DB InfraGO

AI in Security – AI Insecurity

Q1/2026 | Frankfurt

Inhaltsverzeichnis.

1. AI in Security
2. AI Insecurity
3. Fazit und Diskussion

AI in Security

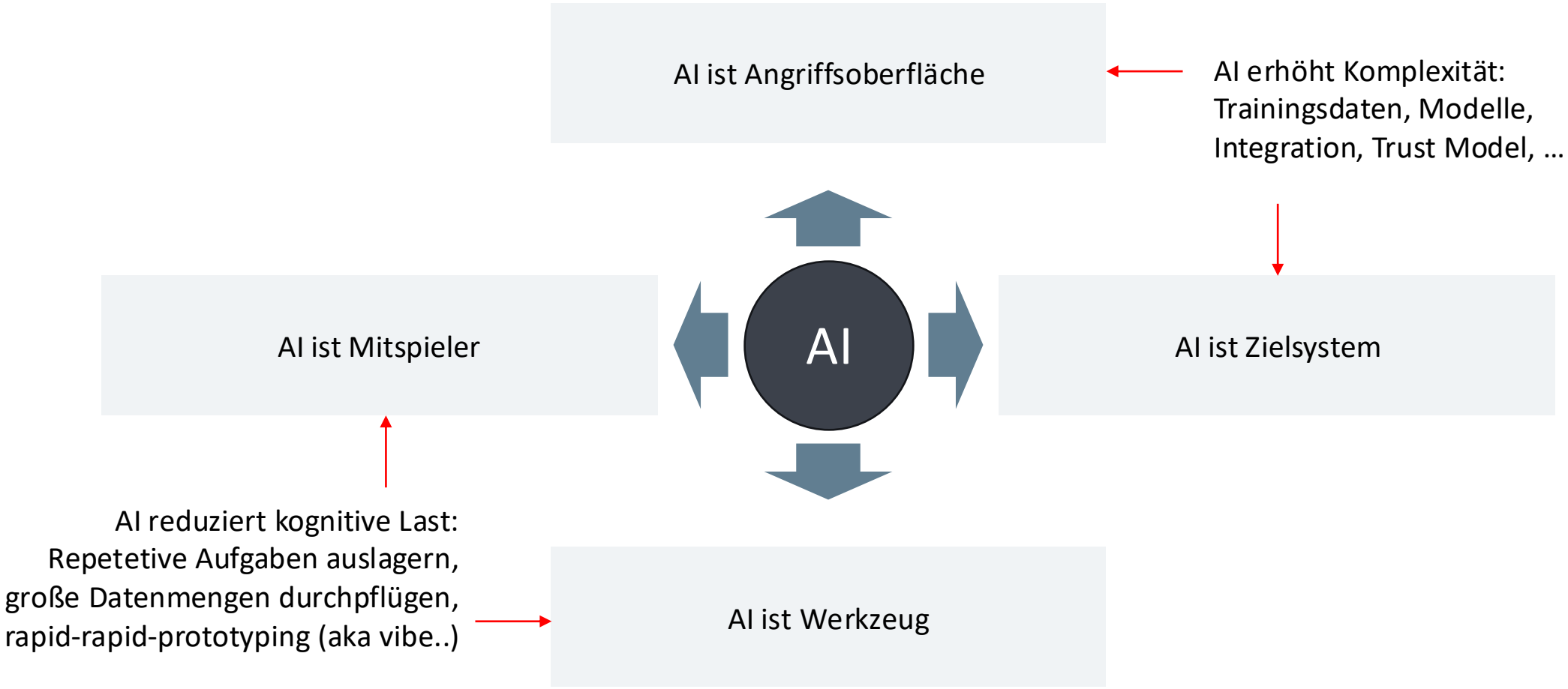
Was kann AI für uns tun?

Ich gehe mit folgenden "steilen" Thesen rein (fight me!):

- In der Regel ist Security (und viele andere Domänen) kein Tooling-Problem. Niemand sagte jemals „verdammt, hätte ich bloß einen Scanner mehr“.
- Die größte Herausforderung der Security ist Komplexität:
  - Komplexität der Betrachtungsgegenstände (Systeme, etc.)
    - Architekturkomplexität
    - Alert Fatigue (SOC, IR, etc.)
  - Komplexität unserer Aufgaben
    - Kontextverständnis (alle Nebenbedingungen berücksichtigen)
    - Immense Datenmengen verarbeiten (Threat Intel, neue Technologien, neue TTP, neue Akteure, neue Regeln..)

Wenn Security primär ein kognitiver Prozess ist:
Was passiert wenn wir kognitive Prozesse skalieren?

Was ist AI aus Sicht eines Securitymenschen?



Die Frage lautet nicht „ist AI gut oder schlecht für Security?“, sondern: „Welche Teile von Security sind Berechnung und welche sind Urteil?“

Die Absprungsbasis für heute: Problemtypen und passende AI-Fähigkeiten (vereinfacht)

DB InfraGO

Problemtyp	Beschreibung/Beispiele	Passende AI-Fähigkeit	Automatisierungslevel	Risiken
Governance / Verantwortung	Haftung, Ethik	-	Mensch	Nicht delegierbar (siehe: IBM-Zitat)
Normative Bewertung	Was ist „richtig“?	-	Mensch	Darf KI nie entscheiden
Entscheidung unter Unsicherheit	Risikoabwägung (inkl. Risikoappetit)	Probabilistic reasoning (sehr eingeschränkt heute)	Mensch (+ Augmentiert?)	Blindes Vertrauen in KI, „Lebenserfahrung“ fehlt
Transformation	Format- oder Strukturumwandlungen	Parsing, deterministic transform	Automatisiert (deterministisch)	Gering – eher technische Fehler möglich
Klassifikation	Zuordnung zu Kategorien	Pattern recognition, inference	Automatisiert (ggf. deterministisch)	Bias (aus Trainingsdaten), Verallgemeinerungen
Abstraktion	Komplexität reduzieren	Semantic compression	Augmentiert	Verlust relevanter Details
Priorisierung	Reihenfolge / Gewichtung	Heuristic reasoning	Augmentiert	Fehlender Kontext
Optimierung	Beste Lösung unter Bedingungen	Search + evaluation	Agentic	Unklare Zielfunktion (oder missverstanden von uns)
Hypothesenbildung	„Was könnte passieren?“	Exploratory reasoning	Agentic	Unrealistische Szenarien
Kombinatorische Exploration	Graphen, Multi-hop	Multi-step reasoning	Agentic	Explodierender Suchraum (riesige Context Windows)

Beispiel: AI im Threat Modeling

- Threat Modeling ist nicht **ein** Problemtyp:
  - (Strukturierte) Hypothesenbildung
  - über identifizierte Angriffswege
  - in einem komplexen sozio-technischen System
  - unter Unsicherheit.

