No CS Degree.

No Problem.

How a Construction Worker Built 5 AI Products
From a Live Server Under Attack

Slawomir Luzny

Founder, FixFlex LTD • West London

sentinel-ai.info • fixflex.co.uk • 24ad.info

Published by FixFlex LTD, West London, United Kingdom.

First edition, May 2026.

Contents

Chapter 1

The Idea

West London, 2021.

I was standing on a construction site, holding a cup of tea that had gone cold an hour ago, watching a crew argue about where a wall should go.

That was my life. Schedules, suppliers, problems that only exist at 7am when half the crew hasn't shown up and the client is already calling.

But my head was somewhere else entirely.

I'd been thinking about classifieds platforms for months. Not because I had some grand vision or a business plan with projections. It was simpler than that — I could see a gap. A way to connect buyers and sellers that felt easier, smarter, more global than what existed.

The idea wouldn't leave me alone.

The problem was obvious: I had zero programming knowledge. Zero. I didn't know what a database was. I'd never written a line of code. My entire technical experience consisted of being reasonably good at not breaking my own phone.

But I had something else.

I had stubbornness.

The First Mistake (That Everyone Makes)

The logical move, I thought, was to buy my way in.

I found a ready-made classifieds script online. It looked professional. It had features. It cost money — but not that much money, and surely this was the smart shortcut, right?

I bought it.

Then I hired a freelancer to customise it.

Then another one, when the first one disappeared mid-project.

Then another, when the second one delivered something that half-worked on a good day and crashed on a bad one.

Each 'expert' arrived with confidence and left with my money. None of them seemed to understand what I actually needed. Or maybe they did understand — and just didn't care enough to get it right.

Bots started attacking the server within weeks of going live. Databases degraded. Cloudflare helped but didn't catch everything. I'd wake up to emails about downtime and spend evenings trying to understand what had gone wrong, armed with error messages I couldn't read and logs that meant nothing to me.

After a year of this, I did the maths.

The money I'd spent on freelancers, on scripts, on fixes for the fixes — it added up to a number that made me feel sick. And the product still wasn't right.

The Decision

There was a moment — I remember it clearly, standing in the kitchen at about 11pm, laptop open, staring at another error message — when I made a decision.

Either I figure this out myself, or this project never launches.

Not 'I'll try harder with better freelancers.' Not 'I'll find a technical co-founder.' Just: I'm going to understand what's happening on my own server.

I had no idea what that would actually involve.

I had no idea that decision would eventually lead to a registered UK company, five live software products, and a server that has blocked over 283,000 cyberattacks.

But that night in the kitchen, I just knew one thing: the idea was worth fighting for.

Chapter 2

The Expensive Lesson

There's a specific kind of frustration that comes from paying someone to solve a problem — and ending up with two problems instead of one.

The Freelancer Economy (From the Inside)

Here's what nobody tells you about hiring freelancers when you don't know what you're doing: you can't evaluate the work.

That's the trap. If you don't understand code, you can't tell the difference between a freelancer who wrote clean, maintainable logic and one who duct-taped something together that works today and collapses in three months.

They both show you the same thing: a screen where the button does what the button is supposed to do.

So you pay. You say thank you. You move on.

And then, three months later, the button stops working — and the freelancer is gone.

I went through this cycle more times than I want to admit. Each time, I told myself this one would be different. Some of them were genuinely trying. But the results were the results. Half-working features. Inconsistent behaviour. A database nobody had bothered to index properly. Security that was basically ceremonial.

The Bots Don't Care About Your Budget

The bot attacks started almost immediately after the site went live.

I didn't understand what was happening at first. I just knew the server was slow. Pages were timing out. The error logs were filling up with something.

A freelancer explained it eventually: automated scripts, scanning the internet for vulnerable endpoints, trying combinations of usernames and passwords, submitting forms thousands of times per hour.

'It's normal,' he said. 'It happens to everyone.'

What he didn't tell me was that 'it happens to everyone' and 'your server is configured to handle it' are two very different statements.

Mine wasn't configured to handle it.

The bots found the contact form. Then the registration page. Then the login endpoint. Each one became a small fire. We patched one, another started.

The Calculation

Late one night, I did a calculation. Not a financial one. A different kind.

How much longer could I keep going like this?

The construction work was steady. But the nights were being consumed by a project that kept bleeding money and progress without producing either.

I had to change the approach. Not the goal — the goal was still right. But the method wasn't working.

I needed to understand what I was building.

