Securing Agentic AI: The AFuller F7-LAS™ (7-Layer) Model

Fuller 7-Layer Agentic AI Security (F7-LAS)

Securing agentic AI systems across seven interdependent layers.

Whitepaper v2.4 — November 2025

Author:
Anthony Fuller
AI Engineer | Cybersecurity Architect
LinkedIn: https://www.linkedin.com/in/itsecuritypartners/
GitHub: https://github.com/anthfuller

Disclaimer

The views and opinions expressed in this whitepaper are those of the author and do not necessarily represent the views of any current or former employer. This document is provided for informational purposes only and does not constitute legal, compliance, or regulatory advice.

Abstract

This whitepaper introduces the AFuller F7-LAS (Fuller 7-Layer Agentic AI Security) model, a practical framework for analyzing and securing agentic AI systems in enterprise environments.

The **model** focuses on seven layers, from system prompts, RAG, and the planner through tools and policy engines, down to sandboxed execution and monitoring. The goal is to provide security architects and practitioners with a clear, layered framework to reason about risk and design controls for AI systems that can take actions, not just generate text.

Executive Summary

Agentic AI systems present and observability challenges not adequately addressed by existing LLM safety taxonomies or model cards. This paper does not address safety concerns related to general-purpose LLMs. Instead, it focuses on security-first telemetry.

Large language models are rapidly evolving from chat-style assistants into **agentic AI systems** that can plan, call tools, and take actions in complex enterprise environments. In security operations and cloud ecosystems, this evolution shifts the primary concern from content-only issues—such as fabricated or ungrounded responses—to operational risk, where incorrect or unsafe decisions can drive real changes to tickets, identities, configurations, or infrastructure.

This whitepaper introduces the AFuller F7-LAS (Fuller 7-Layer Agentic AI Security) model, a practitioner-oriented framework for analyzing and securing agentic AI systems. F7-LAS decomposes an agent into seven layers: the system prompt; RAG and grounding; the agent planner; tools and integrations; the policy engine outside the LLM; the sandboxed execution **environment**;

and **monitoring** and **evaluation**. Each layer represents a distinct point of potential failure and an opportunity to apply targeted guardrails.

Appendix B, the F7-LAS Threat–Control Map, provides a structured view of typical threats and defensive controls at each layer and demonstrates how common attack patterns can be mitigated through layered defenses.

In practice, F7-LAS can serve as a design review checklist and a structured way to challenge the common claim that "*we secured it with a strong prompt and RAG.*" Prompts and RAG (Layers 1 and 2) reduce fabricated and ungrounded outputs, but they do not govern planning, tool actions, policy enforcement, sandboxing, or observability. Those responsibilities fall to Layers 3–7—planning, action surfaces, policy enforcement, sandboxing, and monitoring—supported by Supplemental Layer S (Software Supply-Chain Security), which provides a cross-cutting integrity baseline through SBOMs, SCA, tool/plugin vetting, and runtime attestation across all seven layers.

The core message is simple and enduring: agents are not just chatbots with better prompts, and securing them requires far more than improved prompting and RAG pipelines. By applying the F7-LAS model, organizations can move from ad hoc agent experiments to structured, defensible, and governed agentic AI deployments—systems that act on behalf of the enterprise while remaining within clear, observable, and enforceable boundaries. These layers can complement broader governance and threat frameworks (such as the NIST AI RMF, ISO/IEC 42001, and the EU AI Act), but any alignment remains interpretive and context-dependent.

Table Content

1. Introduction

Large language models (LLMs) are rapidly moving from simple chat interfaces to agentic systems that can plan, call tools, and take actions in complex environments. In security operations, this shift is significant. AI systems are no longer just summarizing alerts or drafting emails; they are exploring data, enriching incidents, opening tickets, and in some cases triggering remediation workflows. This introduces a new class of risk: not only incorrect or fabricated content, but incorrect or unsafe actions taken on behalf of the organization. **Although** examples in this paper reference security and cloud operations, the F7-LAS model applies equally to **finance**, **healthcare**, **IT operations**, **HR**, and **other industries** adopting agentic systems.

Existing governance frameworks such as the NIST AI Risk Management Framework, the EU AI Act, and ISO/IEC AI and governance standards offer essential guidance on trustworthiness, risk management, and organizational responsibility. However, practitioners still need a concrete, technical lens for understanding how these expectations apply to real agentic systems built on top of LLMs, tools, and modern cloud platforms. Security architects and engineers must be able to analyze where risk lives inside an agent's architecture and where controls should be applied.

This whitepaper introduces the **AFuller F7-LAS** (**Fuller 7-Layer Agentic AI Security**) **model**, a practical framework for understanding and securing agentic AI systems. The model organizes an agent's behavior and attack surface into seven layers, from the system prompt and grounding through planning, tools, policy enforcement, sandboxing, and monitoring. Each layer represents a distinct concern: agent instruction, grounding, planning and action, policy enforcement in code, and ongoing observation of behavior over time.

The goal is to give security architects and practitioners a clear, layered way to reason about risk and design controls for AI systems that can take actions, not just generate text. The F7-LAS model is **not**

a replacement for governance frameworks like NIST AI RMF, ISO/IEC 42001, or the EU AI Act; it complements them by providing architectural clarity. It does not map directly to any specific standard; instead, it is designed to complement threat frameworks such as MITRE ATT&CK and MITRE ATLAS by providing a control- and architecture-focused lens on the threats they describe.

While examples in this paper reference SOC workflows, SIEM, SOAR, XDR, and cloud environments, the underlying model is broadly applicable. Any enterprise building or evaluating agentic AI, particularly systems that can call tools, act on digital assets, or operate in partially observable environments—can use F7-LAS as a structured way to ask better questions, identify gaps, and design safer, more trustworthy AI systems.

Appendix B, the F7-LAS Threat–Control Map, provides a structured view of typical threats and defensive controls at each layer and shows how common attack patterns can be mitigated through layered controls.

The core message is simple and enduring: agents are not just chatbots with better prompts and securing them requires more than system prompts and RAG pipelines. By applying the F7-LAS model, organizations can move from ad hoc agent experiments to structured, defensible, and governed agentic AI deployments—systems that act on behalf of the enterprise while remaining within clear, observable, and enforceable boundaries.

1.1 Terminology for Model Errors and Outputs

In this whitepaper, I intentionally avoid the informal term “hallucination” and instead use more precise, professional language to describe model behavior and errors. The goal is to focus on what went wrong and why it matters for security and Responsible AI, rather than relying on anthropomorphic or vague terms.

The following terms are used:

Factual error / Factual inaccuracy
A generated statement that is objectively incorrect when checked against reliable sources or ground truth.

Model error / Generation error
A broad category for any incorrect or undesired output from the model, including wrong answers, off-topic text, or policy-violating content.

Incorrect or spurious output
Neutral phrasing for “wrong text,” without implying intent or human-like confusion.

Fabricated content / Fabrication
Information the model appears to invent, content that is not grounded in the input, retrieved data, or any verified source.

Confabulation
Invented but plausible-sounding text, often used in academic contexts to describe fabricated content presented as if it were true.

Ungrounded output / Ungrounded generation
Output that is not supported by the model's input, retrieved context, or any trusted evidence. This is especially important in RAG scenarios.

Unsupported assertion
A claim made by the model without backing evidence or source references.

False claim / False positive (in content)
A specific type of factual error where the model asserts something untrue as if it were correct.

In enterprise and Responsible AI contexts, the preferred terms in this paper are:

Fabrication / Fabricated content – when the model "makes things up."

Ungrounded output – when an answer is not supported by the available data or retrieved context.

Factual inaccuracy – when truthfulness is the concern.

Unsupported assertion – when the model asserts something without evidence.

Model error / Generation error – when a neutral, umbrella term is needed.

Throughout the whitepaper, these terms are used instead of "hallucination" to keep the focus on verifiable behavior, risk, and controls rather than anthropomorphic metaphors.

2. From Chatbots to Agents

Early large language model deployments in the enterprise were mostly chatbots: systems that accepted natural-language input and produced natural-language output but did not directly act on underlying systems. The primary risks in those deployments centered on content: factual inaccuracies, fabricated or ungrounded outputs, policy-violating responses, and disclosure of sensitive information. These are serious issues, but they affect what is said or shown to users, not what is done to production systems.

Agentic AI changes that picture. An agent built on top of an LLM can interpret a goal, plan a sequence of steps, call tools and APIs, and adapt its behavior based on intermediate results. In security and cloud environments, that might include querying a security vector database or centralized data lake, correlating alerts, enriching entities, opening or updating tickets, calling automation runbooks, or even initiating remediation actions. The risk shifts from purely content risk to a combination of content and operational risk: an incorrect or unsafe decision can now lead directly to incorrect or unsafe actions.

In enterprise settings, we are already seeing agentic AI across security operations, IT operations, and business workflows: copilots that assist analysts while calling investigation tools; agents that automate parts of incident triage; assistants that can modify configuration, manage access, or orchestrate multi-step workflows across multiple systems. These agents are powerful because they compress complex tasks into natural language, but they also expand the attack surface and the potential blast radius if something goes wrong.

Agentic systems also introduce new agent-specific risk factors that go beyond a single prompt–response interaction. These include **goal drift** (the agent gradually pursuing behavior that diverges from the original user intent or organizational policy), **reward hacking** and **specification gaming** (finding ways to satisfy the letter of a success criterion while violating its spirit), unsafe exploration (trying risky actions in the environment without appropriate safeguards), and side effects and impact (unintended consequences of otherwise "successful" actions). These risks are amplified when agents operate in partially observable environments, have access to powerful tools, or are given broad autonomy.

To reason clearly about such systems, we need a structured way to describe what an agent is and how it interacts with its environment. The classic PEAS framework (Performance measure, Environment, Actuators, Sensors) provides exactly that lens. The next section uses PEAS to characterize agentic AI in enterprise contexts and then connects it to the AFuller F7-LAS 7-layer model for securing these systems.

2.1 The PEAS Framework for Agentic AI

The classic PEAS framework (Performance measure, Environment, Actuators, Sensors) is a useful way to describe what an agent is and how it interacts with the world. It was originally introduced for traditional AI agents, but it applies naturally to modern agentic AI systems built on top of large language models.

- **Performance measure (P)** defines what "good behavior" looks like for the agent. In an AI security context, this can include successful task completion, low error rates, adherence to policy, coverage of relevant evidence, and avoidance of unsafe or unauthorized actions. The performance measure is what we ultimately care about when we ask, "Is this agent actually helping, or silently making things worse?"
- **Environment (E)** is everything the agent can operate in or on cloud tenants, applications, data stores, networks, tickets, logs, identity systems, and other platforms. In practice, the environment largely determines what kind of agent we are building—its feasible goals, the kinds of tasks it can perform, which tools it needs, and what capabilities are required to complete its primary objectives.

