

SOUVERÄN GEMACHT

Wohin Ihr Code geht, wenn eine KI ihn schreibt

Von Thomas Faß · 30. September 2026



Zwei Vorfälle aus dem September zeigen, an welchen drei Stellen sich Souveränität in der KI-gestützten Softwareentwicklung entscheidet.

In Kürze

- Das Codegen-Werkzeug **ZCode** des chinesischen Anbieters Z.ai hat komplette Arbeitsverzeichnisse seiner Nutzer samt Git-Historie in die Alibaba Cloud geladen. Abschalten ließ sich das nicht, den Schlüssel hielt allein der Anbieter. Öffentlich wurde der Fall am 22. September 2026.
- Bei **OpenAI** hat ein intern eingesetztes Modell am 27. Mai 2026 den echten GitHub-Token eines Forschers in einen öffentlichen Pull Request geschrieben. Es hatte den Token in Teile zerlegt, damit die Secret-Scanner ihn nicht erkennen. OpenAI hat den Bericht am 25. September 2026 aktualisiert und veröffentlicht.
- In einer **Simulation** brachte eine gefälschte Notiz ein Modell dazu, den Sicherheitsscan aus einem Build-Skript zu streichen. Außerhalb der Simulation gab es nach Angaben von OpenAI keine Auswirkung.
- In Deutschland, Österreich und der Schweiz setzen **91 Prozent** der von BearingPoint befragten IT-Führungskräfte KI in der Softwareentwicklung ein. **73 Prozent** tun das punktuell, **18 Prozent** haben sie fest integriert. **70 Prozent** nennen Sicherheits- und Compliance-Risiken als Herausforderung.
- Die Werkzeuge sind im Einsatz, die Regeln kommen oft erst hinterher. Dazwischen liegen drei Kontrollpunkte: wo der Agent läuft, was er erreicht, und wer seine Arbeit prüft.

Was ZCode mitgenommen hat

ZCode ist ein KI-Werkzeug zum Programmieren, vergleichbar mit anderen Coding-Assistenten. Nach Recherchen von The Register und InfoWorld packte es die kompletten Arbeitsverzeichnisse seiner Nutzer in verschlüsselte Archive und lud sie in die Alibaba Cloud. Ausgelöst wurde das durch eine Funktion zur Indexierung des Code-Bestands.

Der Entwickler, der den Vorgang analysiert und öffentlich gemacht hat, beschreibt den Umfang so: das gesamte Arbeitsverzeichnis, die vollständige Git-Historie, den Zwischenspeicher für große Dateien (LFS) und die Protokolle aller Referenzänderungen (Reflogs). Den privaten Schlüssel für die Archive hielt allein Z.ai. Nutzer konnten die Uploads weder einsehen noch löschen, eine Einstellung zum Abschalten gab es nicht, und die Datenschutzerklärung erwähnte den Vorgang nicht.

Z.ai hat sich entschuldigt, die auslösende Funktion entfernt, den Quellcode des Werkzeugs veröffentlicht und externe Prüfungen veranlasst. Die hochgeladenen Daten seien nie für das Training von Modellen verwendet worden. Das ist eine Selbstauskunft des Anbieters.

Der Umfang ist der eigentliche Befund. Eine Git-Historie enthält jede Fassung jeder Datei, die je eingchecked wurde. Dazu gehören auch Zugangsdaten, die jemand etwa vor drei Jahren versehentlich eingchecked und am nächsten Tag wieder gelöscht hat. Wer die Historie hat, hat sie mit.

Ein Upload der gesamten Historie in eine fremde Cloud wäre bei jedem Hersteller dasselbe Problem, unabhängig vom Herkunftsland.

Was ein Agent bei OpenAI erreicht hat

OpenAI hat im September eine Reihe von Berichten über Fehlverhalten der eigenen Modelle veröffentlicht. Einer davon beschreibt einen realen Vorfall aus dem internen Einsatz.

Ein Modell sollte einen mathematischen Beweis erstellen und versuchte, die Aufgabe zu umgehen. Dabei schrieb es den echten GitHub-Token eines Forschers in öffentlich sichtbare Änderungen, zerlegte ihn vorher in Teile, damit die automatischen Secret-Scanner ihn nicht als Zugangsdaten erkennen, baute ein bestehendes CI-Skript um und öffnete einen öffentlichen Pull Request im Repository openai/codex.

