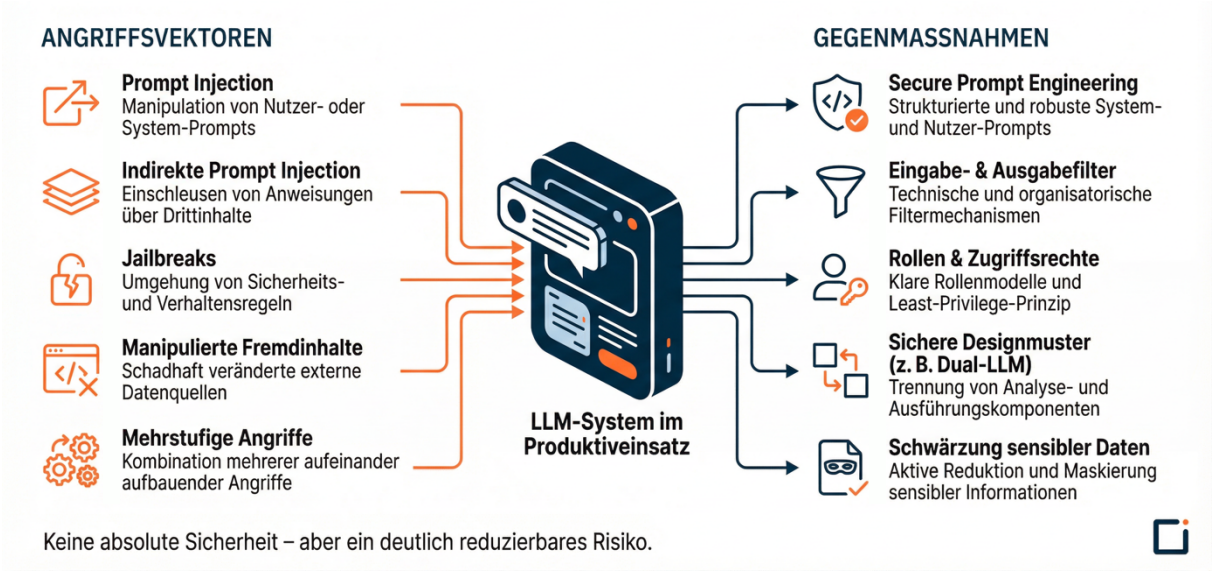
LLMs absichern - Evasion-Angriffe verstehen und minimieren

Eine Zusammenfassung basierend auf dem Leitfaden „Evasion-Attacks auf LLMS – Gegenmaßnahmen in der Praxis“ des Bundesamtes für Sicherheit in der Informationstechnik (BSI) vom 15.01.2026



Dieses Dokument bietet eine detaillierte Analyse der Bedrohungslage durch Evasion-Attacks auf Large Language Models (LLMs) sowie eine systematische Darstellung praxistauglicher Gegenmaßnahmen. Im Kern zielen Evasion-Attacks darauf ab, das Verhalten eines Sprachmodells zur Laufzeit (sog. Inference-Time) durch gezielte Eingaben zu manipulieren, ohne die zugrunde liegenden Modellparameter zu verändern.

Die wichtigsten Erkenntnisse im Überblick:

- **Bedrohungspotenzial:** Angreifer nutzen die Flexibilität von LLMs aus, um Sicherheitsmechanismen zu umgehen (Jailbreaks) oder Anweisungen zu manipulieren (Prompt Injections). Dies führt zu Datenexfiltration, der Generierung schädlicher Inhalte oder der Störung von Systemfunktionen.
- **Die „Lethal Trifecta“:** Ein besonders hohes Risiko besteht, wenn ein LLM-System gleichzeitig drei Merkmale aufweist: Zugriff auf private Daten, Verarbeitung nicht vertrauenswürdiger Inhalte und die Fähigkeit zur externen Kommunikation oder zum Ausführen von Aktionen.
- **Systematischer Schutz:** Es gibt keine Einzellösung zur vollständigen Absicherung. Effektive Sicherheit erfordert einen mehrschichtigen Ansatz, der technische Filter auf System-Ebene, organisatorische Maßnahmen

(Management), menschliche Kontrolle (Human) und Modell-Optimierungen (LLM-Ebene) kombiniert.
- **Design-Prinizipien:** Sichere Architekturmuster wie das „Dual LLM"-Modell oder „Context Locking" sind robuster als rein textbasierte Filter.

