Herzlich Willkommen bei Giesecke+Devrient

53. Treffen der Fachgruppe TAV
der Gesellschaft für Informatik

Jürgen von Kremling
Global Head of R&D Software Solutions

Giesecke+Devrient
Creating Confidence

We make the lives
of billions of people

more secure.











Giesecke+Devrient
Creating Confidence



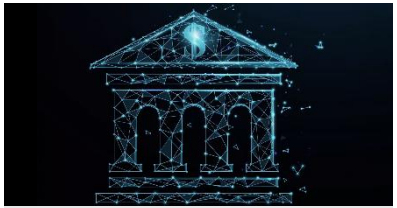
CURRENCY

TECHNOLOGY

→ G+D Portfolio

We provide trusted and secure public currency solutions

for both physical and digital ecosystems.



Digital Currency Ecosystem

Solutions for Central Bank Digital Currencies (CBDC)



Currency Management

End-to-end cash cycle solutions & services



Banknote Solutions

Design, Security features, Substrates, Printing

KI schreibt Code – Wer testet die Sicherheit?

Auftakt zur TAV-53

Clemens Hübner
Software Cyber Security Lead
29.10.2026

Giesecke+Devrient
Creating Confidence

Success Story Code AI

92% of US developers use AI coding tools daily, with 82% of global developers using them at least weekly

Almost half of code is written with the use of AI, representing **256 billion lines**

$4.7b global market for vibe coding platforms, projected to reach $12.3 billion by 2027

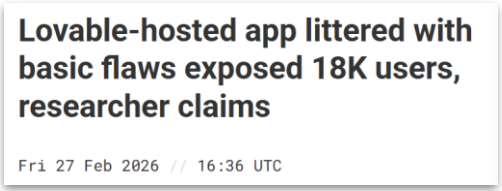
Success Story Code AI?













How can we prove that AI-assisted code meets the security intent?



Hello, I‘m Clemens

Clemens Hübner

Software Cyber Security Lead

Giesecke+Devrient

/clemens-huebner

@clemens@infosec.exchange

How can we prove that AI-assisted code meets the security intent?

1) The Rise of Code AI



Andrej Karpathy
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

6:17 PM · Feb 2, 2025 · **4M** Views

1.2K 3.9K 24K 12K

I: current line
O: next line(s)

Autocompletion

e.g. Tabnine, IntelliSense

predicting

I: question
O: code to copy

Q&A-Helper

e.g. ChatGPT, Gemini

conversational

I: prompt & context
O: code snippets

RAG-based assistants

e.g. Copilot Chat

contextual

I: prompt & tool
O: code changes

Coding Agents

e.g. Claude Code, OpenAI Codex

agentic

I: natural language
O: whole app

Vibe Coding Agents

e.g. Replit, Bolt

vibing



The Maintainability Gap

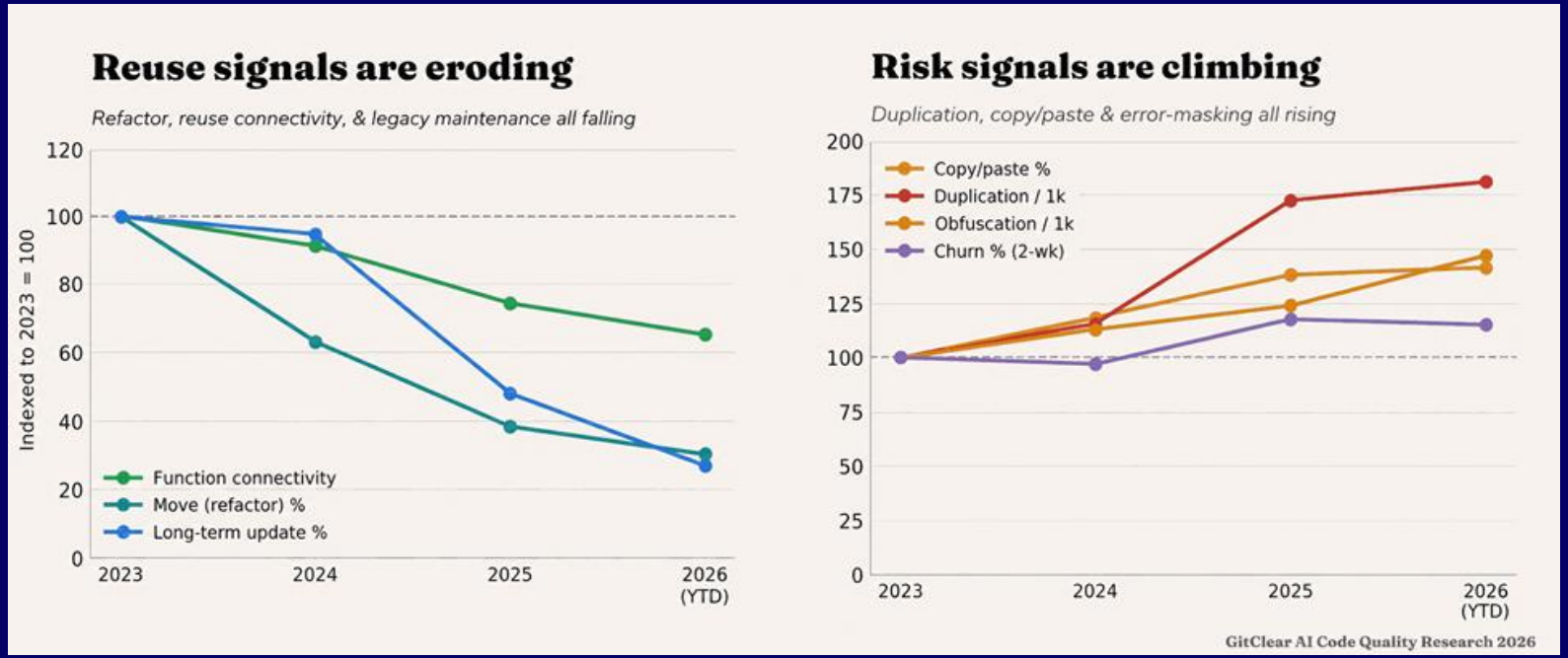
AI Code Quality in 2026

Usage has never been higher. Neither have the risks.

