ENTERPRISE AI PLAYBOOK · DEEP DIVE · PART 7 · SECURITY

Securing the AI Attack Surface

The threat model is genuinely new — prompt injection, data poisoning, agents with real permissions. Five pillars for defending enterprise AI end to end.

CROSS-CUTTING CAPABILITY · RUNS THROUGH EVERY STAGE

- **Input Defense — untrusted input is the attack**
- **Data & Supply Chain — trust what you train and ground on**
- **Access & Least Privilege — an agent's reach is its risk**
- **Output & Content Safety — what comes out can hurt too**
- **Detection & Response — assume breach; rehearse it**

Tom Ward
Enterprise Architect · Cloud & AI

linkedin.com/in/tomward
July 2026

PILLAR 1 Input Defense

"Untrusted input is the attack"

The defining new threat of the AI era is prompt injection: untrusted content — a web page, an email, a document — carrying instructions that hijack the model into ignoring its rules. It has no perfect defense today, which is the single most important fact for a security team to internalize. You cannot filter your way to safety; you contain the damage through layered controls and least-privilege design so that a successful injection can't reach anything valuable.

Input defense is the first layer of that containment: prompt-injection shields, jailbreak screening, and validation of everything entering the model — especially content the model will treat as instructions. Treat every input the model consumes as potentially hostile, the way you treat user input to any other system.

KEY DECISIONS

- Which inputs are trusted vs. untrusted, and are they clearly separated?
- What shields screen for injection and jailbreaks, and how are they updated?
- Given no perfect filter, what does least-privilege containment look like?

WATCH OUT FOR

- × Assuming a filter makes prompt injection "solved" — it doesn't.
- × Mixing trusted instructions and untrusted content in one context.
- × Retrieved documents treated as safe because they're "internal."

PILLAR 2 Data & Supply Chain

"Trust what you train and ground on"

An AI system is only as trustworthy as what went into it — the training data, the fine-tuning set, the retrieval corpus, and the chain of models and libraries it depends on. Data poisoning (corrupting what the model learns or retrieves) and supply-chain compromise (a tampered model or dependency) are attacks with no analog in a traditional app, and they're invisible unless you've built the provenance to detect them.

The defense is the same discipline that secures any software supply chain, extended to models and data: know where every model came from, verify its integrity, vet dependencies, and control who can write to the corpora your system grounds on. Provenance is not paperwork — it's how you answer "can we trust this output" after an incident.

KEY DECISIONS

- What is the provenance and integrity check for every model you run?
- Who can write to the training and retrieval corpora, and how is it controlled?
- How are model and library dependencies vetted and monitored?

WATCH OUT FOR

- × Pulling models from unverified sources with no integrity check.
- × Retrieval corpora anyone can write to, enabling indirect injection.
- × No inventory of which models and versions are actually in production.

PILLAR 3 Access & Least Privilege

"An agent's reach is its risk"

Because prompt injection can't be fully prevented, the security of an AI system comes down to what a compromised model can actually *do* — and that is a question of access, not of prompts. An agent inherits the blast radius of every credential and tool you hand it, and unlike a phished employee, it can act thousands of times a minute. This is the pillar that most directly contains agentic risk, and it ties straight to the autonomy ladder from Part 4.

The rule is unforgiving and simple: an agent should never hold permissions its accountable human owner doesn't, every tool it can call should be scoped to the task, and write access to a system of record should be as deliberately granted as a production deployment. Constrain the tools, not just the prompts.

KEY DECISIONS

- What is the blast radius if this agent is fully compromised?
- Are credentials and tool permissions scoped to the task, or inherited wholesale?
- Does any agent hold access its human owner does not?

WATCH OUT FOR

- × Broad service-account credentials handed to an agent for convenience.
- × Write access to systems of record with no rate limit or approval.
- × Tool permissions that quietly expand as capabilities are added.

PILLAR 4 Output & Content Safety

"What comes out can hurt too"

The model's output is an attack surface in its own right. It can leak PII or secrets it saw in context, emit content that creates legal or brand harm, or — most subtly — produce output that a downstream system executes without sanitizing (insecure output handling: think a model that returns SQL, a shell command, or HTML that's run as-is). Every output should be treated as untrusted until filtered, exactly like input.

Output controls are the mirror of input defense: content safety and PII redaction on the way out, plus strict handling of anything the model produces that another system will act on. If a model's output can reach a database, a browser, or a shell, that path deserves the same scrutiny as any injection point.

KEY DECISIONS

- What output filtering and PII redaction sit between the model and the user?
- Where does model output flow into a system that will execute it?
- How is content safety enforced, and mapped to your policy?

