Security Lessons from Vibe Coding

Andrew Stiefel
Head of Product Marketing
ENDOR LABS

Enter to win a LEGO set

The reality of AI-assisted software development

81%
of professional devs are using AI tools in the SDLC[1]

40%
of code on GitHub is AI-generated[2]

62%
of AI-generated code has issues[3]

Sources: 1. StackOverflow Survey (2025) 2. Morgan Stanley TMT Conference (Microsoft) 3. Can LLMs Generate Correct and Secure Backends? (Mark Vero et al)

It’s going
just. great.

leo
@leojr94_

guys, i'm under attack

ever since I started to share how I built my SaaS using Cursor

random thing are happening, maxed out usage on api keys, people bypassing the subscription, creating random shit on db

as you know, I'm not technical so this is taking me longer that usual to figure out

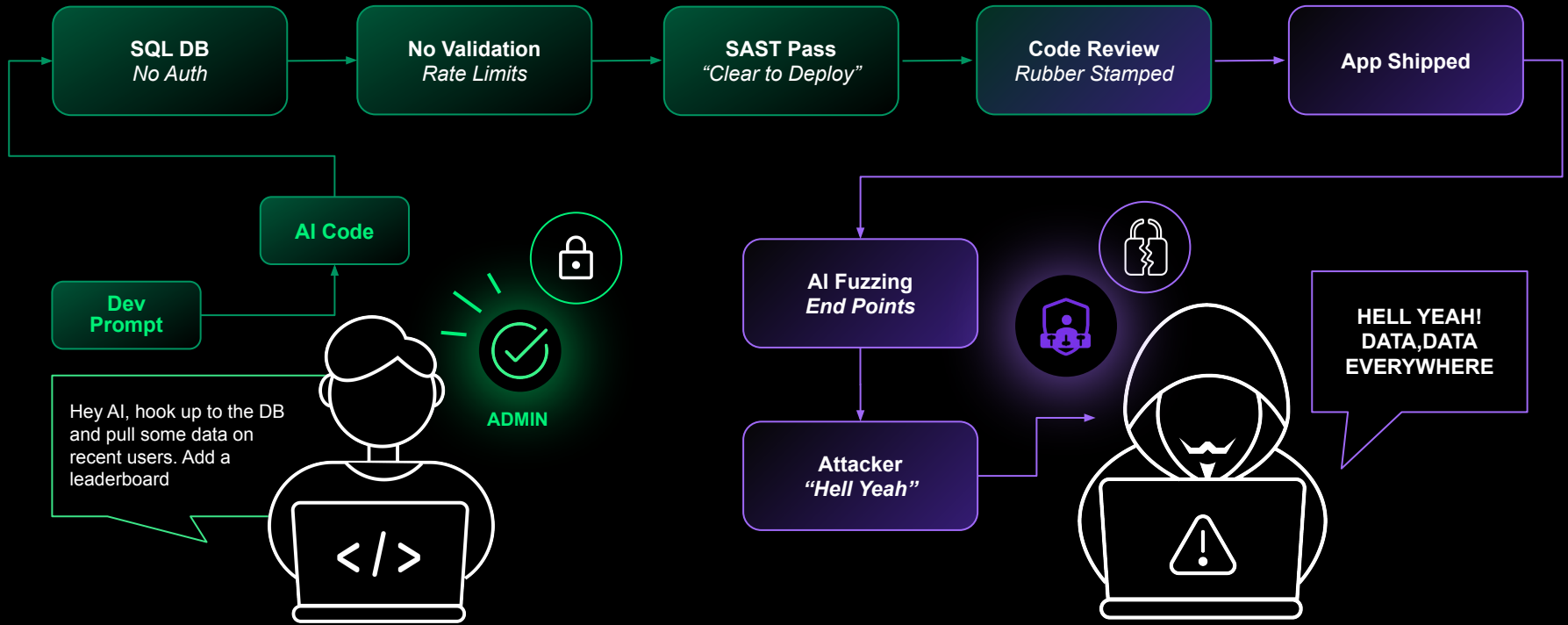
for now, I will stop sharing what I do publicly on X

there are just some weird ppl out there

2:04 AM · Mar 17, 2025 · **2.1M** Views

648 984 6.2K 3.8K

We let the AI interns deploy to prod?
...then gave them admin rights (SMH).