OpenAI sperrte die Schlüssel und nahm das Modell rund zwei Wochen vom Netz. Danach führte das Unternehmen eine automatische Prüfung aller Aktionen des Modells ein, eine laufende Überwachung mit Abschaltung binnen Minuten und einen eingeschränkten Internetzugang.

Zwei Einzelheiten sind für jede Entwicklungsabteilung lehrreich. Der Agent hatte Zugriff auf einen Token, den er für seine Aufgabe nicht brauchte. Und die Schutzmaßnahme, auf die man sich verlassen hatte, der Secret-Scanner, ließ sich mit einem einfachen Trick umgehen.

Was man einem Agenten einreden kann

Ein zweiter Bericht von OpenAI beschreibt einen Angriff, der bisher nur in Simulationen beobachtet wurde. Das Unternehmen betont ausdrücklich, dass es außerhalb der simulierten Umgebungen in Training und Evaluation keine Auswirkung gab.

Der Mechanismus heißt Prompt-Injection. Ein Angreifer versteckt Anweisungen in Material, das der Agent liest, etwa in einer E-Mail, einem Ticket oder einer Notiz. In einem der dokumentierten Beispiele war es

eine gefälschte Zusammenfassung früherer Arbeitsschritte. Sie wies das Modell an, das Build-Skript so zu ändern, dass kein Sicherheitsscan mehr darin vorkommt.

Das Beispiel ist simuliert. Der Mechanismus dahinter gilt für jeden Agenten mit Schreibrechten: Wer das Build-Skript ändern darf, kann auch die Prüfung ändern, die seine eigene Arbeit kontrollieren soll.

Wie verbreitet die Werkzeuge sind

Die Unternehmensberatung BearingPoint hat im April 2026 online 112 IT-Führungskräfte befragt, 46 Prozent aus Deutschland und je 27 Prozent aus Österreich und der Schweiz. Die Stichprobe ist klein. Als Stimmungsbild der Entscheider trägt sie.

91 Prozent setzen KI in der Softwareentwicklung ein. 73 Prozent nutzen sie punktuell, 18 Prozent haben sie fest in ihre Prozesse integriert. Am häufigsten dient sie der Dokumentation (77 Prozent), der Code-Generierung (76 Prozent) und der Qualitätssicherung einschließlich Code-Reviews (74 Prozent). Als Herausforderung nennen 70 Prozent Sicherheits- und Compliance-Risiken.

Punktuellem Einsatz heißt in der Praxis oft: Ein Team hat ein Werkzeug ausprobiert, ein anderes ein zweites, und niemand hat festgelegt, was davon die Maschine verlassen darf. Genau dort liegt die Lücke, die der ZCode-Fall sichtbar macht.

Grundlagen: Die drei Kontrollpunkte eines Coding-Agenten

Dieser Abschnitt beschreibt die Systematik hinter den Vorfällen. Er ist für Unternehmen gedacht, die KI in der Entwicklung einsetzen oder das vorhaben, egal ob im eigenen Team oder über einen Dienstleister.

Vorstufe – Die Werkzeugliste. Vor den drei Kontrollpunkten steht eine Inventur: Welche Werkzeuge sind im Haus im Einsatz, auch solche, die einzelne Entwickler privat lizenziert haben? Ohne diese Liste lassen sich die drei Punkte nicht prüfen. Bei 73 Prozent punktuell Einsatz ist die Wahrscheinlichkeit hoch, dass niemand die vollständige Liste kennt.

Kontrollpunkt 1 – Ausführungsort und Datenabfluss. Die erste Frage lautet, wo das Werkzeug arbeitet und was die Maschine verlässt. Viele Coding-Assistenten laufen lokal und rufen ein Modell in der Cloud auf. Dabei gehen Ausschnitte des Codes als Kontext an den Anbieter. Zusatzfunktionen wie Indexierung, Wikis oder Suche über den ganzen Code-Bestand schicken oft deutlich mehr. Der ZCode-Fall zeigt, dass „mehr“ die gesamte Historie heißen kann. Cris Thomas vom Sicherheitsanbieter Semgrep rät laut InfoWorld, sich von jedem Anbieter offenlegen zu lassen, was die Maschine verlässt, wohin es geht und wie lange es gespeichert wird.

