Wenn KI den Code schreibt – wer prüft dann, ob er stimmt?

Ein Gedanke, der die Softwareentwicklung grundlegend verändern könnte

Vitalik Buterin – Mitgründer von Ethereum und eine der einflussreichsten Stimmen in der Kryptographie- und Blockchain-Forschung – hat im Mai 2026 einen bemerkenswerten Essay veröffentlicht. Er beschreibt darin, warum formale Verifikation in einer Welt KI-generierten Codes nicht nur nützlich, sondern notwendig wird. Die folgende Analyse greift seine Kernthesen auf und ordnet sie ein.

Stellen Sie sich vor, Sie beauftragen einen Architekten mit einem Haus. Er liefert Pläne in Rekordzeit – dank KI-Unterstützung. Aber niemand prüft, ob die Statik stimmt. Das Haus steht. Bis es nicht mehr steht.

Genau in dieser Situation befinden wir uns gerade in der Softwareentwicklung.

KI-Systeme schreiben heute Code in einem Tempo und einer Menge, die menschliche Entwickler schlicht nicht mehr vollständig überblicken können. Das ist produktivitätstechnisch beeindruckend. Es ist sicherheitstechnisch ein Problem. Denn mehr Code bedeutet mehr potenzielle Fehler – und in manchen Bereichen kann ein einziger Fehler katastrophale Folgen haben.

Das stille Versagen im Code

Nicht jeder Softwarefehler ist harmlos. Im Alltag bedeutet ein Bug vielleicht eine falsche Anzeige oder ein abgestürztes Programm. In anderen Kontexten bedeutet er etwas anderes:

Ein Fehler im Verschlüsselungsprotokoll eines Messengers macht private Nachrichten lesbar. Ein Fehler in einem Smart Contract auf einer Blockchain kann dazu führen, dass Millionenbeträge automatisch und unwiderruflich verschwinden. Ein Fehler in kryptographischen Systemen – die immer häufiger in Finanzinfrastrukturen eingesetzt werden – kann von Angreifern still ausgenutzt werden, ohne dass überhaupt bemerkt wird, wann oder wie.

Das Unbehagen wächst. Ein erfahrener Entwickler beschreibt es so: Mit den besten KI-nativen Teams hat sich die Codebasis zu etwas entwickelt, *von dem man glaubt, dass es funktioniert – aber mit einer Wahrscheinlichkeit, die sich nicht mehr präzise angeben lässt.*

Das ist kein Randphänomen. Das ist die neue Normalität.

Ein Werkzeug, das fast niemand kennt

Es gibt eine Disziplin, die seit fast 60 Jahren existiert und trotzdem ein Nischenthema geblieben ist: die **formale Verifikation**.

Das Grundprinzip ist verblüffend einfach zu erklären. In der Mathematik beweist man Sätze. Man zeigt nicht nur, dass etwas für tausend Beispiele gilt – man beweist, dass es *immer* gilt, für alle denkbaren Eingaben, ohne Ausnahme. Genau dasselbe kann man mit Software machen.

Ein Computerprogramm ist letztlich ein mathematisches Objekt. Man kann mathematisch beweisen, dass es sich auf eine bestimmte Weise verhält. Und mit modernen Systemen wie **Lean** – einer Programmiersprache speziell für mathematische Beweise – kann ein Computer diesen Beweis automatisch überprüfen.

Das klingt abstrakt. Ein konkretes Beispiel: Eine Forschergruppe hat mathematisch bewiesen, dass das Verschlüsselungsprotokoll des Messengers Signal unter bestimmten kryptographischen Annahmen tatsächlich sicher ist. Nicht "wahrscheinlich sicher" oder "bisher nicht geknackt" – sondern *bewiesen* sicher. Wer die Passivkommunikation von Signal abhören kann, kann trotzdem den Sitzungsschlüssel nicht von einem zufälligen Schlüssel unterscheiden. Das ist keine Behauptung. Das ist ein Theorem.

Warum jetzt?

Wenn diese Technologie so mächtig ist – warum wird sie kaum genutzt?

Der Hauptgrund: Beweise von Hand zu schreiben ist unglaublich aufwendig. Ein einziges Sicherheitslemma kann hunderte Zeilen formalen Beweistext erfordern, geschrieben in einer Sprache, die selbst für erfahrene Entwickler zunächst unlesbar wirkt. Das Verhältnis von Aufwand zu Nutzen war lange ungünstig.

Das ändert sich gerade. KI kann diese Beweise schreiben.