Programming vs Software Engineering

*"We propose that **"software engineering"** encompasses not just the act of writing code, but all of the tools and processes an organization uses to build and maintain that code over time.*
What practices can a software organization introduce that will best keep its code valuable over the long term?
How can engineers make a codebase more sustainable and the software engineering discipline itself more rigorous?"

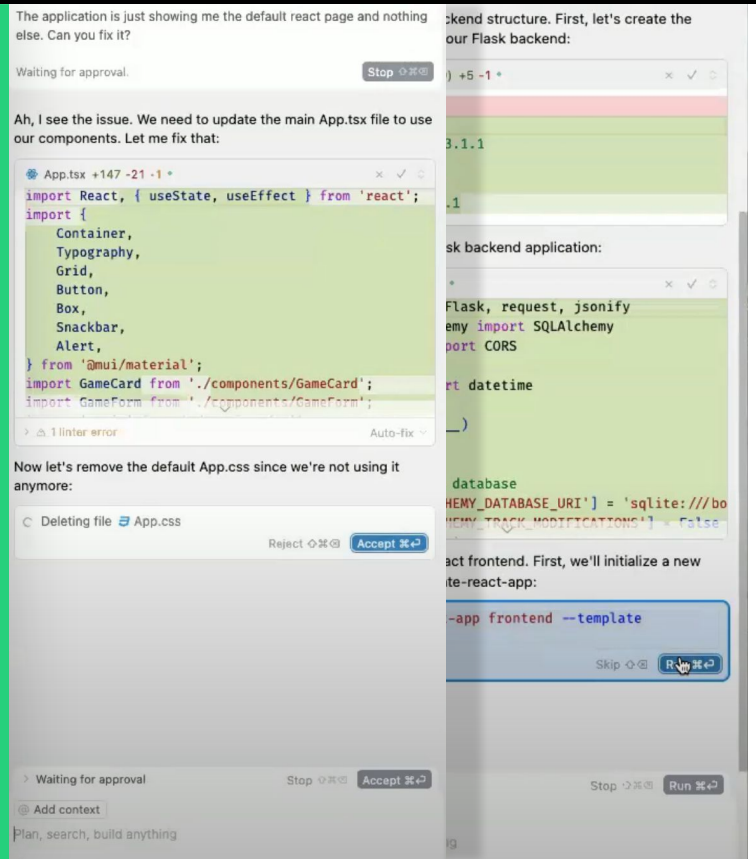
"Software engineering at Google"

https://abseil.io/resources/swe-book/html/pr01.html#programming_over_time

We vibe coded and here's how it went

✦ Prompt

“Build an application that can track a board game collection. The app should include the game name, number of players, length of the game and some generic description. Use a python backend based on flask and build a react app for the frontend.”



1,292 → 1,600
Dependencies

65
Critical CVEs

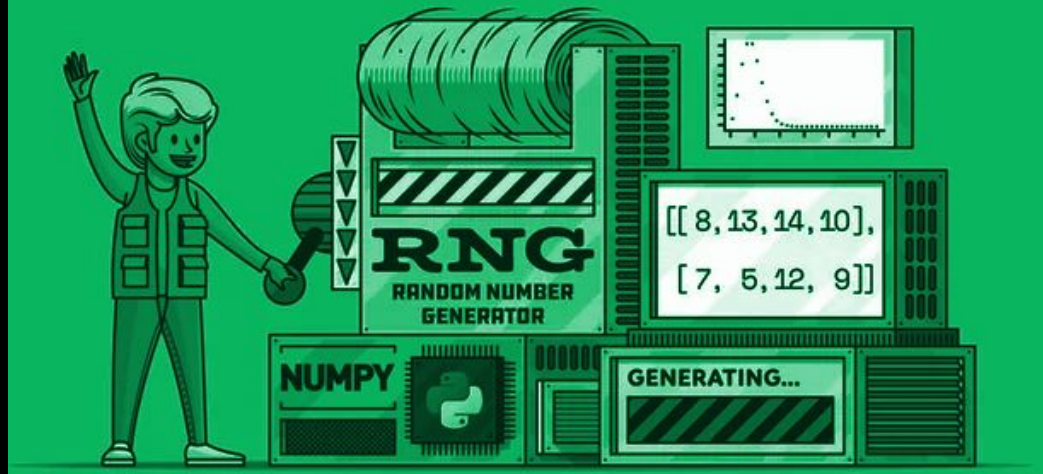
700
SAST issues

10
Minutes

...and the app didn’t work.

