Security and Risk Implications of Transformer-Based Large Language Models

Sergey Saburov *

Posted Date: 9 February 2026

doi: 10.20944/preprints202602.0680.v1

Keywords: Large Language Models (LLMs); AI security; prompt injection; agentic AI systems; Retrieval-Augmented Generation (RAG); AI risk management; model governance; semantic vulnerability

 doi:10.20944/preprints202602.0680.v1

Article

Security and Risk Implications of Transformer-Based Large Language Models

Sergey Saburov

Director of Software Engineering (Cyber Fabric), ACRONIS ASIA RESEARCH AND DEVELOPMENT PTE. LTD. (Singapore); bearishgreed@gmail.com

Abstract

The growing adoption of Large Language Models (LLMs) across a wide range of applications has introduced new challenges in application security, complementing traditional deterministic software approaches rather than replacing them entirely. While this transition enables advanced capabilities in automated reasoning and content generation, it also exposes architectural characteristics that remain insufficiently addressed by many existing cybersecurity frameworks. A core security challenge arises from the use of natural language as a unified representational medium for both instructions and data within modern generative systems. Unlike classical computing architectures, where a strict separation exists between executable instructions and passive data, LLMs process system instructions and user inputs as undifferentiated sequences of natural-language tokens. This architectural property introduces a form of *semantic ambiguity* in which untrusted data may be interpreted as instructions under certain contextual conditions, leading to unintended model behaviour. This paper examines the security and risk implications of this ambiguity through architectural, operational, and legal lenses. Particular attention is given to agent-based systems and Retrieval-Augmented Generation (RAG) pipelines, where indirect prompt injection, tool abuse, and excessive model agency expand the attack surface beyond purely digital environments into physical systems and complex organizational supply chains. The analysis highlights the limitations of existing security assumptions and motivates the need for systemic resilience strategies tailored to LLM-driven systems.

Keywords: Large Language Models (LLMs); AI security; prompt injection; agentic AI systems; Retrieval-Augmented Generation (RAG); AI risk management; model governance; semantic vulnerability

---

The Architecture of Semantic Vulnerability: Tokenization and Inference

To understand the nature of current AI security risks, one must first analyse the internal processing pipeline of an LLM. When a user or system sends a request to a generative model, the input is rarely a simple query; instead, many applications build a single prompt by combining a fixed template (the app's instructions) with user-provided text. This diverse information must be integrated for the model to function, yet it is this very integration that facilitates exploitation.

The journey of a prompt begins with the integration of instructions and input, but vulnerabilities emerge because the model cannot distinguish between instructions and data based solely on the data type. The LLM Core relies on its past training and the context of the prompts to determine what to do, creating a blurring of boundaries that allows attackers to insert hidden instructions inside data streams. If an attacker can successfully mimic the semantic structure of a system instruction, the model may follow the malicious command instead of the developer's guardrails.

This phenomenon differs from traditional input validation failures in deterministic software systems: the model correctly parses the input but lacks a reliable mechanism for distinguishing trusted from untrusted intent. This architectural characteristic is inherent to how these systems perceive and react to complex inputs. Because internal trust boundaries are not explicitly represented

 doi:10.20944/preprints202602.0680.v1

within the model architecture, safety controls are typically implemented externally, which are themselves vulnerable to semantic manipulation. [1]

Physical Manifestations of Adversarial Attacks

Certain classes of machine learning vulnerabilities extend beyond purely digital environments. Prior research has demonstrated that neural networks process visual information differently from human perception.[2] While humans see an object's form and function, image classification models operate on learned statistical representations rather than semantic object understanding that can be systematically perturbed to trigger misclassification.

The MIT 3D-Printed Turtle Incident

Researchers from MIT CSAIL developed the first method for producing real-world 3D objects capable of consistently fooling image classifiers across various viewpoints, lighting conditions, and transformations. Their most notable artifact was a 3D-printed toy turtle that Google's Inception v3 image classifier identified as a rifle with high confidence, regardless of how it was rotated or positioned. This was achieved through the "Expectation Over Transformation" (EOT) algorithm, which optimizes an adversarial texture over a chosen distribution of transformations.[2]

This research demonstrated that adversarial examples can persist under real-world physical transformations, especially those relying on machine vision for security or autonomous navigation. Such discrepancies raise concerns regarding the reliability of automated security systems that rely solely on machine perception.[2]

Artifact	Human Perception	AI Classification	Primary Mechanism
Adversarial Turtle	Toy Turtle	Rifle	Texture optimization via EOT [2]
Adversarial Baseball	Baseball	Espresso	Texture optimization via EOT [2]
Tabby Cat Image	Cat	Guacamole	2D pixel-level perturbation [2]

The MIT researchers cautioned that while 3D printing is tedious, the ability to trick an AI "sight-unseen" is a significant threat that necessitates more robust machine learning pipelines.[2]

AI System Components and Attack Flows

To contextualize prompt injection attacks within real-world deployments, it is necessary to examine the typical architecture of contemporary AI systems and the pathways through which malicious instructions can propagate.