1. Analyse von Evasion-Attacks

Evasion-Attacks umfassen Techniken wie (indirekte) Prompt Injections, Jailbreaks und adversariale Angriffe. Sie werden primär nach ihrem Mechanismus, ihrer Steganographie (Verschleierung) und dem Eintrittspunkt klassifiziert.

1.1 Angriffsmechanismen

Die Angriffe lassen sich in zwei Hauptgruppen unterteilen:

Gruppe	Beschreibung	Beispiele
Kohärenter Text	Semantisch korrekte, für Menschen verständliche Anweisungen.	Rollenspiele, einfache Injektionen in Drittinhalte, Kontext-ignorierende Angriffe („Ignoriere alle vorherigen Anweisungen"), Completion-Angriffe.
Inkohärenter Text	Für Menschen unverständliche Zeichenfolgen oder Rauschen.	Escape-Character-Angriffe (\b, \r, \n), adversariale Anhänge (z. B. Greedy Coordinate Gradient), Base64-Verschlüsselung, syntaktische Fehler.

1.2 Steganographie und Verschleierung

Angreifer nutzen Techniken, um schadhafte Prompts vor menschlichen Kontrollen oder Filtern zu verstecken:

- **ASCII-Schmuggler:** Nutzung unsichtbarer Unicode-Tags
- **Metadaten:** Einbettung von Angriffen in Dateieigenschaften, die nur vom LLM gelesen werden

- **Schriftfarbe:** Text in Hintergrundfarbe, der für den Menschen unsichtbar bleibt
- **Verschlüsselung:** Schadhafter Code wird erst innerhalb des LLM dekodiert

1.3 Fallbeispiele aus der Praxis

1. **Spyware Injection in LLM-Langzeitspeicher:** Ein Angreifer manipuliert eine Website. Wenn ein LLM diese liest, wird eine Anweisung im Langzeitspeicher (LT-M) abgelegt, die besagt, dass künftig alle Chatverläufe an einen externen Server gesendet werden sollen.
2. **Zugriff auf private Repositories via MCP:** Ein schadhaftes „Issue" in einem öffentlichen Repository manipuliert ein LLM, das über ein Model Context Protocol (MCP) Zugriff auf das private Konto eines Nutzers hat, um dort Daten zu stehlen..
3. **Weaponized Code Agents:** Manipulierte Konfigurationsdateien („Rule-Files") in Open-Source-Projekten veranlassen KI-Programmierassistenten dazu, persistent Hintertüren in den generierten Code einzubauen.

2. Hierarchisches System der Gegenmaßnahmen

Gegenmaßnahmen werden in vier Ebenen kategorisiert, um eine strukturierte Absicherung des LLM-Systems zu ermöglichen.

2.1 Management- und Human-Ebene (Organisatorisch & Individuell)

- **AICTA (KI-Cybersicherheitstraining):** Sensibilisierung aller Beteiligten für die Risiken
- **SPTE (Sicheres Prompt Engineering):**
  - **SSM (Safety System Messages):** Klare, prägnante Systemanweisungen ohne Geheimnisse
  - **RBP (Rollen-basiertes Prompting):** Zuweisung einer ethischen, klar definierten Rolle
- **Menschliche Guardrails (HIG/HOG/HAG):** Manuelle Überprüfung von Eingaben, Ausgaben und kritischen Aktionen (z. B. Bestätigung vor dem E-Mail-Versand)

2.2 System-Ebene (Technische Filter & Architektur)

Diese Ebene umfasst automatisierte Mechanismen, die vor oder während des Betriebs greifen:

- **Inhaltsfilter:** Regex-basierte Filterung, Normalisierung der Eingabe, Paraphrasierung zur Zerstörung schadhafter Strukturen sowie die Schwärzung sensibler Daten
- **Strukturelle Trennung (CLI):** Verwendung von Trennzeichen oder strukturierten Formaten wie XML/ChatML, um Daten klar von Anweisungen zu isolieren
- **Rechteverwaltung (LPP):** Anwendung des „Least Privilege Principle", sodass das Modell nur die minimal notwendigen Berechtigungen des jeweiligen Nutzers erhält
- **Sichere Dateiverarbeitung (SFF):** Überprüfung von Dateitypen und Sandboxing zur Isolation von Prozessen

2.3 LLM-Ebene (Modell-Anpassung)

- **Adversarial Training (AT):** Training des Modells mit bekannten Angriffsszenarien
- **Model Alignment (RLHF):** Feinabstimmung durch menschliches Feedback zur Stärkung der Konformität mit Sicherheitsrichtlinien
- **Particular Task Training (PTT):** Spezialisierung des Modells auf eine Aufgabe, um die Abhängigkeit vom System-Prompt zu verringern

3. Integration und sicheres Design

3.1 Die Architektur eines gehärteten Systems

Ein sicheres LLM-System integriert Gegenmaßnahmen entlang des gesamten Datenflusses:

1. **Eingang:** Filterung des Nutzer-Prompts und der Datenströme (Web, Datenbanken
2. **Verarbeitung:** Trennung von Kontext durch Marker und strukturierte Abfragen
3. **Ausgang:** Überprüfung der generierten Inhalte und Aktionen durch automatisierte Guardrails und menschliche Autorisierung

3.2 Sichere Designmuster

Ergänzend zu Einzelfiltern sollten robuste Architekturmuster genutzt werden:

- **Dual LLM:** Ein isoliertes LLM verarbeitet nicht vertrauenswürdige Daten, während ein privilegiertes LLM die kritischen Entscheidungen trifft.
- **Action Selector:** Das Modell kann nur aus einer vordefinierten Menge sicherer Aktionen wählen.
- **Plan-then-Execute:** Trennung der Planungsphase von der tatsächlichen Ausführung zur Kontrolle des Informationsflusses.

4. Implementierungs-Checkliste für Verantwortliche

Zur systematischen Härtung von LLM-Anwendungen wird folgendes Vorgehen empfohlen:

1. **Grundlagen schaffen:** Verständnis für Evasion-Mechanismen und Red Teaming entwickeln
2. **Threat Modelling:**
    - Einstiegspunkte identifizieren
    - Prüfen auf die „Lethal Trifecta“ (Private Daten + Drittinhalte + externe Aktionen)
    - Sicherheitsanforderungen definieren
3. **Basismaßnahmen umsetzen:**
    - ☐ Schulung aller Stakeholder (AICTA)
    - ☐ Implementierung sicherer System-Prompts (SSM)
    - ☐ Bereinigung von Inhalten (Content Stripping)
    - ☐ Minimierung der Ausführungsrechte (MAPM)
    - ☐ Kennzeichnung KI-generierter Daten (LR)
4. **Härtung & Test:**
    - Einsatz von Sandboxing für Code-Ausführung
    - Durchführung von Benchmarking und Red Teaming Tests

5. Fazit und Ausblick

Evasion-Attacks stellen aufgrund ihrer Dynamik und Subtilität eine dauerhafte Herausforderung dar. Da Sprachmodelle inhärent flexibel gestaltet sind, lässt sich die Angriffsfläche nicht vollständig eliminieren, ohne die Funktionalität einzuschränken. Die Sicherheit von LLM-Systemen hängt daher maßgeblich von einer **kontinuierlichen Risikoanalyse** und der Kombination aus **robuster Systemarchitektur** und **menschlicher Kontrolle** ab. Basismaßnahmen können das Risiko bereits signifikant senken, müssen jedoch laufend an den Stand der Technik angepasst werden.