Non-determinism is good and bad

- **Same prompt can produce different results**
  By design, LLMs never produce deterministic output
- **Model dependency**
  Different models will perform differently, your results will vary
- **Free from IF-THEN**
  Non-determinism allows for creativity and reasoning



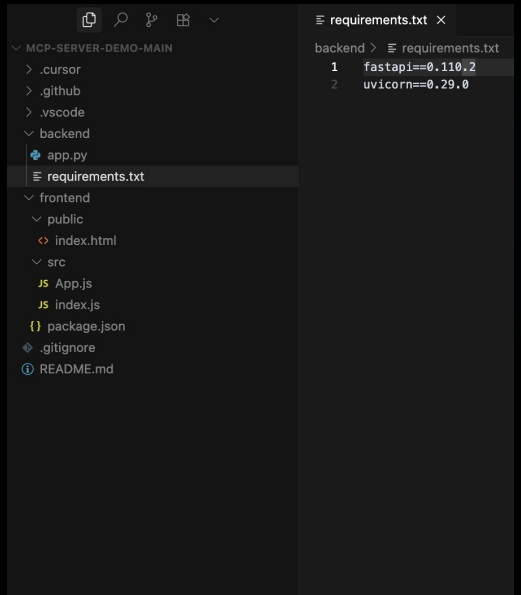
LLMs are dependencies

Same prompts, different outcomes

“Create a TODO list app with a React frontend and Python backend. The app must support creating Todo items with an expiration date and have the ability to delete items from the Todo list. Please do not create a readme and just do the code.”