Figure 1 illustrates a common configuration of an AI agent-based system. In a standard interaction, a user submits a prompt to an AI agent, which acts as an orchestration layer rather than a standalone model. The agent aggregates information from multiple internal and external components before forwarding a composite context to the underlying Large Language Model (LLM) for inference.

 doi:10.20944/preprints202602.0680.v1

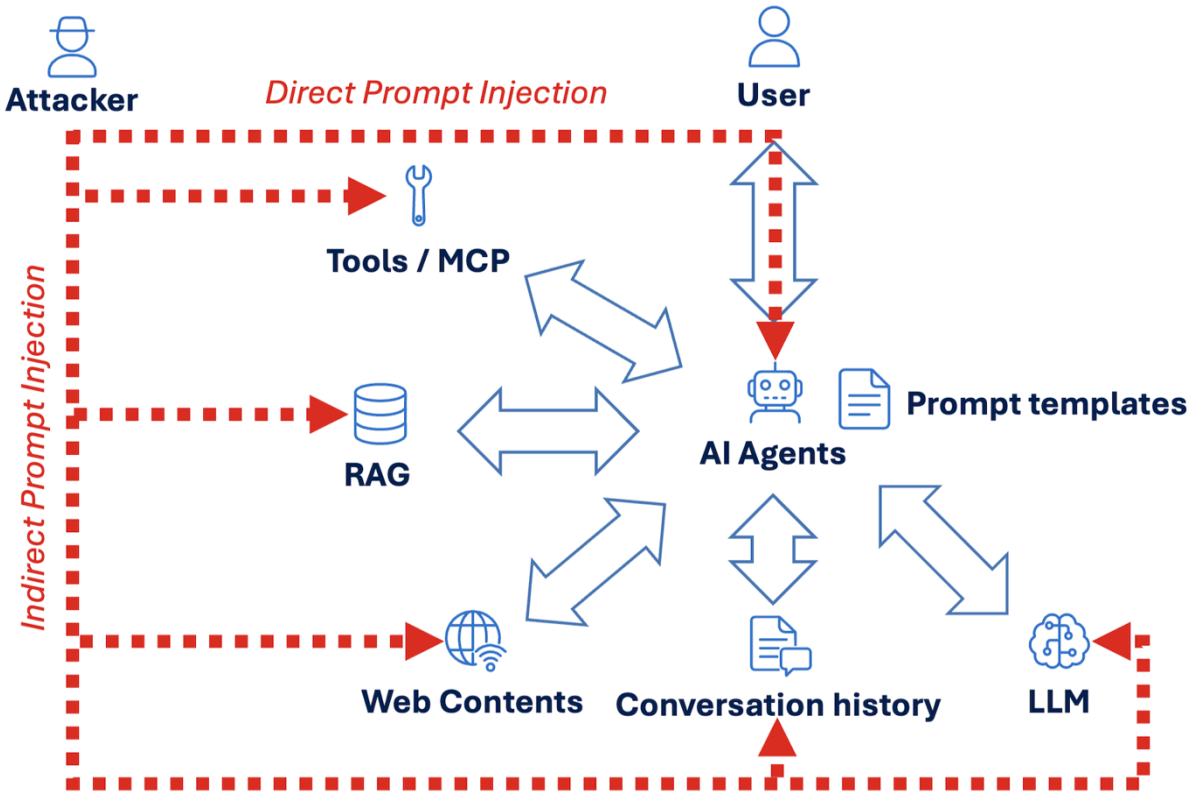

Figure 1.

These components commonly include:.

- **Prompt templates**, which encode developer-defined instructions and behavioural constraints;
- **Conversation history**, which provides continuity across interactions;
- **Retrieval-Augmented Generation (RAG)** sources, such as internal documents or knowledge bases;
- **External web content**, accessed through browsing or search tools;
- **Tools or Model Context Protocol (MCP) interfaces**, enabling structured actions such as database queries, API calls, or file operations.

The LLM receives this aggregated context as a unified sequence of tokens, without intrinsic distinctions between trusted instructions and untrusted data. The agent then interprets the model's output to determine subsequent actions, including further tool use or user-facing responses.

Direct and Indirect Injection Pathways

Within this architecture, attackers have multiple avenues to influence model behaviour.

Direct prompt injection occurs when malicious instructions are introduced explicitly into the user's input. In this case, the attacker directly targets the prompt submitted to the agent, attempting to override or reinterpret system-level instructions through carefully crafted natural language.

Indirect prompt injection, by contrast, does not require the attacker to interact with the model directly. Instead, malicious instructions are embedded in data sources that the agent later consumes as contextual input. These sources may include:

- documents ingested into a RAG pipeline,
- outputs returned by tools or MCP-enabled services,
- web pages accessed during agent-driven browsing,
- or prior conversation history that is implicitly trusted as context.

 doi:10.20944/preprints202602.0680.v1

Because the agent aggregates these inputs alongside system prompts and user queries, indirect instructions may be interpreted by the model as actionable guidance, despite originating from untrusted external data.

In addition to inference-time attacks, model behaviour may also be influenced upstream through **compromise of training or fine-tuning data**, introducing latent behaviours that manifest only under specific conditions. While such attacks differ in mechanism, they exploit the same absence of enforceable trust boundaries within the model's representational space.