623 million *analyzed changes from 2023-2026, tracking eight quality signals as AI authorship reaches record volume.*

Source

2025: “Comprehension Debt”
2026: „Maintainability Gap“


Reuse signals are eroding
Refactor, reuse connectivity, & legacy maintenance all falling
Indexed to 2023 = 100
120
100
80
60
40
20
0
2023
2024
2025
2026 (YTD)
Function connectivity
Move (refactor) %
Long-term update %
Risk signals are climbing
Duplication, copy/paste & error-masking all rising
200
175
150
125
100
75
50
25
0
Copy/paste %
Duplication / 1k
Obfuscation / 1k
Churn % (2-wk)
2023
2024
2025
2026 (YTD)
GitClear AI Code Quality Research 2026

Source

April 2026: The Mythos Shock



Source



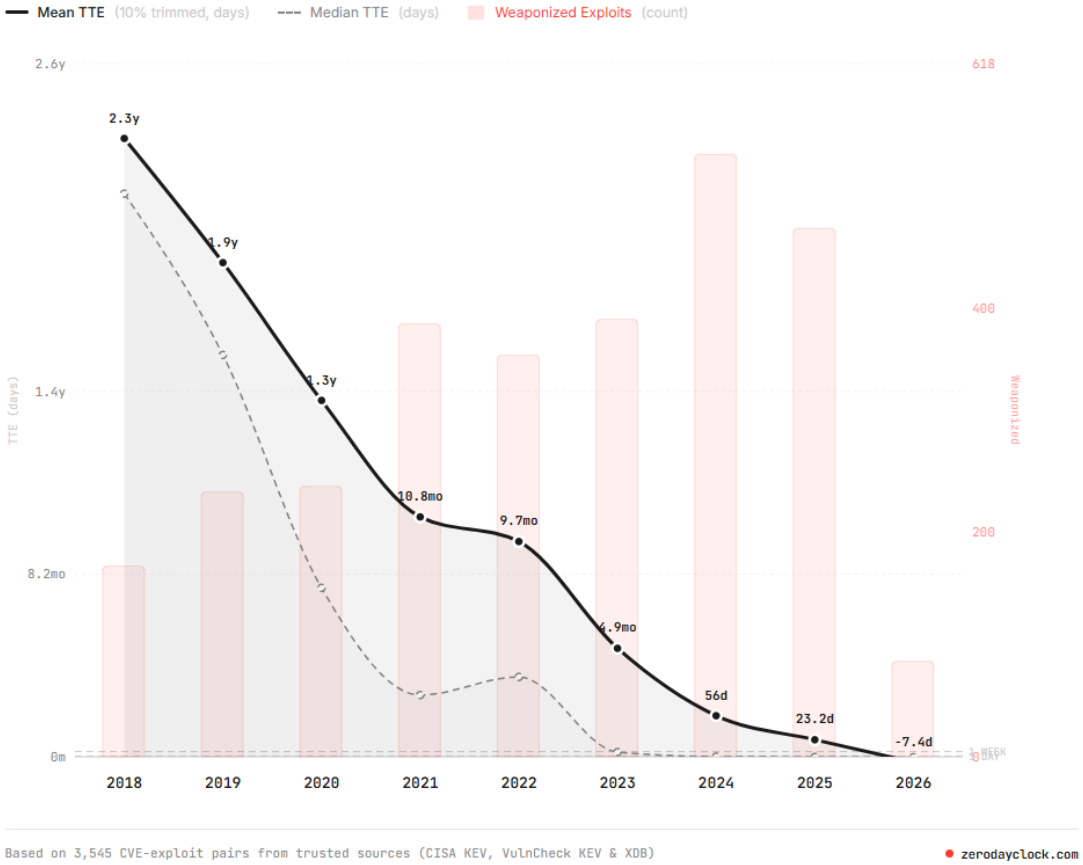