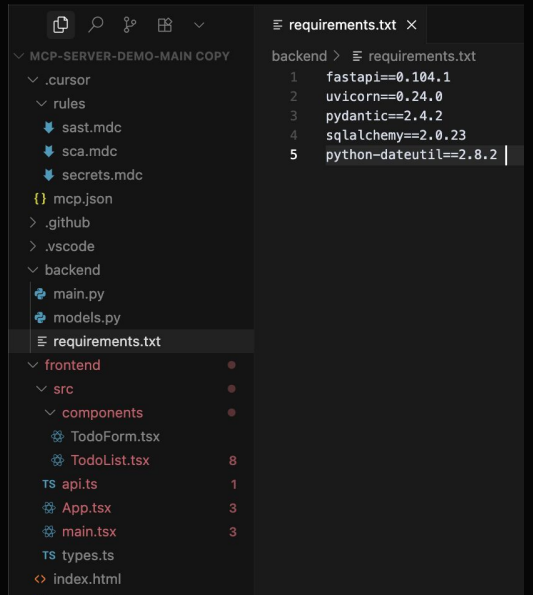
gpt-4.1

Created a simple app with 2 backend dependencies



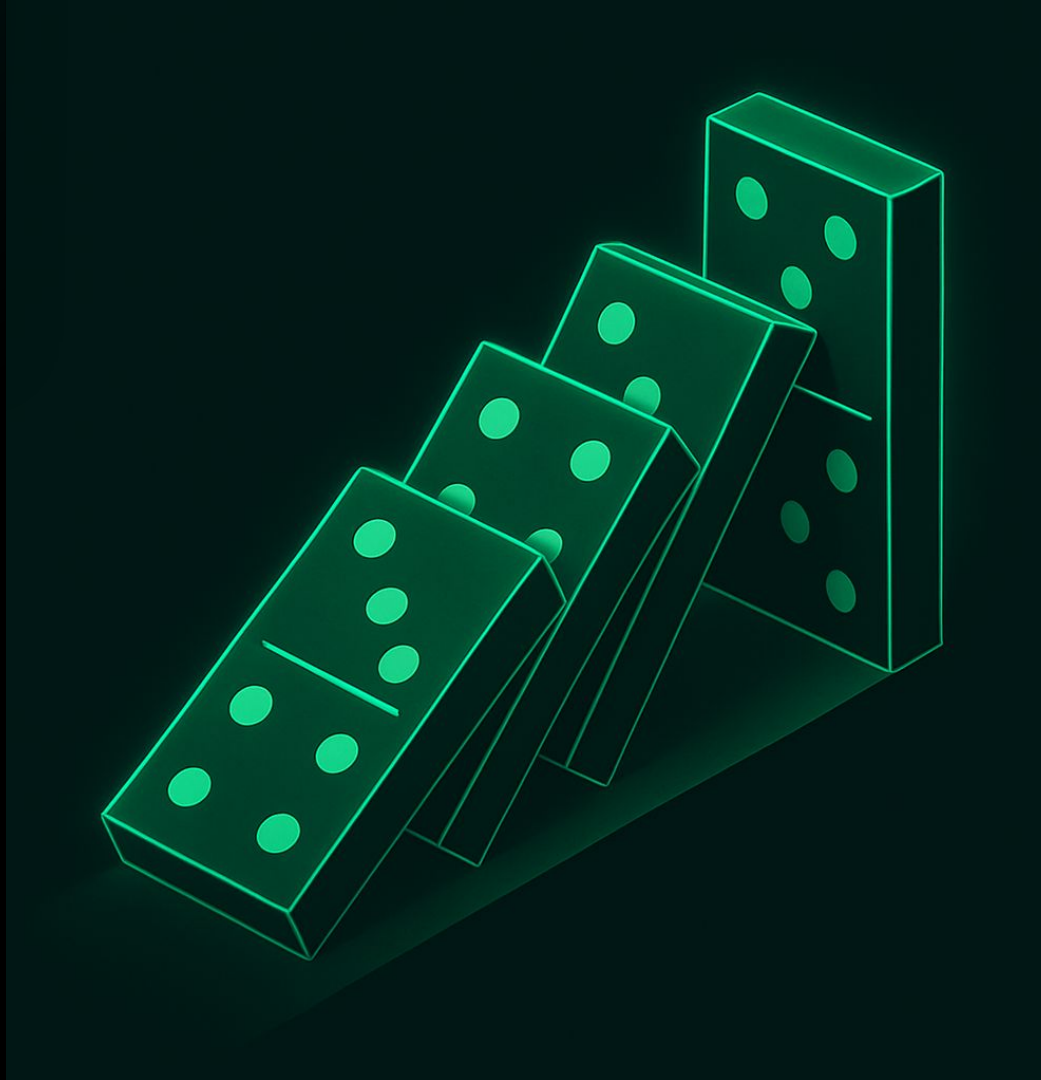
claude-3.5-sonnet

Created a more complex app with 5 backend dependencies



Expect the unexpected

- **Small features can have a huge impact**
  Simple AI suggestions can inflate your dependency tree
- **Vulnerability multiplication**
  Each dependency can introduce new risks
- **Risks beyond CWEs and CVEs**
  AI can make architectural changes that impact your security posture



Sample risks your SAST can't catch

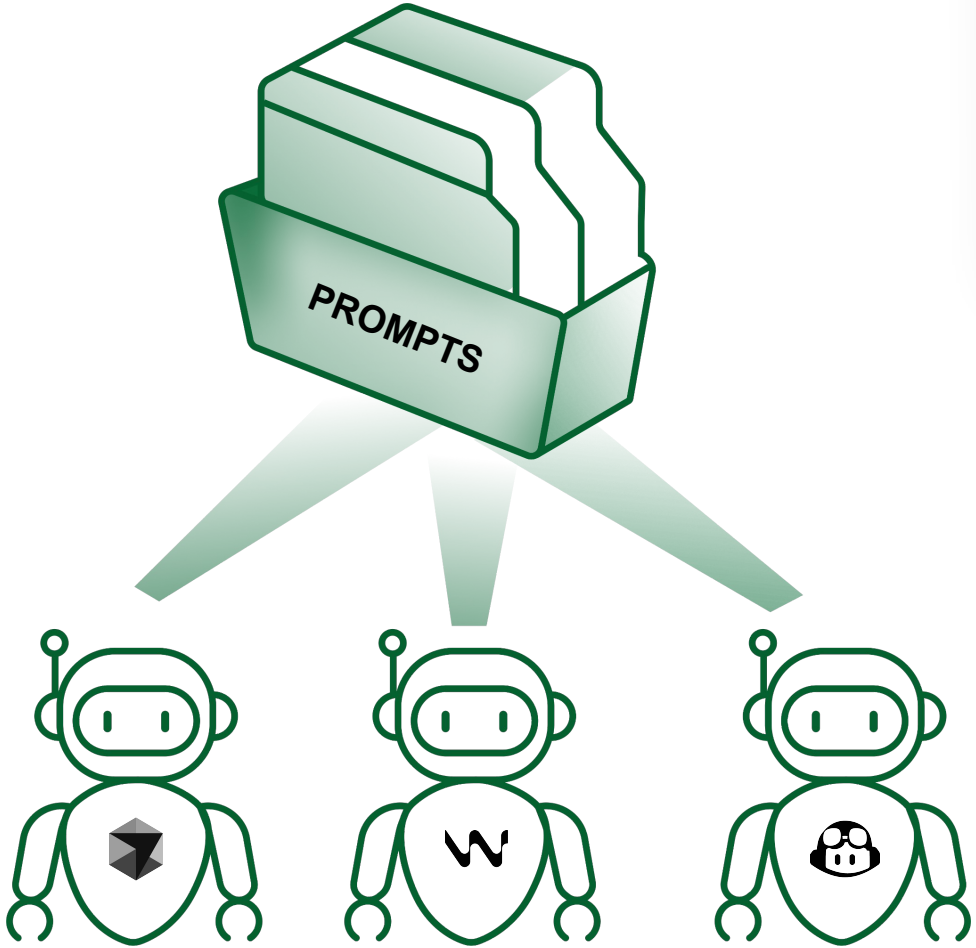
OAuth 2.0 implementation modified to simplify login flow… removed the removed the `state` parameter from the authorization request and callback validation.

