** seeded bugs with its false-positive rate published. Two more labs cover JWT internals and LLM/MCP attack surfaces. English C1, Spanish native, German B1. Remote (LATAM/EU/US) or open to relocation.

Security Engineering Projects

authzscan | *Source Code* | *Overview* **TypeScript | ts-morph | SARIF**

Autonomous IDOR/BOLA review for Next.js App Router codebases, driven by Claude agents.

- Four phases: a deterministic **ts-morph** inventory of route handlers, Server Actions and auth-library usage; a trace agent that follows each client-controlled identifier to the query it reaches; and a verify agent that knocks down what trace found. Reports come out as **SARIF**, Markdown or JSON, with exit codes CI can gate on.
- The accuracy is a number, not a feeling: **16** seeded IDOR/BOLA bugs plus **6** correctly authorized twins that exist to catch false positives, gated on recall and precision. An oracle runner checks the harness scores right no matter how good the model is.
- It fails loudly. An endpoint it could not analyze is reported as *not analyzed*, never as clean.

JWT Security Lab & **jwt-scan** | *Source Code* **TypeScript | Node.js | Docker**

Offense-and-defense JWT lab, shipped as an npm CLI scanner.

- Wrote JWT signing and verification from scratch in TypeScript, no libraries, to reproduce five flaws that reach production: **alg=none** bypass, **HS256/RS256 key confusion**, weak-secret offline brute force, **kid** header path traversal, and missing **iss/aud/exp** validation. Every payload runs against both APIs and is checked each way.
- Shipped the checks as **jwt-scan**, an npm CLI with token and live-endpoint modes and CI-friendly exit codes. The hardened mirror removes whole classes of bug at once: one allowed algorithm (RS256), a fixed **kid** registry with rotation, generic errors so nothing leaks through an oracle, scrypt for passwords.

LLM/RAG Security Lab | *Source Code* **Python | FastAPI | MCP | Docker**

Five OWASP LLM Top 10 attacks as a pytest suite against a vulnerable/hardened FastAPI pair.

- Reproduced and fixed five LLM-specific classes on one RAG surface: **cross-tenant retrieval** (confused deputy), **indirect prompt injection** through retrieved documents, vector-store **poisoning** via forged ingest metadata, **excessive agency** over a live **MCP** tool server, and stored **XSS** straight out of the model.
- Stopped excessive agency with a deny-by-default authorization hook on the host side, so every MCP tool call is authorized before dispatch. A deterministic mock LLM turns "every attack fails against the secure API" into something CI can rely on.

Materiales La Bodega | *Live Site* **Next.js | PostgreSQL | Mercado Pago**

Solo-built e-commerce platform, live in production for a hardware retailer.

- The only engineer on a live storefront that moves about **$1.5M COP/day**: requirements, architecture, the real-time inventory catalog, deployment, and the maintenance it still gets.
- Built the authentication, session security and **RBAC** that keep the staff side separate from the customer side. Hardened against the **OWASP Top 10** with parameterized queries, server-side validation, CSRF protection on state-changing routes, and least-privilege database roles.

Work Experience

Tambora **Jul 2025 – Sep 2025**

Frontend Developer (Contract) *Remote*

- Migrated business-critical legacy modules from jQuery to **React** and cut bundle size by **40%** on the rebuilt surfaces, dropping the ad-hoc DOM string injection along with it.
- Consolidated the rebuilt UI into a modular **Atomic Design** component library, so input handling and escaping sit in one reviewable layer instead of scattered per page.
- Replaced manual SSH deploys with **Azure CI/CD** pipelines and took deployment time from over **two hours to under fifteen minutes**. The automated test gates are where SAST and dependency scanning hook in.

Certifications & Security Training

Completed: Google Professional Cybersecurity Certificate, PortSwigger Web Security Academy (all free labs).

In progress: Burp Suite Certified Practitioner (BSCP), CompTIA Security+. **Practice**: HackTheBox (@davidldv), TryHackMe (dlondon.dev), CTF challenges.

Technical Skills

Security: OWASP Top 10 (Web + LLM), IDOR/BOLA & broken access control, vulnerability research, penetration testing, OAuth 2.0 / OIDC, JWT cryptography, RBAC, CSRF, XSS and SQL injection (SQLi) defense, secure session handling, secrets management, threat modeling (STRIDE), secure code review, prompt-injection defense.

Tooling: Burp Suite, Semgrep, Trivy, nmap, Wireshark, SARIF / GitHub code scanning, Docker, Linux, Git.

Languages: TypeScript, JavaScript, Python, C, SQL, Bash.

Frameworks & Data: Next.js, React, Node.js, FastAPI, Prisma, PostgreSQL, MongoDB, REST, WebSockets, Jest, Zod.

Cloud & DevOps: AWS, Azure, CI/CD (Azure Pipelines, GitHub Actions), containerization.

Education

Universidad Tecnológica de Pereira, Pereira, Colombia. Systems & Computing Engineering, in progress (Feb 2022 to present).

EliteStack Bootcamp, Pereira, Colombia. Full-Stack Development (Jun 2024 to Jul 2024).