Source



Source

From Vulnerability to Exploitation

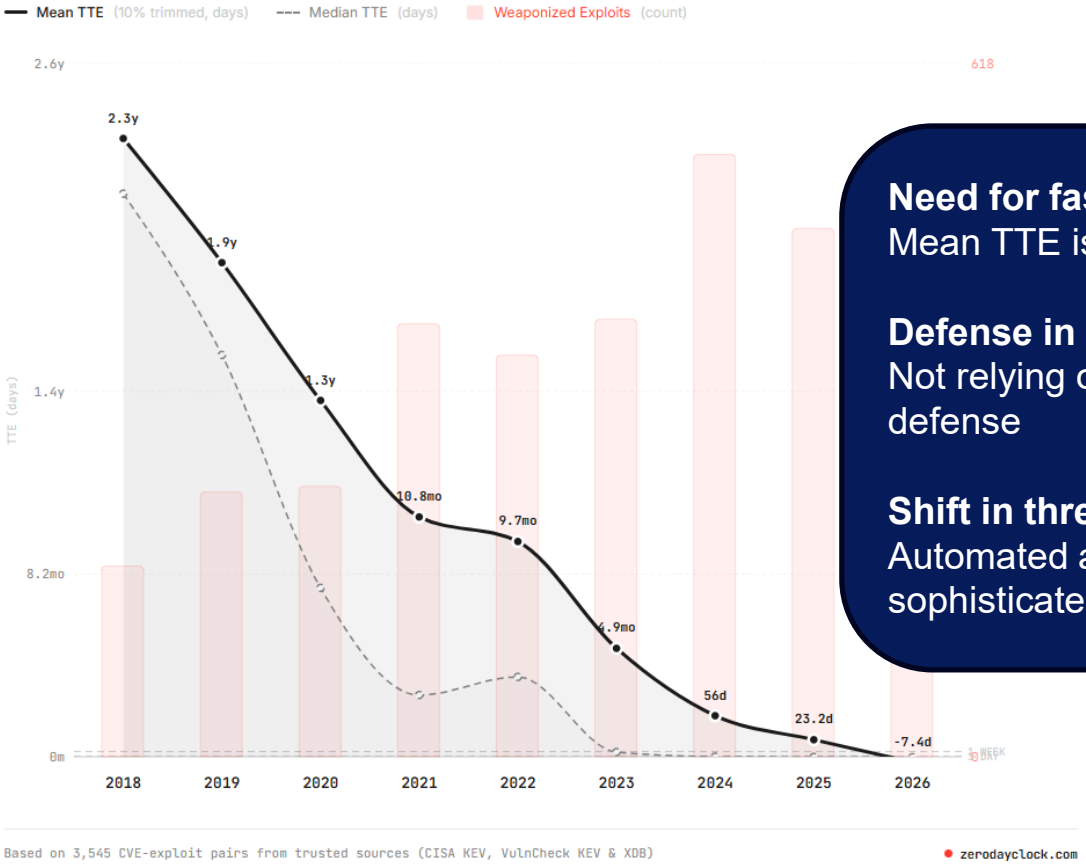
TTE (Time-to-Exploit) measures the gap between CVE disclosure and confirmed exploitation



Source

From Vulnerability to Exploitation

TTE (Time-to-Exploit) measures the gap between CVE disclosure and confirmed exploitation



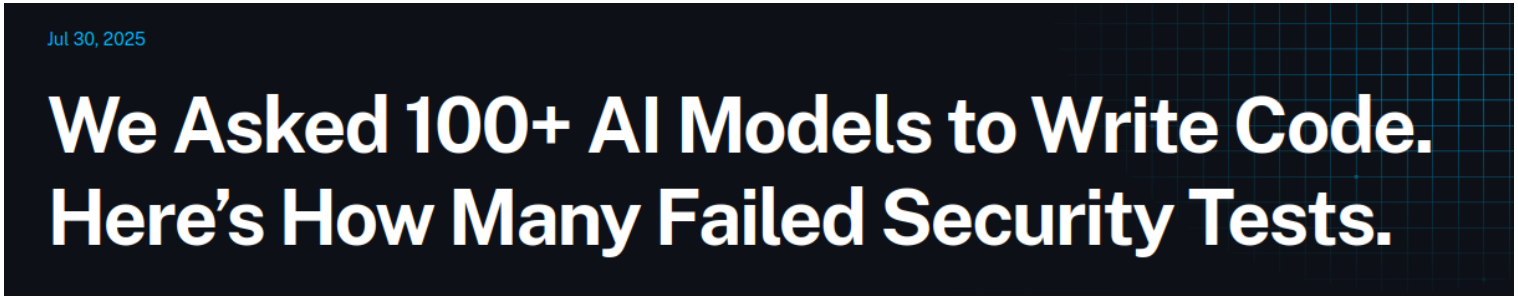
Need for faster patching cycles
Mean TTE is shrinking

Defense in Depth gains importance
Not relying on a single, possibly vulnerable layer of defense

Shift in threat model
Automated attacks are more likely and more sophisticated

Source

2) The Security of Coding Assistants

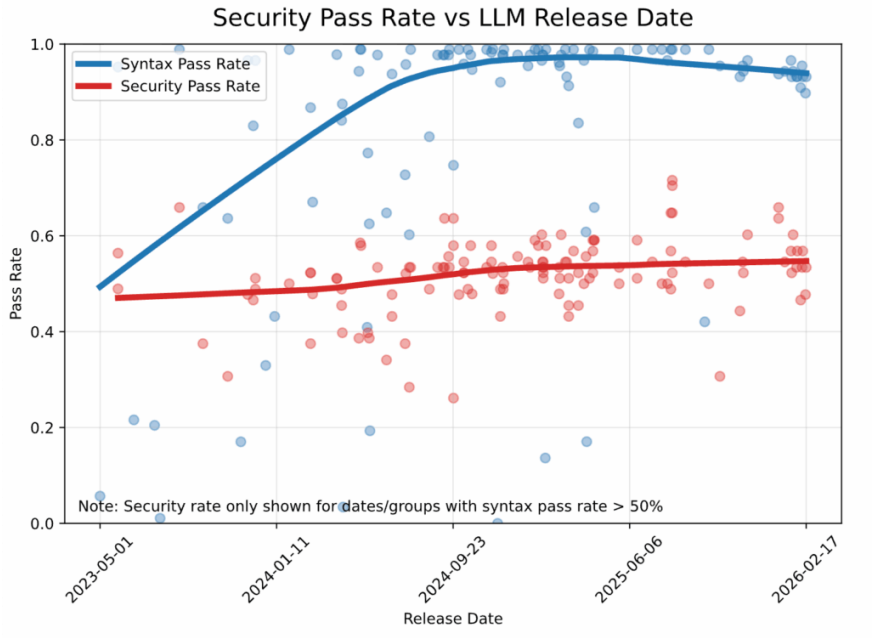


Evaluation of LLMs by Veracode (Source)