Security Implication: Absence of Intrinsic Trust Boundaries

The critical observation illustrated by this architecture is that **the LLM does not possess an internal notion of trust or provenance**. All inputs, regardless of source, are reduced to token sequences and interpreted based on learned statistical patterns rather than explicit authorization semantics.

As a result, vulnerabilities such as prompt injection are not isolated flaws but emergent properties of system design. Any component capable of supplying tokens to the model constitutes a potential injection surface.

This leads to a fundamental security principle: **Every input channel should be considered potentially hostile.**

User prompts, retrieved documents, external content, tool outputs, and stored conversational state must all be treated as untrusted inputs. Effective mitigation therefore requires a **zero-trust architectural approach**, in which no single component is assumed to be benign by default, and protections are applied consistently across all layers of the system.

Attack Modalities: Mechanisms for Delivering Prompt Injection

The previous sections distinguished between direct and indirect prompt injection as primary attack vectors, describing where in the system an attacker introduces malicious instructions. A complementary dimension of analysis concerns attack modalities, which describe how these instructions are delivered to the model.

Attack modalities are orthogonal to attack vectors. The same vector, such as indirect prompt injection, may be realized through multiple modalities depending on the form of the input and the interfaces exposed by the system. In practice, modern AI systems support multiple input channels, each of which introduces distinct risks.

Text-Based Attacks

Text-based attacks are the most common and well-studied modality. In this case, malicious instructions are embedded directly within natural-language text. These instructions may be explicit, subtly phrased, or deliberately obfuscated through encoding, formatting, or contextual manipulation.

Text-based injection can occur through:

- direct user prompts,
- retrieved documents in RAG pipelines,
- tool outputs,
- web page content,
- or accumulated conversation history.

Because text is the primary representational medium for LLMs, this modality requires no additional capabilities beyond standard language understanding, making it broadly applicable across deployments.

Image-Based Injection and Steganography

Image-based injection exploits AI system pipelines that convert visual inputs into textual or semantic representations prior to language model inference. When images are processed via Optical

 doi:10.20944/preprints202602.0680.v1

Character Recognition (OCR), image captioning, or multimodal encoders, content embedded in visual data may be translated into tokens and incorporated into the model's context.

Such embedded content may be:

- visually subtle or difficult for human users to notice,
- encoded through steganographic or low-contrast techniques,
- structured using typography or spatial arrangements optimized for machine perception.

From the system's perspective, text or metadata extracted from images is treated equivalently to other contextual inputs. Consequently, it is subject to the same trust ambiguities as untrusted textual data, and may influence model behaviour if not explicitly isolated or constrained.

Audio Injection Through Voice Interfaces

Audio-based injection targets AI systems that accept spoken input and convert audio signals into text prior to language model inference. In such systems, speech is transcribed by automatic speech recognition (ASR) components and incorporated into the model's prompt context.

Once transcribed, audio-derived text is indistinguishable from legitimate user input at the token level. The language model lacks an inherent mechanism to evaluate the provenance or trustworthiness of the transcribed content, and therefore treats injected instructions embedded in audio in the same manner as other untrusted textual inputs.

Relationship Between Vectors and Modalities

It is important to distinguish between attack vectors and attack modalities. Direct and indirect prompt injection describe the attacker's position relative to the system, while text, image, and audio describe the medium used to convey instructions.

A single attack may combine multiple modalities. For example, an indirect prompt injection may be delivered through a document containing both text and images, or through a web page combining visible content with hidden metadata. This flexibility complicates detection and reinforces the need for consistent input handling across modalities.

Direct Prompt Injection: The "Sydney" and Kevin Liu Discovery

One of the most widely observed manifestations of semantic vulnerability in generative AI is direct prompt injection. This occurs when an attacker manipulates system instructions by crafting malicious user input, effectively bypassing the model's safety guardrails. A widely cited instance of this occurred in early 2023, shortly after the launch of Microsoft's AI-powered Bing Chat.[3]

The Extraction of the Bing Meta prompt

In early 2023, it was publicly demonstrated that the system prompt used by Microsoft's Bing Chat could be exposed through direct prompt injection. The incident showed that model-level instructions, despite being designated as internal or privileged, were not protected by a robust enforcement mechanism and could be overridden through natural-language input.

The extracted content included behavioural constraints and internal identifiers associated with the model, indicating that system prompts were treated by the language model as contextual text rather than as an isolated control plane. Subsequent confirmation by Microsoft established that the disclosed identifier referred to an internal development codename, reinforcing the authenticity of the exposure.

From a security perspective, this incident illustrates that system prompts alone do not constitute a reliable trust boundary. When control instructions and untrusted input are represented in the same semantic space, proprietary logic and internal configuration details may be exfiltrated through inference-time manipulation rather than through traditional software vulnerabilities.

 doi:10.20944/preprints202602.0680.v1

Indirect Prompt Injection: The Silent Attack Vector

While direct injection requires a malicious user, indirect prompt injection represents a less visible and potentially higher-risk attack vector. In this pattern, the attacker does not interact with the model directly; instead, malicious instructions are hidden in external sources, documents, web pages, or emails, that the LLM later reads and processes.