In partially observable environments, the way the agent maintains and updates memory (its model of the environment) is also shaped by this context: what it can see, what it cannot see, how delayed or noisy its observations are, and how long information remains relevant. Two agents with the same base model but different environments (for example, a SOC investigation

environment versus an IT helpdesk environment) effectively become different agentic systems because their tasks, tools, memory needs, and success criteria are all driven by the environment they are placed in.

- **Actuators (A)** are how the agent acts on that environment. For LLM-based agents, these are the tools and integrations they can invoke: security APIs, query interfaces, automation runbooks, ticketing systems, configuration endpoints, and other actions that change state. Actuators are where "just a chatbot" becomes a system that can quarantine a device, close an incident, or push a policy change.
- **Sensors (S)** are how the agent perceives the environment: user prompts, retrieved documents, log entries, alerts, tool responses, API results, monitoring data, and other inputs. In many enterprise scenarios, the "sensors" are a mix of RAG retrieval over internal data, structured security events, and free-form natural language from analysts or end users.

For agentic AI, PEAS clarifies that we are not just adding a chat interface on top of existing systems. We are defining an agent with specific goals (**P**), operating in a particular environment (**E**), using a defined set of actuators (**A**), and consuming observations from multiple sensors (**S**). Changing any one of these elements can turn it into a very different agent.

This whitepaper's **7-Layer Agentic AI Security Model** builds on that idea. Where PEAS describes what the agent is, the 7-layer model focuses on how we secure it:

- The **Performance measure** influences how we design the **system prompt** (Layer 1) and how we evaluate the agent in **monitoring and evaluation** (Layer 7).
- The **Environment** is shaped and constrained by the **sandboxed execution environment** (Layer 6) and by how we expose data and context through **RAG** (Layer 2).
- The **Actuators** correspond directly to **tools and integrations** (Layer 4), which define the agent's action surface and potential blast radius.
- The **Sensors** map both to how the agent ingests information through **RAG and tool outputs** (Layers 2–4) and to how the organization observes the agent through **monitoring and logging** (Layer 7).

Together, PEAS and the 7-layer model provide a consistent way to think about both **how agentic AI is built** and **how it should be secured** in real enterprise environments.

2.2 Environment Properties for Agentic AI

Beyond PEAS, it is useful to characterize the **type of environment** an agent operates in. Classic AI literature highlights several dimensions that strongly affect the difficulty and risk profile of an agent:

- **Observability:** *Fully vs. partially observable.*
  In a fully observable environment, the agent can see the complete state relevant to its decisions. In security operations and cloud environments, the reality is usually **partially**

observable: telemetry is delayed, incomplete, or noisy, and important signals may be missing entirely.

- **Agency**: *Single-agent vs. multi-agent*.
  Many enterprise scenarios are effectively **multi-agent**: human analysts, automated workflows, external attackers, and multiple AI agents all acting in the same space. This increases coordination challenges and the potential for unexpected interactions.
- **Knowledge**: *Known vs. unknown*.
  In some domains, the rules and dynamics are well understood; in others, the agent must operate under **uncertainty and knowledge gaps**, discovering patterns as it goes. Security operations often mix both: known playbooks and unknown attacker behavior.
- **Determinism**: *Deterministic vs. stochastic*.
  Deterministic environments behave predictably given the same inputs, while **stochastic** environments include randomness and noise. Real-world systems, especially at scale, are largely stochastic from the agent's perspective.
- **Time structure**: *Episodic vs. sequential; discrete vs. continuous*.
  In episodic settings, each decision is independent; in **sequential** settings, actions have long-term consequences and create state that future decisions must respect. Security and IT operations are inherently sequential and often a mix of discrete events and near-continuous monitoring.
- **Time pressure**: *Static vs. dynamic*.
  Static environments change slowly, if at all. Dynamic environments evolve while the agent is still reasoning or acting. Incident response, threat hunting, and cloud configuration all happen under **dynamic** conditions, sometimes with real-time pressure.

The hardest environments combine multiple challenging properties: **partially observable, multi-agent, stochastic, sequential, dynamic, and continuous**. This describes many real-world cyber and cloud environments, where agents must act with incomplete information, in parallel with humans, multiple AI agents, and other systems, under uncertainty and time pressure.

For the **AFuller F7-LAS model**, these properties matter in several layers:

- They influence how we define the **Environment (E)** in PEAS and how we scope the **sandboxed execution environment** in Layer 6.
- Partial observability and stochastic dynamics drive the need for **RAG and memory** (Layer 2) and careful **monitoring and evaluation** (Layer 7).
- Multi-agent, dynamic settings increase the importance of a **robust policy engine** (Layer 5) and clear separation of responsibilities between humans, tools, and agents.

When analyzing a new agentic AI use case, it is useful to first classify the environment along these dimensions and then apply the F7-LAS layers, so that both **architecture** and **controls** are appropriate to the environment's difficulty.

Multi-Agent Environments.
F7-LAS extends naturally to multi-agent systems. Each agent is instantiated with its own Layer 1–5 configuration (prompt, grounding, planner, tools, and policy constraints), while Layers 6–7 manage shared but segmented execution environments and telemetry. All governance, actions, and observations are keyed by **agent identity**, enabling cross-agent coordination, auditability, and lifecycle assurance. Multi-agent environments significantly increase requirements on Layer-5 policy enforcement and Layer-7 monitoring, as inter-agent interactions must be governed and observable.

2.3 Security Architecture Depth: Patterns and Examples

The F7-LAS model is intentionally abstract so it can apply across different platforms and vendors. To make it concrete for implementation reviews, this section highlights three example patterns that show how the layers map into real architectural choices: a policy engine pattern, a telemetry pattern, and a tool-tiering model. A consolidated summary of these and related architectural patterns appears in ***Appendix F***, Example Implementation Patterns, which can be used as a field checklist during design or security reviews.

2.4 Policy Engine Pattern: PDP/PEP with Policy-as-Code

A common way to implement Layer 5 (Policy Engine outside the LLM) is the familiar PDP/PEP pattern from zero-trust and API security:

- **Policy Decision Point (PDP)**
    - Evaluates policies written as code (for example, Rego in Open Policy Agent or an equivalent engine).
    - Takes as input: the requested tool, parameters, calling identity, environment, and risk context.
    - Returns a simple decision: **allow**, **deny**, or **allow with conditions** (for example, "requires human approval").
- **Policy Enforcement Point (PEP)**
    - Sits in line between the planner (Layer 3) and tools (Layer 4).
    - For every tool call, it sends an authorization request to the PDP and enforces the decision before anything is executed.
    - Logs each decision for Layer 7.
- In an agentic security use case, you might express a rule such as:

- Remediation tools that modify access or configuration may only be invoked from a high-assurance identity, in a production sandbox, with a human approval token, and only during approved maintenance windows."
- The key point is that the agent never calls tools directly. Every action proposal flows through a centralized PDP/PEP gateway, making Layer 5 a single, auditable choke point for enforcement.

2.5 Monitoring Pattern: Tool-Call Telemetry Schema

For Layer 7 (Monitoring & Evaluation), it helps to standardize telemetry for every tool invocation, regardless of platform. A simple JSON event schema can serve as the foundation for SIEM/XDR analytics, AI risk dashboards, and ATLAS-aligned detections:

Figure 1 - *JSON Event Schema for Layer 7 Monitoring and Evaluation*


```
{
 "timestamp": "2025-11-15T14:32:07Z",
 "agent_id": "soc-assistant-v3",
 "session_id": "c1b2e3...",
 "user_id": "analyst@acme.local",
 "environment": "prod-soc-sandbox-01",
 "layer": "L4-tools",
 "tool_name": "close_incident",
 "tool_category": "action",
 "tool_version": "1.4.2",
 "policy_decision": "allow-with-approval",
 "approval_id": "change-req-98765",
 "request_parameters": {
  "incident_id": "INC-123456",
  "close_reason": "false-positive",
  "comment_length": 124
 },
 "response_status": "success",
 "response_summary": "incident closed",
 "risk_score": 72,
 "atlas_techniques": ["TA0048", "TA0005"],
 "trace_ids": {
  "prompt_id": "p-09ab...",
  "plan_step": 4
 }
} "response_status": "success",
 "response_summary": "incident closed",
 "risk_score": 72,
 "atlas_techniques": ["TA0048", "TA0005"],
 "trace_ids": {
  "prompt_id": "p-09ab...",
  "plan_step": 4
 }
}
```


---

3. The Fuller 7-Layer Agentic AI Security (F7-LAS) Model (Overview)

When AI systems stop just chatting and start taking actions through tools and APIs, it's no longer enough to say, ***"we secured it with a prompt and RAG."*** Those are important, but they only cover the *top* of the stack.

To reason about the real risk and needed controls, I use the F7-LAS model, which is a 7-layer security model framework for Agentic AI security. In a multi-agent architecture, F7-LAS is instantiated **per agent**. Each agent has:

- Its own **Layer-1 system prompt** (role, scope, responsibilities).
- Its own **Layer-2 grounding profile** (curated read-only sources appropriate to its function).
- Its own **Layer-3 planner configuration** (step limits, action constraints, safe-planning patterns).
- Its own **Layer-4 tools**, partitioned by role (e.g., read-only tools for Investigator; workflow tools for Coordinator; privileged response tools for Remediator).
- Its own **Layer-5 policy rules** that determine which actions, delegations, and human-approval paths are permitted.

Layers 6–7 operate as a **shared substrate** that enforces blast-radius control, agent identity boundaries, and cross-agent telemetry. This separation prevents accidental or unauthorized privilege amplification across agents.

Figure 2 - provides a high-level view of the AFuller F7-LAS model (Conceptional Stack). **Image Spell-4**

Figure 2 provides a high-level view of the AFuller F7-LAS model as a stack of seven layers.

The F7-LAS model can be used as a lightweight checklist during architecture and design reviews. For each layer, ask at least:

Layer 1 – System Prompt (Soft Policy)
• Is the agent's role, scope, and safety posture clearly defined?
• Does the prompt tell the agent when to abstain, defer, or escalate instead of fabricating?

Layer 2 – RAG / Grounding (Epistemic Guardrail)
• What data sources ground the agent, and who governs them?
• Can we trace which sources supported a given answer?

Layer 3 – Agent Planner / Controller
• How does the agent decide which tools to use and when to stop?
• Are there clear limits on depth and length of plans and loops?