Which meant I needed to start learning.

Chapter 3

Learning by Fire

The first thing I learned about Linux is that it will tell you exactly what's wrong. The second thing I learned is that understanding what it's telling you takes considerably longer.

The Terminal

My first real SSH session into my own server was approximately fifteen minutes of typing things incorrectly, being told I didn't have permission to do them, and then being told I also didn't have permission to give myself permission.

I didn't quit. Not because I was brave. Because I was angry. The server was mine. I was paying for it. And I was going to understand what was on it even if it took all night.

It took considerably more than one night.

Learning Without a Teacher

There is no curriculum when you're learning this way. With a course, someone has decided what you need to know and in what order. Without that, you learn whatever the problem in front of you requires.

I learned SSH because I needed to get into the server. I learned basic Linux commands because I needed to find files and read logs. I learned about Fail2Ban because someone mentioned it in a forum post about bot attacks.

I learned about web servers by breaking them. Nginx first. Then Apache. Then LiteSpeed. Then Varnish. Then H2O, briefly.

Eventually I found Caddy. Caddy is what happens when someone designs a web server to be used by people who are not web server experts. Its configuration file is written in something approaching plain English. I stayed with Caddy. Still use it today.

The Git Problem

I should mention Git. Everyone uses Git. It's how developers track changes to code, collaborate, roll back mistakes.

I didn't use Git. Not for the first two years.

Instead, I took snapshots. Before making any significant change to the server — before editing a config file, before trying anything I wasn't sure about — I'd take a full snapshot through the hosting panel.

It was slow. It was inelegant. But it worked. If something went catastrophically wrong — and things went catastrophically wrong with some regularity — I could roll back.

It's not how professionals do it. It's how I survived long enough to eventually learn how professionals do it.

The Methodology

I developed a process, though I wouldn't have called it that at the time.

Something breaks, or needs to be changed. I describe the problem — in plain language, as precisely as I can — into an AI chat window. The AI gives me something. Code, a command, an explanation.

I read it. I try to understand what it's supposed to do before I run it. Then I run it on the live server.

This is not best practice. Every developer reading this is wincing. But I tested on production, with snapshots as my safety net. And I learned faster than I would have otherwise, because every mistake had real consequences.

There's no better teacher than a mistake that actually matters.

Chapter 4

The Attack

I don't remember the exact date. I remember the feeling.

3am

The alert came through at just after three in the morning.

I opened the laptop. The server was unresponsive. SSH was timing out. The hosting panel showed CPU usage at something I'd never seen before — not a spike, not a brief surge, but sustained, brutal load that had been building for hours while I was asleep.

What I found was a Laravel application being hammered by a coordinated bot campaign. Not the low-level scanning I'd become used to — this was targeted, sustained, and it had found something.

The contact form.

My contact form had no rate limiting. No CAPTCHA. No validation beyond the basics. To an automated script, it was an open door.

Nearly a million emails, it turned out, before I managed to shut it down.

The Damage

The server was blacklisted. Not by one service — by multiple. Email providers, spam databases, reputation services.

I spent the next week in a kind of focused misery. Days on the construction site and then evenings trying to repair what had happened. Contacting blacklist operators. Waiting for reviews. Implementing fixes that should have been there from the beginning.

Rate limiting on every form. CAPTCHA where it mattered. SPF, DKIM, DMARC records properly configured. Monitoring on outbound email volume.

Each fix felt like closing a door that should never have been open.

Building the Answer

While doing this repair work, I started making a list. It grew naturally from the process of fixing things.

By the end of the week, I had a comprehensive picture of everything that could go wrong on a server like mine, and what would need to be true to catch it.

You'd need to monitor the contact form, the login endpoint, the registration page, the email queue, the database connections, the CPU load, the disk usage, the SSL certificate expiry, the running services.

Nothing I found did all of this. So I started building it.

The Number

The first version of what would eventually become Sentinel was not impressive. It was a Python script that ran on a cron job every few minutes and checked a list of things.

But I ran it. And it found things. A service that had quietly crashed seven times in the past hour. A database query taking twelve seconds. A disk usage trend heading toward full.

Small things. Not crises. But the kind of small things that become crises if nobody notices them.

I noticed them.

As I write this, the total attacks blocked counter reads: 283,103. That's every banned IP, every blocked request, every Fail2Ban trigger, accumulated over the time Sentinel has been running.

Two hundred and eighty-three thousand times, something tried to get in. Two hundred and eighty-three thousand times, it didn't.