WATCH OUT FOR

- × Model output executed by a downstream system without sanitizing.
- × Secrets or PII from context leaking into a response.
- × Content-safety gaps discovered by a customer, not a test.

PILLAR 5 Detection & Response

"Assume breach; rehearse it"

Because you can't prevent every attack, you have to be able to see and stop one. Detection and response is the pillar that turns a security incident from an existential event into a managed one: adversarial red-teaming to find weaknesses before attackers do, monitoring that surfaces anomalous prompts and actions, and an incident-response plan that has actually been rehearsed against an AI-specific scenario — not just adapted from the generic runbook.

Red-teaming deserves special emphasis because AI systems fail in ways traditional testing misses. Someone whose job is to attack your own assistants — to jailbreak them, poison their retrieval, escalate an agent's privileges — will find the holes a functional test never will. Make it a standing cadence, not a one-time pre-launch exercise.

KEY DECISIONS

- Who red-teams your AI systems, and on what cadence?
- What does monitoring watch — prompts, actions, outputs, anomalies?
- Has the AI incident-response plan been rehearsed against a real scenario?

WATCH OUT FOR

- × Red-teaming treated as a one-time gate, not an ongoing practice.
- × Monitoring infrastructure health but not model behavior or misuse.
- × An incident plan that assumes a traditional breach, not an AI one.

Security at every stage

Security isn't a gate at the end — it's a capability that widens as the system grows. It shows up at each of the four stages, and the attack surface expands as you climb.

Assess · Part 1	Threat-model the stack layer by layer — every tier from infrastructure to interface has an attack surface.
Decide · Part 2	The risk tier of a use case sets how much security scrutiny it earns before you commit.
Build · Part 3	Prompt shields, guardrails, and output filtering are security controls built into the pipeline.
Advance · Part 4	Agents that act multiply the blast radius — security must scale with every rung of autonomy.

The AI Security Baseline

Ten controls that turn "we take AI security seriously" into something you can evidence. Mapped to the OWASP Top 10 for LLM Applications and MITRE ATLAS — the vocabulary your security team already recognizes.

- ☐ **Trusted and untrusted inputs are separated**, and injection/jailbreak shields screen everything the model treats as instructions.
- ☐ **Least-privilege by design** — no agent holds access its human owner doesn't, and every tool is scoped to the task.
- ☐ Every action an agent can take has a **known, bounded blast radius**, with rate limits on writes to systems of record.
- ☐ **Model provenance is verified** — you know where each model came from and check its integrity before it runs.
- ☐ **Retrieval corpora are write-controlled**, so an attacker can't plant indirect-injection content in your knowledge base.
- ☐ **Output is filtered like input** — PII redaction and content safety on the way out, and no model output is executed unsanitized.
- ☐ **Secrets never enter model context** where they could leak into a response.
- ☐ **Adversarial red-teaming runs on a standing cadence**, not just once before launch.
- ☐ **Monitoring watches behavior** — anomalous prompts, actions, and outputs — not just infrastructure health.
- ☐ An **AI-specific incident-response plan has been rehearsed** against a real scenario.

Going Deeper: References

The frameworks and threat taxonomies to build your AI security program on.

OWASP Top 10 for LLM Applications

The canonical LLM threat list — prompt injection, insecure output handling, data poisoning, excessive agency. Start here.

owasp.org/www-project-top-10-for-large-language-model-applications

MITRE ATLAS

The adversarial threat landscape for AI systems, in the familiar ATT&CK style — your red team's map.

atlas.mitre.org

NIST — Adversarial Machine Learning Taxonomy (AI 100-2)

The authoritative taxonomy of attacks and mitigations for ML and generative systems.

csrc.nist.gov/pubs/ai/100/2/e2023/final

Google — Secure AI Framework (SAIF)

A practitioner's framework for securing AI systems across their lifecycle.

safety.google/cybersecurity-advancements/saif

Microsoft — AI Red Team

Practical guidance on adversarially testing AI systems, from a team doing it at scale.

learn.microsoft.com/en-us/security/ai-red-team

Cloud Security Alliance — AI

Vendor-neutral research and guidance on securing AI in the enterprise.

cloudsecurityalliance.org/research/topics/artificial-intelligence

NIST AI Risk Management Framework

Ties security into the broader governance model — where AI risk is owned and managed.

nist.gov/itl/ai-risk-management-framework

About the Author

Tom Ward is an Enterprise Architect with 20+ years across Fortune 500 financial services, retail, and government — including Wells Fargo and Bank of America — specializing in EA governance, reference architectures, hybrid cloud, and legacy modernization, with recent hands-on GenAI and mobile delivery work.

The full series: archreviews.com/playbook · Connect: linkedin.com/in/tomward