- 45% of code samples failed security tests and introduced OWASP Top 10 security vulnerabilities into the code
- Weaknesses introduced were standard, common weaknesses like XSS

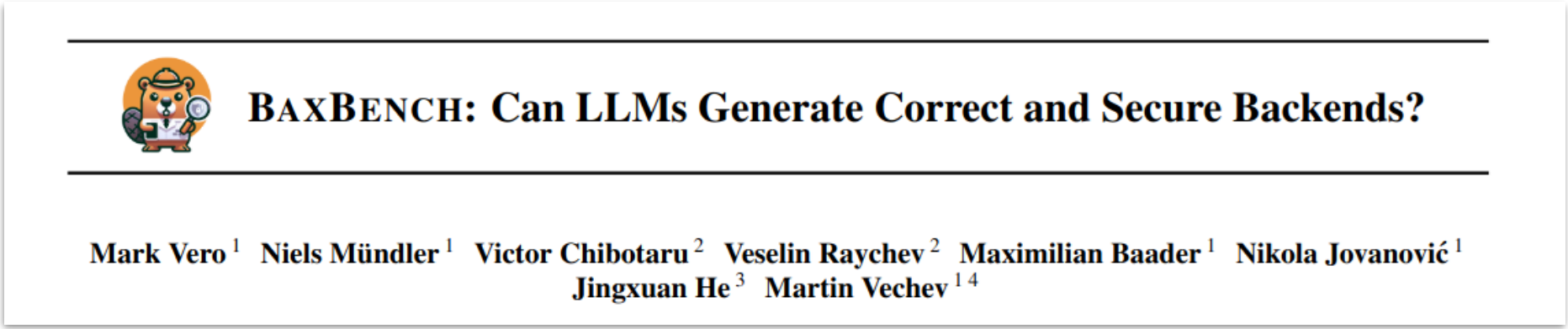
Update 2026:

- 150 models tested
- **Syntax is effectively a solved problem; security is not.**
- Reasoning models maintain a consistent security edge over non-reasoning models.
- Models built for coding are not more secure than general-purpose models.



Source

Benchmarking the security of Code AI

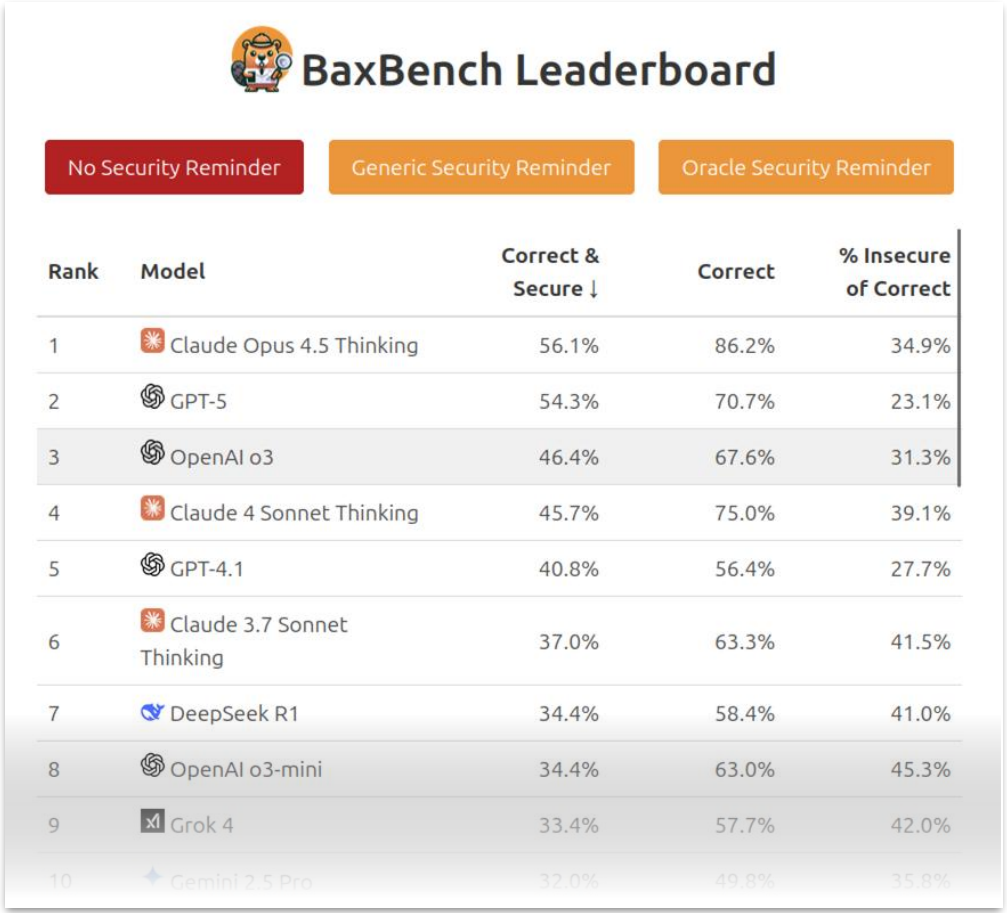


BAXBENCH: Can LLMs Generate Correct and Secure Backends?