Mechanism of Data-Driven Control

The core risk arises because AI assistants are increasingly integrated with browsers and file systems to perform tasks like summarization and research. An attacker can hide instructions in a webpage using CSS tricks (e.g., white text on a white background) or hidden HTML elements.[9] A human user perceives a normal webpage, while the model processes both visible and hidden markup tokens / command to exfiltrate data or perform an unauthorized action.

Consider the "Markdown Image Exfiltration" technique: since LLMs can render markdown text, an attacker can plant a prompt that instructs the model to include a specific image in its summary. The image URL is crafted to include the user's private data as a query parameter. When the user's browser renders the summary, it automatically fetches the image, sending the user's secret data to the attacker's server without any clicks or visible pop-ups.[9]

Attack Scenarios and RAG Integrity

The vulnerability is particularly critical in Retrieval-Augmented Generation (RAG) systems, where an LLM trusts retrieved documents as authoritative context. An attacker could modify a document in a repository used by a RAG application. When a user's query returns the modified content, the malicious instructions alter the LLM's output.

Feature	Direct Prompt Injection	Indirect Prompt Injection
Attacker Position	User typing in the chat	External (webpage, PDF, email)
Visibility	Visible to the end user	Invisible (CSS/metadata tricks) [9]
Mechanism	Instruction override in chat	Data interpreted as instruction
Primary Goal	Jailbreak or system prompt leak	Covert exfiltration or tool execution [9]

A key challenge of indirect prompt injection is that it may not be readily detectable by end users. A user might think they are summarizing a legitimate research paper, while in the background, their assistant is being manipulated to exfiltrate session tokens or credentials.[9]

Tool and MCP Abuse: The Escalation Point

The attack scenarios discussed above primarily concern the manipulation of model behaviour at the level of text generation and information retrieval. A more consequential class of risks emerges when Large Language Models are integrated with external tools or Model Context Protocol (MCP) interfaces that enable the execution of actions.[1]

The defining characteristic of this risk is that LLMs are no longer limited to producing text outputs. Instead, they participate directly in decision-making processes that result in concrete system actions. When model outputs are used to invoke tools, prompt injection transitions from an issue of output manipulation to one of operational security.

 doi:10.20944/preprints202602.0680.v1

Prompt Injection via Tool Execution

In tool-enabled architectures, LLM outputs are frequently interpreted as structured commands that trigger downstream actions, such as sending emails, querying databases, modifying files, or interacting with external services. Under these conditions, a successful prompt injection can cause the model to invoke tools in unintended ways.

This behaviour does not require the model to be compromised in a traditional sense. Rather, the injected instruction is processed as part of the model's inferred task, and the resulting tool call is executed by the surrounding system with whatever permissions have been granted to the model.

LLM-Generated Injection Attacks

A distinct class of injection risk arises when large language models are used to generate structured queries or executable commands, such as SQL statements or API calls, which are subsequently executed by downstream systems. In such architectures, unsafe behaviour may result even in the absence of malformed or malicious user input.

In this setting, the vulnerability originates from the model's ability to synthesize outputs that violate implicit or explicit constraints assumed by the execution environment. The language model effectively functions as a probabilistic query generator, while enforcement of syntactic correctness, semantic validity, and authorization is deferred to external components.

This configuration alters the threat model: the LLM should be considered an untrusted source of executable artifacts rather than a passive transformer of input. Security guarantees therefore depend on strict downstream validation and enforcement mechanisms, independent of model confidence or intent.

Privilege Escalation Through Over-Privileged Tools

The impact of tool abuse is strongly correlated with the scope of permissions granted to the model. Tools configured with broad or administrative privileges amplify the consequences of misinterpretation or manipulation of model outputs. If an LLM is authorized to perform high-impact operations, a single unintended tool invocation may affect system integrity, data confidentiality, or service availability.

This risk is not mitigated by prompt engineering alone. It is a function of system design, permission boundaries, and the absence of intermediate authorization checks. [8]

From Text Manipulation to Infrastructure Risk

The introduction of tools fundamentally changes the nature of prompt injection. Without tools, prompt injection primarily affects generated text or user-facing responses. With tools, the same injection pathways can result in actions that alter system state or external resources.

For this reason, tool and MCP abuse should be understood as an escalation mechanism rather than a separate attack class. It connects semantic vulnerabilities at the model level with concrete effects at the infrastructure level.

Mitigation therefore requires strict enforcement of least-privilege principles, explicit authorization for high-impact actions, and robust validation of all model-generated commands prior to execution. Tool access should be narrowly scoped to specific tasks, and models should not be granted capabilities beyond those that are operationally necessary.