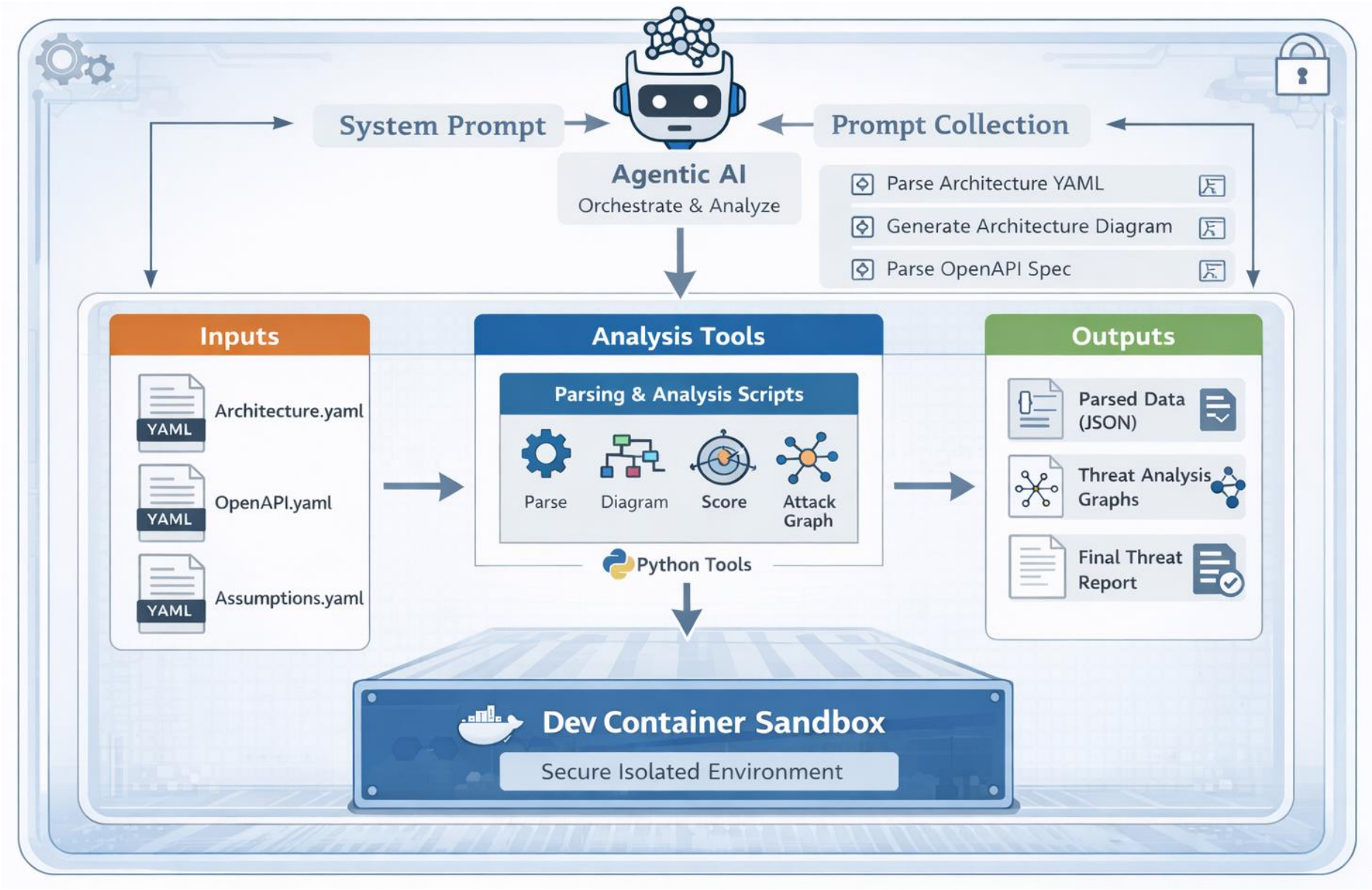
- AI-Unterstützung für
  - Analyse von Architekturen/Systemen
  - Brainstorming und Erstellen von Szenarien
  - Methoden (STRIDE, Attack Trees,…)
  - Berichterstellung, Zusammenfassung, Übersetzung

Verbindung der konkreten Problemtypen mit (sinnvollen) Umsetzungsoptionen, mal mit, mal ohne AI

DB InfraGO

Schritt	Problemtyp	AAA (Automated, Augmented, Agentic)	Konkret
1: Architekturbeschreibung formalisieren	Normalisierung	Automated (deterministic transform)	Pythonskript
2: Architekturdiagramm lesen/bauen	Visualisierung	Automated (template rendering)	Pythonskript
3: Spezifikationen berücksichtigen	Identifikation (Attack Surface)	Automated (Schema parsing)	Pythonskript
4: Bedrohungsmodellierung	Multiple Problemtypen	Agentic (multi-step/explorative reasoning)	**Agenten-Prompt**
5: Bedrohungen bewerten	Risikoklassifizierung	Automated vs Augmented (inference)	Pythonskript
6: Attack Graph/Tree analysieren	Angriffspfadanalyse	Augmented (structured reasoning)	Pythonskript
7: Fehlende Mitigationen aufdecken	Identifikation (Kontrolllücken)	Augmented (mapping)	Pythonskript
8: Geeignete Mitigationen suchen	Klassifikation / Mapping	Agentic (RAG + pattern inference)	**Agenten-Prompt**
9: Bedrohungsmodell finalisieren	Narrative Synthese	Agentic (structured summarization)	**Agenten-Prompt**
10: neue Erkenntnisse einarbeiten	Bedingungen anwenden	Agentic (constraint reasoning)	**Agenten-Prompt**
11: Re-Score …	Neubewertung (Delta)	Automated (weighted calculation)	Pythonskript
12: Re-Analyse …	Delta-Analyse	Agentic (constraint propagation)	Agenten-Prompt
13: Re-Run ...	Generierung (Artefakte)	Agentic (tool-use)	Agenten-Prompt

Das Setup


System Prompt
Agentic AI
Orchestrate & Analyze
Prompt Collection
Parse Architecture YAML
Generate Architecture Diagram
Parse OpenAPI Spec
Inputs
YAML
Architecture.yaml
YAML
OpenAPI.yaml
YAML
Assumptions.yaml
Analysis Tools
Parsing & Analysis Scripts
Parse
Diagram
Score
Attack Graph
Python Tools
Outputs
Parsed Data (JSON)
Threat Analysis Graphs
Final Threat Report
Dev Container Sandbox
Secure Isolated Environment

