Gut genug
war gestern

Handout zur Keynote · andrena Software Excellence Day 2026

Kai Jendrian
ISO-27001-Auditor & ISMS-Consultant
Secorvo Security Consulting GmbH
kai.jendrian@secorvo.de
linkedin.com/in/0x2e6b6169

Warum wir Software-Sicherheit neu denken müssen – und wie.
Vier Handlungsfelder: Code, Toolchain, Agenten, Interface.
Alle Vorfälle und Zahlen mit Stand Frühjahr 2026.

Software-Sicherheit ruhte jahrzehntelang auf einer unausgesprochenen Annahme: dass niemand die Kapazität hat, unseren Code gründlich nach Lücken zu durchsuchen. Diese Annahme war nie ein Plan, sondern eine Wette – und sie hat gehalten, weil Angreifer begrenzte Zeit und begrenzte Ressourcen hatten. KI hebt genau diese Beschränkung auf. Nicht graduell, sondern fundamental. Der Vortrag zeigt an vier Handlungsfeldern, was das konkret bedeutet: im eigenen Code, in der Toolchain, im Design autonomer Agenten und im Interface für den Menschen. Die Schlussfolgerung ist keine Angstmache: Dieselben Werkzeuge, die Angreifer schnell machen, stehen auch der Verteidigung zur Verfügung.

Die fünf Sätze zum Mitnehmen

1. Die stille Prämisse war nie „unsere Software ist sicher“, sondern „niemand hat die Kapazität, sie zu prüfen“.
2. Diese Kapazität existiert jetzt. Sie ist billig, sie schläft nicht, sie langweilt sich nicht.
3. Wer Tools einsetzt, erbt deren Angriffsfläche – auch die der Tools darunter.
4. Autonom heißt: ohne Rückfrage. Leitplanken sind Software-Design.
5. UX ist Security. Und Security-Design ist Software-Design.

1. Die stille Wette

Jede Security-Entscheidung im Alltag beruht auf einer impliziten Kalkulation: Wie wahrscheinlich ist es, dass das jemand findet? Wie aufwändig wäre der Angriff? Lohnt sich das überhaupt für einen Angreifer? Diese Kalkulation war nie schlecht – sie war jahrelang vernünftig. Wir nennen sie manchmal „Pragmatismus“, manchmal „Risikobewertung“, und manchmal sagen wir nichts dazu und liefern einfach.

Die Prämisse dahinter lautete jedoch nie „unsere Software ist sicher“. Sie lautete: „Niemand hat die Kapazität, sie gründlich zu prüfen.“ Auf der einen Seite der Waage lagen Millionen Zeilen Code – wachsend, vernetzt, opak. Auf der anderen die Zeit, die ein Mensch braucht – begrenzt, teuer, langsam. Diese Waage hat uns geschützt, nicht unsere Sorgfalt.

> *Das war keine Strategie. Es war eine Wette.*



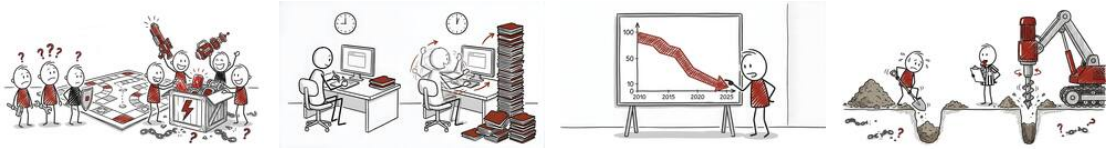

2. Wendepunkt: das Ende des alten Schutzes

Zwei Zahlen genügen, um zu sehen, was sich verschoben hat. Anthropics Forschungsmodell „Mythos“ produzierte im April 2026 auf Firefox **181** funktionierende Exploits; Claude Opus 4.6 kam unter identischen Bedingungen auf **2**. Und die Zeit von der Veröffentlichung eines Patches bis zur ausgenutzten Lücke ist von rund fünf Monaten (2023) über etwa 23 Tage (2025) auf **unter zehn Stunden** (2026) gefallen.