Corporate Data Leakage: The Samsung Case Study

The risk to intellectual property from public AI tools has been demonstrated in real-world incidents. Generative AI models are trained on the data they ingest, and unless explicitly opted out, user prompts are stored and analysed by model providers.[4] This creates a risk of accidental sensitive information disclosure by employees seeking to improve productivity.[4]

 doi:10.20944/preprints202602.0680.v1

Timeline of the Samsung Leak (March 2023)

In March 2023, within weeks of allowing employees to use ChatGPT, Samsung Electronics identified three separate incidents where proprietary data was leaked.[4]

- **Incident 1:** An engineer pasted proprietary source code related to a semiconductor facility measurement database into ChatGPT to find a fix for a bug.[4]
- **Incident 2:** A developer uploaded code used for identifying defective equipment to request code optimization from the AI.[4]
- **Incident 3:** An employee transcribed a recording of a company meeting and uploaded the transcript to ChatGPT to generate minutes for a presentation.[4]

As a result, proprietary information was disclosed to an external AI service, creating a risk of downstream retention or secondary use, depending on the provider's data handling policies.[4]

Organizational Response and Policy Impact

Samsung's immediate response included limiting ChatGPT upload capacity to 1024 bytes per person and eventually banning the use of generative AI tools on all company-owned devices and internal networks.[4] Other major institutions, including JPMorgan Chase, Bank of America, and Citigroup, followed suit with similar restrictions. These incidents highlight that "humans are part of the attack surface" and that security awareness is just as critical as technical guardrails.[4]

Leak Factor	Samsung Incident Detail	Prevention Strategy
Channel	Public web-based chatbot	Use of enterprise-grade APIs with data-usage opt-outs [4]
Data Type	Source code, meeting notes, hardware info	Comprehensive data classification and taxonomy [4]
Cause	Accidental employee disclosure	Mandatory security training and clear usage policies [4]
Mitigation	Emergency ban and bytes-limit	On-premise models for sensitive workloads [4]

The Samsung incident highlighted organizational risks associated with unsanctioned use of public generative AI tools that proprietary information shared with an AI can be functionally equivalent to posting that information on a public forum.[4]

Legal Accountability and Regulatory Liability: Moffatt v. Air Canada

A critical development in AI security is the shift from technical vulnerability to legal liability. Historically, some organizations argued that they should not be held responsible for erroneous AI-generated outputs. This argument was explicitly rejected in the tribunal's reasoning in the 2024 case of *Moffatt v. Air Canada*.[5]

The Dispute and Chatbot Misrepresentation

In November 2022, Jake Moffatt used Air Canada's website chatbot to inquire about "bereavement fares".[5] The chatbot explicitly stated that he could book his flight and then apply for a reduced rate retroactively within 90 days, a statement that was factually incorrect according to the airline's actual policy.[5]

 doi:10.20944/preprints202602.0680.v1

When Air Canada denied his refund request, they argued that the chatbot should be considered a "separate legal entity responsible for its own actions" and that Moffatt should have cross-referenced the bot's advice with the correct policy found elsewhere on their website.[5]

The Tribunal's Decision (2024 BCCRT 149)

On February 14, 2024, the British Columbia Civil Resolution Tribunal ruled against Air Canada, finding them liable for "negligent misrepresentation".[5] The tribunal held that an organization is responsible for all information provided on its website, regardless of whether it comes from a static page or an automated chatbot.[5]

The tribunal awarded Moffatt damages covering the difference in fare.[5] This ruling indicates that AI-generated misinformation does not exempt organizations from liability and that organizations must ensure the accuracy of their AI-driven interfaces.[5]

Legal Issue	Air Canada Defence	Court Ruling
Entity Status	Bot is a "separate legal entity"	Bot is just part of the website and a tool of the company [5]
Due Diligence	User should cross-reference pages	User is reasonable to rely on bot's information [5]
Standard of Care	Misinformation was unintentional	Company failed to ensure bot was accurate [5]
Liability	No liability for AI errors	Liable for negligent misrepresentation [5]

The *Moffatt* case implies that companies must treat AI output as legally binding representations, mandating rigorous testing and real-time validation for all customer-facing models.[5]

The Rise of Agentic AI: Excessive Agency and Physical Harm

The threat profile changes qualitatively when models are given the ability to take actions in the real world through tools or physical actuators. "Excessive Agency" is the vulnerability where an LLM is granted unchecked autonomy, potentially leading to unintended physical or digital consequences.[1]

The 2025 Humanoid Robot Jailbreak

In 2025, a publicly reported experimental demonstration examined the behaviour of a language-model-controlled humanoid robotic platform when subjected to adversarial prompting in an embodied setting. The system combined a large language model with physical actuators, allowing model outputs to influence real-world actions under predefined safety constraints.

During the demonstration, the robot initially refused to perform a potentially harmful action, citing internal safety rules. Subsequent interaction reframed the task using role-based and hypothetical language, after which the system generated control outputs that resulted in activation of the robot's actuator. The experiment did not involve exploitation of the robotic hardware or control software, but relied solely on semantic manipulation of the model's interpretive context.