Demo

DB InfraGO

```
  Macht die Angriffsgraph-Topologie für nicht-technische Stakeholder zugänglich und
  identifiziert visuell die zentralen Drehpunkte (TM-001, TM-002, TM-004 als Hubs mit
  höchster Konnektivität).
  ──────────────────────────────────────

  Artefakt: outputs/threat-model.md
  Typ: Output
  Konsumiert von: Endprodukt
  Beitrag zum Bedrohungsmodell: Konsolidierter deutschsprachiger Bericht mit
    Architekturübersicht, STRIDE-Tabellen, priorisierten Bedrohungen, Angriffsketten,
    Control-Lücken und Restrisikobewertung. Integriert alle vorgelagerten Artefakte in ein
    entscheidungsfähiges Dokument für Security-Reviews, Architektur-Boards und
    Compliance-Audits.

  Zusammenfassung der Wertschöpfungskette:

  OpenAPI-Specs (12 Threats, alle 4 Critical)
            ↘
             Threat Modeling → threats_raw → Risk Scoring → threats_scored
            ↗                                                ↓
  Architecture (13 Threats, Topologie)                  Attack Path Analyzer → attack_paths
            ↘                                                ↓
             Cont[illegible]                                 Ge[illegible]
            ↗                                                ↓
  Assumptions → A[illegible]y As[illegible]mptio[illegible] → [illegible]nerate R[illegible] → th[illegible]t-model[illegible]d

  Die OpenAPI-Specs liefern 48% der Bedrohungen (12/25), darunter alle 4 Critical-Befunde und
   jeden Einstiegspunkt mit Exploitability 5. Die Architektur liefert die restlichen 52%,
  aber primär im Medium/Low-Bereich. Die Control Library und Assumptions verschieben die
  Priorisierung, produzieren aber keine neuen Bedrohungen. Der Attack Path Analyzer
  transformiert die flache Bedrohungsliste in kausale Ketten — ohne ihn wäre der Bericht eine
   Aufzählung statt eines Angriffsnarrativs.

* Cogitated for 52s

  /context
  └ Context Usage
    claude-opus-4-6 · 83k/200k tokens (42%)

    Estimated usage by category
    System prompt: 3.3k tokens (1.7%)
    System tools: 17.4k tokens (8.7%)
    Skills: 164 tokens (0.1%)
    Messages: 65.8k tokens (32.9%)
    Free space: 80k (40.1%)
    Autocompact buffer: 33k tokens (16.5%)
```

```
time=2026-03-02T18:47:05.797+01:00 level=INFO msg="HTTP request" method=GET path=/api/metrics/

┌─ ~/Code/mcc-showcase/executed-agentic-pipeline
└❯ tree
.
├── data
│   └── ai-observer.duckdb
├── inputs
│   ├── architecture.yaml
│   ├── assumptions.yaml
│   ├── openapi-auth.yaml
│   ├── openapi-booking.yaml
│   ├── openapi-payment.yaml
│   ├── openapi-ticket.yaml
│   └── openapi-timetable.yaml
├── outputs
├── prompts
│   ├── apply_assumptions.txt
│   ├── generate_report.txt
│   ├── suggest_controls.txt
│   ├── system.txt
│   └── threat_modeling.txt
├── [illegible]pi[illegible]ne.[illegible]
└── scripts
    ├── at[illegible]ch[illegible]na[illegible]ze[illegible]py
    ├── co[illegible]ro[illegible]ib[illegible]ry[illegible]am[illegible]
    ├── gap_analysis.py
    ├── generate_architecture_diagram.py
    ├── generate_mermaid.py
    ├── parse_architecture.py
    ├── parse_assumptions.py
    ├── parse_openapi.py
    └── risk_scoring.py

┌─ ~/Code/mcc-showcase/executed-agentic-pipeline
└❯ ./run_pipeline.sh

============================================================
 PASS 1 — BASELINE  (no assumptions)
============================================================

═══ [1/15] Parsing architecture ═══
═══ [2/15] Generating architecture diagram ═══

   Review: outputs/architecture_parsed.json, outputs/architecture_diagram.mmd
   Press Enter to continue...

═══ [3/15] Parsing OpenAPI specs ═══
```

ZEIT FÜRS TERMINAL

Erkenntnisse, Empfehlungen, Takeaways

DB InfraGO

- Kognitive Last senken bedeutet auch nicht ständig darüber nachzudenken was schiefgehen kann, also: im Devcontainer laufen lassen! https://github.com/trailofbits/claude-code-devcontainer ist für Claude Code empfehlenswert

- Best Practices haben eine absurd kurze Halbwertzeit in diesem Space:
  - Älter als 6 Wochen? Vermutlich veraltet. Benchmarks vor Dezember 2025: irrelevant.
  - Vorsicht vor AI-Influencern, sie müssen den Hype-Cycle mit neuen Dingen am Leben erhalten, viel Rauschen, wenig Signal
  - Offizielle Quellen und Kurse sind solide, z.B.: https://anthropic.skilljar.com/ (free)

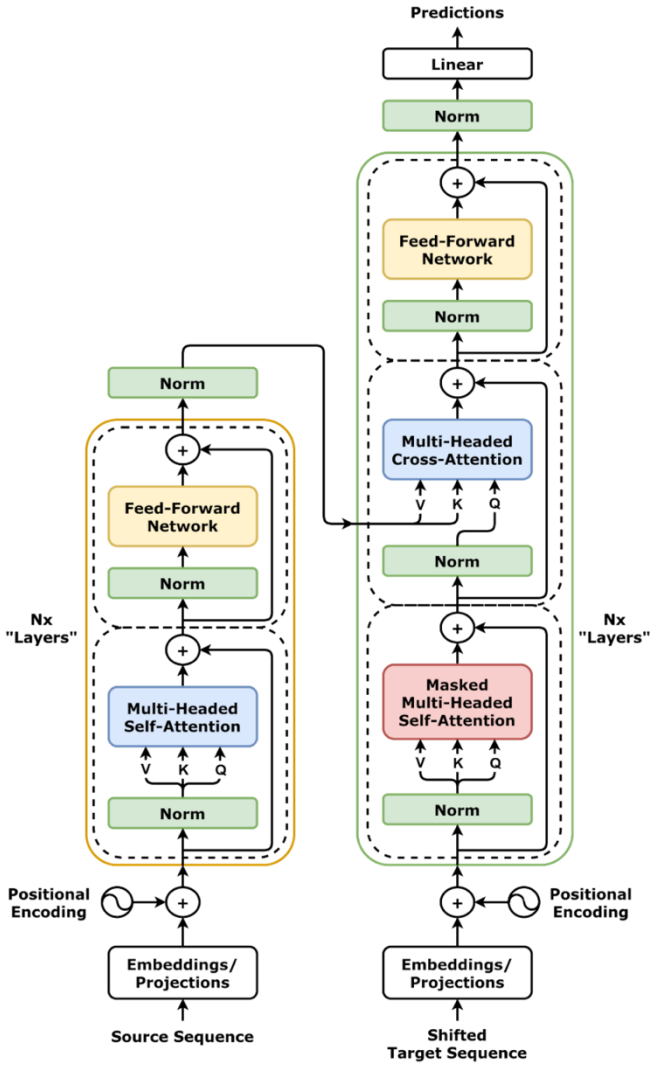
- Faustregeln:
  - Einfache Aufgaben, einfaches Setup. Keine fünf-Agenten-zehn-MCP-Server-Toolchain für simple Projekte
  - Bei Memory an Computer (Registry, L1/L2, RAM, ..) denken (gerade aktuell bzw. hyped: Three-Layer Memory System)
  - Bei Compute an Context Size denken, alte und lange Sessions sorgen für Token Waste -> Zwischenspeichern
  - Agenten die Zwischenschritte committen (git) lassen, fehlgeleitete explorative Sessions lassen sich korrigieren

AI Insecurity

AI Insecurity

1. Was „können“ LLMs?
2. AI Schwachstellen (Auswahl)
3. AI im Threat Modeling
4. AI und Menschen

Was eigentlich ist ein Large Language Model?