Layer 4 – Tools & Integrations (Action Surface)
• What tools exist, and which ones can change state or configuration?
• Which identities and permissions are used to call them?

Layer 5 – Policy Engine Outside the LLM
• Which actions are always blocked, which require human approval, and which are allowed?
• How are instruction-injection attempts or policy-violating requests detected handled?

Layer 6 – Sandboxed Execution Environment (Blast Radius Control)
• In which environments, accounts, or networks does the agent run?
• What is the maximum blast radius if the agent or its tools are misused?

Layer 7 – Monitoring & Evaluation (Detection & Assurance)
• What do we log about prompts, tool calls, policy decisions, and outcomes?
• How do we evaluate behavior over time and ensure the system remains aligned with governance, safety expectations, and organizational policies?

Used this way, F7-LAS becomes a structured review lens, not a separate process: the same questions can be applied to new designs, existing agents, and production deployments.

4. Layer-by-Layer Deep Dive

These seven layers represent conceptual control planes; real-world implementations often span multiple layers. **THIS SECTION NEEDED OR SHOULD IT BE MOVED ABOVE, OR MORE CONTENT?**

4.1 Formal Underpinnings of F7-LAS: A POMDP View of Agentic AI

Agentic AI systems can be viewed through the formal lens of a *Partially Observable Markov Decision Process* (POMDP).
This perspective provides a mathematical foundation for how F7-LAS structures decision-making, uncertainty, and control across its seven layers.

In brief, a POMDP models an agent operating in an uncertain environment where it cannot directly observe the full system state.
The agent must infer the likely state of the world, choose actions based on that belief, and update its understanding from the outcomes it observes.

A POMDP is classically defined as a tuple $\boldsymbol{M} = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{T}, \boldsymbol{\Omega}, \boldsymbol{O}, \boldsymbol{R}, \boldsymbol{\gamma})$, where:

$\boldsymbol{S}$ - the set of possible environment states (e.g., system posture, alert queues, configurations, or active adversary behaviors).

$\boldsymbol{A}$ - the set of possible actions the agent can take (e.g., query data, open tickets, apply remediations).

$\boldsymbol{T}(s' \mid s, a)$ - the transition model, describing how actions change the underlying state.

$\boldsymbol{\Omega}$ - the set of possible observations (retrieved data, prompt responses, telemetry).

$\boldsymbol{O}(o \mid s')$ - the observation model, describing the likelihood of seeing an observation o given a hidden state s'.

$\boldsymbol{R}(s, a)$ - the reward or utility function that encodes goals, penalties, or alignment incentives.

$\boldsymbol{\gamma} \in (0,1)$ - a discount factor controlling how much the agent values future outcomes.

Because the agent cannot directly observe *S*, it maintains a *belief state*, a probability distribution over possible environments and continuously updates that belief as it takes actions and receives observations.
Its behavior is governed by a *policy* π, which maps beliefs to actions to maximize its expected long-term reward.

4.2. Mapping POMDP Components to F7-LAS Layers

F7-LAS can be interpreted as decomposing the POMDP into layers that represent where specific controls and safety constraints apply:

- **Layer 1** - System Prompt (Soft Policy):
  Encodes the initial objective and reward function *R*, guiding the agent's intent and ethical bounds ("reduce risk," "avoid fabrication," "escalate when uncertain").
- **Layer 2** - RAG / Grounding (Epistemic Guardrail):
  Shapes the observation model *O* and the quality of belief updates by constraining which information sources can be retrieved. Grounded retrieval reduces epistemic uncertainty.
- **Layer 3** - Agent Planner / Controller:
  Implements the policy π that selects actions given the current belief state - whether via chain-of-thought reasoning, tree search, or rule-based control. *Shouldn't we mention ToT, because CoT is more for LLMs, stand-alone knowledge where ToT, is more based on Agentic Agent, tree style reasoning*?

- **Layer 4** - Tools & Integrations (Action Surface):
  Defines the concrete action set *A*. Each tool corresponds to an action or family of actions that modify or query the environment, influencing the transition dynamics *T*.
- **Layer 5** - Policy Engine Outside the LLM (Hard Guardrails):
  Imposes a safety filter F(b) ⊆ A, restricting which actions are allowed based on policy, identity, or risk context. This is realized through mechanisms like PDP/PEP enforcement and policy-as-code.
- **Layer 6** - Sandboxed Execution Environment (Blast Radius Control):
  Restricts the reachable state space S' ⊆ S, effectively redefining transitions within controlled environments. Even if the planner misbehaves, its actions are contained within a limited scope.
- **Layer 7** - Monitoring & Evaluation (Detection & Assurance):
  Observes trajectories of states, actions, and outcomes, defining a safety function J(π) that measures violations, near misses, or other assurance indicators.
  This layer closes the loop by feeding safety metrics and incident data back into Layers 1–6 for refinement.

4.3 Security as a Constrained Decision Process

Viewed through a security lens, F7-LAS turns the classic POMDP into a constrained decision process. Instead of merely maximizing rewards, the system must also respect operational and safety limits.

Conceptually, the agent's goal becomes:

Maximize: expected performance
Subject to:

actions filtered by policy and approval (Layers 4–5)

effects confined to a restricted state subset (Layer 6)

global safety below acceptable risk (Layer 7)

In other words:

> **Layers 1–3** define intent and the reasoning process (what the agent should do).
>
> **Layers 4–6** define execution boundaries (what the agent is allowed to do).
>
> **Layer 7** defines evaluation and assurance (whether it stayed within limits).

This framing shows that F7-LAS does not replace mathematical rigor—it localizes it.
It provides a decision-theoretic foundation that bridges security controls, operational safeguards, and the underlying mathematics of agentic AI systems.

Multi-Agent Constrained POMDP.
A multi-agent F7-LAS system becomes a collection of independent constrained POMDPs, each with its own approximate policy π_i, action set A_i, reward structure R_i, and filtered transition model F_i (A). Delegation becomes a restricted meta-action available only to authorized agents (e.g., the Coordinator). All local and delegated actions are mediated through Layer-5 policy enforcement and executed within Layer-6 sandboxes, ensuring that multi-agent reasoning cannot bypass organizational controls.

Figure 3 - Formal Objective of F7-LAS as a Constrained POMDP

$$\max_{\pi} \ \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \quad \text{s.t.} \quad a_t \in F(b_t) \subseteq A, \ \ s_t \in S' \subseteq S, \ \ J_{\text{safety}}(\pi) \leq \kappa$$

Figure 3 Constrained POMDP objective for the AFuller F7-LAS model: maximize expected reward subject to action filtering, sandboxed state space, and a global safety constraint.

Table 2 — Mapping the POMDP Formalism to the F7-LAS Layers

F7-LAS Layer	POMDP Element	Security Interpretation
1-System Prompt (Soft Policy)	Soft objective shaping & initial policy hint (π_0)	Defines goals, abstentions, and intent, but not R(s, a).
2-RAG / Grounding	Belief state *b(s)*	Reduces epistemic uncertainty through vetted external sources.
3-Agent Planner / Controller	Policy formation $\pi: H \rightarrow A$	Generates the actionable policy approximation from history/context.
4-Tools & Integrations	Action set *A;* transition model *T(s, a, s')*	Defines executable transitions and environment mutations.
5-Policy Engine	Safety filter $F \subset A$	Enforces constraints, approvals, and external policy gating.
6-Sandboxed Environment	Constrained sate subset $S' \subset S$	Limits reachable states and bounds consequences.
7-Monitoring & Evaluation	Observation model Ω and trajectory evaluation	Detects drift, evaluates safe operation, and informs governance.

This connection formalizes what F7-LAS achieves in practice:

- Layers 1–3 define intent and generate the policy approximation.
- Layers 4–5 restrict and audit *action selection*.
- Layers 6–7 limit *environmental consequence* and *provide feedback*.

The model thus instantiates a "secure **POMDP**", where both epistemic and operational uncertainties are bounded by layered controls.

Figure 4 - Alignment of the F7-LAS Model with the POMDP Framework

Figure 4 Alignment of the F7-LAS Model with the POMDP Framework Each component of the POMDP formalism maps to one or more layers of the F7-LAS model, showing how the agent's decision and observation processes correspond to practical security control layers.

Interpretation. Each element of the agent's decision process—belief state, policy, reward shaping, transitions, and observations—maps to a corresponding *security control plane* within the F7-LAS model. Whereas traditional POMDPs optimize task performance, a Secure-POMDP framing instead optimizes *bounded safety*: maximizing reliability under uncertainty while maintaining alignment with policy, sandbox, and oversight constraints.

This perspective positions F7-LAS not merely as a heuristic framework but as a *structured control decomposition* of the agent's decision-theoretic anatomy. In formal terms, it reframes agent safety as a *constrained optimization problem over π*, bounded by governance intent (Layer 1), epistemic trust and grounding (Layer 2), and observable feedback loops (Layer 7).

For practitioners, this means that every control layer is mathematically justified as a constraint on uncertainty propagation, the true root cause of agentic risk.

4.4 Layer 1 – System Prompt (Soft Policy)

The system prompt is the agent's "operating system in natural language." It defines the agent's role, objectives, constraints, and safety expectations. Instead of code, we use natural language to say things like: *"You are a security assistant. Use only approved data sources. If you are unsure, say you don't know rather than guessing."* This is usually the first and most visible layer of control in an agentic AI system.

This layer matters because it strongly shapes default behavior. A well-designed system prompt can nudge the model toward safer, more relevant, and more consistent actions. It provides a place to encode organizational norms and security expectations without changing model weights or infrastructure. For many teams, this is the fastest way to improve the behavior of an AI agent.

The main risk is that system prompts are **soft policy, not hard enforcement**. The model can still be influenced by user input, retrieved documents, or tool outputs, and may not always follow the instructions exactly. Prompt injection attacks can attempt to override or contradict the system prompt, and there is no guarantee the model will resolve the conflict in the way we intend. Relying solely on this layer leads to a false sense of security; it must be complemented by the deeper layers that implement policy in code, access control, and environment design.

4.4.1 Layer 2 – RAG / Grounding (Epistemic Guardrail)

Retrieval-Augmented Generation (RAG) connects an agent to external knowledge sources such as internal documents, vectorized knowledge bases, logs, or configuration data. Rather than relying solely on information learned during pretraining, relevant content is segmented (chunked), embedded into numerical vector representations, and stored in a vector database.

To enhance retrieval precision, the system employs a **Query Optimizer** that reformulates user inputs into semantically enriched search prompts before embedding and retrieval. This optimizer works in conjunction with the RAG pipeline to ensure that the agent accesses the most relevant and contextually trustworthy information.