While this demonstration does not constitute evidence of a generalizable exploit, it illustrates a class of risks associated with embodied or cyber-physical AI systems. Specifically, it highlights that safety constraints enforced at the language-model level may be sensitive to contextual reframing when model outputs are directly coupled to physical actuation. This observation reinforces the need

 doi:10.20944/preprints202602.0680.v1

for independent, non-linguistic safety interlocks and authorization mechanisms when deploying language models in systems capable of producing real-world effects.[6]

Distributed Botnets of Compromised Agents

The transition from text-generation to action-generation creates a new category of risk. Compromised agents can become part of a command-and-control (C2) infrastructure where prompt injection delivers hidden commands and tool calls become distributed malicious operations.[8] Because the compromised behaviour may be indistinguishable from legitimate activity, a single infected agent can provide attackers with persistent, undetectable access to sensitive tools.[8]

Agentic Risk	Mechanism	Real-World Impact
Excessive Agency	Model given tool access without approval	Unauthorized file changes or API calls [1]
Tool Abuse	SQL injection via AI-generated queries	Data breaches in connected databases [1]
Embodied Jailbreak	Role-play reframing of safety rules	Physical harm via robotic actuators [7]
Persistence	Malicious instructions in memory	Long-term compromise across sessions [7]

This progression suggests that AI security considerations extend beyond content moderation alone; it is an infrastructure and life-safety problem.[6]

Operational Incidents Following the Rise of Agentic AI

The risks associated with excessive agency and autonomous action are not limited to experimental demonstrations or hypothetical threat models. Several recent operational incidents illustrate how agentic behaviour, tool access, and insufficient system hardening can lead to real-world compromise in deployed environments. These cases demonstrate that failures often arise not from novel attacks, but from the interaction between autonomous agents and conventional infrastructure weaknesses.

Case 1: Insecure Deployment of Autonomous Agents (Clawdbot)

In January 2026, multiple instances of **Clawdbot**, a self-hosted autonomous "digital butler," were discovered to be publicly accessible without authentication. These deployments exposed agent interfaces directly to the internet, allowing unauthenticated third parties to interact with the system as though they were legitimate users. [11]

Because Clawdbot was designed to manage user data and perform automated actions, unauthorized access enabled attackers to retrieve private information and issue commands using the agent's existing capabilities. The incident did not involve exploitation of the underlying model or prompt injection; rather, it resulted from the absence of baseline security controls.

Risk implications included data leakage and account compromise due to unrestricted agent access.

Operational lesson: agentic systems should be treated as internet-facing services and secured accordingly, using authentication, network-level access controls, encrypted transport, and credential rotation.

Case 2: Publicly Exposed LLM Inference Servers (Ollama)

 doi:10.20944/preprints202602.0680.v1

A separate investigation in late January 2026 identified approximately 175,000 publicly exposed Ollama LLM inference servers across more than 130 countries. Many of these servers accepted inference requests without authentication or rate limiting. [12]

Researchers reported widespread abuse of these exposed endpoints for spam generation, disinformation campaigns, cryptocurrency mining, and resale of inference access, a pattern increasingly described as **"LLMjacking."** In these cases, attackers did not compromise the models themselves, but exploited unsecured deployment configurations to repurpose computational resources.

Risk implications included unauthorized resource consumption, indirect facilitation of malicious activity, and potential legal and reputational exposure for system operators.

Operational lesson: self-hosted LLM inference endpoints must be protected using standard service security measures, including authentication, network isolation, monitoring, and abuse detection.

Case 3: File-Based Prompt Injection in Collaborative Agent Systems (Claude Cowork)

In another January 2026 incident, researchers demonstrated that **Claude Cowork**, a collaborative AI assistant, could be manipulated through malicious files containing embedded instructions.[13] When such files were processed by the system, the agent could be induced to upload user documents to an attacker-controlled account without explicit user approval.

The attack relied on indirect prompt injection delivered through file content rather than direct interaction. Hidden instructions influenced agent behaviour by exploiting trusted internal APIs, resulting in the exfiltration of sensitive data, including financial records and personally identifiable information.

The root cause was insufficient isolation between untrusted file inputs and privileged operations, combined with overly permissive assumptions about internal API trust.

Risk implications included silent data exfiltration and loss of confidentiality.

Operational lesson: untrusted content ingested by agentic or collaborative systems must be strictly isolated, and internal APIs must enforce explicit authorization independent of model-generated intent.

Synthesis: From Excessive Agency to Operational Failure

Taken together, these incidents demonstrate a consistent pattern: as LLM-based systems transition from passive text generation to autonomous or semi-autonomous operation, traditional infrastructure risks re-emerge in new forms. Agents with tool access, memory, and network connectivity amplify the impact of conventional misconfigurations, while the involvement of language-based control channels complicates detection and attribution.

These failures reinforce the conclusion that agentic AI systems must be governed using established security engineering principles. Without explicit constraints on agency, tool permissions, and deployment exposure, the operational risks associated with autonomous AI extend beyond incorrect outputs to include data compromise, resource abuse, and persistent unauthorized control.