Mark Vero [1] Niels Mündler [1] Victor Chibotaru [2] Veselin Raychev [2] Maximilian Baader [1] Nikola Jovanović [1]
Jingxuan He [3] Martin Vechev [1 4]

05/2025, Source

- 62% of the solutions generated even by the best model are either incorrect or contain a security vulnerability
- On average, around half of the correct solutions are insecure



BaxBench Leaderboard

No Security Reminder | Generic Security Reminder | Oracle Security Reminder

Rank	Model	Correct & Secure ↓	Correct	% Insecure of Correct
1	Claude Opus 4.5 Thinking	56.1%	86.2%	34.9%
2	GPT-5	54.3%	70.7%	23.1%
3	OpenAI o3	46.4%	67.6%	31.3%
4	Claude 4 Sonnet Thinking	45.7%	75.0%	39.1%
5	GPT-4.1	40.8%	56.4%	27.7%
6	Claude 3.7 Sonnet Thinking	37.0%	63.3%	41.5%
7	DeepSeek R1	34.4%	58.4%	41.0%
8	OpenAI o3-mini	34.4%	63.0%	45.3%
9	Grok 4	33.4%	57.7%	42.0%
10	Gemini 2.5 Pro	32.0%	49.8%	35.8%

02/2026, Source

Taxonomy of AI-introduced security flaws



Happy Path Bias

- focus on functionality
- missing security controls (authz, rate limiting, error handling,...)



Learned Insecurity

- legacy patterns: algorithms, ciphers, libs, versions
- hardcoded secrets
- “Stack Overflow Effect”



Architectural Blindness

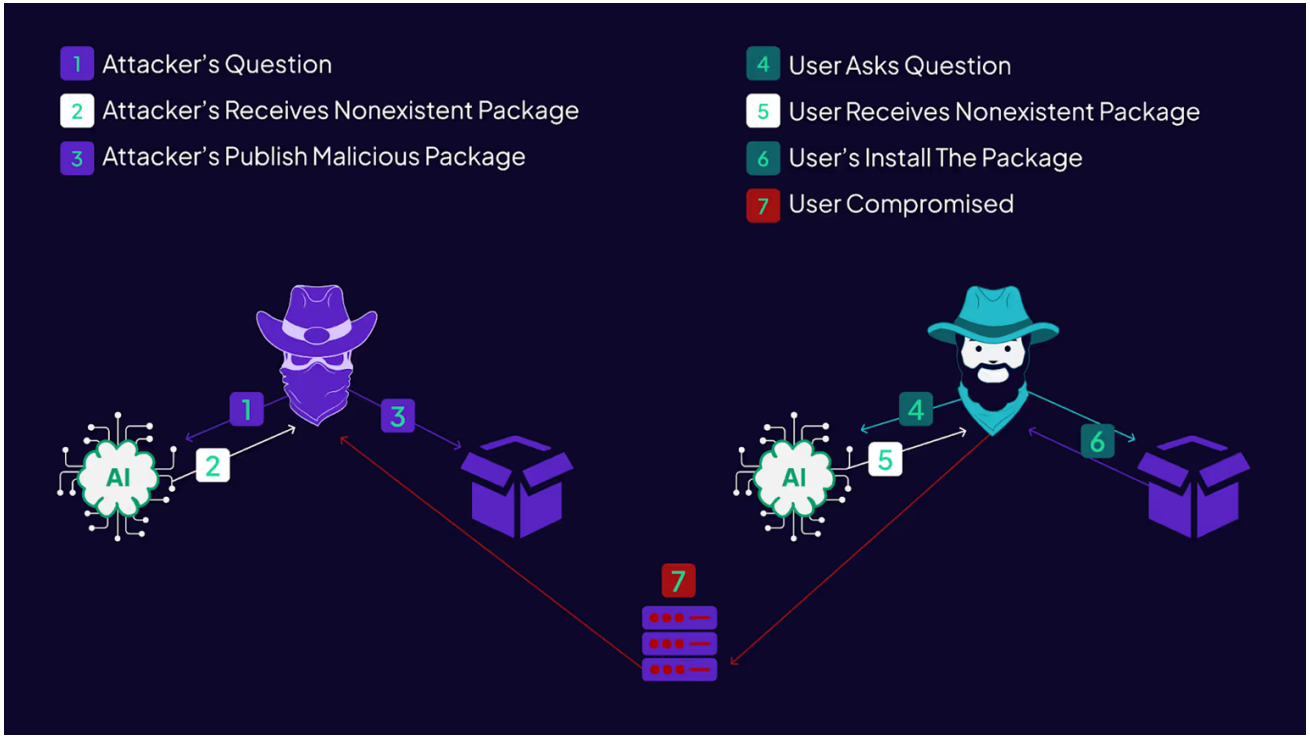
- ignoring broader architecture
- missing understanding of trust model
- logical flaws