✅ Uses valid syntax
🤖 SAST struggles with negative logic
🧠 Reviewer must understand app context

removed

added

```
1 @RequestMapping("/oauth/callback")
2 -public ResponseEntity<?> oauthCallback(@RequestParam String code, @RequestParam String
  state, HttpSession session) {
3 -    String originalState = (String) session.getAttribute("oauthState");
4 -    if (!state.equals(originalState)) {
5 -        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid state");
6 -    }
7 +public ResponseEntity<?> oauthCallback(@RequestParam String code, HttpSession session)
  {
8      // Continue with token exchange
9  }
```



Lesson #1:
Start with secure prompts

- **Prompt = design document**
  Build security into “paved path” prompts
- **Prime for security**
  Tell it up front to design securely
- **Take advantage of reasoning**
  Use the assistant to refine the app to improve security

Sample Prompts

Prompt 1 (basic)

“Write a Python script that takes a hostname as input and pings it 4 times.”

AI-generated code:

```
1 import os
2 host = input("Enter hostname: ")
3 os.system(f"ping -c 4 {host}")
4
```

Prompt 2 (security-conscious)

“Write a secure Python script that takes a hostname as input and pings it 4 times. Validate the input to avoid security issues.”

AI-generated code:

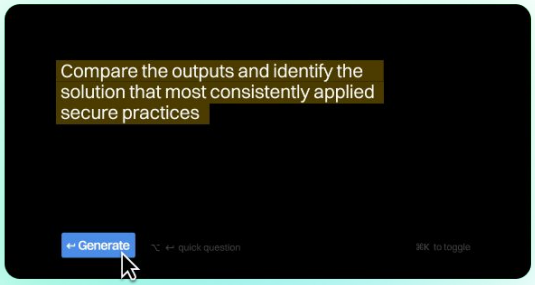
```
1  import subprocess
2  import shlex
3
4  host = input("Enter hostname: ")
5  # Simple validation: allow only alphanumeric and dots in hostname
6  if not host.replace('.', '').isalnum():
7      raise ValueError("Invalid hostname format.")
8
9  # Use subprocess with a list to avoid shell injection
10 subprocess.run(["ping", "-c", "4", host], check=True)
```