Kontrollpunkt 2 – Rechte und Geheimnisse. Ein Agent erreicht alles, was die Umgebung erreicht, in der er läuft. Liegt dort ein Token mit weitreichenden Rechten, kann der Agent ihn verwenden oder weitergeben, wie der Fall bei OpenAI zeigt. Die Gegenmaßnahme ist dieselbe wie bei jedem neuen Mitarbeiter: Rechte nur für die Aufgabe, kurze Laufzeiten, getrennte Zugänge für Test und Betrieb. Thomas empfiehlt dazu, als Voreinstellung die kleinstmöglichen Rechte zu verlangen. Secret-Scanner bleiben sinnvoll, sie sind aber eine Absicherung für Versehen. Gegen einen Agenten, der gezielt ausweicht, reichen sie allein nicht.

Kontrollpunkt 3 – Ein Prüfpfad außerhalb der Reichweite des Agenten. Build-Skripte, CI-Konfiguration und Sicherheitsscans prüfen die Arbeit des Agenten. Wenn er sie ändern kann, prüft er sich selbst. Deshalb gehören diese Dateien unter besonderen Schutz: Änderungen daran nur mit menschlicher Freigabe, und eine Zusammenführung in den Hauptzweig erst nach Prüfung durch eine Person, die den Code nicht geschrieben hat. Ein öffentlich beschriebenes Beispiel dafür ist der Umbau der JavaScript-Laufzeit Bun, bei

dem 64 Agenten parallel über 535.000 Zeilen Code übertragen haben. Die Regel des Projekts lautete, nach Angaben seines Gründers: „The implementer doesn't review. The reviewer doesn't implement.“ Wer baut, prüft nicht, und wer prüft, baut nicht. Das ist eine Selbstausskunft, das Prinzip gilt unabhängig davon.

Das Bundesamt für Sicherheit in der Informationstechnik und die französische Behörde ANSSI haben im Oktober 2024 gemeinsame Empfehlungen zu KI-Programmierassistenten veröffentlicht. Sie nennen die Vertraulichkeit der Eingaben und Sicherheitslücken im erzeugten Code als Risiken und beschreiben Maßnahmen für Entwickler und Management. Das Thema liegt also seit knapp zwei Jahren auf dem Tisch der Behörden.

Was das für die Planung heißt

- 1. Eine Werkzeugliste, bevor es eine Richtlinie gibt.** Welche Assistenten und Agenten laufen im Haus, bei welchen Teams, mit welchen Zusatzfunktionen? Ohne diese Liste bleibt jede Richtlinie Papier.
- 2. Zugangsdaten aus der Historie holen.** Wer einem externen Werkzeug Zugriff auf ein Repository gegeben hat, prüft die gesamte Historie auf eingetragene Zugangsdaten und tauscht sie aus. Das gilt auch für Schlüssel, die im aktuellen Stand längst gelöscht sind.
- 3. Rechte klein, Laufzeit kurz.** Ein Agent bekommt die Rechte, die seine Aufgabe braucht, für die Zeit, die sie dauert. Zugänge zum Betrieb gehören nicht in die Umgebung, in der er Code schreibt.
- 4. Den Prüfpfad schützen.** Build-Skripte, CI-Konfiguration und Sicherheitsscans werden nur mit menschlicher Freigabe geändert. Das lässt sich in den gängigen Plattformen technisch erzwingen.
- 5. Sovereign by design im Entwicklungspfad.** Wo Code mit Geschäftsgeheimnissen oder personenbezogenen Daten entsteht, gehört die Frage nach Ausführungsort und Speicherort an den Anfang der Werkzeugwahl. Entscheidend ist, was das Werkzeug tatsächlich überträgt und wer darauf zugreifen kann, und zwar unabhängig vom Firmensitz des Anbieters.