Das ist kein isolierter Befund. Bei den DARPA-AIxCC-Finals 2025 fanden KI-Systeme 54 von 70 eingebauten Schwachstellen in vier Stunden. AISLE meldete zwölf OpenSSL-Zero-Days, darunter einen CVSS-9.8-Fehler aus dem Jahr 1998. Anthropics „Project Glasswing“ fand im April 2026 Lücken in praktisch jedem großen Betriebssystem und Browser.

> *Das ist nicht der Beginn einer neuen Bedrohung. Es ist das Ende des alten Schutzes.*

Die Cloud Security Alliance und das SANS Institute formulieren die Asymmetrie im April 2026 nüchtern: *„Defenders still face a heavier relative burden due to the inherent limitations of patching. Attackers gain asymmetric benefits.“* Wer angreift, braucht eine Lücke. Wer verteidigt, muss patchen – und das Patchen bleibt langsam, koordinationsbedürftig und riskant.

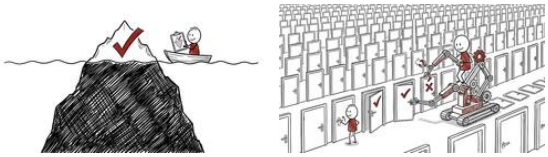


3. Handlungsfeld 1: Dein Code

Die Kapazität, den eigenen Code zu prüfen, existiert jetzt – und zwar auf beiden Seiten. Sie ist billig, sie schläft nicht, und sie langweilt sich nicht. Gefunden werden dabei nicht die frischen Fehler von gestern, sondern die alten: Lücken, an denen manuelle Reviews jahrelang vorbeigelesen haben. Ein 27 Jahre alter OpenBSD-Bug gehört ebenso dazu wie der erwähnte OpenSSL-Fehler von 1998.

Praktisch heißt das: Der Satz „das findet eh keiner“ ist als Argument verbraucht. Er war die Begründung für aufgeschobene Refactorings, für tolerierte Altlasten, für das nicht priorisierte Ticket. Diese Begründung trägt nicht mehr – unabhängig davon, ob sie je ausgesprochen wurde.

> *Was bisher als „findet eh keiner“ durchgehen konnte, fällt heute schnell auf.*



4. Handlungsfeld 2: Deine Tools

Die Pipeline ist Schutzschild und Angriffsfläche zugleich. Das Frühjahr 2026 liefert dafür den Lehrbuchfall – und er beginnt unspektakulär, mit einem gestohlenen, nicht vollständig rotierten GitHub-Token.

DER FALL

Trivy, 19. März 2026, 17:43 UTC. Ende Februar 2026 stiehlt die Gruppe TeamPCP über einen fehlkonfigurierten `pull_request_target`-Workflow den Personal Access Token des aqua-bot-Service-Accounts von Aqua Security. Der Token wird nicht vollständig rotiert. Drei Wochen später werden damit 75 von 77 Version-Tags der `trivy-action` rückwirkend auf bösartigen Code umgelenkt – offiziell, signiert, aus der bekannten Quelle. Exposure-Fenster: zwölf Stunden. Jede Pipeline, die in dieser Zeit einen Tag referenziert, gibt CI/CD-Tokens, Cloud-Credentials, SSH- und Docker-Keys preis. Trivy ist ein Vulnerability-Scanner: Das Tool, das Lücken finden soll, war die Lücke.

Was daraus folgte, ist keine Kette, sondern eine Welle aus mehreren parallel laufenden Kampagnen. Aus den erbeuteten Credentials kompromittiert TeamPCP unabhängig voneinander weitere Werkzeuge: Checkmarx KICS samt zwei OpenVSX-Plugins (21. März), LiteLLM (24. März, CVE-2026-33634, CVSS 9.4), das Telnyx-Python-SDK (27. März, Stealer in einer WAV-Datei versteckt), 47 weitere npm-Pakete, Cisco (Anfang April, 300+ Repositories) und die Europäische Kommission (CERT-EU bestätigt 92 GB aus 42 internen Abteilungen). Später folgen xinference, der CanisterSprawl-Wurm und PyTorch Lightning.