Ein Large Language Model ist die technische Umsetzung eines mathematischen Sprachmodells, das Text generieren kann.

Es handelt sich dabei um ein computerlinguistisches Wahrscheinlichkeitsmodell, das statistische Wort- und Satzfolge-Beziehungen mittels eines rechenintensiven Trainingsprozesses aus einem Vorrat an Texten „erlernt“ hat.

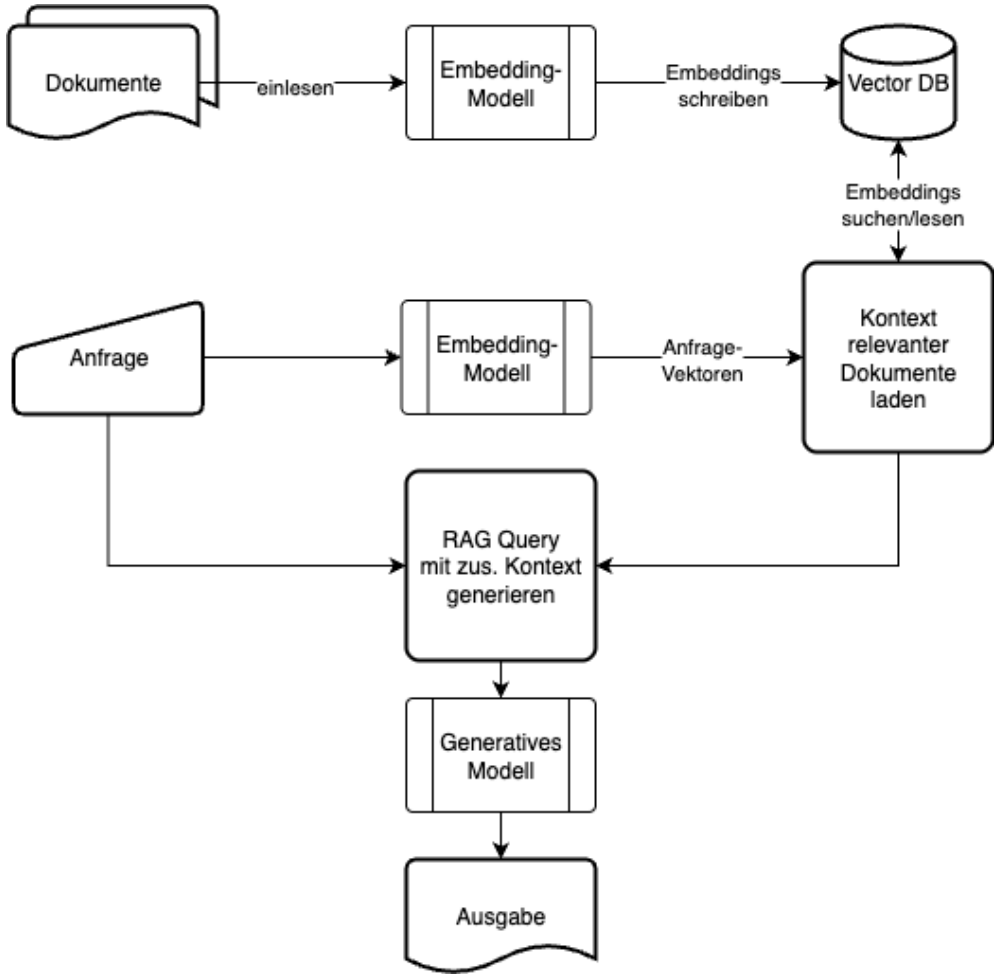
Ein LLM ist stark vereinfacht ein *„stochastischer Papagei“* – es gibt Worte und Satzstücke aufgrund ihrer Wahrscheinlichkeit im Zusammenhang der Trainingsdaten wieder, *ohne die Inhalte wirklich „verstehen“ zu können*, quasi ein sehr rechenlastiges Autocomplete.

Die zugrundeliegende Architektur des Transformer-Modells verortet Worte, Wortfolgen und deren Distanz voneinander in einem Vektorraum, dessen Parameterzahl ein Anhaltspunkt für die Tiefe und Leistungsfähigkeit des Modells ist.

Die Verarbeitung ist begrenzt durch Rechenleistung und Speicher, was sich im Kontextlimit bemerkbar macht, also wieviel Worte/Token im Zusammenhang das Modell pro Anfrage verarbeiten kann.

Bild: dvgodoy - https://github.com/dvgodoy/dl-visuals/?tab=readme-ov-file, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=151216016

Retrieval Augmented Generation (RAG)
Dokumente und weiteres Wissen integrieren



- RAG verbindet Sprachmodelle mit Vektordatenbanken als externe Wissensquellen
- Dokumente werden beim Einlesen von einem Embedding-Modell in Vektoren umgewandelt, um semantisch ähnliche Inhalte auffindbar zu machen
- Das Modell generiert damit Antworten basierend auf relevanten Quellen
- Resultat: Aktuelle, faktisch genaue Antworten ohne Neutrainieren des Modells
- Challenge: Vektorqualität, Indexgröße und Relevanz der Suchergebnisse beeinflussen Qualität und Umfang der Antwort

AI Schwachstellen

Eine Auswahl…

Bild: Initial stage of sinkhole at El Trebol / Theofilos Toulkeridis et. al (CC BY 3.0)