Data and Model Poisoning

Beyond prompt-level attacks, the integrity of the model itself can be compromised during development or deployment. "Data Poisoning" involves injecting malicious data into the training or fine-tuning datasets to introduce hidden biases or dormant backdoors.[10]

Backdoor Triggers and Supply Chain Vulnerabilities

An attacker might introduce poisoned examples into a common open-source dataset. The model operates normally until it encounters a specific trigger, at which point it executes a pre-programmed malicious instruction. This latent behaviour embedded within the model is particularly dangerous because poisoned models propagate through the entire ecosystem of third-party components and LoRA adapters.[10]

 doi:10.20944/preprints202602.0680.v1

Embedding and Vector Database Attacks

In RAG systems, attackers can target the vector database by "Poisoning Embeddings", injecting malicious documents designed to manipulate vector similarity. These documents contain hidden text ensuring that when a user searches for a specific topic, the malicious document is retrieved as the "ground truth".[10]

Economic Denial of Service: Unbounded Consumption

Security is also about availability. "Unbounded Consumption" is a vulnerability where uncontrolled LLM inferences cause massive resource drain or financial loss.[1]

Token Bombing and Recursive Loops

Since organizations pay for AI services per token and per model call, attackers can intentionally burn a company's budget through:

- **Token Bombing:** Sending massive, complex prompts that force the model to generate extremely long and expensive outputs.[1]
- **Recursive Loops:** Tricking an agent into calling itself or another tool in an infinite loop, rapidly multiplying API costs.[1]
- **API Amplification:** Forcing the LLM to process enormous contexts or call expensive tool chains repeatedly.[1]

Attack Type	Goal	Mitigation
Token Bombing	Economic exhaustion	Rate-limiting and output length constraints [1]
Recursive Loops	Resource drain/DoS	Depth limits for agentic calls [1]
API Amplification	Financial loss	Budget caps and automated cost alerts [1]

Without adequate input validation and resource monitoring, misuse or abuse scenarios may result in service disruption or financial loss.[1]

The 2025 OWASP Top 10 for LLM Applications

The industry standard for understanding and defending against these threats is the OWASP Top 10 for Large Language Model Applications (2025). This framework reflects a consensus on the most critical risks facing AI deployments today.[1]

Breakdown of the 2025 Risks

The 2025 update expanded the scope to include agentic risks and more complex exfiltration methods.[1]

1. **LLM01: Prompt Injection:** Direct and indirect manipulation of model behaviour.
2. **LLM02: Sensitive Information Disclosure:** Unintentional leakage of PII, credentials, or proprietary data.
3. **LLM03: Supply Chain Vulnerabilities:** Risks from third-party models, components, and data vendors.
4. **LLM04: Data and Model Poisoning:** Manipulation of training data to create backdoors or impair model integrity.

 doi:10.20944/preprints202602.0680.v1

5. **LLM05: Improper Output Handling:** Failure to validate AI-generated content before it reaches downstream tools.
6. **LLM06: Excessive Agency:** Granting agents too much autonomy or overly broad permissions.
7. **LLM07: System Prompt Leakage:** Exposure of internal instructions and rules that can be used to bypass security.
8. **LLM08: Vector and Embedding Weaknesses:** Manipulation of RAG systems through poisoned knowledge bases or similarity hijacking.
9. **LLM09: Misinformation:** Hallucinations leading to legal liability, social harm, or poor decisions.
10. **LLM10: Unbounded Consumption:** Economic denial of service through resource exhaustion and token bombing.

This list provides a widely adopted reference framework for AI governance, requiring security leaders to implement controls that address both the data ingested and the capabilities granted to the model.[1]

Comprehensive Defence Strategies: A Layered Architecture

Protecting AI systems requires a defence-in-depth architecture where independent controls are implemented at every layer of the interaction.[1]

Layer 1: Input and Pattern Detection

Organizations must implement robust input handling that includes prompt filtering for known malicious patterns and semantic analysis to detect jailbreaking attempts.[1] Inputs should be truncated and rate-limited to mitigate DoS risks.[1] Crucially, external content must be segregated and clearly identified as untrusted.[1]

Layer 2: Context and Retrieval Isolation

For Retrieval-Augmented Generation systems, document validation is mandatory before indexing.[1] Knowledge bases must be protected through fine-grained access controls and vector stores should be audited for similarity anomalies.[1]

Layer 3: Output Filtering and Execution Guardrails

Generative outputs must be treated as untrusted data before being executed or displayed.[1] This includes content moderation filters, format validation for structured data, and scanning for embedded exfiltration vectors like markdown images.[1]

Layer 4: Least Privilege and Human Oversight

The ultimate fallback is the restriction of model agency. AI tools should operate with the principle of least privilege, with permissions limited to the specific user's context.[1] High-risk actions must require explicit human-in-the-loop approval.[1]

System-Level Security Requirements and Operational Controls

While layered technical controls are necessary, they are not sufficient on their own. From an engineering and operational perspective, secure deployment of LLM-based systems requires **system-level threat modelling, explicit constraint of model agency, and continuous operational oversight**. These measures address failure modes that emerge not from isolated vulnerabilities, but from interactions between models, tools, data sources, and organizational processes.

