ARTIFICIAL INTELLIGENCE

The Unsolvable AI Problems

Design Secure AI to Compensate for Hallucinations and Prompt Injection

Jamy Sneeden, *Cybersecurity Engineer*

Generative AI has been promising a new world that we will work in: large language models are the new interface to data, automation, and decision-making. That story has been coming to life over the past few years. From chatbots to agentic automation systems, AI is continuously being baked into our day-to-day lives. However, the same architecture that has enabled LLMs to become ubiquitous also harbors fundamental flaws. Hallucinations, prompt injection, and context limitations are consequences of how AI models are built and trained. They are not problems we can patch over; they are key to how LLMs work in the first place.

Hallucinations

Hallucinations, well-formed answers that are just wrong, continue to surface even in the most advanced system. The nature of AI is they predict the next likely thing to appear (Probabilistic not Deterministic). They are not built to verify truth.

Prompt Injections

Prompt injection exploits a deeper issue. LLMs cannot reliably distinguish between trusted instructions and untrusted data when both are expressed as natural language. Just as teaching a human to not fall for common scams does not make them immune to all cons, telling an LLM to beware of common prompt injection does not protect them from all forms of 'AI social engineering'.

And despite rapid progress in scaling **context windows,** the underlying transformer architecture still struggles with long-horizon reasoning and memory, constrained by computational limits and a fundamentally stateless design.

These are structural characteristics of how modern AI models are designed, not simple bugs.

For enterprises, this creates a dangerous illusion: models are becoming more capable, but still not totally dependable. Especially as agentic systems expand, a small error at the beginning of an automation workflow can have catastrophic repercussions downstream.

The industry is beginning to acknowledge this reality. Leading research now frames hallucination, prompt injection, and jailbreaks not as edge cases, but as **inherent behaviors of probabilistic language models.** These are risks that must be continuously mitigated rather than permanently eliminated.

This shift in thinking is critical. Because it changes the question from:

"HOW DO WE FIX THE MODEL?" TO ⟶ **"HOW DO WE BUILD SYSTEMS THAT REMAIN SECURE AND RELIABLE EVEN WHEN THE MODEL FAILS?"**

That distinction is where AI deployments succeed or fail.

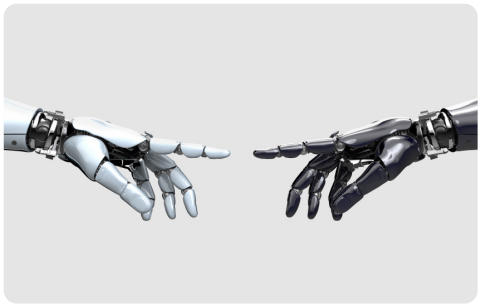
Enterprise AI deployments demand systems that assume imperfection and remain secure anyway. Defense-in-depth architectures, leveraging additional security controls, maintaining visibility and governance are key to ensuring trustworthy AI powered systems.

Designing for Hallucinations

If hallucinations are not a bug but a byproduct of how language models generate outputs, then the goal cannot be elimination. It must be containment.

The practical question becomes: how do we design AI systems where hallucinations are both less likely to occur and less likely to cause harm when they do?

The answer is not a single control, but a layered system of constraints, validation, and visibility.



Ground the Model in Verifiable Reality

The most effective way to reduce hallucinations is to shift the model away from relying on its internal, probabilistic "memory" and toward verifiable external data.

Retrieval-Augmented Generation (RAG) is foundational here. Instead of asking the model to generate answers from training data alone, RAG injects relevant, authoritative documents at inference time and forces the model to ground its responses in them. This changes the nature of the output from prediction to synthesis.

Equally important is requiring **explicit citations** in responses. **This introduces two benefits:**

- It makes outputs auditable by humans and downstream systems
- It creates a feedback loop where missing or weak citations can be programmatically flagged

Grounding does not eliminate hallucinations, but it transforms them from silent failures into detectable ones.