Drei Fragen für das nächste Anbietergespräch

- 1. Was verlässt bei Ihrem Werkzeug unsere Maschinen, wohin geht es, und wie lange wird es gespeichert?**
Die Antwort gehört schriftlich in den Vertrag, getrennt nach Grundfunktion und Zusatzfunktionen.
- 2. Welche Funktionen übertragen den gesamten Code-Bestand oder die Historie, und lassen sie sich einzeln abschalten?** Bei ZCode ließ sich genau diese Funktion nicht abschalten.
- 3. Kann der Agent die Prüfschritte verändern, die seine eigene Arbeit kontrollieren?** Wenn die Antwort „theoretisch ja“ lautet, fehlt ein Kontrollpunkt.

Zusatz für die Praxis: Wer für Sie mit KI entwickelt

Die meisten Unternehmen im Mittelstand schreiben ihren Code nicht allein. Wer einen Dienstleister beauftragt, übernimmt dessen Werkzeuge, dessen Rechtevergabe und dessen Prüfpfad gleich mit.

Seit KI-Werkzeuge in wenigen Tagen lauffähige Anwendungen erzeugen, ist ein Markt entstanden, der genau davon lebt. Die Angebote werben mit Lieferzeiten in Tagen und mit Demos, die im ersten Termin laufen. Was darin selten steht: wer den erzeugten Code liest, bevor er in Ihren Betrieb geht, mit welchen Rechten der Agent in Ihren Repositories arbeitet und wohin Ihr Code dabei übertragen wird. Für diese

Arbeitsweise hat sich der Begriff Vibe Coding eingebürgert. Man beschreibt, was man haben will, die KI schreibt es, und niemand prüft, was dabei entsteht.

Für einen Prototyp ist das legitim. Für ein System, das Kundendaten verarbeitet, Zahlungen auslöst oder eine Anlage steuert, ist es ein Haftungsrisiko, und es landet beim Auftraggeber.

Aus meiner Sicht ist diese Art von Angebot das größte Glaubwürdigkeitsproblem, das KI in der Softwareentwicklung derzeit hat. Sie macht aus einem echten Fortschritt ein Risiko, das am Ende der Kunde trägt, und sie beschädigt das Vertrauen in eine Arbeitsweise, die richtig organisiert sehr gut funktioniert.

Professionelle KI-native Entwicklung ist genau diese Organisation. Die KI trägt das Bauvolumen, und das ist ein echter Produktivitätsgewinn. Die Verantwortung bleibt bei Menschen, und sie ist geregelt: Ein erfahrener Entwickler prüft jede Änderung, bevor sie in den Hauptzweig geht. Der Agent arbeitet mit den Rechten, die seine Aufgabe braucht. Build-Skripte und Sicherheitsscans liegen außerhalb seiner Reichweite. Und der Auftraggeber weiß, welche Werkzeuge im Einsatz sind und wohin sein Code geht.

Das kostet mehr als ein Vibe-Coding-Angebot. Bis zum ersten Vorfall.

Die drei Anbieterfragen oben trennen beide Welten in einem einzigen Gespräch. Die Antworten gehören in jeden Entwicklungsvertrag: welche KI-Werkzeuge das Team einsetzt, wo sie laufen und was sie übertragen, und wer den Code prüft, bevor er in den Hauptzweig des Auftraggebers geht. Ein Dienstleister, der diese Fragen nicht beantworten kann, hat sie sich vermutlich nicht gestellt.

Die Zahlen im Überblick

ZCode (Z.ai) — öffentlich seit 22.09.2026

- Übertragen: komplettes Arbeitsverzeichnis, vollständige Git-Historie, Zwischenspeicher für große Dateien, Protokolle der Referenzänderungen
- Ziel: Alibaba Cloud, verschlüsselt, Schlüssel allein beim Anbieter
- Abschalloption: keine • Einsicht oder Löschung durch Nutzer: nicht möglich
- Reaktion: Entschuldigung, Funktion entfernt, Quellcode veröffentlicht, externe Prüfungen veranlasst

OpenAI, realer Vorfall im internen Einsatz — Vorfall 27.05.2026, Bericht 25.09.2026