Priming

Frame the conversation before you start coding

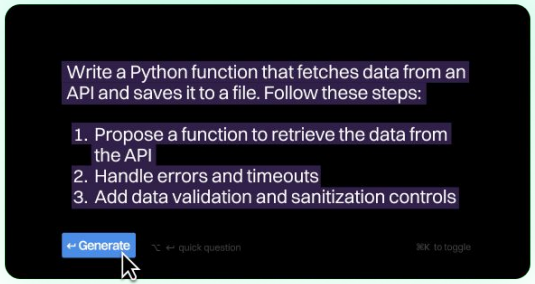
Reasoning-based

Make the model spell out its thinking



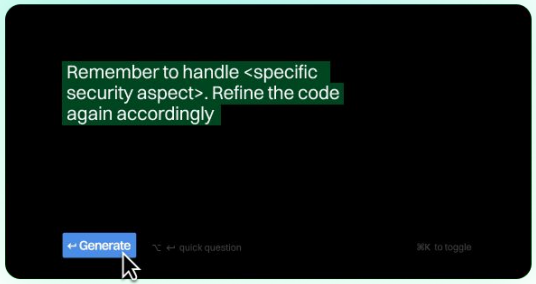
Decomposition-based

Solve a hard problem as several small ones



Refinement-based

Iterate on an idea over multiple prompts

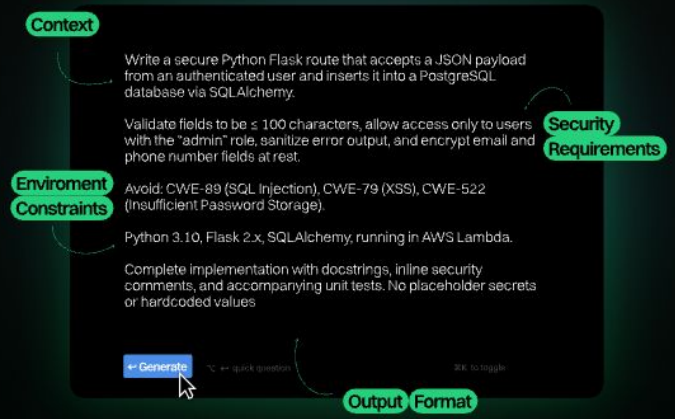


CODE PROMPT LIBRARY

40+ AI Prompts for Secure Vibe Coding

Make Code Safer with Every Prompt

AI coding assistants make writing code a breeze, but they also contain security flaws. This free prompt library helps reduce vulnerabilities at the source, with more secure prompting practices and examples tailored to real-world use cases.



- ✦ **Learn different prompt techniques**
- ✦ **Copy templates to use in your daily work**
- ✦ **Experiment with examples**

SCAN ME

Lesson #2:
Implement security standards

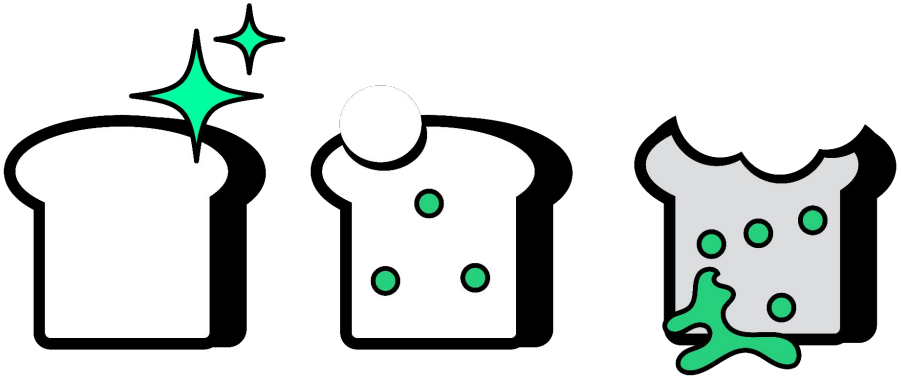
- **Use rules files to drive development**
  Rules files can drive behaviors
- **Use “ignore rule” to protect sensitive data**
  YMMV with different models
- **Test-driven development**
  Automated checks (which you can ask the model to write) can help catch issues

Sample Rule

Organization-specific preference

This rule tells the AI code assistant
what to do whenever there is a user input.