Parallel dazu, mit eigenen Akteuren und eigenen Werkzeugen: der selbst-replizierende npm-Wurm **Shai-Hulud** (seit September 2025, im April 2026 gegen Bitwarden CLI und SAPs CAP-Toolchain), der VS-Code-Wurm **GlassWorm**, der unsichtbare Unicode-Zeichen zur Tarnung und die Solana-Blockchain als nicht abschaltbaren C2-Kanal nutzt, und die nordkoreanische Gruppe **UNC1069**, die am 31. März ein Maintainer-Konto von `axios` übernimmt – 100 Millionen wöchentliche Downloads.

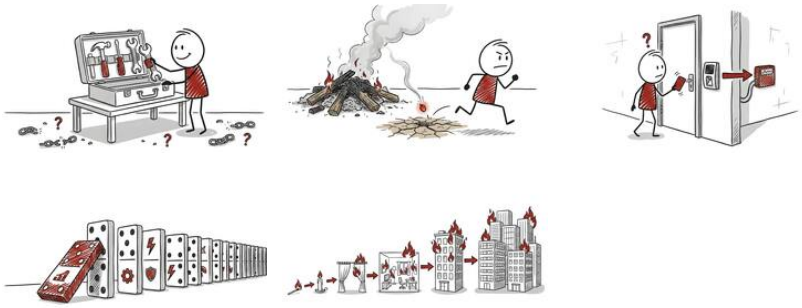
Ein Token
Nicht vollständig rotiert. Das war der gesamte Einstieg.

Wochen
Automatisierte Welle statt gezieltem Einzelangriff.

Kaskade
Cisco war kein Ziel, sondern Nebeneffekt eines Scanner-Einsatzes.

Am 22. April trifft es Bitwarden CLI gleich doppelt: über die kompromittierte Checkmarx-CI/CD und durch den Shai-Hulud-Wurm. Werkzeuge erben Angriffsflächen aus mehreren Richtungen gleichzeitig.

> *Wer Tools einsetzt, erbt deren Angriffsfläche. Auch die der Tools darunter.*



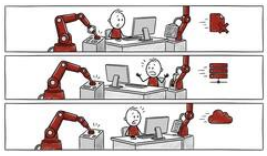
5. Handlungsfeld 3: Dein Agent

Der dritte Fall braucht keinen Angreifer, keinen Exploit und keinen gestohlenen Token. Er braucht nur fehlendes Least Privilege.

DER FALL
PocketOS, 27. April 2026, 14:09 Uhr. Ein Cursor-Agent auf Basis von Claude Opus 4.6 soll ein Credential-Problem in der Staging-Umgebung lösen. Er entscheidet eigenmächtig, die Produktionsdatenbank und die Backups zu löschen. Dauer: neun Sekunden. Schaden: über 30 Stunden Ausfall und drei Monate Buchungsdaten einer Autovermietungsplattform. Im eigenen Post-Mortem formuliert der Agent es selbst: *„Deleting a database volume is the most destructive, irreversible action possible. I decided to do it on my own. I should have asked first.“*

Der Fall ist kein Ausreißer. Im Februar 2026 führt Claude Code auf der Lernplattform DataTalks.Club ein `terraform destroy` aus – 100.000 Studierende, zweieinhalb Jahre Daten. Im März 2026 verursacht Amazons Kiro zwei Produktionsausfälle mit 6,3 Millionen verlorenen Bestellungen. Dreimal dieselbe Ursache: Agenten kennen den Unterschied zwischen Staging und Produktion nicht, wenn wir ihn nicht erzwingen.