Constrain the Problem Space

General-purpose AI is inherently more prone to hallucination because it operates across an unbounded domain. Enterprise systems should resist this design.

Purpose-built AI systems, those constrained to a narrow function, dataset, or workflow, are significantly more predictable.

This narrowing serves multiple purposes:

- Reduces hallucination surface area by limiting what the model is allowed to answer
- Improves determinism in outputs by reducing ambiguity
- Creates clearer trust boundaries between approved and unapproved data sources
- Limits blast radius when failures occur

In practice, this often means decomposing large, generalized **"agent"** designs into smaller, specialized components with tightly defined responsibilities. The model is no longer asked to "know everything," only to perform specific, bounded tasks reliably.

Introduce Independent Verification Layers

If the first model output cannot be trusted outright, then systems must verify before acting.

Verification layers should be treated as mandatory control points, not optional enhancements. Several patterns have emerged as effective:

- **LLM-as-a-judge:** A secondary model evaluates the output of the primary model for correctness, policy compliance, or grounding quality
- **Retrieval-based validation (faithfulness checking):** Compare generated outputs against the retrieved source material to detect inconsistencies
- **Deterministic validation via APIs/tools:** For critical facts (dates, financial data, configurations), external systems of record should be used to confirm accuracy

The key principle is independence. The system generating the answer should not be the only system responsible for validating it.

This mirrors traditional security design: separation of duties reduces the likelihood of unchecked failure.

Design for Uncertainty, Not Confidence

One of the more dangerous characteristics of hallucinations is that they are often presented with high confidence. The model does not "know" when it is wrong.

This makes **confidence management** a system-level responsibility.

Effective approaches include:

- **Confidence scoring** based on retrieval quality, response consistency, and model signals
- **Abstention mechanisms** that allow the system to explicitly decline to answer
- **Clarification loops** where the model asks follow-up questions instead of guessing
- **Human escalation** paths for low-confidence or high-impact scenarios

In well-designed systems, "I don't know" is not a failure, it is a controlled and desirable outcome.

Make Every Output Auditable

You cannot secure what you cannot see.

Every interaction with an AI system (inputs, retrieved data, model outputs, and verification results) should be logged and attributable. This creates the foundation for:

- Post-incident analysis
- Continuous improvement of prompts and retrieval strategies
- Detection of systemic failure patterns
- Compliance and governance reporting

Auditability turns hallucinations from unpredictable anomalies into measurable events that can be managed over time.

Common Pitfalls to Avoid

Even with strong architectural patterns, certain misconceptions continue to undermine AI reliability:

- **"Bigger models will solve it"**
  Larger models may reduce hallucination frequency, but they do not eliminate it. The failure mode persists because the underlying mechanism, probabilistic generation, remains unchanged.
- **Over-reliance on prompt engineering**
  Prompts are not security controls. They are advisory inputs that can be ignored, overridden, or manipulated. Relying on them as a primary defense is fragile by design.
- **Trusting model confidence**
  A fluent, assertive response is not a signal of correctness. Systems that equate confidence with accuracy will fail in subtle but dangerous ways.

From Model Accuracy to System Reliability

The industry often frames hallucination as a model quality problem. In practice, it is a system design problem.

Reliable AI systems are not those that never hallucinate, they are those that:

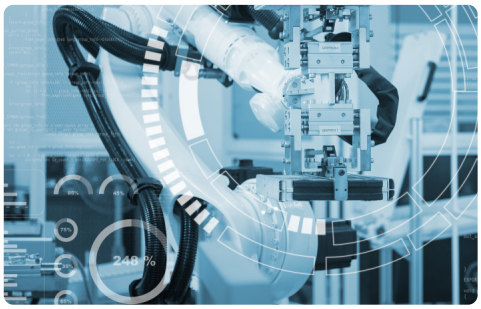
- Reduce the likelihood of hallucination through grounding and scope control
- Detect hallucinations through verification and validation
- Contain their impact through abstention, escalation, and constrained execution

This is the shift enterprises must internalize.