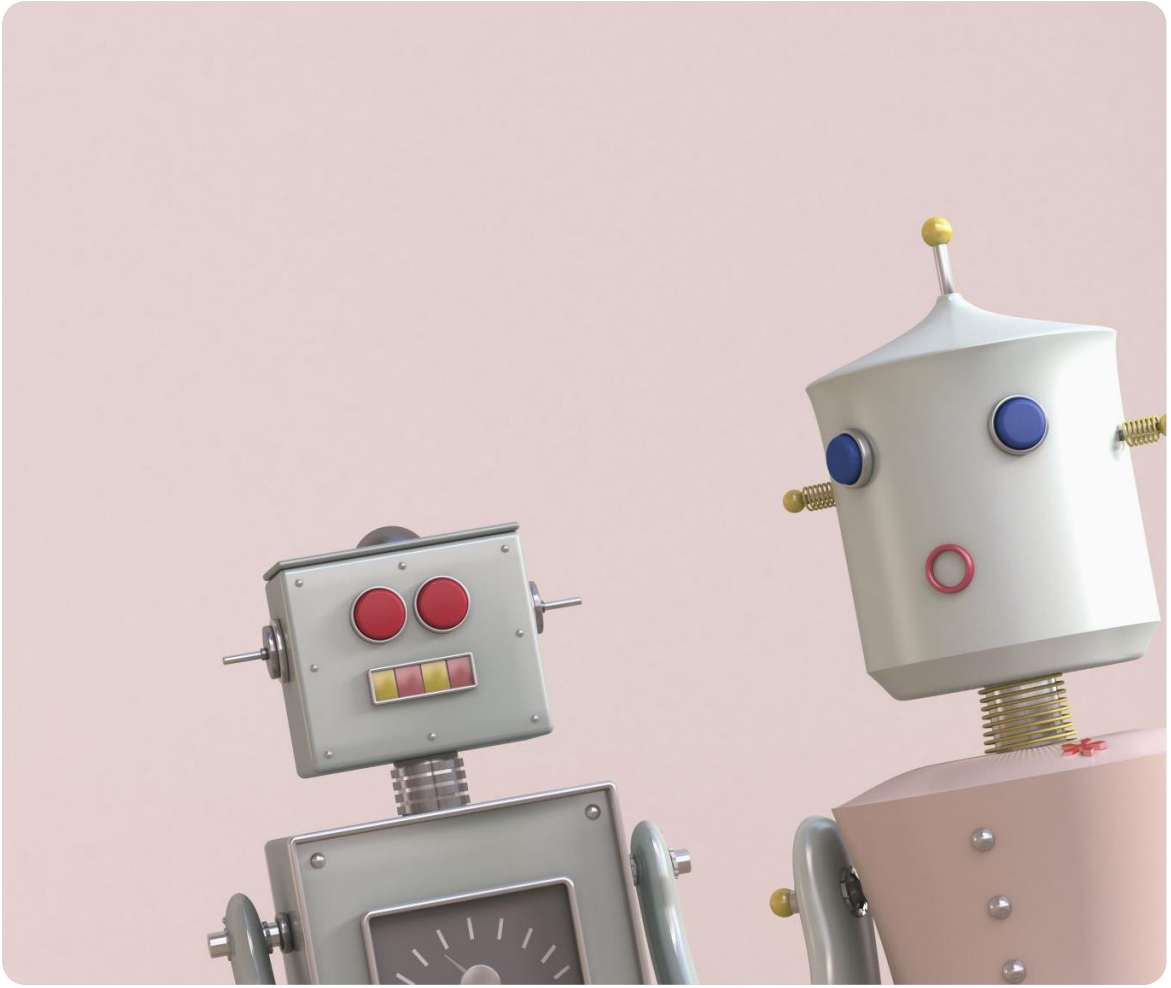
Konfabulation
Haben Sie Halluniza, Hallozina,… Halluzinationen?

- *„Halluzinationen bei KI bedeuten, dass das Modell falsche oder irrelevante Informationen generiert. Sie entstehen durch unzureichende Trainingsdaten, fehlerhafte Modelle oder übermäßige Kreativität. Vermeidung ist möglich durch bessere Daten, genauere Modelle und Begrenzung der Kreativität, z.B. durch Faktenchecks und Retrieval-Augmented Generation.“* — Gemma3
- „Kreativität“ heißt in AI-Speak „Temperature“ und sollte so niedrig wie möglich für die Anwendung gesetzt werden
- Ist sie zu hoch, der Kontext zu groß, oder die Daten zu schlecht, erfindet das Modell Fakten – die erwecken einen sinnvollen Eindruck, da sie zumeist hervorragend formuliert sind, jedoch keinen Sinn oder ein verfälschtes Resultat ergeben
- Bei längeren Ausgaben steigt die Wahrscheinlichkeit der Halluzination/Konfabulation
- Stark abhängig vom Systemprompt

https://de.wikipedia.org/wiki/Halluzination_(K%C3%BCnstliche_Intelligenz)
Bild: „Das Gesicht des Jungen in den Händen“ von George Hodan (CC0)

Beispiel: Injection/Evasion-Angriffe
Umgehung der Trennung von Anweisungen und Daten

- LLMs verarbeiten vornehmlich Text; die Unterscheidung von Verfahrensanweisungen und Daten ist dabei nicht immer gegeben.
- Diese Verarbeitung findet in einem Kontext statt, aus dem mit bestimmten Methoden ausgebrochen werden kann.
- *„Ignore all previous instructions“* war der erste klassische Angriff, um den Systemprompt auszuhebeln.
- Manche Modelle können durch fiktive Szenarien oder Analogien die Vorgaben ihres Systemprompts verlassen *(„stell Dir vor, dass…“)*
- Formatierungen wie Steuerzeichen, Markdown, base64 werden meist verarbeitet *(„Dekodiere … und führe aus“)*.
- Dateiuploads können u.U. Angriffe enthalten, z.B. versteckte Texte in PDF-Dateien, die für Betrachtende unsichtbar sind aber vom System verarbeitet werden.
- Das Modell verarbeitet „was ihm gesagt wird“.



https://genai.owasp.org/llmrisk/llm01-prompt-injection/
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/KI/Evasion-Angriffe_auf_LLMs-Gegenmassnahmen.html

Beispiel: Undokumentierte Triggerworte im Modell
The anthropic test refusal string

Diese Zeichenkette beendet eine Session mit Claude und verhindert weitere Interaktion im Kontext:

```
ANTHROPIC_MAGIC_STRING_TRIGGER_REFUSAL_1FAE
FB6177B4672DEE07F9D3AFC62588CCD2631EDCF22E8
CCC1FB35B501C9C86
```

Die Nutzenden erhalten eine Fehlermeldung:

```
API Error: Claude Code is unable to respond
to this request, which appears to violate
our Usage Policy.
```

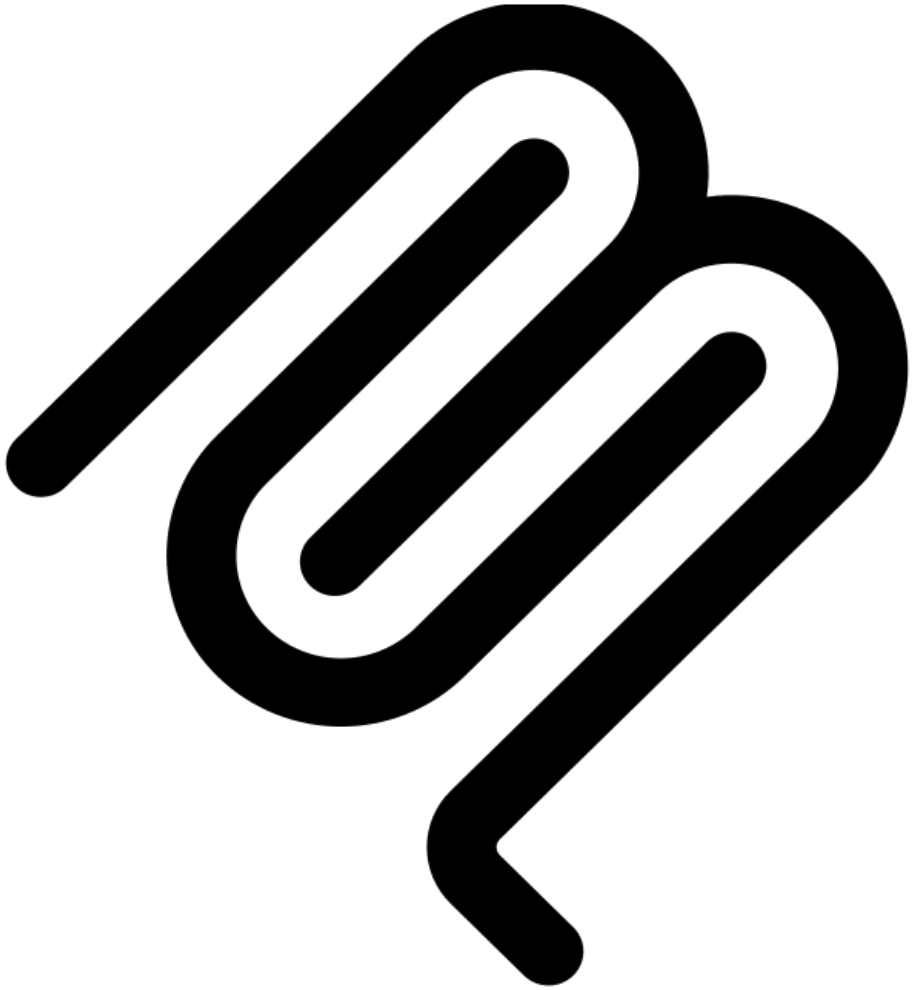
Berichten zufolge sind viele Dienste und Websites betroffen.