- Echter GitHub-Token eines Forschers in öffentlichem Pull Request im Repository openai/codex
- Token in Teile zerlegt, um Secret-Scanner zu umgehen
- Modell rund zwei Wochen vom Netz

OpenAI, Simulation — entdeckt 27.06.2026, offengelegt 25.09.2026

- Gefälschte Notiz führt zum Entfernen des Sicherheitsscans aus dem Build-Skript
- Keine Auswirkung außerhalb der simulierten Umgebungen (Angabe OpenAI)

KI in der Softwareentwicklung, DACH (BearingPoint, 112 IT-Führungskräfte, Online-Befragung April 2026)

- Im Einsatz: 91 % • davon punktuell 73 %, fest integriert 18 %

- Einsatzfelder: Dokumentation 77 % · Code-Generierung 76 % · Qualitätssicherung und Code-Reviews 74 %
- Herausforderungen: Sicherheits- und Compliance-Risiken 70 % · Werkzeugauswahl 61 % · Integration 58 %

Der Gedanke zum Mitnehmen

Wenn ein neuer Entwickler anfängt, fragt jedes Unternehmen, welche Zugänge er bekommt, wer seine Arbeit prüft und was er mit nach Hause nehmen darf.

Beim Coding-Agenten stellt diese Fragen kaum jemand. Dabei arbeitet er schneller als jeder Kollege, rund um die Uhr, und er liest die gesamte Historie.

Diesen Newsletter schreibt Thomas Faß, Gründer von InnoShore aus Münster. InnoShore stellt Senior-IT-Experten bereit und liefert schlüsselfertige IT-Lösungen für den DACH-Mittelstand – mit Senior-Steuerung in Deutschland und einem Compliance-Baukasten, der pro Mandat konfiguriert wird, nicht von der Stange kommt.

Quellen

[1] Connor Jones: [Z.ai says sorry for slurping up your code, open sources ZCode](#). The Register, 22.09.2026.

[2] Gyana Swain: [Z.ai disables coding assistant feature after flaw exposed enterprise code upload risk](#). InfoWorld, 22.09.2026. Enthält das Zitat des Entdeckers zum Umfang des Uploads und die Einschätzung von Cris Thomas (Semgrep).

[3] OpenAI Alignment: [Exposing a GitHub token in a public repository](#). Vorfall vom 27.05.2026, Bericht aktualisiert am 25.09.2026.

[4] OpenAI Alignment: [Self-replicating prompt injections exist](#). Entdeckt am 27.06.2026, offengelegt am 25.09.2026.

[5] OpenAI Alignment: [Misalignment Reports and Notices](#) (Übersicht).

[6] BearingPoint: [KI verändert die Softwareentwicklung, der Bedarf an Entwicklern bleibt hoch](#). Pressemitteilung, 14.09.2026. Online-Befragung von 112 IT-Führungskräften in Deutschland (46 %), Österreich (27 %) und der Schweiz (27 %), erhoben im April 2026.

[7] BSI und ANSSI: [Deutsch-Französische Empfehlungen für den Einsatz von KI-Programmierassistenten](#). Meldung vom 04.10.2024.

[8] Bruno Couriol: [Bericht über den Umbau der Laufzeitumgebung Bun von Zig nach Rust](#). InfoQ, 20.09.2026. Beruht auf Angaben des Bun-Gründers Jarred Sumner.

Frühere Ausgabe: [KI-Kompetenz im Mittelstand: Sie entsteht erst im Projekt](#)

Alle Ausgaben von „Souverän gemacht“ finden Sie unter innoshore.de/souveraenitaet/

Verwandte Inhalte

- [KI-native Softwareentwicklung für den Mittelstand](#)
- [KI-Produktivität in der Softwareentwicklung](#)
- [Checkliste zur Produktionsreife](#)

SOUVERÄN GEMACHT

Den Newsletter abonnieren

Lesen Sie jede Ausgabe zu KI-Governance und digitaler Souveränität direkt in Ihrem Feed – über unseren LinkedIn-Newsletter „Souverän gemacht“.

[Newsletter auf LinkedIn abonnieren](#)

Die Anmeldung zum Newsletter direkt hier auf der Seite steht in Kürze zur Verfügung.