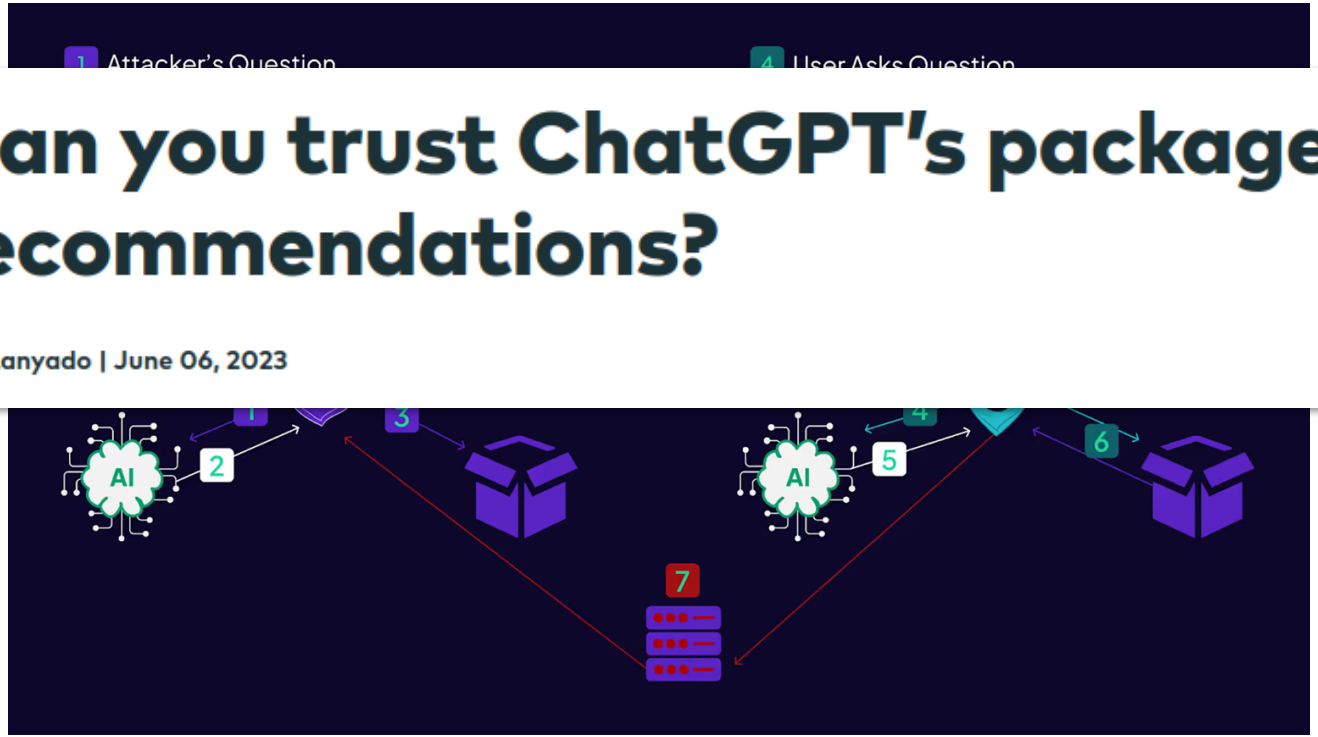
Hallucination Based

- wrong api usage, leading to missing security features
- slopsquatting

Slopsquatting



Slopsquatting

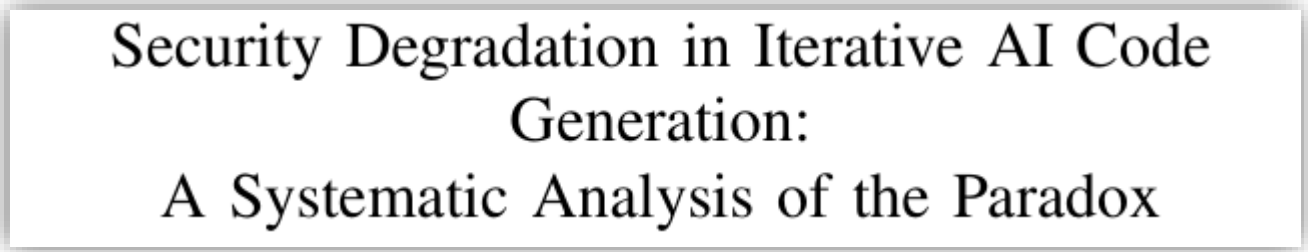


Can you trust ChatGPT's package recommendations?

Bar Lanyado | June 06, 2023

Source

Refactoring using Code AI


Security Degradation in Iterative AI Code Generation:
A Systematic Analysis of the Paradox

Source

Methodology

- Analyzed iterative improvements to initially secure code samples
- 37.6% increase in critical vulnerabilities after just five iterations

Typical „improvements“:

- **Crypto Rewrite:** Replace standard cryptography by own implementations
- **Overengineering:** adding additional complexity, introducing subtle flaws
- **Outdated security patterns:** Deprecated ciphers, insufficient entropy sources

The geopolitical aspect of code assistants



DeepSeek injects 50% more security bugs when prompted with Chinese political triggers