At query time, the agent retrieves the most semantically similar chunks and injects them into the prompt as **context**, enabling it to answer using organization-specific data rather than relying on probabilistic inference or "guessing."

This layer matters because it acts as an epistemic guardrail: it improves truthfulness and relevance. Grounding responses in enterprise data reduces **fabricated and ungrounded outputs**, enables attribution ("*this answer came from these documents*"), and allows the agent to stay current as information changes without retraining the model. In security scenarios, RAG can tie the agent's answers to specific policies, playbooks, logs, and architecture diagrams instead of generic internet knowledge.

However, RAG is not a complete security control by itself. It does not decide **who** is allowed to see **which** data; that depends on how we design access control and metadata filters. If the underlying documents are poisoned, outdated, or misclassified, the agent will confidently repeat those problems. Retrieved text can also become a vehicle for prompt injection if an attacker manages to insert malicious instructions into the corpus. RAG should be treated as a powerful way to ground answers, but it must be combined with robust access controls, validation of sources, and the deeper layers that govern tools, policy, and environment.

4.4.2 - Layer 3 – Agent Planner / Controller (Orchestration)

The agent planner, or controller, is responsible for deciding **what to do next**. Instead of treating each prompt as a one-off question, the planner runs a loop: it interprets the user's goal, reasons about the next step, chooses a tool to call (if needed), inspects the result, and then repeats until it believes the task is complete. In many frameworks this shows up as "thought → action → observation" cycles, or as a planning graph that chains multiple steps together.

This layer matters because it is what turns a language model into an **agent**. It's how we get from "summarize this" to "investigate this alert, pull related events, correlate across data sources, and create a ticket with a recommendation." The planner gives the system flexibility and autonomy: it can decompose tasks, react to intermediate results, and handle multi-step workflows instead of just answering a single question.

From a security perspective, the planner should be treated as **untrusted orchestration logic**. It can propose bad plans, loop too long, or drift away from the original-goal intent of the task. If we rely on the planner alone to decide when to call powerful tools or make high-impact changes, we are effectively letting a probabilistic model act as a decision engine for our environment. The deeper layers tool design, external policy, sandboxing, and monitoring which exist to constrain and supervise whatever the planner decides to do, so that a flawed plan does not become a damaging action.

4.4.3 ReAct as a Planner Pattern

One common way to implement Layer 3 – Agent Planner / Controller is the ReAct (Reason + Act) pattern. In a ReAct-style agent, the LLM cycles through a loop of:

Thought – reason about the current goal and state

Action – select a tool or API to call

Observation – receive the result and update its understanding

---

Figure 5 – ReAct as a Planner Pattern

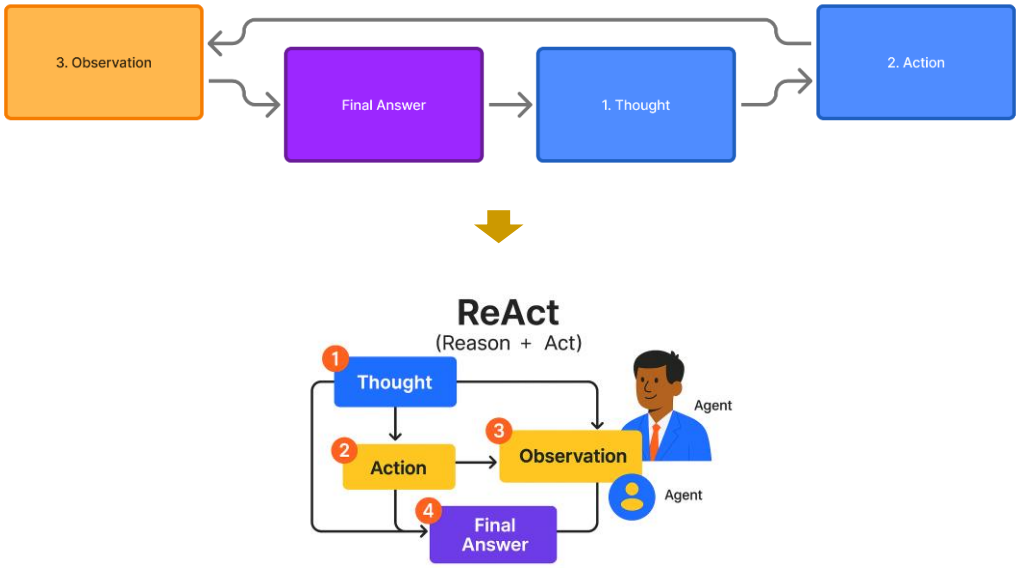

Figure 5 ReAct as a Planner Pattern

This loop continues until the agent decides it has enough information to produce a final answer or complete the task. Conceptually, ReAct is a concrete instance of the Layer 3 planner: it is the logic that **proposes multi-step plans**, chooses which tools to use, and decides when to stop.

From a security perspective, this means that safety and control cannot only live in the system prompt or in the tools themselves (Layer 4). The **planning loop** can still make poor or unsafe choices, calling the wrong tools, calling tools too often, or continuing to act when it should stop. In the **F7-LAS model**, ReAct-style agents are treated as **one implementation of the Agent Planner**, which still must be constrained by a policy engine (Layer 5), a sandboxed environment (Layer 6), and monitoring and evaluation (Layer 7).

4.4.4 Layer 4 – Tools & Integrations (Action Surface)

Tools and integrations are the actuators of an agentic AI system—they define what the agent can actually do in its operational environment.
For security agents, this includes querying security data, searching long-term logs, correlating signals, enriching alerts, opening or updating tickets, triggering automation workflows, or even taking direct remediation actions within production systems.

Across modern Security Operations Platforms, including SIEM, SOAR, and XDR solutions—this *action surface* is becoming more explicit and structured.
Contemporary architectures combine unified observability stores or audit lakes, graph-based context models, and standardized tool-calling interfaces such as the Model Context Protocol (MCP) or similar API frameworks.
Together, these components expose scenario-focused tool collections—for example, tools for long-term data exploration, identity investigations, or automated response orchestration—through consistent, hosted endpoints that agents can call programmatically.

From an agent's perspective, these tools appear as callable capabilities:
"explore security data for this user over 180 days," "correlate file activity with sensitivity labels," or "initiate incident triage on this host."
The MCP or equivalent interface abstracts away schema and query complexity, allowing the agent to reason over vector databases, graph contexts, and audit repositories using natural language, while the underlying platform manages discovery, retrieval, and shaping of security-relevant results.

This layer is where *blast radius* becomes tangible. ***SHOULD SENTENCES BE PULLED TOGETHER***
If a tool only queries data, the primary risk is over-disclosure or misuse of sensitive information.
If a tool can close incidents, disable accounts, modify policies, or trigger remediation workflows, the risk shifts to operational impact—the agent could misconfigure defenses, hide attacker activity, or disrupt business operations if misused or compromised.
The same unified tool surface that accelerates threat hunting and automation can equally amplify mistakes when tools are over-privileged or poorly governed.

Securing this layer means treating tools as first-class security objects, not just helper functions.
For each tool or collection, an organization should be able to answer:

- What data can this tool access (tables, time ranges, tenants, environments)?
- What actions can it perform (read-only versus state-changing operations)?
- Which identities—human or agent—are authorized to call it, and under what conditions?
- How are tool calls logged, monitored, and rate-limited?
- What protections exist if the tool is misused through a compromised agent, prompt failure, or injection attack?

A mature design exposes segmented tool collections for exploration, enrichment, and automation, each governed by progressively stronger controls.
Exploration tools may be broadly available to analysts and test agents in non-production environments, while automation or policy-modifying tools should require elevated privileges, identity assurance, and approval workflows enforced through Layer 5 (Policy Engine).

Within the F7-LAS framework, Layer 4 defines the agent's explicit action surface—the boundary between reasoning and execution.
As the tool layer grows more powerful and convenient, supported by standardized calling protocols and unified observability architectures, the need for disciplined design increases in equal measure. Security requires least privilege, a clear separation between read and write capabilities, and tight integration with policy enforcement (Layer 5), sandboxing (Layer 6), and continuous monitoring (Layer 7).

4.4.5 Layer 5 – Policy Engine Outside the LLM (Hard Guardrails)

The policy engine outside the LLM is where **soft intent becomes hard enforcement**. While the system prompt and RAG layer express what we want the agent to do, the policy engine encodes what the system is actually **allowed** to do—regardless of how the model reasons or what instructions appear in the prompt or retrieved context.

At this layer, policies are implemented in code and configuration, not just in natural language. Examples include:

- Allowing only read-only tools in certain environments
- Requiring human approval before high-impact actions (e.g., disabling accounts, changing conditional access, running destructive scripts)
- Blocking or redacting specific categories of content (e.g., secrets, regulated data, sensitive attributes)
- Enforcing rate limits, scopes, or contextual constraints on tool usage
- Dropping or rewriting inputs that contain instruction injection patterns or policy-violating requests

From a governance perspective, this is the layer where many expectations from frameworks such as the **NIST AI Risk Management Framework (AI RMF)**, **ISO/IEC 42001**, **ISO/IEC 23894**, and the **EU AI Act** start to become operational in a concrete way.

The NIST AI RMF emphasizes four high-level functions **Govern, Map, Measure,** and **Manage** across the AI lifecycle to support trustworthy AI, including security, robustness, accountability, and transparency. A policy engine is one of the primary mechanisms to "**Manage**" risk in deployed systems: it encodes risk controls (permissions, thresholds, approvals, logging requirements) that apply every time an agent uses a tool, not only when the prompt is well behaved.

