AI Coding Assistants Have a Blind Spot: Security

6 ways they create vulnerabilities in your code

The Story

A non-technical founder shared how he built his SaaS using Cursor.

Within days, attackers:

- ⚠️ Maxed out his API usage
- ⚠️ Bypassed subscriptions
- ⚠️ Messed with his database

His post went viral with 2.2M views.

1

Hardcoded Secrets in Code

This is the most common mistake.

```
const apiKey = "sk-live-abc123...";
```

Push this to GitHub and bots will find it **within minutes**.

One student got a **$55,000 bill** from a leaked API key.

The Fix

Always use environment variables.

❌ `const secret = 'supersecretkey';`

✅
```
const secret =
process.env.SECRET;
```

Environment variables keep secrets out of your codebase.

2

Committing .env Files to Git

AI often creates .env files but doesn't always add them to .gitignore.

Once it's in Git history, it's there **forever**.

(Unless you rewrite history)

- ✓ Add `.env*` to .gitignore immediately
- ✓ If committed, invalidate ALL credentials
- ✓ Back up .env files elsewhere

3

Fallback Secrets

This one is sneaky.

```
const jwtSecret = process.env.JWT_SECRET ||
'supersecretjwtkey';
```

Looks safe because it uses an env variable.

But if the env var isn't set, it falls back to a **hardcoded secret**.

A time bomb waiting to explode.

4

Insecure Random Numbers

AI generated this OTP function:

```
function generateOTP() {
  return Math.floor(100000 + Math.random() *
900000)
}
```

Math.random() is NOT cryptographically secure.

Attackers could potentially **predict future OTPs**.

The Fix

Use cryptographically secure randomness:

❌ `Math.random()`

✅ `crypto.randomInt(100000, 1000000)`

The AI knew this was wrong when I asked it to review. But it generated the insecure version first.

5

Unsanitized Input Enables Phishing

AI-generated forms rarely sanitize user input.

Attacker signs up with name:

```
John <a href="[malicious link]">Click for a special
offer!</a>
```

Your welcome email becomes:

"Hi John Click for a special offer!, thanks for signing up!"

Your signup form becomes a phishing delivery system.

6

AI Treats Rules as "Suggestions"

You can tell your AI:

> *"Never hardcode secrets. Always use environment variables."*

You can add it to `.cursor/rules` or `CLAUDE.md`.

But AI treats these as **suggestions**, not guarantees.

It will break its own rules when the context gets complex.

The Bottom Line

AI coding assistants optimize for **"working code"** not **"secure code."**

- ✓ Review AI-generated code for security issues
- ✓ Set up security rules (even if AI doesn't always follow them)
- ✓ Use environment variables for all secrets
- ✓ Scan your codebase for exposed credentials
- ✓ Understand that AI makes mistakes

Build fast, but build securely.

→

Want These Rules for Your AI?

I created a **security.md** file you can drop into your project to guide your AI coding assistant.

Comment **"Security"**
and connect with me
to get a copy of the rules.

Let's Connect

I share practical tips about AI, coding, and security.

💬 Follow me for more insights

🔔 Repost to help others

♻️ Save this for future reference

#AI #Security #VibeCoding