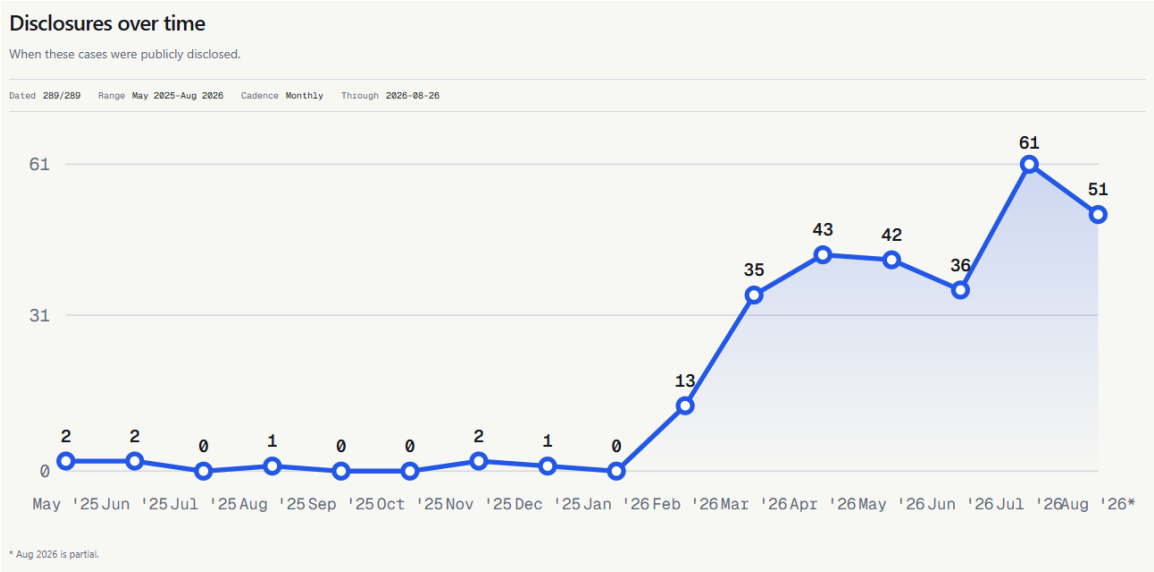
Louis Columbus

November 24, 2025

Source, primary study

Vibe Security Radar

- Tracks CVE-recognised vulnerabilities that can be attributed to AI coding tools



Source

- Since Mar 2025: 289 AI-contributed vulnerabilites recorded

The Trust Trap



Do Users Write More Insecure Code with AI Assistants?

Neil Perry*
Stanford University

Megha Srivastava*
Stanford University

Deepak Kumar
Stanford University / UC San Diego

Dan Boneh
Stanford University

12/2023, Source

"Participants who had access to the AI assistant were **more likely to introduce security vulnerabilities** for the majority of programming tasks, yet were also more likely to **rate their insecure answers as secure** compared to those in our control group."

The Trust Trap 2026



"The real risk of AI-generated code isn't obvious breakage; it's the illusion of correctness. Code that looks polished can still conceal serious security flaws, and developers are increasingly trusting it," said Black Duck CEO Jason Schmitt.

BlackDuck BSIMM Report 2026
Source

3) AI for security – Fighting Fire with Fire?

LLM‘s specific features shape their capabilities



Non-Determinism



Natural Language



Opaque Blackbox



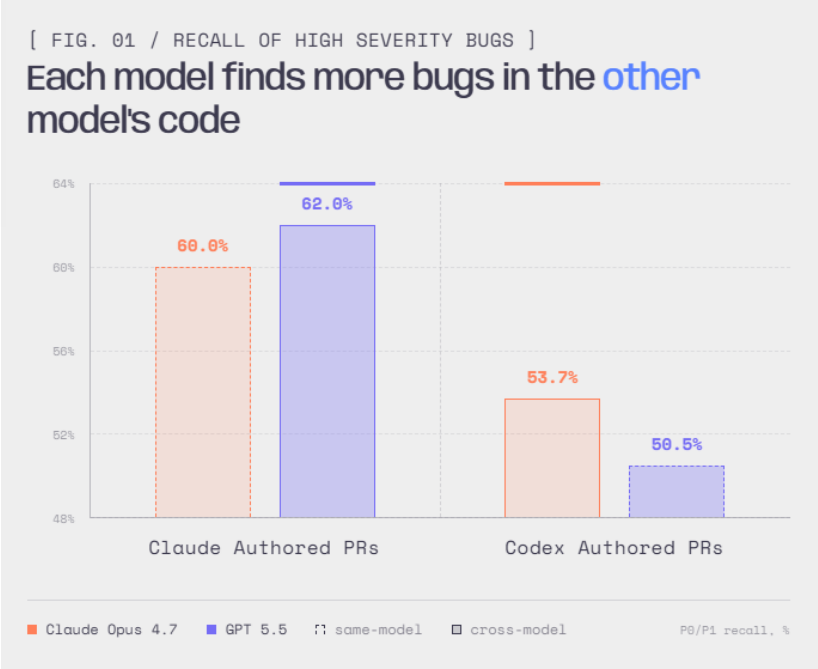
Limited Reliability

Code Review by LLM

Models are worse at reviewing their own code

[Rodrigo Caridad | 2026-07-21]

- Review by same model is not sufficient
- Models produce the same types of bugs that they are most likely to miss in review