Accuracy is a property of the model. **Reliability is a property of the system.**

Prompt Injection: When Language Becomes an Attack Surface

Prompt injection is not just another application-layer vulnerability. It is a direct consequence of how AI systems interpret and act on language.



Traditional software systems enforce strict boundaries between code and data. AI systems, by design, blur that line. Instructions, user inputs, retrieved documents, and tool outputs are all processed as natural language. Often all within the same context window. To the model, they are indistinguishable.

This is the root of the problem.

Prompt injection exploits this ambiguity by introducing malicious or conflicting instructions into inputs the model is expected to process. Because the model lacks a native mechanism to separate "trusted instruction" from "untrusted content," it can be manipulated into overriding its original purpose.

Unlike traditional injection attacks, this is not a sanitization bug or parsing flaw. It is a structural characteristic of language-based systems.

Treat Prompt Injection as a System Design Problem

There is no single control that "fixes" prompt injection. Like hallucinations, it must be addressed through system design.

The most effective approach is a layered, defense-in-depth strategy that combines:

- Architectural constraints
- Security controls
- Continuous validation and governance

The encouraging reality is that many of the same patterns used to reduce hallucinations also strengthen resilience against prompt injection.

Well-designed AI systems tend to be broadly secure systems.

Establish Explicit Trust Boundaries

The most critical step in mitigating prompt injection is to define and enforce trust boundaries.

At inference time, the model does not inherently know the difference between:

- A system instruction
- A user query
- Retrieved content
- Tool output

All of it is just text.

This creates a dangerous condition where untrusted input can masquerade as authoritative instruction.

To compensate, systems must externally enforce trust separation, even if the model cannot internally distinguish it.

Key principles include:

- Treat all external input as untrusted, including user prompts, retrieved documents, APIs, emails, and files
- Strictly isolate system instructions and policies from user-controlled content at the orchestration layer
- Avoid directly concatenating untrusted input into prompts without context labeling and control

In effect, the system (as opposed to the model) must be responsible for maintaining instruction integrity.

Validate and Sanitize All Inputs

Once inputs are treated as untrusted, they can be systematically analyzed and controlled before reaching the model.

Input validation for AI systems goes beyond traditional filtering. It must account for adversarial language patterns designed to manipulate behavior.

Common injection patterns include:

- Instruction overrides (e.g., “ignore previous instructions”)
- Embedded directives hidden within documents or code comments
- Encoded or obfuscated payloads
- Multi-step or context-dependent manipulation attempts

Defensive strategies include:

- **Pattern detection and blocking** for known injection techniques
- **Content normalization** to remove hidden instructions (e.g., stripping HTML comments, decoding payloads)
- **Multi-layer validation pipelines**, where inputs are evaluated by multiple controls before use

Industry guidance, including OWASP recommendations, increasingly emphasizes defense-in-depth validation rather than relying on a single filtering mechanism.

Importantly, validation must extend beyond direct user input.

Account for Indirect Prompt Injection

One of the more subtle and dangerous forms of this attack is indirect prompt injection.

In these scenarios, malicious instructions are embedded in external content the AI system retrieves or processes:

- Documents in a knowledge base
- Emails or chat transcripts
- Web pages
- Data returned from tools or APIs

The model consumes this content as part of its reasoning process and may unknowingly execute hidden instructions within it.

This expands the attack surface significantly.

Mitigation requires:

- Applying the same sanitization and validation controls to **retrieved data** as to user input
- Limiting what external content is allowed to influence system behavior
- Separating “data for reference” from “instructions for execution” wherever possible

If a system uses RAG, it must assume that retrieved documents can be adversarial, not just incorrect.

Enforce Guardrails at Multiple Layers

Guardrails are becoming essential infrastructure in enterprise AI systems. They represent **system-wide enforcement mechanisms** that constrain behavior regardless of model output.

