



Security Guide for Vibe Coders

10 rules for everyone who builds apps with AI tools, in plain language.

Version 1.0 · October 2026 · switech-solution.com

With tools like Cursor, Lovable, Bolt or v0 you can build an app today without knowing how to code. That is great. But the AI does not automatically write secure code: it does what you ask and often forgets about security along the way.

This guide explains the ten most important rules without jargon. For each rule you get a sentence you can give straight to your AI.

01 Secret keys never belong in your code

WHY

API keys (e.g. from OpenAI or Stripe) in the code or the frontend can be read by any visitor and used at your expense.

WHAT TO DO

- Put keys in a .env file and add that file to .gitignore.
- Careful: anything with NEXT_PUBLIC_ or VITE_ in its name is publicly visible.
- If a key has leaked, revoke it with the provider and create a new one immediately. Deleting the commit is not enough.

ASK YOUR AI

«Check my project for API keys or passwords in the code and in the frontend. Move them into environment variables that are only available on the server.»

02 Lock down your database

WHY

Many AI apps use Supabase or Firebase. Without rules, anyone with the public key can read and change all data.

WHAT TO DO

- Supabase: enable Row Level Security (RLS) on every table and define policies.
- Firebase: set Security Rules and never use "allow read, write: if true".
- Test it: use a second account and check whether you can see other people's data.

ASK YOUR AI

«Enable Row Level Security on all tables and write policies so that each user can only read and change their own data.»

03 Always check permissions on the server

WHY

A hidden button protects nothing. Anyone who sends the request directly bypasses the frontend.

WHAT TO DO

- For every action, check on the server: is the user logged in, and does this object belong to them?
- Only allow admin functions for admins, checked on the server.

ASK YOUR AI

«Review all API routes: is it checked on the server that the user is logged in and allowed to access the requested object?»

04 Do not build your own login

WHY

Self-built login systems are a common source of security holes.

WHAT TO DO

- Use proven providers such as Supabase Auth, Clerk, Auth0 or Firebase Auth.
- Enable two-factor authentication at least for your own admin account.

ASK YOUR AI

«Replace the custom login logic with [provider] and remove all self-built password functions.»

05 Trust no input

WHY

Forms and uploads can be manipulated. AI-generated code often only validates input in the browser.

WHAT TO DO

- Validate input on the server as well (e.g. with zod).
- Limit file type and size for uploads.

ASK YOUR AI

«Add server-side validation to all forms and API routes, including length and format checks.»

06 Protect expensive APIs from abuse

WHY

An open AI endpoint can run up a large bill overnight.

WHAT TO DO

- Route AI calls through your server only, never directly from the browser.
- Add rate limiting per user.
- Set a spending limit and budget alerts with the provider.

ASK YOUR AI

«Add rate limiting to all routes that call paid APIs, for example a maximum of 20 requests per user per hour.»

07 Be careful with packages

WHY

AI tools sometimes suggest packages that do not exist. Attackers register such names with malicious code.

WHAT TO DO

- Before installing, check: does the package exist on npmjs.com, is it maintained and widely used?
- Update packages regularly and run npm audit.

ASK YOUR AI

«List all installed packages. Flag unknown, outdated or unused ones.»

08 Secure the AI features in your app

WHY

Users may try to manipulate your AI with clever text (prompt injection).

WHAT TO DO

- Never give the AI more rights than the user has.
- Do not put secrets in the system prompt.
- Require confirmation for critical AI actions (delete, send, pay).

ASK YOUR AI

«Review the AI features for prompt injection. Which actions can the AI trigger, and are they limited to the user's permissions?»

09 Backups and separate environments

WHY

One wrong command from the AI can delete data.

WHAT TO DO

- Enable automatic database backups and test a restore once.
- Try changes in a test environment first, not directly in the live app.

ASK YOUR AI

«Set up separate development and production databases and explain how I restore a backup.»

10 Get a review before real customers arrive

WHY

The AI finds many problems, but not all of them.

WHAT TO DO

- Ask the AI for a security review regularly.
- Before launching with real customer data, have an experienced person take a look.

ASK YOUR AI

«Do a security review of my project based on the OWASP Top 10. List the issues by urgency and suggest concrete fixes.»

QUICK CHECK BEFORE LAUNCH

- ☐ No API keys in the code, the frontend or the Git history
- ☐ Database rules (RLS or Security Rules) active on all tables
- ☐ Tested with a second account: no other users' data visible
- ☐ Login via a proven provider, 2FA active on the admin account
- ☐ Input is validated on the server
- ☐ Rate limiting and spending limits for AI and payment APIs
- ☐ All packages known and up to date
- ☐ Backups active and restore tested
- ☐ Privacy policy and legal notice in place
- ☐ Security review done by the AI and by a human

This guide covers the basics and does not replace a professional security assessment, especially for apps with sensitive data or payments.

Questions or need support?

I help small businesses and founders build secure AI tools and automations. The intro call is free.

www.switech-solution.com