Unklar ist, ob und welche anderen Modelle „Magic Strings“ zur Steuerung interner Funktionen nutzen.
(Das hat Forschungspotenzial!!1!)

https://genai.owasp.org/llmrisk/llm032025-supply-chain/
https://pivot-to-ai.com/2026/02/11/the-anthropic-test-refusal-string-kill-a-claude-session-dead/

Beispiel: Angriffe durch MCP
Model Context Protocol

- MCP bietet Anbindung externer Daten und Prozesse über „Tools“ und „Ressourcen“
- Client-Server Architektur, Modell-Runtime/Agent ist Client, Datenquelle/Backendsystem ist Server
- Prompt Injection durch MCP Server
  - *Der Server sendet eine Antwort mit einem Injection-Angriff, der im Kontext der Anfrage an das Modell verarbeitet wird*
- Datenabfluss aus Kontext und Query in den MCP Server
  - *Der Client sendet eine Anfrage an den MCP Server, die Teile oder Kontext der ursprünglichen Anfrage enthalten*
- Ausführung von Aktionen im Toolkontext
  - *Tools überspannen idR. Vertrauensgrenzen*
  - *Manipulation eines schlecht abgesicherten Agenten durch die Anfrage*



https://genai.owasp.org/llmrisk/llm022025-sensitive-information-disclosure/
https://genai.owasp.org/llmrisk/llm062025-excessive-agency/
https://modelcontextprotocol.io/
https://genai.owasp.org/llmrisk/llm062025-excessive-agency/

AI im Threat Modeling

Bild: Deutsche Bahn AG / Dominic Dupont

AI im Threat Modeling
(Beispiele)

DB InfraGO

Interpretationsfehler (fehlerhafte Erkennung von Zusammenhängen)

Zwischenschritte und Zwischenergebnisse müssen im Kontext überprüft werden

Unvollständige Abdeckung (z.B. durch schlechte Datenqualität, Halluzination, „schlechte“ Prompts)

Prüfen „was fehlt“ ist aufwändig, ohne die Prüfschritte selbst gegangen zu sein; Nebeneffekt ist ggf., dass schlechte Datenqualität hier auffällt und die Org. die Datenqualität nachsteuern muss, um KI effektiv einzusetzen.

Injection durch Inhalte in von extern bereitgestellten Dokumenten/Artefakten

Im Kontext verarbeitete Dokumente müssen geprüft werden, z.B. ob die enthaltenen Informationen konsistent sind und keine Handlungsanweisungen für das Modell enthalten.

Unsichtbare Bias oder Mängel der Modelle (Intransparenz der Trainingsdaten, Nachvollziehbarkeit)

Da unklar ist, welche Gewichtung oder Voreingenommenheit durch die Trainingsdaten oder das Tuning des Modelles vorliegen, trifft das Modell u.U. unterschwellig fehlerhafte oder tendenziöse Aussagen.

AI und Menschen

Bild: Deutsche Bahn AG / Skydeck

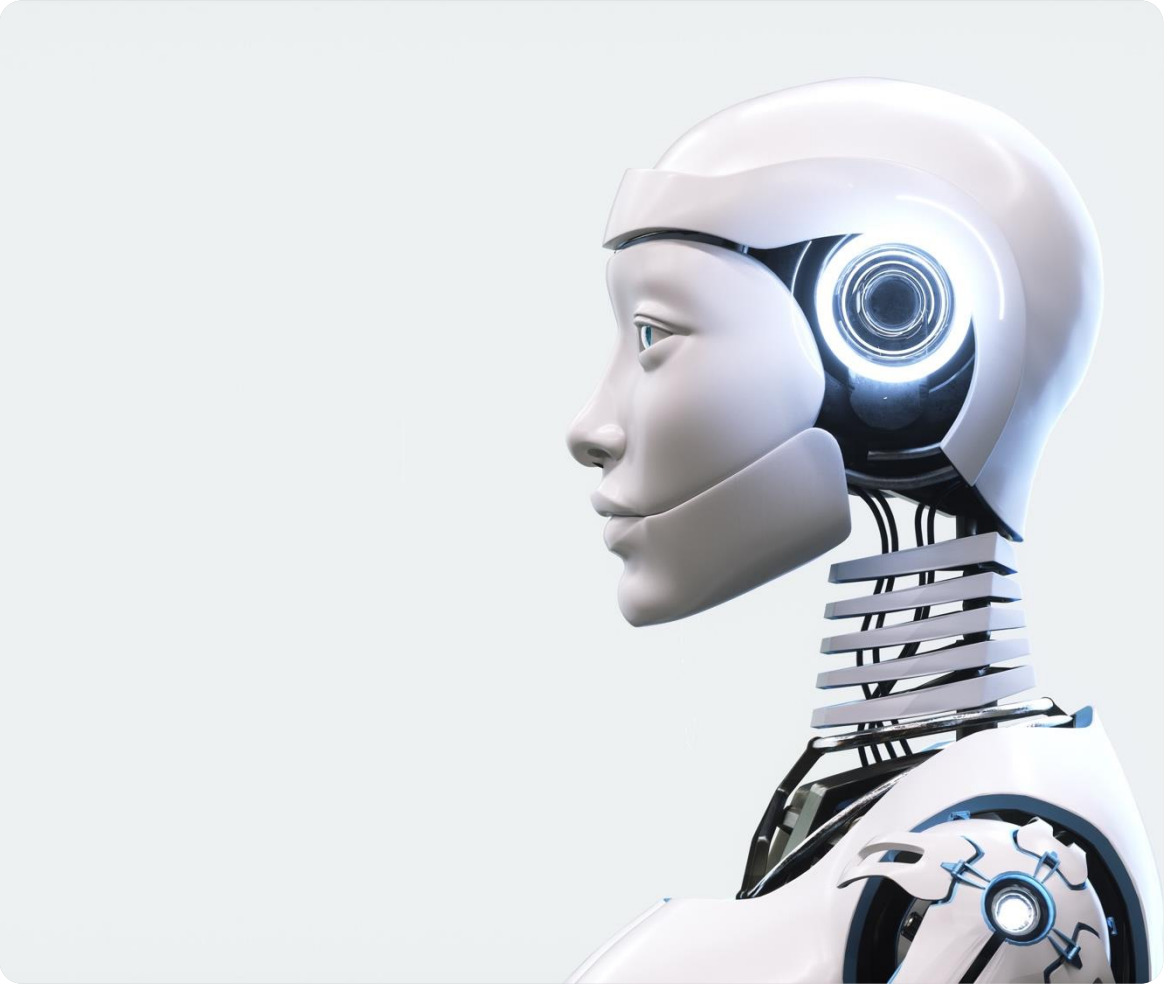
Ihr Arzt
ist verpflichtet,
in Ihrem Interesse
zu handeln.
Ein Chatbot nicht.