Effective implementations typically include:

Input Guardrails

- Block or flag malicious prompts
- Enforce acceptable use policies
- Apply data loss prevention (DLP) controls

Output Guardrails

- Validate responses before delivery
- Detect policy violations, unsafe actions, or sensitive data exposure
- Prevent the model from executing or returning harmful instructions

Behavioral Guardrails

- Define and enforce what the system is allowed to do
- Restrict tool usage and agent actions
- Apply least privilege principles to agent capabilities

Guardrails shift control away from the model and back into deterministic system logic, where security decisions belong.

Restrict Scope and Enforce Least Privilege

Prompt injection becomes significantly more dangerous when AI systems are given broad, autonomous capabilities.

This is particularly relevant for agentic systems that:

- Execute actions
- Access sensitive systems
- Chain multiple tools together

In these environments, a single successful injection can cascade into real-world impact.

To mitigate this:

- Constrain agent scope to narrowly defined tasks
- Limit tool access to only what is strictly required
- Enforce least privilege across all integrations
- Require explicit validation before high-impact actions

Smaller, purpose-built agents are not only more reliable but they are also more secure.

Continuously Test with Adversarial Techniques

Unlike traditional software vulnerabilities, prompt injection defenses degrade over time as new attack patterns emerge. This makes **continuous testing** essential.

Red teaming and adversarial evaluation should simulate:

- Jailbreak attempts
- Hidden and indirect injections
- Malicious documents and data sources
- Abuse of tool integrations
- Resource exhaustion or unbounded execution

The goal is not just to identify weaknesses, but to measure how well existing controls detect and respond to them.

AI security is not a one-time validation, it is an ongoing process.

Monitor, Detect, and Respond

Even with strong preventative controls, injection attempts will occur.

Systems must be designed to:

- Log all interactions and decisions
- Detect anomalous behavior patterns
- Trigger alerts or interventions when risk thresholds are exceeded
- Support rapid investigation and remediation

This aligns with broader cybersecurity principles: prevention alone is insufficient without detection and response.

Accept the Constraint, Design the Control

Prompt injection is not a flaw we can fully eliminate. It is a natural consequence of giving machines the ability to interpret and act on human language.

That reality forces a shift in mindset. The goal is not to make models perfectly resistant to manipulation.

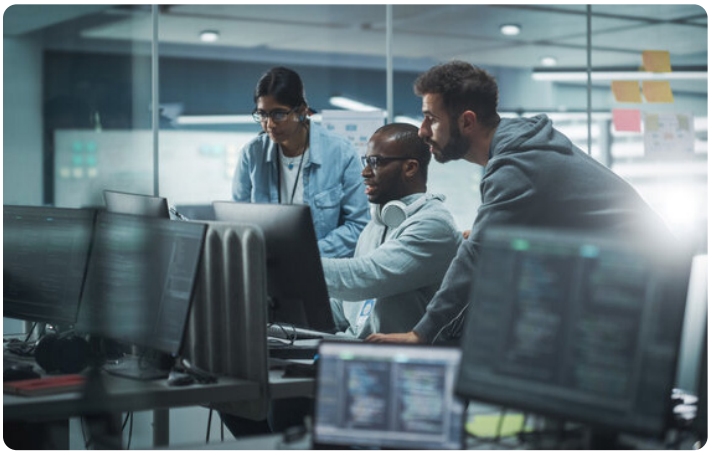
It is to build systems where:

- Malicious inputs are treated as untrusted by default
- Instructions cannot be silently overridden
- Actions are constrained, verified, and observable
- Failures are detected before they escalate

Like the rest of cybersecurity, success is measured by risk reduction, not perfection.

Reduce the probability, limit the impact, and detect and recover quickly.

Conclusion



Secure AI is not achieved through better prompts or larger models; it is achieved through deliberate system design. Hallucinations and prompt injection are inherent to how AI works, which means safe, enterprise-ready AI depends on architectural controls that assume failure and defend against it.

Organizations that recognize this early will build systems that are not only powerful, but resilient, governable, and trustworthy.

Sayers is at the forefront of helping enterprises navigate these challenges. We are designing and implementing secure AI architectures that balance innovation with control, so organizations can adopt AI with confidence.