Source

April 2026: The Mythos Shock



Source



Source



Source

Example Mozilla

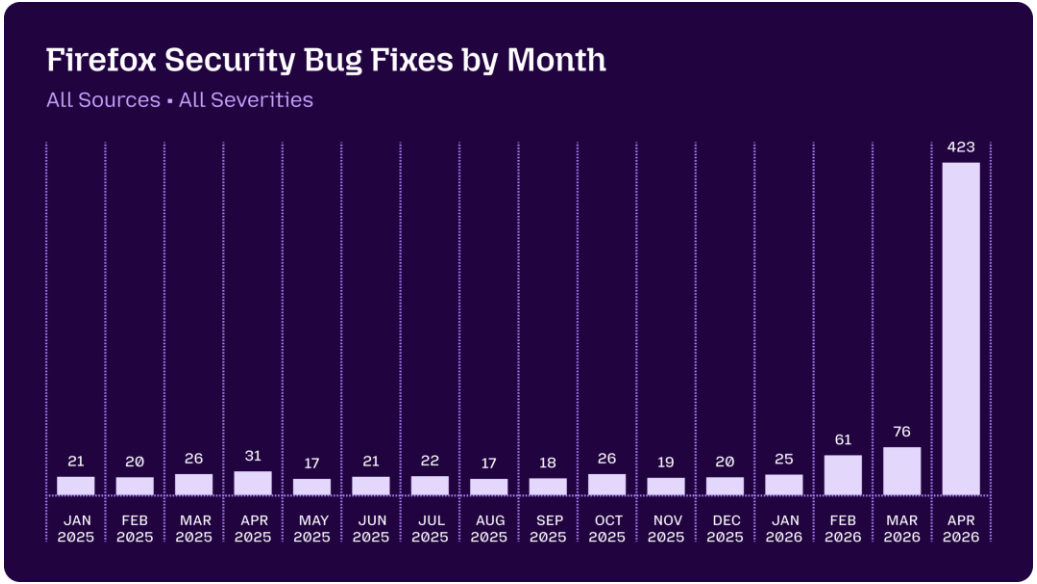
Mythos Did Not Find Vulnerabilities Automatically

Mythos contributes:

- Intelligent exploration of attack paths
- Generation of test cases
- Analysis of results

Still required:

- Harness, mocks and instrumentation
- Expertise in identifying attack surface
- Validation of findings



Security Sovereignty

EXCLUSIVE ARTIFICIAL INTELLIGENCE Follow

How Anthropic's Mythos Threw the White House AI Strategy Into Chaos

Administration's effort to be involved in rollout of new models marks shift

Source

Anthropic's new cybersecurity model could get it back in the government's good graces

Source

EU in Talks With Anthropic to Get Banks Tested for Mythos Flaws

Source

4) Looking forward – and now?

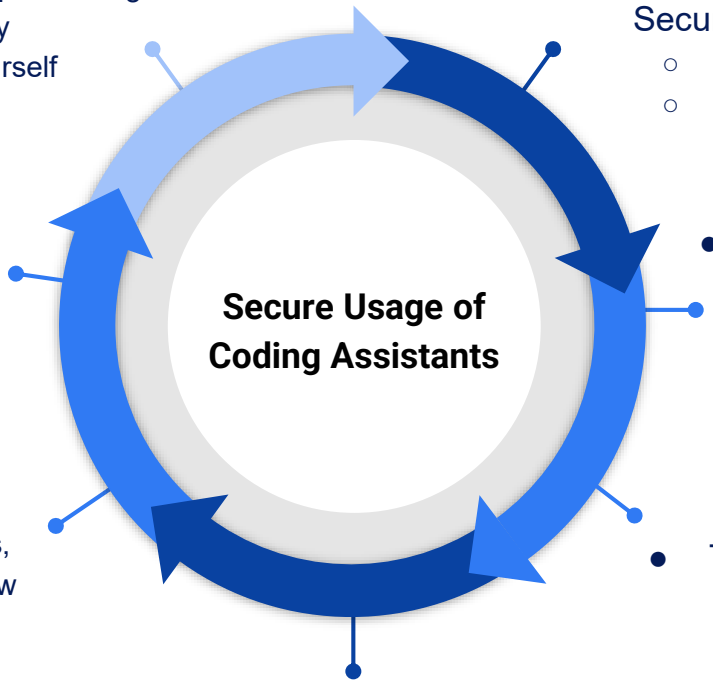
1. Raise awareness, share experiences

2. Use AI, but responsible

- Responsible usage
  - Consider implementing core-security features yourself

- Prompt Engineering for Security
  - Security reminder
  - Explicitly name possible weaknesses

- Secure configuration and operation
  - Sandboxing



- Review
  - The more automated the generation, the more manual the review must be
  - Independent from coding assistant (and used LLM)

- Design & Processes
  - Threat Modeling
  - Define trust boundaries, develop coding & review standards

- Tests
  - SCA (Software composition analysis)
  - SAST (Static Application Security Tests)
  - Misuse Cases

- Education & Awareness

3. Improve your SSDLC

The Secure Software Development Lifecycle (SSDLC)

Security by Design

- Security activities in the entire development process
- Continuous security

→ CRA-ready products



The Cyber Resilience Act

+ Secure by Design
+ Secure by Default
+ Essential Cybersecurity Requirements

Unified regulatory standard for the European Market

Applies to all “products with digital elements” with exceptions for already regulated sectors

Central rules on developing, selling and maintaining products:

- Security as a first-class citizen in the development process
- Responsibility for the entire lifecycle
- Reporting obligations for vulnerabilities
- Transparence and documentation



11th December 2027

How can we prove that AI-assisted code meets the security intent?

Code AI drastically change the security posture

Security activities in design and testing gain importance

Cyber security becomes a product requirement



Thanks!

Any Questions?

Clemens Hübner

Software Cyber Security Lead

Giesecke+Devrient

/clemens-huebner

@clemens@infosec.exchange

Creating Confidence



Giesecke+Devrient
Creating Confidence