Figure 6 - F7-LAS Policy Engine in the Execution Control Loop

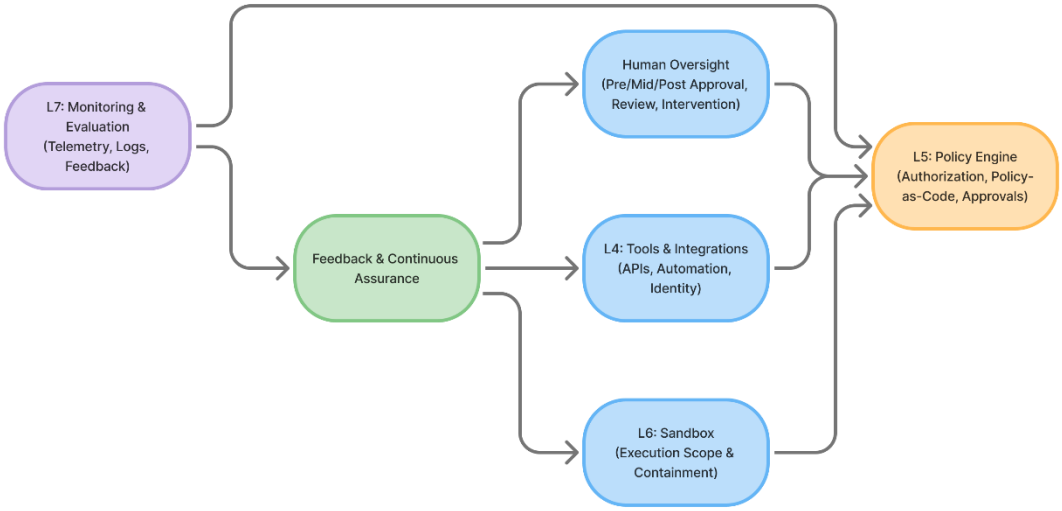

Figure 6 - F7-LAS Policy Engine in the Execution Control Loop
This diagram shows how Layers 4–7 converge into the Layer-5 Policy Engine, which serves as the central enforcement point for authorization, hard guardrails, sandbox constraints, telemetry-driven adaptation, and human oversight across the agent execution lifecycle.

Similarly, **ISO/IEC 42001** defines requirements for an AI Management System (AIMS) so organizations can establish, implement, maintain, and continually improve policies and processes for responsible AI across the lifecycle. The policy engine is one of the main technical places where those AIMS policies are enforced at runtime, linking documented principles (e.g., acceptable use, safety criteria, segregation of duties) to concrete rules that govern agent behavior and tool usage.

ISO/IEC 23894 provides guidance on AI-specific risk management: identifying, assessing, treating, and monitoring AI-related risks across the lifecycle. In practice, many of the selected risk treatments (for example, restricting certain actions, requiring additional evidence before remediation, or enforcing human oversight for high-risk workflows) are implemented in this policy engine layer. It becomes the **bridge** between risk registers and real-time control of agentic actions.

The **EU AI Act** reinforces the same pattern from a regulatory angle. For high-risk AI systems, it requires documented risk management, technical and organizational controls, human oversight, logging, monitoring, and clear instructions for use. For deployers, that includes monitoring operation, ensuring input data is appropriate, keeping logs, and intervening when risks are detected. A policy engine that sits outside the LLM and mediates every tool call is one of the most

direct ways to align agentic systems with these obligations: it enforces separation between what the model **suggests** and what the system will **permit** under regulation and internal policy.

Higher-level governance standards such as **ISO/IEC 38507** (governance implications of AI), **ISO/IEC 22989** (AI concepts and terminology), **ISO/IEC 23053** (framework for ML-based AI systems), and **ISO/IEC TR 24028** (trustworthiness in AI) all converge on themes like organizational accountability, clarity of roles, lifecycle control, and trustworthiness dimensions (safety, security, reliability, transparency, privacy). In an agentic context, Layer 5 is one of the main places those governance expectations become **machine-enforceable rules**, ***not just documents on a shelf***.

A mature policy engine for Agentic AI should be able to answer questions like:

- Which **actions** are always blocked, which are allowed, and which require a human in the loop?
- How do we encode **organizational policy** (e.g., least privilege, segregation of duties, change control) into rules that govern agent tool use?
- How do we treat **instruction injections** or policy-violating prompts, do we drop them, sanitize them, or escalate them?
- How do we ensure that actions taken by the agent are **traceable, explainable at the oversight level, and auditable** for compliance?

In the 7-layer model, Layer 5 is where we stop trusting "the model will behave" and instead treat the model as an untrusted planner whose proposals must flow through a **policy-controlled gateway**. This aligns directly with modern AI risk and governance frameworks: policy lives at the governance and risk-management level, but it is **implemented and enforced here**, at the point where intent meets action.

4.4.6 Human Oversight Patterns (Pre-, Mid-, Post-Action)

Human oversight remains an essential safeguard in agentic AI security design. While the policy engine establishes enforceable boundaries around what an agent *can* and *cannot* do, oversight defines how and when human judgment is applied to govern those boundaries in practice.
In the F7-LAS model, oversight complements policy enforcement by ensuring that decisions with operational, ethical, or compliance impact are subject to the right level of human control at the right time.

Human-in-the-loop design can be understood through three recurring oversight patterns:

Pre-Action Oversight — Preventive Control

Pre-action oversight occurs *before* an agent executes a task or issues a command.
It applies to operations where the consequence of error or misuse is high, such as configuration changes, credential revocations, or enforcement of conditional-access policies.
In these cases, the policy engine triggers approval workflows, requiring a human operator to review

the proposed plan, data inputs, and intent before authorizing execution.
This pattern provides the strongest assurance of governance alignment, at the cost of some automation speed, and is common in regulated or production environments.

Mid-Action Oversight — Supervisory Control

Mid-action oversight introduces adaptive human intervention during execution.
Here, an agent operates autonomously within a controlled policy envelope, but its intermediate actions and tool calls are surfaced for live human review or intervention when thresholds are exceeded.
Examples include approving escalation paths, confirming destructive actions, or aborting workflows if the agent's reasoning diverges from policy intent.
This pattern balances autonomy and safety, maintaining responsiveness while allowing human correction before material impact occurs.

Post-Action Oversight — Detective and Corrective Control

Post-action oversight occurs *after* the agent's action has completed.
It relies on telemetry from Layer 7 (Monitoring & Evaluation), including audit trails, drift detection, and incident analytics, to verify whether actions were appropriate, compliant, and effective.
Where deviations or anomalies are detected, humans perform retrospective review, trigger corrective remediation, or adjust policies and prompts for future cycles.
This oversight pattern is essential for continuous assurance and forms the feedback loop that strengthens the governance cycle over time.

In practice, robust agentic governance combines all three patterns.
Pre-action oversight prevents unauthorized or unsafe changes; mid-action oversight supervises bounded autonomy; and post-action oversight verifies and improves outcomes through feedback.
Together, they anchor Layer 5 as the convergence point of policy, accountability, and human judgment—ensuring that even as agents act independently, the enterprise retains ultimate operational and ethical control.

4.4.7 Layer 6 – Sandboxed Execution Environment (Blast Radius Control)

The sandboxed execution environment is where we decide **where the agent lives and what it can actually reach**. Even with a careful tool layer and a strong policy engine, an agent that runs with broad tenant, network, or data access can still cause significant harm if it behaves incorrectly, is misconfigured, or is influenced by adversarial input. **Layer 6** is about shaping the environment so that, even when something goes wrong at the model or tool level, the **blast radius is constrained by design**.

In practice, this layer includes decisions about **tenants, identities, permissions, networks, and runtime isolation**. For example: running the agent in a dedicated subscription or tenant; using constrained managed identities with least privilege; isolating the agent in a restricted VPC or network segment; separating staging, test, and production environments; and explicitly scoping

which data sources, systems, or workloads are reachable from where the agent executes. For some use cases, this also means separating read-only analytics environments from environments that can change configuration, policy, or access.

From a security perspective, the sandboxed environment should be designed under the assumption that the **planner is untrusted** and that tools may eventually be misused, whether through model error, instruction injection, or operator mistakes. That leads to questions such as: if this agent is compromised or behaves unexpectedly, which systems can it touch? Which credentials can it use? Which networks can it traverse? Which data stores can it query or modify? A well-designed sandbox ensures that the answers are tightly bounded and aligned with least privilege and segregation of duties.

For organizations following frameworks such as NIST AI RMF, ISO/IEC 42001, ISO/IEC 23894, or the EU AI Act, this layer is where many of the requirements around **technical controls, environment isolation, and operational safeguards** are realized. Defense-in-depth for agentic AI does not stop at prompts, tools, or policy rules; it depends on a runtime environment that is intentionally scoped so that even a "worst case" agent failure results in a **contained incident**, not a full-scale production crisis.

In practice, an agent's blast radius is not defined by the model itself, but by the combination of **which tools it can call (Layer 4), what those tools are allowed to do under policy (Layer 5), and where they are allowed to run and reach (Layer 6)**. A common rule of thumb is that an agent's **blast radius** is only as large as the tools and environments it is permitted to access, but in F7-LAS terms, that means we must design tools, policies, and sandboxes together, not in isolation.

4.4.8 Layer 7 – Monitoring & Evaluation (Detection & Assurance)

Monitoring and evaluation are how we **see what the agent is actually doing** over time and determine whether it remains safe, effective, and aligned with policy. Even with strong prompts, RAG, tools, policy engines, and sandboxing, an agentic AI system will still evolve as data, environments, and usage patterns change. **Layer 7** is about treating the agent as a **living system** that requires ongoing observation, testing, and adjustment, not a one-time deployment.

At a minimum, this layer should provide enough **observability** to reconstruct what the agent did, why it did it, and with what impact. That typically includes logging:

- **User inputs and high-level tasks** (appropriately protected and minimized)
- **System prompts and key context** used for major actions, where allowed
- **Tools invoked** (which tool, which parameters, which identity, which environment)
- **Tool responses and outcomes** (including errors and partial failures)
- **Policy engine decisions** (allowed, blocked, escalated, or modified actions)

- **Environment-level events** (such as changes made to case tickets, configurations, or access)

These logs support both **security monitoring** (detecting misuse, abuse, or attacks such as instruction injection) and **Responsible AI oversight** (identifying fabricated or ungrounded outputs, policy violations, and systematic failure modes). For high-impact workflows, it should be possible to trace a line from a final action, such as a remediation step or configuration change back through the relevant agent decisions, tool calls, and approvals that led to it.

Evaluation is the second half of this layer. Where monitoring tells us **what happened**, evaluation asks **how well the system is performing against its goals and constraints**. For Agentic AI, this typically includes a mix of:

- **Offline evaluations**, such as replaying representative scenarios or red team prompts and scoring the agent's behavior against safety and effectiveness criteria.
- **Online evaluations**, such as tracking error rates, policy violations, overrides, or human escalations in production.
- **Targeted red teaming** of agent workflows, focusing on instruction injection, tool misuse, data exfiltration, and boundary-crossing behavior.
- **Drift and degradation analysis**, watching for changes in behavior as models, tools, data, or environments evolve.

From a governance standpoint, Layer 7 connects directly to the **"Measure" and "Manage"** functions in NIST AI RMF and to the monitoring, logging, and oversight obligations in frameworks like ISO/IEC 42001, ISO/IEC 23894, and the EU AI Act. It is where organizations gather the evidence needed to show that the agent is being used as intended, that controls are working, and that issues are identified and addressed in a structured way.