Und das ist der entscheidende Punkt: Wenn KI einen formalen Beweis schreibt und Lean ihn überprüft, hat man etwas Besonderes. Das System prüft sich selbst. Entweder der Beweis stimmt – dann ist die Eigenschaft garantiert. Oder er stimmt nicht – dann wird er abgelehnt, egal wie überzeugend er aussieht. KI kann hier nicht "halluzinieren" und unbemerkt durchkommen.

Die nüchterne Gegenseite

Formale Verifikation ist kein Allheilmittel. Das muss man ehrlich sagen.

Erstens kann man nur das beweisen, was man auch spezifiziert. Wer vergisst, eine kritische Eigenschaft zu formulieren, bekommt einen Beweis für etwas, das nicht das Entscheidende

war. Das ist kein theoretisches Problem – es ist in realen Projekten nachweislich passiert, auch in aktuellen Post-Quanten-Kryptographie-Bibliotheken aus dem Jahr 2025.

Zweitens schützen mathematische Beweise nicht vor Hardwareangriffen. Ein Seitenkanalangriff – zum Beispiel das Messen des Stromverbrauchs eines Prozessors während einer Verschlüsselung – kann theoretisch perfekt geschützte Systeme praktisch angreifbar machen.

Drittens bleibt ein Restrisiko: Selbst das Verifikationssystem Lean könnte Fehler enthalten.

Formale Verifikation reduziert Fehler drastisch. Sie eliminiert sie nicht vollständig.

Das eigentlich wichtige Konzept: Redundanz der Absicht

Hinter formaler Verifikation steckt ein Gedanke, der über die Technologie selbst hinausgeht.

Wenn man eine Absicht auf möglichst viele, möglichst unterschiedliche Weisen beschreibt – als Code, als Tests, als Typsystem, als formalen Beweis – und dann automatisch prüft, ob alle Beschreibungen zueinander konsistent sind, dann steigt die Wahrscheinlichkeit erheblich, dass das System tatsächlich das tut, was man will.

Das ist kein neuer Gedanke. Tests machen das. Typsysteme machen das. Formale Verifikation macht es nur gründlicher und für komplexere Eigenschaften.

Für KI-generierten Code ist dieser Ansatz besonders wertvoll: Die KI schreibt den Code *und* den Beweis. Ein Mensch prüft nur noch, ob die bewiesene Eigenschaft seinen Erwartungen entspricht – nicht mehr jede Zeile Code.

Zwei Welten in einer Software

Das führt zu einer Architektur, die sich bereits abzeichnet – und die die Zukunft sicherheitskritischer Software darstellen könnte:

Ein kleiner, formal verifizierter Kern – für alle kritischen Operationen zuständig, mathematisch geprüft, bewusst klein gehalten.

Eine größere, nicht verifizierte Peripherie – läuft in isolierten Sandboxes, hat nur minimale Berechtigungen, darf Fehler enthalten.

Fällt etwas in der Peripherie aus, schützt der Kern. Nur wenn der Kern versagt, ist wirklich alles gefährdet. Deshalb wird auf den Kern alle Sorgfalt verwendet.

Dieses Prinzip gilt für Betriebssysteme (der Linux-Kernel als sichere Basis), für Blockchains wie Ethereum, für kryptographische Infrastrukturen – und strukturell auch für jedes andere System, in dem Fehler existenzielle Konsequenzen hätten.

Was bedeutet das für KI?

KI steigert die Produktivität in der Softwareentwicklung – das ist unbestreitbar. Aber unkontrolliert eingesetzt, steigert sie auch die Fehlerrate. Das ist die andere Seite derselben Medaille.

Formale Verifikation ist die natürliche Antwort darauf. Nicht als Bremse, sondern als Gegenwicht. KI schreibt schnell. Formale Verifikation stellt sicher, dass das Ergebnis verlässlich ist.

Die eigentliche These lautet daher nicht "KI vs. Sicherheit" – sondern: **KI und formale Verifikation sind komplementäre Technologien.** Wie Blockchain und Zero-Knowledge-Beweise. Oder Geschwindigkeit und Bremsen.

Für Bereiche, in denen Softwarefehler irreversible Schäden anrichten – Kryptographie, Finanzinfrastrukturen, kritische Protokolle – könnte diese Kombination langfristig ähnlich bedeutsam werden wie die Einführung von Verschlüsselung selbst.

Quelle: Vitalik Buterin: *"A quick primer on formal verification"*, 18. Mai 2026. Verfügbar unter: vitalik.eth.limo — Die hier beschriebenen Konzepte – redundante Spezifikation, Kern-Peripherie-Architektur, KI-gestützte Beweisführung – haben eine Relevanz, die weit über Blockchain hinausgeht.