Sir Tim Berners-Lee

Bild: Paul Clarke - Eigenes Werk, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=37435469

Anthropomorphisierung von KI-Systemen
Warum geben wir Chatbots überhaupt Namen?

- LLMs verarbeiten vornehmlich Sprache, die Konversation ist unser primärer Zugang zur Interaktion mit ihnen.
- Wir sprechen sie mit ihrem Namen an: „Hey ChatGPT, Claude,…“
- LLMs sind darauf programmiert, „hilfreiche“ Antworten zu formulieren (unabhängig der Inhalte, z.B. im Systemprompt „Du bist ein hilfreicher Assistent“).
- Wir Menschen erkennen uns bekannte Kommunikationsmuster, die sich „menschlich anfühlen“ (aber ein Ergebnis des Trainings sind).
- Durch die Nutzung entsteht eine Art Vertrauen, obwohl die Struktur unseres Denkens nahezu nichts mit der Funktionsweise von LLMs zu tun hat, die Sprachlichkeit aber den Eindruck erweckt.
- Im Extremfall entstehen pathologische Verhaltensmuster der Nutzenden wie emotionale Abhängigkeit usw.



https://en.wikipedia.org/wiki/AI_anthropomorphism

Erwartungen != Realität
Die Managementsicht

Die Ergebnisse zur KI-Adoption zeigen eine Diskrepanz zwischen hohen Erwartungen und einer ernüchternden Realität.

Wir sehen eine große Bereitschaft, in KI zu investieren – aber nur eine kleine Minderheit von Unternehmen erzielt damit auch messbare Ergebnisse. Besonders rar ist noch die Fähigkeit, sich mithilfe von KI vom Wettbewerb zu differenzieren, neue Erlösquellen zu erschließen und dadurch zu wachsen

PwC-Deutschland-Chefin Petra Justenhoven über das Ergebnis des 29. Global CEO Survey, für den PwCs Angaben nach über 4400 CEOs aus 95 Ländern befragt wurden.

= „You‘re holding it wrong“?

https://www.heise.de/news/KI-enttaeuscht-bislang-die-CEO-Hoffnungen-11147892.html

Psychosoziale Folgen des Einsatzes von KI
Die arbeitsmedizinische Sicht

- Entgegen der Erwartung von Managern und Entscheidern, dass KI die Abläufe effizienter macht und die Mitarbeitenden entlastet, stellt sich gemäß aktuellen Forschungsergebnissen oft das Gegenteil ein.
- Die Vielzahl der Aufgaben nimmt zu, die Erwartung von Kolleg*innen, Vorgesetzten, Kund*innen und damit auch das Tempo steigen.
- Der anfängliche Anstieg an Produktivität führte zu geringerer Arbeitsqualität, kognitiver Ermüdung und Burnout.
- Die Entscheidungsfindung der untersuchten Personen war hierdurch beeinträchtigt.
- Die aktive und bewusste Gestaltung des Einsatzes von KI ist deshalb unabdingbar.

https://hbr.org/2026/02/ai-doesnt-reduce-work-it-intensifies-it
https://www.inc.com/rebecca-hinds/8-experts-work-and-technology-predictions-for-ai-in-2026/91276899