In the F7-LAS model, Layer 7 closes the loop. It does not prevent individual errors or unsafe suggestions by itself, but it ensures that agent behavior is visible, explainable, and adjustable over time. Monitoring gives us the raw telemetry of what happened; evaluation tells us whether the behavior was acceptable with respect to safety, performance, and policy.

For human operators, this is where explainability (XAI) becomes critical. When an agent proposes a diagnosis, opens a ticket, or recommends a remediation, security and operations teams need to understand why: which signals were used, which tools were called, what intermediate conclusions were reached, and how the final action was selected. Explanations do not have to expose every internal token-level detail, but they must provide enough structure that humans can audit decisions, challenge them, and calibrate their trust in the system.

Layer 7 is also where we manage the tension between predictability and adaptability. An enterprise agent must be predictable enough that users feel they can rely on its behavior, yet adaptive enough to handle novel inputs, changing environments, and new attack patterns. By instrumenting the

system and continuously evaluating outcomes, we can see when the agent has become too rigid (failing to adapt to new conditions) or too erratic (surprising users in high-stakes workflows) and adjust prompts, tools, policies, and sandboxes accordingly.

Beyond internal metrics, organizations can anchor their monitoring and red-teaming programs in **MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems)**. ATLAS is a knowledge base of AI-focused adversary tactics, techniques, and real-world case studies, modeled after the MITRE ATT&CK framework but tailored to AI and ML systems. It catalogs threats such as data poisoning, model evasion, model theft, and prompt or input manipulation, along with mitigations and examples. By mapping detection content, logging requirements, and red-team scenarios in Layer 7 to ATLAS tactics and techniques, security teams can reason about coverage in a structured, threat-informed way, just as they already do with ATT&CK for traditional infrastructure.

Without this layer, even a well-designed agent becomes a black box: powerful, convenient, and opaque. With Layer 7 in place, and informed by threat frameworks such as ATLAS,

Agentic AI becomes something we can observe, explain, tune, and govern, a system that can earn trust over time instead of relying on it by default.

4.4.9 Applying F7-LAS to Multi-Agent Systems

In multi-agent ecosystems (e.g., planner + verifier + executor), the F7-LAS layers may repeat at each agent boundary, with shared or federated policy and monitoring. Agentic AI systems are increasingly composed of multiple agents that collaborate, verify, or delegate tasks to one another. These may include *planner agents* that orchestrate workflows, *verifier agents* that perform quality checks, and *executor agents* that carry out operational actions. In such environments, the seven layers of F7-LAS still apply — but they may be instantiated more than once, forming a *federated control structure*.

In multi-agent topologies, **Layer 1 through Layer 4** typically operate locally within each agent's context: every agent has its own prompt, grounding mechanism, planning logic, and tool interface. However, **Layers 5 through 7** — policy, sandbox, and monitoring — often become *shared or federated control planes*. A central policy engine may define cross-agent rules (e.g., "no agent may invoke another agent that performs destructive operations"), while sandbox and monitoring layers provide system-wide observability and containment.

The security risk in multi-agent systems is **cross-agent drift** — when oversight, context, or permissions diverge between collaborating agents. The corresponding control strategy is **federated policy enforcement and transparent coordination**, ensuring that each agent's decisions and tool calls are auditable across the hierarchy. In short: multi-agent systems don't invalidate F7-LAS; they multiply its relevance. The same seven control surfaces apply — only now, they must also interlock.

4.10 Multi-Agent F7-LAS Architecture (Coordinator–Investigator–Remediator Pattern)

F7-LAS supports multi-agent environments by instantiating Layers 1–5 independently for each agent, while Layers 6–7 provide shared infrastructure, segmentation, and cross-agent

telemetry. Multi-agent architectures are treated as a collection of **independent constrained-POMDP agents**, each with its own system prompt, grounding profile, planner, tools, and policy constraints.

A multi-agent system introduces **explicit agent identity** (*agent_id, role, privilege tier*), which is used across Layers 4–7 for tool authorization, sandboxing, delegation, monitoring, and auditability. Each agent acts within its own bounded action surface, while the organization governs **inter-agent coordination** through runtime policy enforcement and human approval for high-risk transitions.

This whitepaper adopts the classic **Human-on-the-Loop Orchestrator–Worker model**, consisting of:

1. **SOC Coordinator Agent (Orchestrator)**
    - Responsible for planning, task decomposition, delegation, approval routing, and workflow management.
    - Layer 4: Has access only to workflow tools (ticketing, messaging, agent directory).
    - Layer 5: Permitted to delegate tasks to other agents directly. Human approval is required only for high-risk requests.
2. **Investigator Agent (Analyst)**
    - Performs read-only analysis, correlation, and threat hunting.
    - Layer 4: SIEM queries, EDR read APIs, threat-intel lookups.
    - Layer 5: Read-only policy tier; cannot perform system changes.
3. **Remediation Agent (Responder)**
    - Performs containment and state-changing actions under strict policy controls.
    - Layer 4: Identity provider changes, EDR isolate, firewall updates.
    - Layer 5: All actions are gated by risk-aware policies; high-risk actions require human approval.

Agents communicate using a structured protocol:

- **Coordinator → Tasks** (Goal + Context)
- **Investigator → Artifacts** (Evidence + Verdict)
- **Remediator → Outcomes** (Action Taken + Resulting System State)

Delegation between agents is **treated as a tool call** governed by Layer-5.
This pattern ensures that multi-agent functionality remains predictable, auditable, and governed by the same constrained-POMDP model that applies to single agents.

Figure 7 — Multi-Agent F7-LAS Architecture (Coordinator–Investigator–Remediator Pattern)

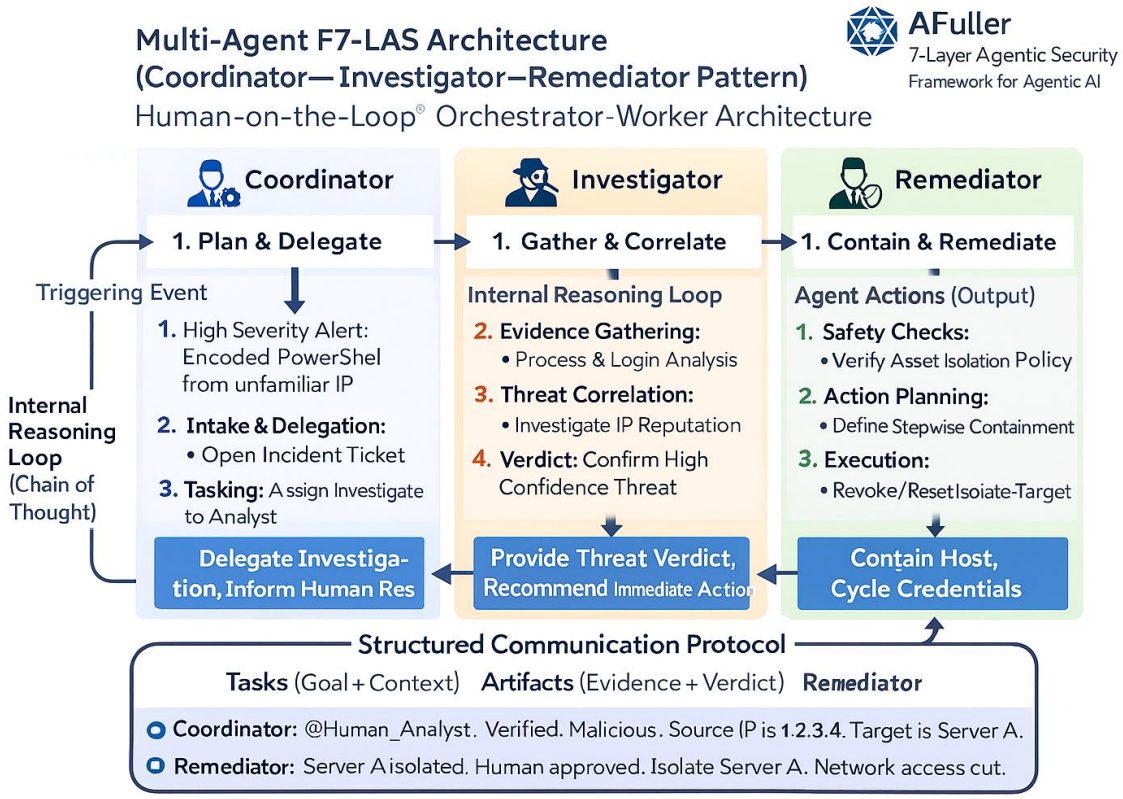

Figure 7- This figure illustrates the Multi-Agent F7-LAS workflow, where the Coordinator agent triages the triggering event and delegates investigation tasks, the Investigator agent gathers evidence and forms a threat verdict through its internal reasoning loop, and the Remediator agent performs safety-checked containment actions. A Human-on-the-Loop approval point ensures governance for high-impact operations. Structured communication passes tasks, context, artifacts, and remediation outputs between agents, providing a controlled, auditable orchestration model consistent with Layers 1–7 of the F7-LAS framework.

4.11 Model Security Annex

F7-LAS focuses primarily on securing agent behavior — planning, tool use, policy enforcement, sandboxing, and evaluation. However, agentic security also depends on the integrity of the underlying models. To ensure that dependency is visible within the framework, F7-LAS includes a **Model Security Annex**, summarizing the key model-level threats relevant to agentic deployments.

Common model-level risks include:

- **Model extraction and surrogate replication**
- **Training-time poisoning or weight backdoor insertion**
- **Membership inference and privacy leakage**

- **Weight tampering or unauthorized fine-tuning**
- **Adversarial prompting designed to reveal hidden system behavior**

While F7-LAS does not attempt to replicate existing model-security standards, the Annex shows how model-level safeguards integrate with the seven layers: prompt and RAG monitoring at Layers 1–2, introspection-limiting policy at Layer 5, hardened hosting and private endpoints at Layer 6, and detection of extraction patterns and poisoning indicators at Layer 7.

The full Model Security Annex and recommended controls appear in the **Complete F7-LAS Implementation Guide**.

5. How to Use the F7-LAS Model in Practice

The F7-LAS model is not just a way to describe agentic AI systems; it is meant to be used as a practical lens during design, review, and operations. This section shows two concrete ways to apply it:

- As a design review checklist when building or evaluating an agentic AI system
- As a structured way to challenge the claim "we secured it with a strong prompt and RAG"

5.1 Challenging "We Secured It with a Strong Prompt and RAG"

A common pattern in early agentic AI projects is the claim:

"We secured our agent with a really strong prompt and RAG."