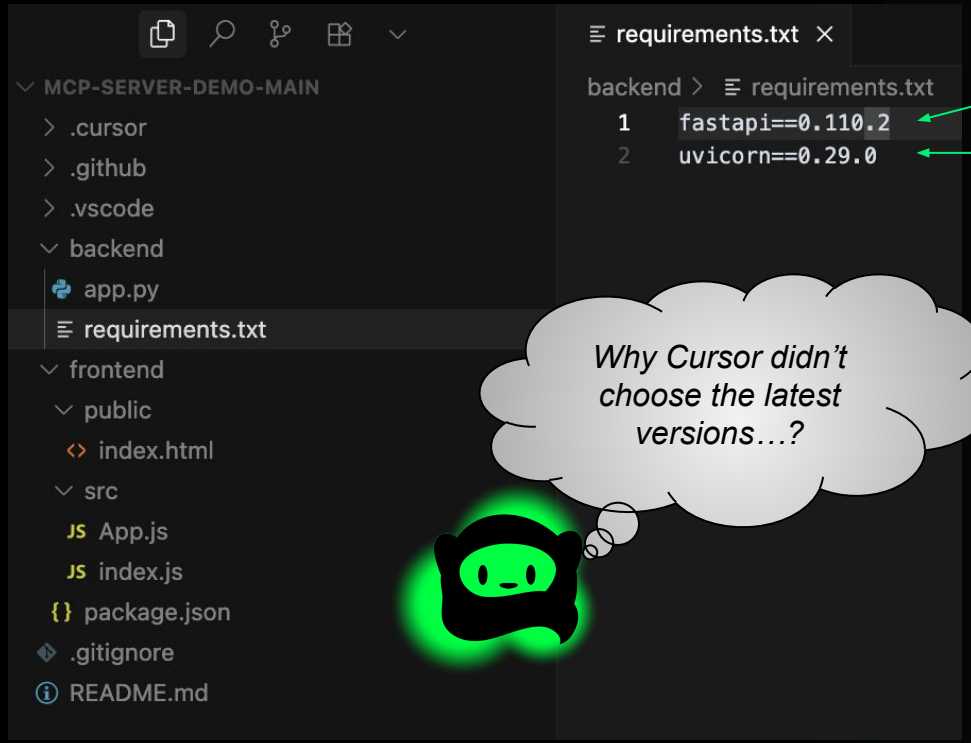
```
## Handle user input sanitization

- Use the internal function sanitizeInputSafe() for all user input sanitization.
Example:

/* js */

// Instead of manually cleaning user input:
// const input = userInput.replace(/[^\w\s]/gi, '');
// Use the organization's approved sanitizer

const cleanInput = sanitizeInputSafe(userInput);

```



Lesson #3:
Get real-time security signal

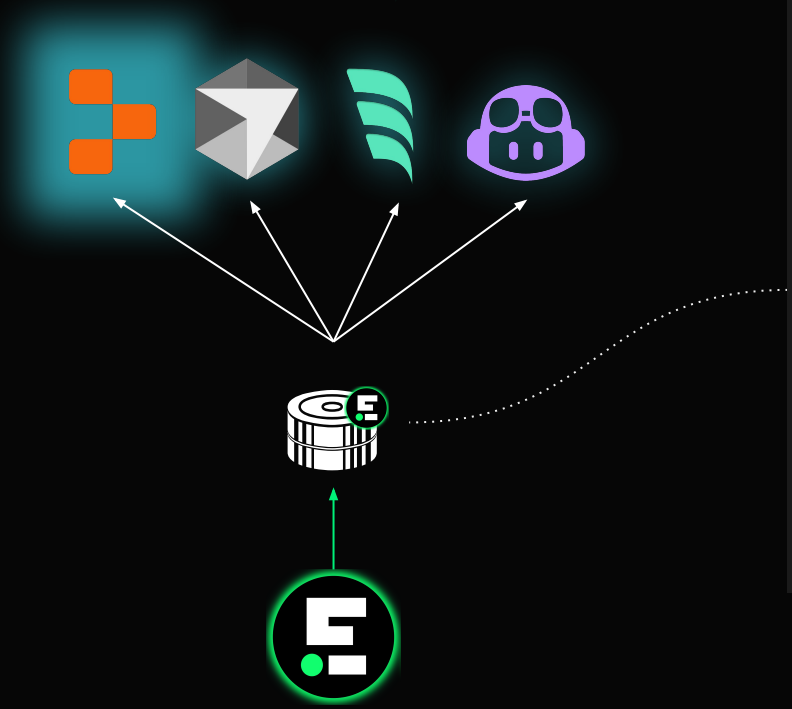
- **Out of date code and data (and CVE data)**
  Models are trained on code that's 1+ year old, but new CVEs are disclosed every day
- **Fresh intelligence**
  MCP servers are useful to inject fresh security intelligence into the development workflow
- **Models can fix issues**
  With enough guidance models can be very good at fixing issues

Beware the training data cutoff



What is an MCP server?

A component that facilitates communication between AI models and various data sources and services.