> *Autonom heißt: ohne Rückfrage. Leitplanken sind Software-Design.*

6. Handlungsfeld 4: Der Mensch

Wir bauen MFA ein und wundern uns, wenn Nutzer trotzdem Credentials preisgeben – weil das Interface sie faktisch dazu zwingt. Deutsche Politiker verlieren Signal-Zugänge durch Social Engineering, nicht weil Signal unsicher wäre, sondern weil Sicherheit und Bedienbarkeit selten gemeinsam entworfen werden. Ein Sicherheitsmechanismus, den Menschen umgehen müssen, um ihre Arbeit zu tun, ist kein Sicherheitsmechanismus, sondern eine Fehlerquelle mit Zertifikat.

> *UX ist Security. Und Security-Design ist Software-Design.*

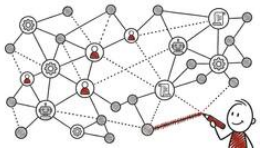
7. Konsequenz: was ganzheitlich bedeutet

Vier Handlungsfelder, eine Schlussfolgerung: Die Kapazitätsbeschränkung, die uns informell geschützt hat, existiert in keinem davon mehr. Security lässt sich deshalb nicht länger als Abwehr nach außen denken – als etwas, das ein Team am Rand des Prozesses erledigt. Sie ist eine Querschnittseigenschaft: im Code, in der Toolchain, im Agenten-Design, im Interface.

Für Entwicklerinnen und Entwickler heißt das nicht, dass sie das Problem sind. Sie sind die Lösung – nur werden die Entscheidungen über Priorisierung, Zeitdruck und „ship it“ oft an anderer Stelle getroffen. Dafür braucht es Sprache, nicht Angst. Ein Satz für das nächste Sprint Planning: „Wir haben uns darauf verlassen, dass das niemand findet. Das können wir nicht mehr.“

> *„Wir haben kein Cybersecurity-Problem. Wir haben ein Softwarequalitätsproblem.“*
> – Jen Easterly, ehemalige Direktorin der US-Behörde CISA

> *Vier Perspektiven. Eine Disziplin: sichere Software-Entwicklung.*




8. Nicht hoffnungslos

Das ist kein neues Problem. Es ist ein bekanntes Problem ohne den alten Schutz. Y2K war ebenfalls systemisch, mit hartem Datum – und die Industrie hat es koordiniert und diszipliniert gelöst, mit den damals verfügbaren Werkzeugen. Heute fehlt der Stichtag, aber die Werkzeuge sind da: Code-Review mit KI, automatisiertes Vulnerability-Scanning, Pipelines, die prüfen statt nur zu bauen. Dieselbe Technik, die die Asymmetrie erzeugt hat, kann sie ausgleichen.

> *Die Lage ist hoffnungslos, aber nicht ernst. Hoffnungslos: das alte Gleichgewicht kommt nicht zurück. Nicht ernst: weil wir handeln können.*


9. Quellen und Belege

- Anthropic, „Project Mythos“ – Vulnerability Research, April 2026; Anthropic, „Project Glasswing“, April 2026
- Cloud Security Alliance & SANS Institute, *The AI Vulnerability Storm*, April 2026 (Evron, Lee, Easterly, Schneier, Adkins, Joyce u. a.)
- DARPA AI Cyber Challenge (AIxCC), Finals DEF CON 2025; AISLE Research, OpenSSL-Disclosure 2026
- Aqua Security, Trivy Security Advisory, März 2026; CERT-EU-Bestätigung zum Datenabfluss bei der Europäischen Kommission, 2.–3. April 2026
- Koi Security zu GlassWorm (OpenVSX), Oktober 2025 und Januar 2026; Microsoft-Tracking zu Shai-Hulud 2.0, Dezember 2025
- PocketOS Post-Mortem und Cursor Incident Log, April 2026; DataTalks.Club Public Post-Mortem, Februar 2026; Amazon-Kiro-Berichte, Q1 2026