Chapter 5

The Pivot

At some point, the frustration stopped feeling like an obstacle and started feeling like information.

What Changed

I had a server I understood. I had a monitoring tool that worked. I had a classifieds platform that was, after everything, actually running.

But I was still slow. Every feature took too long. The gap between idea and execution was still too wide.

Then I found Windsurf. Then Claude. And the process changed.

The difference wasn't the quality of the answers. The difference was the conversation.

With Claude, I could think out loud. I could describe a problem imprecisely and have the imprecision noticed and questioned. For someone still building mental models that experienced developers have automatically — that was transformative.

I stopped feeling like I was fighting my own ignorance. I started feeling like I had a collaborator who could compensate for it.

The Methodology, Refined

I learned to be more precise in my descriptions. Not technical precision — I still often couldn't give you the exact name for what I wanted. But functional precision: here is what exists, here is what I need to exist, here is how I'll know it's working.

I learned to verify before implementing. Not to run things blindly but to read them, understand what they were supposed to do, and only then apply them.

I learned to ask why, not just how.

And I learned to trust my own judgment about when something was wrong. The AI would sometimes suggest solutions that technically worked but felt off. I learned to push back.

That back and forth — human judgment combined with AI capability — was what actually produced good results.

Throwing Things Away

Around this time, I made a decision that felt dramatic at the time and obvious in retrospect: I abandoned Laravel.

I'd inherited a Laravel codebase I didn't fully understand, built by freelancers who'd made choices I couldn't always explain.

So I rewrote. Over several months, moving functionality from the old system to a new one built on React 19, TypeScript, tRPC, and Node.js.

The new version was faster. Cleaner. More mine. I understood every part because I'd built every part — or at least understood what the AI had built and why.

I rewrote breathtime.info in six hours. That would have been unthinkable eighteen months earlier.

The Productisation Moment

Sentinel had been running on my servers for a while. It had grown from the original script. It had a proper dashboard, fleet view, AI integration.

I'd built it for myself. But people kept asking about it.

I started thinking about what it would mean to make it a real product. With pricing. With documentation. With a support commitment.

Free Basic tier for one server — because the barrier to trying it should be as low as possible. Pro at $49 a month. Enterprise at $149 a month.

Real pricing. Real commitment. Real product.

Chapter 6

Where I Am Now

I set an alarm for 5:45am. I've been setting it for the same time for years.

The Morning

The routine hasn't changed much. Up before six. Construction site by seven. Managing schedules, dealing with suppliers, solving the kind of problems that exist only in the physical world.

By the time most tech founders are sitting down with their first coffee, I've already been working for hours.

That part of my life is stable. The construction company runs well. The work is real in a way that software sometimes isn't: at the end of the day, there's something that exists in the world that didn't exist before.

I find that grounding.

The Portfolio

Five products, currently live:

Sentinel — the server guardian. 283,000 attacks blocked. Still running.

24ad.info — the classifieds platform. The original idea, finally built properly. React 19, TypeScript, tRPC. AI-powered listing creation across 25 countries and 20+ languages.

SpamSlap — the spam filter. Born from the contact form attack. AI-powered detection for web forms.

PostPilot — automated social media content generation and scheduling.

ArticlePilot — the content platform. Generates long-form articles, manages multiple blogs, handles cross-posting and RSS. The machine that runs in the background

while I sleep.

None of them started as products. All of them started as tools I needed.

What I've Learned

The tools matter less than you think. Find what fits how you work.

Breaking things is the education. The things I remember most clearly are the things that went wrong.

AI is a collaborator, not a replacement. The code is better when I understand it. The solutions are better when I push back.

Shipping beats perfecting. Every product I have is imperfect. Every product I have is live.

The story is the product. Authenticity is the differentiator.

Consistency compounds. Three posts a day, consistently, matters more than a viral moment.

The Invitation

If you're reading this and you have an idea you haven't started because you don't have the technical background — start.

Not with a plan to learn everything first. Not with a commitment to do it properly before doing it at all. Just start. With whatever you have.

The knowledge follows the doing.

I didn't learn to manage a server and then build products. I built products and learned to manage a server because I had to.

The credential doesn't come before the work. It comes from it.

"I don't know programming languages at an expert level.
I don't need to.

I imagine, describe, verify — AI codes. I decide."

— Slawomir Luzny

Founder, FixFlex LTD • West London

sentinel-ai.info • fixflex.co.uk • 24ad.info