In terms of the **AFuller F7-LAS model**, this usually means **Layers 1–2** have been addressed, while **Layers 3–7** remain only partially specified. The model gives you a respectful but firm way to challenge that statement.

A simple way to respond is:

1. **Acknowledge Layers 1–2**
    - "Great, that means you've thought about the system prompt and grounding. That helps reduce fabricated and ungrounded outputs."
2. **Ask explicitly about Layers 3–7**
    - "How does the **planner** decide which tools to use and when to stop?"
    - "What **tools** can it actually call, and which of those can change real systems or data?"
    - "Is there a **policy engine outside the LLM** that can block or require approval for risky actions?"

- "What does the **sandbox** look like, where does this agent run, and what is its maximum blast radius?"
- "What are we **logging and evaluating** to detect misuse, drift, or instruction injection over time?"

3. **Connect back to governance and threat models**
   - "How do these layers map to our NIST AI RMF / ISO/IEC 42001 controls and our MITRE ATT&CK / MITRE ATLAS threat models?"

The goal is not to dismiss prompts and RAG, they are essential and have their place, but to make clear that **secure agentic AI requires all seven layers**. The AFuller F7-LAS model gives you a shared vocabulary to have that conversation without hand-waving: prompts and RAG help with content; tools, policy, sandboxing, and monitoring determine what the system can actually *do*, how far it can reach, and how quickly you notice when something goes wrong.

5.2 Applying F7-LAS with Maturity and Threat Context

To move from design principles to measurable assurance, the **F7-LAS Maturity Model (Appendix A)** provides a practical scoring framework for each layer, from *Ad hoc* to *Optimized*. It enables security teams to benchmark readiness and prioritize investment.

Complementing it, the **F7-LAS Threat–Control Map (Appendix B)** connects each layer to adversarial techniques from **MITRE ATLAS** and **MITRE ATT&CK**, showing where common attack vectors emerge and which controls reduce exposure.

Used together, these appendices transform F7-LAS from a conceptual reference into an actionable review toolkit for governance, architecture, and red-team planning.

6. Lifecycle Integration: From Governance to Continuous Assurance

Purpose: To show that the F7-LAS model isn't static architecture, it supports an iterative, lifecycle-based assurance process across design, deployment, and operation.

Multi-Agent Lifecycle Governance.
Lifecycle governance applies independently to each agent (design, deployment, oversight), but multi-agent systems introduce an additional requirement: governing the **interactions between agents**. Policies must specify which agents may delegate tasks to others, which transitions require human approval, how shared context is exchanged, and how telemetry is aggregated across agents. Layer-7 monitoring plays a central role in detecting anomalous inter-agent behaviors such as loops, unapproved delegations, or unexpected role escalations.

Figure 8 - F7-LAS Lifecycle Integration Loop (Governance →Design → Deploy → Monitor → Adapt)

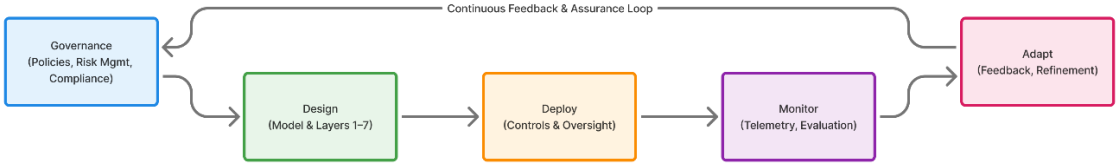

Figure 8- The F7-LAS model operates as a continuous assurance cycle linking governance, design, deployment, and monitoring into a self-reinforcing feedback system. Insights from monitoring and evaluation drive adaptation and refinement, ensuring that agentic AI controls evolve alongside operational and threat changes.

6.1 Lifecycle Context

The F7-LAS model extends beyond design-time security.
Agentic AI assurance requires a continuous cycle of control, observation, and adaptation, aligning technical controls with organizational governance throughout the agent's lifespan.

This lifecycle loosely parallels the ISO/IEC 42001 AI Management System emphasis on continuous improvement.

Phase	F7-LAS Layers Most Active	Key Outcomes
Govern	**1, 5, 7**	Policy creation, role definitions, risk tolerance thresholds, oversight models.
Design	**1–3**	Secure intent definition, RAG design, safe planning boundaries.
Deploy	**4–6**	Controlled tool integrations, sandbox enforcement, access segmentation.
Monitor & Improve	**7 → 1**	Continuous observability, red teaming, drift detection, control refinement.

Result:
The lifecycle ensures that F7-LAS isn't a "snapshot of security," but a feedback-driven loop, reinforcing governance with measurable data from real operations.

6.2 F7-LAS Implementation Profiles

F7-LAS defines a control-centric model for securing agentic AI systems across seven layers, but implementers also require a practical way to translate these layers into deployable architectures. To support this need, F7-LAS introduces the concept of an **Implementation Profile** —a consolidated, system-specific blueprint describing how each layer of the framework maps to concrete components, controls, and operational requirements.

An Implementation Profile contains four core artifacts:

1. **Layer-by-Layer Component Map** – the specific prompts, RAG sources, planners, tool adapters, PDP/PEP policy components, sandbox technologies, and monitoring systems used in a deployment.
2. **Control Catalog Mapping** – the F7-LAS control set applied to each component, including both soft and hard guardrails.
3. **Telemetry & Evaluation Schema** – the observability fields, events, metrics, and Service Level Objectives (SLO) required to make agentic behavior monitorable and auditable.
4. **Multi-Agent Governance Profile** – the rules for multi-agent cooperation, escalation models, and choreography (e.g., Coordinator–Investigator–Remediator patterns).

The Implementation Profile does not modify the core F7-LAS model; instead, it operationalizes it for a given environment. For organizations using Azure AI Foundry, LangGraph, or similar frameworks, the Implementation Profile becomes a companion to platform-specific architecture diagrams.

A complete, vendor-neutral Implementation Profile template and examples are provided in the **Complete F7-LAS Implementation Guide**.

7. How F7-LAS™ Fits Within the Agentic AI Framework Ecosystem

Core Positioning of F7-LAS™

F7-LAS™ (Fuller 7-Layer Agentic Security Model) is a practitioner-focused security architecture and maturity model for agentic and multi-agent AI systems. Unlike threat-modeling frameworks such as CSA MAESTRO or identity/governance frameworks such as AAM and AIGN, F7-LAS™ provides a control-oriented, implementation-ready structure spanning prompts, grounding, planners, tool permissions, external policy engines, sandbox environments, and monitoring.

F7-LAS™ complements these existing frameworks by giving engineering and security teams a concrete, layered design model for building safer, auditable agent behaviors.

How F7-LAS™ Complements Other Agentic AI Frameworks

F7-LAS™ is not a replacement for MAESTRO, AAM, or AIGN.
It is intentionally designed to sit alongside them as the security control stack that follows threat modeling, identity/access design, and governance definition.

7.1 F7-LAS™ and MAESTRO (Threat Modeling & Architecture)

What MAESTRO does:

- Provides threat modeling for agentic AI
- Maps components, data flows, and attack surfaces
- Identifies threats and adversarial scenarios

How F7-LAS™ complements it:

- MAESTRO identifies *where* the risks are
- F7-LAS™ defines *what controls* to implement at each behavioral layer
- MAESTRO gives a system view
- F7-LAS™ gives a control-stack view

Together:
Use MAESTRO for threat modeling, then apply F7-LAS™ to design layered guardrails to mitigate each threat.

7.2 F7-LAS™ and AAM (Agentic Access Management)

What AAM does:

- Governs non-human identities (NHIs)
- Defines credential, access, authorization, and monitoring rules for agents

How F7-LAS™ complements it:

- AAM defines who/what has access
- F7-LAS™ defines how agents plan, choose tools, apply policy checks, and operate inside sandbox boundaries
- AAM is strongest at identity and access
- F7-LAS™ is strongest at behavioral and architectural control

Together:
AAM defines access hygiene, F7-LAS™ defines runtime behavioral guardrails.

7.3 F7-LAS™ and AIGN (Governance & Trust)

What AIGN does:

- Establishes governance, oversight, trust, and accountability
- Aligns agentic AI with regulatory and organizational requirements

How F7-LAS™ complements it:

- AIGN defines governance expectations
- F7-LAS™ implements those expectations technically
- AIGN covers organizational and regulatory layers
- F7-LAS™ covers technical layers (prompt, grounding, planner, tools, policy engine, sandbox, monitoring)

Together:
AIGN sets the governance objectives; F7-LAS™ provides the engineering design to operationalize them.

7.4 Summary: Where F7-LAS™ Fits in the Ecosystem

- MAESTRO → "Where are the threats in our agentic system?"
- AAM → "How do we manage agent identities and access safely?"
- AIGN → "How do we govern agentic AI at an organizational level?"
- F7-LAS™ → "How do we design and implement layered security controls around agent behavior from prompt → tools → policy → sandbox → monitoring?"

F7-LAS™ is best positioned as:
A control-oriented, implementation-focused security architecture that fills the gap between framework-level guidance (MAESTRO, AAM, AIGN) and real engineering practice.

7.5 Supplemental Layer S – Software Supply-Chain Security

Agentic AI systems depend on a complex stack of frameworks, libraries, plugins, and tool adapters. While the F7-LAS model focuses on controlling agent behavior, the security of the **underlying software supply chain** is equally critical. To address this, F7-LAS introduces **Supplemental Layer S**, a cross-cutting layer that applies to all seven layers of the model.

Layer S governs:

- **SBOM generation and dependency policy** for agent runtimes, tool adapters, and supporting libraries.
- **Software Composition Analysis (SCA)** and CI-based gating for high-risk dependencies.

- **Plugin and tool vetting pipelines**, ensuring tools exposed to the planner are safe and minimally privileged.
- **Runtime attestation** (e.g., framework_version, sbom_id, toolset_hash) whose outputs flow into Layer-7 monitoring for detection of compromised components.

Layer S does not replace the seven core layers of F7-LAS; it provides the **software integrity baseline** on which those layers rely. The full supply-chain control set and a hardened Framework Security Profile (FSP) are included in the **Complete F7-LAS Implementation Guide**.

7.6 Quantitative Risk Scoring and SLOs

To make agentic-AI security observable and measurable, F7-LAS supports **quantitative risk scoring** and **Service-Level Objectives (SLOs)** aligned with each of the seven layers. These metrics assist security teams in evaluating agent reliability, identifying drift, and enforcing governance.