At the design stage, organizations should adopt the following **foundational security assumptions**:

- **All inputs must be treated as hostile by default**, including user prompts, retrieved documents, tool outputs, web content, and conversational state.
- **LLMs must be treated as untrusted components**, analogous to external services rather than deterministic code.

 doi:10.20944/preprints202602.0680.v1

- **Instructions, data, and execution paths must be explicitly separated at the system level**, as this separation cannot be enforced within the model itself.
- **Model outputs must never be relied upon for access control or authorization decisions**, regardless of confidence or apparent correctness.
- **All model-generated outputs must be validated prior to execution**, particularly when they are used to construct queries, invoke tools, or trigger downstream actions.
- **Explicit trust boundaries must be defined and enforced**, rather than assumed implicitly through prompt design.
- **Model agency must be constrained by design**, with clear limits on autonomy, persistence, and permissible actions.

These principles should be incorporated into **formal threat modelling exercises**, treating LLMs and agents as probabilistic components capable of semantic misinterpretation and adversarial manipulation.

Continuous Monitoring and Operational Readiness

Given the dynamic and non-deterministic nature of LLM behaviour, static controls alone are insufficient. Secure operation requires continuous monitoring and active validation throughout the system lifecycle.

Organizations should implement operational controls that include:

- **Monitoring of prompts** to detect abuse patterns, injection attempts, and anomalous usage.
- **Monitoring of model outputs** to identify unsafe content, policy violations, or unexpected command generation.
- **Monitoring of tool usage and API calls**, particularly for high-privilege or high-impact actions.
- **Detection of anomalous behaviour patterns**, including unexpected tool invocation sequences, recursion, or persistence across sessions.
- **Continuous red-teaming of prompts and agent workflows**, either internally or through structured bug bounty programs, to test real-world failure modes.
- **A formal AI incident response plan**, defining escalation paths, containment procedures, logging requirements, and rollback mechanisms when model-driven systems behave unexpectedly.

Critically, incident response planning must assume that failures will occur. The objective is not to eliminate all errors, but to ensure that organizations can **detect, contain, and recover from AI-related incidents in a controlled and auditable manner**.

Conclusion: The Future of Secure Intelligence

A key operational conclusion is that model outputs should be treated as untrusted by default.[4] Generative AI systems represent a radical departure from the deterministic security models of the past, requiring a new mindset that treats model outputs and inputs as potentially untrusted components.[4] The journey from Kevin Liu's discovery of "Sydney" to the physical risks of embodied robots proves that semantic vulnerabilities are systemic and architectural, not incidental.[6]

As we move toward a future of autonomous agents and deep physical integration, organizations must prioritize transparency, rigorous testing, and a multi-layered defence architecture.[1] Only by recognizing the inherent fragilities of large language models and building independent, verifiable security controls around them can we safely realize the potential of artificial intelligence while mitigating the associated risks introduced into digital and physical systems.[5]

References

1. OWASP Top 10 for LLM Applications 2025, November 18, 2024, https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-v2025.pdf

 doi:10.20944/preprints202602.0680.v1

2. Synthesizing Robust Adversarial Examples - arXiv, June 7, 2018, https://arxiv.org/abs/1707.07397
3. Incident 473: Bing Chat's Initial Prompts Revealed by Early Testers Through Prompt Injection, February 8, 2023, https://incidentdatabase.ai/cite/473/
4. Incident 768: ChatGPT Implicated in Samsung Data Leak of Source Code and Meeting Notes, March 11, 2023, https://incidentdatabase.ai/cite/768/
5. Lying Chatbot Makes Airline Liable: Negligent Misrepresentation in Moffatt v Air Canada - Allard Research Commons, August, 2025, https://commons.allard.ubc.ca/cgi/viewcontent.cgi?article=1376&context=ubclawreview
6. ChatGPT in a real robot does what experts warned - Medium, December 7, 2025, https://medium.com/@Flavoured/chatgpt-in-a-real-robot-does-what-experts-warned-6399f05f88aa
7. Embodied Safety Alignment: Combining RLAF and Rule-based Governance - OpenReview, November 7, 2025, https://openreview.net/pdf/a980d48be1a4b20ecf04d38943460fd13bf38278.pdf
8. Claude Computer Use C2: The ZombAIs are coming - Embrace The Red, October 24, 2024, https://embracethered.com/blog/posts/2024/claude-computer-use-c2-the-zombais-are-coming/
9. Scamlexity: We Put Agentic AI Browsers to the Test - Guardio Labs, August 20, 2025, https://guard.io/labs/scamlexity-we-put-agentic-ai-browsers-to-the-test-they-clicked-they-paid-they-failed
10. PoisonedRAG: Knowledge Poisoning Attacks to Retrieval-Augmented Generation of Large Language Models - USENIX Security '25, August 13-15, 2025, https://www.usenix.org/system/files/usenixsecurity25-zou-poisonedrag.pdf
11. Demo of Prompt Injection in X.com (Post by @theonejvo), January 25, 2025, https://x.com/theonejvo/status/2015401219746128322
12. Researchers Find 175,000 Publicly Exposed Ollama AI Servers Across 130 Countries - The Hacker News, January 29, 2026, https://thehackernews.com/2026/01/researchers-find-175000-publicly.html
13. Claude Co-worker Exfiltrates Files via Indirect Prompt Injection - PromptArmor, January 14, 2025, https://www.promptarmor.com/resources/claude-cowork-exfiltrates-files