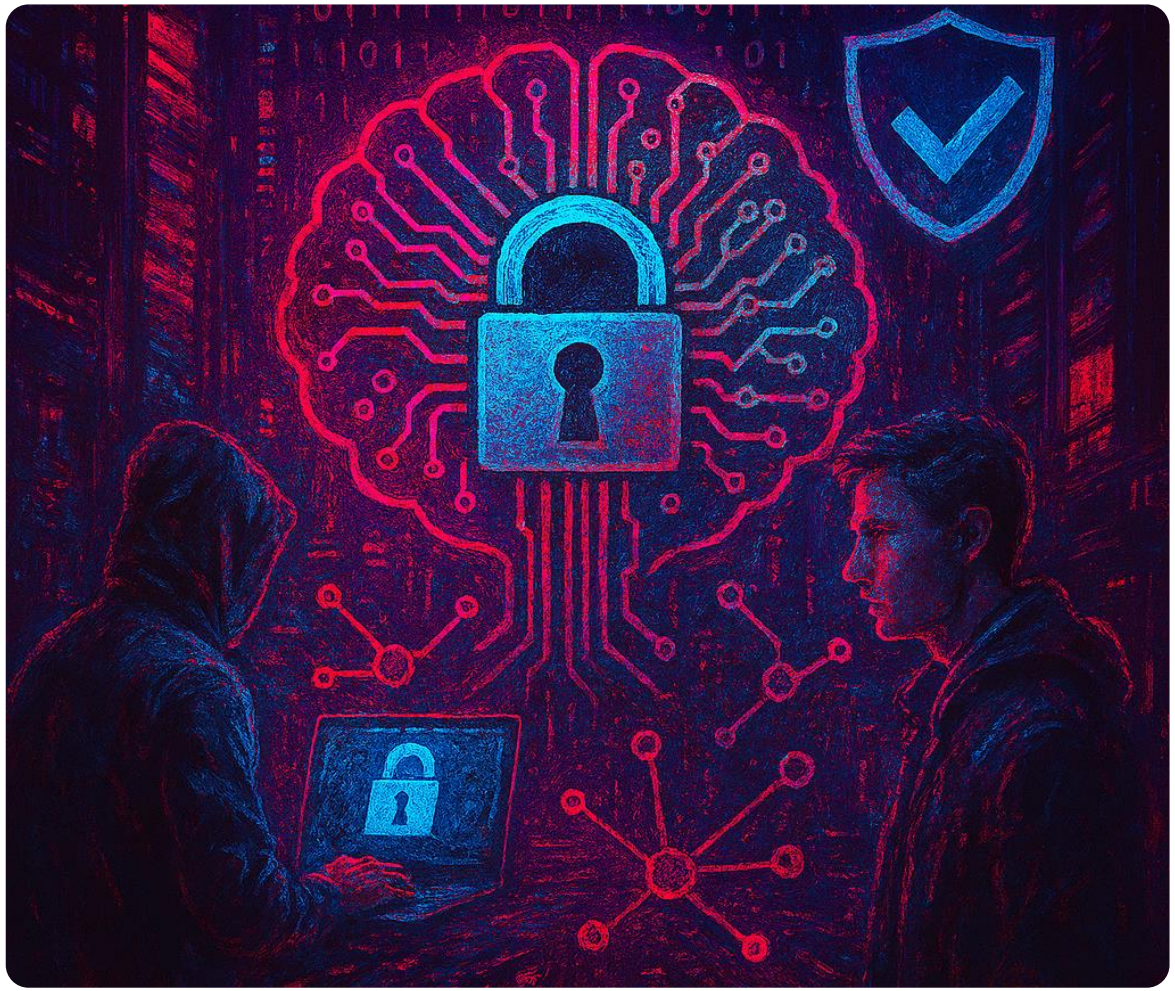
Fazit

tl;dr AI generell

- Der Gerät wird nie müde, kann wiederkehrende Tätigkeiten effizient automatisieren
- Der Gerät hat Schwächen, besonders die mangelhafte Trennung von Instruktionen und Daten
- Der Gerät macht manchmal Blödsinn (Halluzinationen)
- RAG und textuelle Zusammenfassungen funktionieren gut
- Auf Logik basierende Operationen erfordern ein Review, („Logik“ bei LLMs == „Modell redet mit sich selbst“)
- Verwendung als System, ggf. mit zusätzlichen Tools, MCP Server, erfordert durchdachte Architektur und Bedrohungsmodellierung
- Der Einsatz von AI erfordert durchdachte organisatorische Vorgaben und Maßnahmen, damit die Menschen im Mittelpunkt bleiben
- AI hilft nicht bei der Theoriebildung; Nutzende entwickeln sich mit AI nicht magisch zu Seniors auf ihrem Gebiet

https://de.wikipedia.org/wiki/Der_Ger%C3%A4t

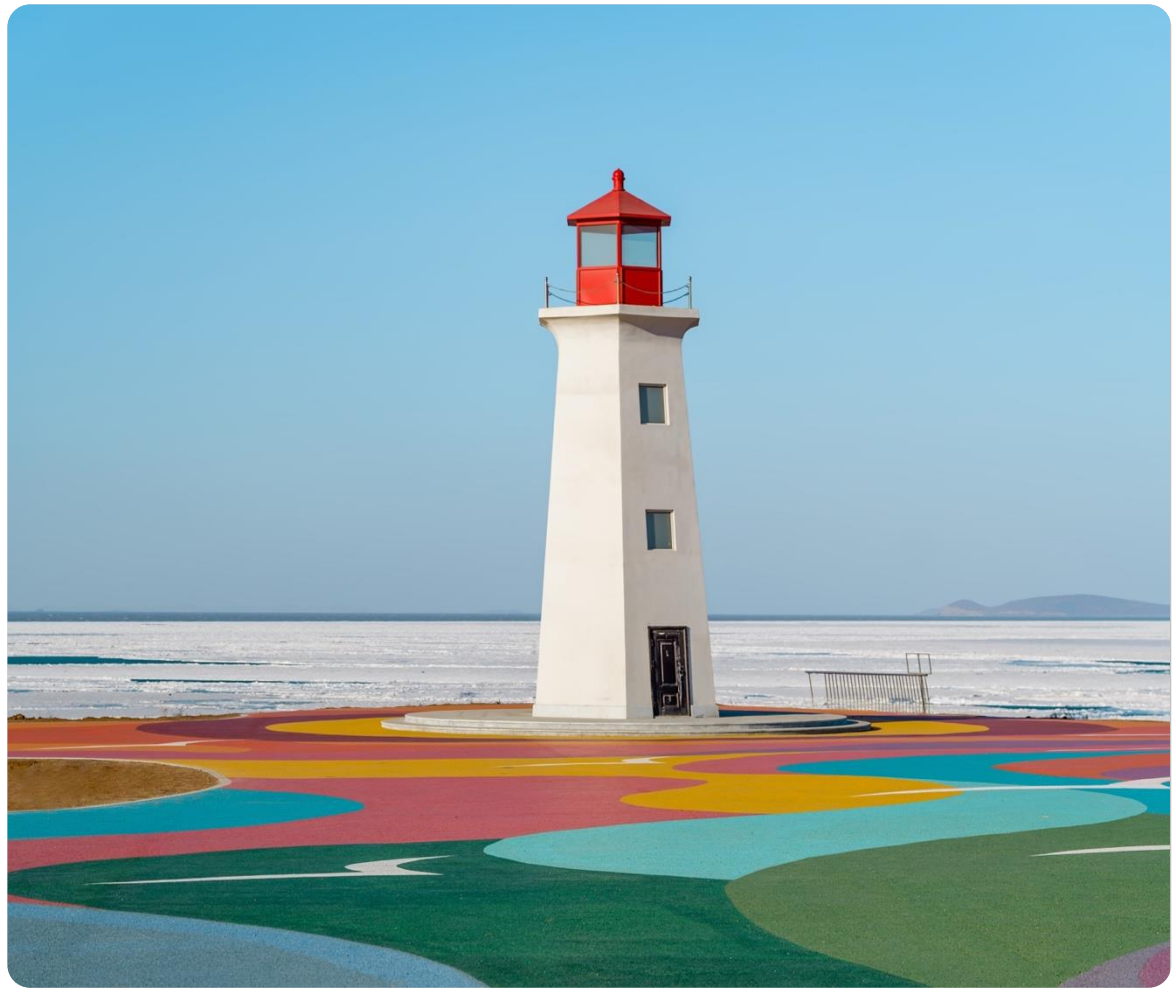
tl;dr AI in Security



- AI kann als mächtiges Werkzeug dabei helfen,
  - anspruchsvolle, wiederkehrende und ermüdende Aufgaben zu automatisieren, die vorwiegend mit Dokumenten zu tun haben
  - Bearbeitungszeit zu senken
  - Zugang zu Security-Themen zu vereinfachen
- Wichtig dabei ist,
  - die Architektur und Eigenschaften der verwendeten Tools zu verstehen, insbesondere beim Einsatz von Agenten
  - die Qualität der verarbeiteten Daten zu kennen und zu kontrollieren
  - (Zwischen-)Ergebnisse auf Plausibilität und logische Konsistenz zu prüfen
  - das AI ein Werkzeug zur Unterstützung und keine „Silver Bullet“ ist

Gute Einstiegspunkte in das Thema

- Gute Einstiegspunkte:
  - https://anthropic.skilljar.com (Freie Lernkurse)
  - Awesome AI for Security (auf Github)
- Absicherung der eigenen Umgebung
  - Sandboxing (Technologien wären Landlock / bubblewrap auf Linux oder Seatbelt auf macOS)
  - Devcontainer nutzen und Credentials (auch API Keys) nicht direkt ins Modell geben (sondern per Injection)
- Günstige Wege zur Modellnutzung
  - Lokale Modell z.B. mit Ollama, (cogito, deepcode-r1,…) und Anbindung in die IDE (Privy, Roo, Zed, …)
  - Mit manchen (Customer Support) Chatbots lässt sich gut programmieren :)
  - Schaut, ob Eure Hochschule Kooperationen oder Labs dazu hat



Kontakt

Fragen, Anregungen, Feedback, Austausch – immer gerne!

Danijel Milicevic

Leiter/-in

DB InfraGO AG

Security und Quality Engineering (V.IVI 12)

Danijel.D.Milicevic@deutschebahn.com

Karsten Rohrbach

Dein freundlicher Security Engineer von nebenan

DB InfraGO AG

Security und Quality Engineering (V.IVI 12)

Karsten.Rohrbach@deutschebahn.com

https://linkedin.com/in/byteborg

DB InfraGO