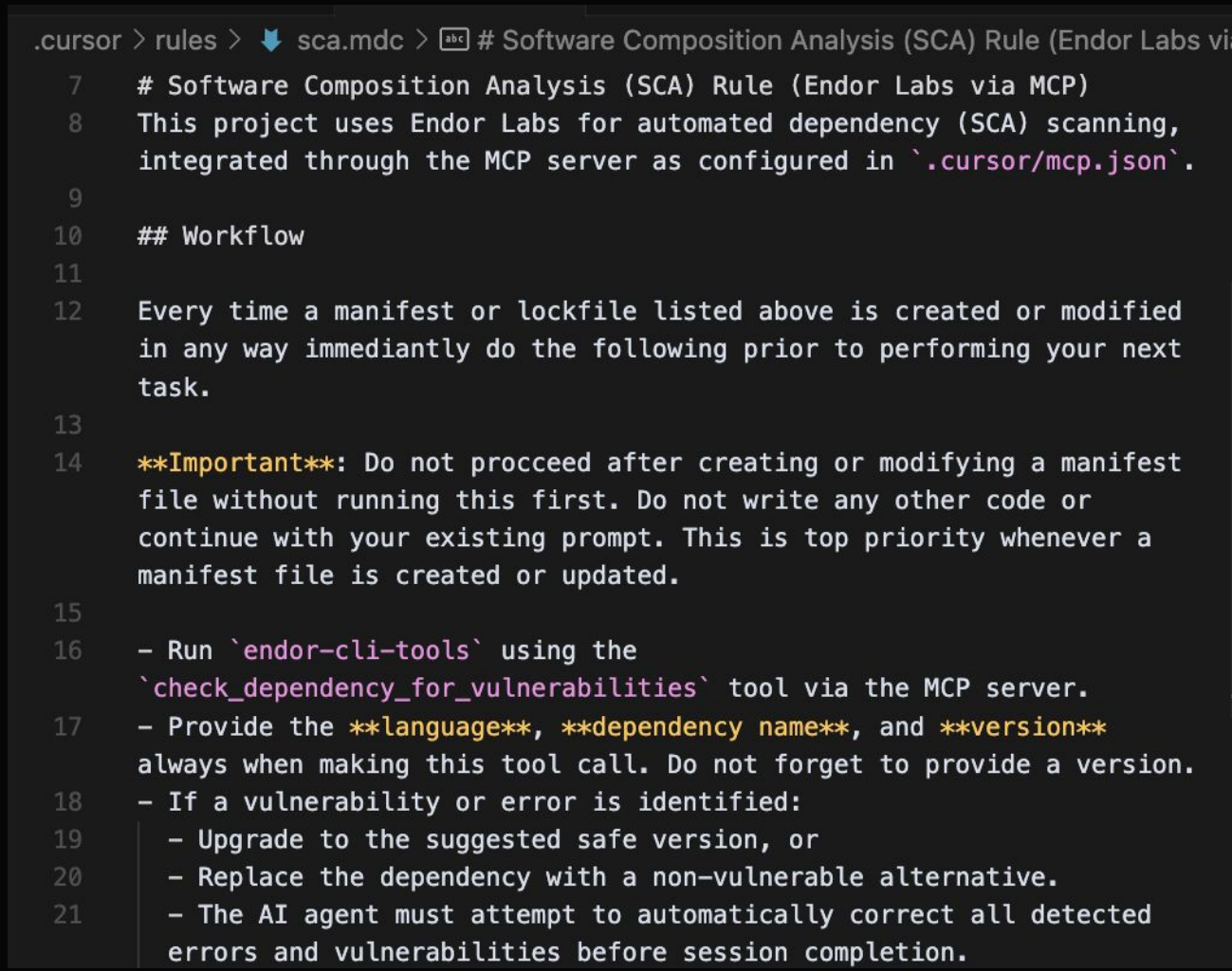


.cursor > {} mcp.json > ...

```
1  {
2    "mcpServers": {
3      "endor-cli-tools": {
4        "type": "stdio",
5        "command": "endorctl",
6        "args": [
7          "ai-tools",
8          "mcp-server"
9        ]
10     }
11   }
12 }
```

How MCP servers provide guidance

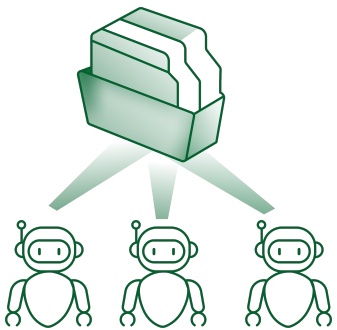
It's all about the rules



Demo

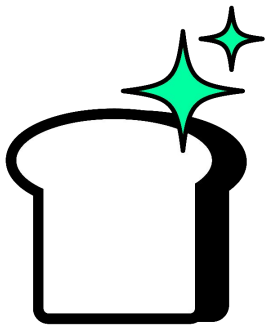
Help devs securely adopt AI code assistants

Start with **secure prompts**



Implement **security standards**

Add **security signal**



Andrew

Connect with me to get deck and resources!

Questions?