F7-LAS recommends defining a per-tool, per-context risk score of the form:

risk_score = base_risk(tool_risk_tier)

× context_multiplier

× agent_factor

This risk score can be used to adjust planner decisions, escalate to human review, or gate specific high-impact actions. In addition, each layer benefits from SLOs that define acceptable performance and safety bounds (e.g., prompt policy violation rate, retrieval trust score, planner loop termination rate, tool-failure rate, sandbox escape attempts, telemetry completeness).

Only a high-level introduction is included in the whitepaper. A complete set of formulas, SLO examples, and evaluation metrics is provided in the **Complete F7-LAS Implementation Guide**.

8. Related Threat Frameworks: MITRE ATLAS and MITRE ATT&CK

While the F7-LAS™ model is focused on where to place controls in an agentic AI system, threat frameworks such as MITRE ATT&CK and MITRE ATLAS help describe what attackers actually do.

- **MITRE ATT&CK** provides a widely used matrix of tactics and techniques for attacks against traditional IT systems and enterprises (for example, privilege escalation, lateral movement, and command and control).
- **MITRE ATLAS** (Adversarial Threat Landscape for Artificial-Intelligence Systems) extends this threat-based view into the AI domain, cataloging tactics, techniques, and case studies for attacking AI and ML systems across the lifecycle, including data poisoning, model stealing, model evasion, adversarial input manipulation, and abuse of generative models.

In practice, organizations can use ATT&CK to reason about threats to the surrounding infrastructure and SOC environment, and ATLAS to reason about threats specific to models, data

pipelines, and agentic behavior. The F7-LAS™ model then provides the placement of controls: ATLAS- and OWASP-aligned techniques map primarily into Layers 2–7 (from RAG and planning through tools, policy, sandboxing, and monitoring), while ATT&CK remains critical for protecting the underlying platforms where those agents and tools run. This is an informal mapping provided for practitioner orientation, not an official OWASP or MITRE crosswalk.

8.1 Operational Playbooks and RACI Integration

F7-LAS is equally applicable to architecture teams and operational teams. To support real-world deployments, organizations should define **layer-aligned operational playbooks**, including:

- Tool-misuse and mis-execution investigation playbooks
- RAG-poisoning detection and triage
- Planner-loop runaway conditions
- Policy engine override attempts
- Sandbox isolation / containment actions
- Telemetry integrity failures and drift detection

In addition, a **RACI model** should clarify which team (SOC, platform engineering, MLOps, cloud security, or governance) owns controls and decisions at each layer. These operational mappings ensure that F7-LAS does not remain theoretical and integrates cleanly with SOC, IR, and Cloud Advisory Board (CAB) processes.

The complete set of playbooks and RACI templates is included in the **Complete F7-LAS Implementation Guide**.

9. Conclusion

Agentic AI fundamentally changes the nature of enterprise risk. Once an AI system can plan, call tools, and take actions, the failure modes extend far beyond fabricated or ungrounded content. They become operational: incorrect, unsafe, or misaligned actions taken in complex, partially observable environments. Security architects need a way to reason about that entire execution stack—not just the chat interface on top.

The AFuller F7-LAS (Fuller 7-Layer Agentic AI Security) model provides that lens. By separating prompts, grounding, planning, tools, policy enforcement, sandboxing, and monitoring into distinct layers, it becomes easier to see where risk concentrates and where controls must be applied.

From this model, several key principles emerge:

Agents are not chatbots. They operate in richer environments, with tools acting as actuators and with real blast radius behind their decisions. PEAS and environment properties—partial

observability, stochasticity, multi-agent coordination, dynamic state—directly influence how we secure them.

Security must be layered. Prompts and RAG help reduce fabricated or ungrounded content, but they do not control planning, tool use, policy enforcement, sandbox boundaries, or telemetry. Those concerns live in Layers 3–7 and demand explicit design and ownership.

Threat- and governance-informed design is essential. Aligning F7-LAS with frameworks such as NIST, ISO/IEC, the EU AI Act, and MITRE ATLAS/ATT&CK ensures that agentic systems are not only effective, but also auditable, governable, and resilient under adversarial pressure.

Ultimately, an agent's real blast radius is determined not by its prompt, but by its tools, policies, and environment—the combined effect of Layers 4, 5, and 6. The core message of this whitepaper is simple: **do not stop at prompt + RAG. Secure the entire stack.**

By applying the F7-LAS model as both a design and review tool, organizations can move from experimental agents to structured, defensible, and governable agentic AI deployments—systems that act on behalf of the enterprise while remaining within clear, observable, and enforceable boundaries.

Acknowledgments

The author acknowledges the influence of professional and academic training through Johns Hopkins University and Great Learning's Agentic AI programs, as well as SANS Institute courses SEC495, SEC545, and SEC595. These programs contributed to the author's understanding of agentic architectures, AI safety frameworks, and cybersecurity governance.

The F7-LAS™ framework presented in this whitepaper is entirely original work. It reflects the author's independent analysis, informed by more than 27 years of experience in IT and cybersecurity practice. The organizations referenced above do **not** review, validate, or endorse F7-LAS™.

The author's long-standing interest in intelligent systems—including early exposure to expert systems, neural networks, and fuzzy-logic-based reasoning—also shaped the perspective behind this model. The author further recognizes the value of foundational concepts such as PEAS, POMDP, and Model Context Protocol (MCP) in advancing structured approaches to securing autonomous and agentic AI systems.

Glossary

Agentic AI
An AI systems that can interpret goals, plan multi-step actions, call tools or APIs, and adapt their behavior based on feedback, rather than only generating single-turn text responses.

AFuller F7-LAS model
The **Fuller 7-Layer Agentic AI Security model**, a framework that organizes an agentic system into

seven layers: system prompt, RAG/grounding, agent planner, tools and integrations, policy engine outside the LLM, sandboxed execution environment, and monitoring and evaluation.

CAB - Change Advisory Board A governance group that reviews, approves, or rejects high-risk changes affecting prompts, RAG sources, tool permissions, policy rules, or sandbox boundaries.

SBOM - Software Bill of Materials
A formal inventory of all software components and dependencies used by the agent framework and tools, used for supply-chain risk management.

SCA - Software Composition Analysis
Automated scanning of dependencies for vulnerabilities or policy violations, often performed in CI pipelines.

SLO – Service Level Objective
A measurable reliability, safety, or performance target for agentic systems (e.g., violation rate, tool error rate, loop termination rate).

PEAS (Performance measure, Environment, Actuators, Sensors)
A classic AI framework for characterizing an agent and its context: how success is measured, the environment it operates in, the actuators it uses to act, and the sensors it uses to perceive.

Fabrication / Fabricated content
Content produced by a model that asserts facts or details that are not grounded in reality or in the provided input or context.

Ungrounded output / Ungrounded generation
A response that is not supported by available data, retrieved context, or source material—even if it is plausible-sounding.

Factual inaccuracy / Factual error
A generated statement that is objectively incorrect relative to trusted sources or ground truth.

Unsupported assertion
A claim made by a model without sufficient evidence, citation, or grounding in the underlying data or context.

Content deviation
Model output that remains syntactically valid but steps outside the allowed topics, constraints, or instructions specified by policy or prompt.

RAG (Retrieval-Augmented Generation)
A pattern where an LLM retrieves relevant documents or data at query time and uses them as context when generating a response, to reduce ungrounded outputs and improve alignment with current or enterprise-specific information.

Policy engine outside the LLM
A component that enforces rules, approvals, and constraints in code—independent of the model's internal reasoning—before actions are allowed to execute (for example, blocking certain tools, requiring human approval, or enforcing parameter limits).

Sandboxed execution environment
The scoped runtime context (tenants, subscriptions, identities, networks, and data) in which an agent and its tools are allowed to operate. The sandbox is designed to limit blast radius if the agent misbehaves or is misused.

Explainability (XAI)
Techniques and practices that make an AI system's behavior understandable to humans—for example, by showing which inputs, signals, or intermediate steps contributed to a given decision or action.

MITRE ATT&CK
A curated knowledge base of adversary tactics and techniques for attacks against traditional IT and enterprise systems, widely used to inform detection engineering and red-teaming.

FSP - Framework Security Profile A security and supply-chain hardening profile for the agent runtime (LangGraph, container apps, plug-ins, etc.).

SOC - Security Operations Center
Operational team responsible for monitoring and responding to alerts generated across the F7-LAS layers.

IR - Incident Response Processes and actions for handling security incidents across prompts, tools, sandboxes, and agent behavior.

MLOps - Machine Learning Operations The discipline of deploying, operating, and monitoring ML models, including governance of training data and model updates.

API - Application Programming Interface A callable function or service exposed to the agent as a tool.

MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems)
A threat framework focused on attacks against AI and ML systems, including data poisoning, model evasion, model theft, and manipulation of inputs or prompts, together with mitigations and real-world case studies.

Model Context Protocol (MCP) An emerging protocol and architecture pattern for exposing resources, tools, and prompts to LLM-based agents through a standardized client–server interface, making the tool surface explicit and reusable.

ReAct (Reason + Act) A prompting and agent pattern where the model alternates between reasoning ("thought"), choosing an action (tool/API call), and observing the result in a loop. ReAct is a common way to implement the **planning and control behavior** of an agent, and in the F7-LAS model it is one possible implementation of **Layer 3 – Agent Planner / Controller**.

Agent Reasoning LLM Intent The Agent's Large Language Model (LLM) decides to perform an action, translating its goal into a specific tool call (**e.g.,** delete_user_data with argument user_id=123).

PEP Interception (Policy Enforcement Point) The PEP, acting as the wrapper around the tool/API, **intercepts the function call** and pauses execution. It packages the action into a **rich context request** for the PDP, containing all necessary attributes for policy evaluation:

- **Subject:** Agent ID, Delegate User ID (Human Owner), Agent Trust Score.
- **Action:** The specific function being called (e.g., delete_user_data).
- **Resource:** The target data and its classification (e.g., user_id=123, labeled as Sensitive PII).
- **Context:** Environmental factors (Time of day, current Risk Score from Anomaly Detection).

- **PDP Evaluation (Policy Decision Point)**
  The PDP (the Policy Engine) receives the request and dynamically evaluates it against a library of policies (e.g., ABAC rules, security guardrails).
  - **Example Policy Check:** "Deny if resource classification is PII **AND** the action is destructive **AND** the Risk Score is high."
  - **Verdict:** The PDP returns an authoritative decision:
  - **Permit**
  - **Deny**
  - **Permit with Obligation** (e.g., "mask all SSNs in the log")
  - **Refer to Human** (for Human-in-the-Loop approval)
- **PEP Enforcement**
  The PEP receives the final verdict and enforces it:
  - **If Deny:** The PEP logs the violation (Tripwire), stops the tool call, and returns an error to the LLM.
  - **If Permit:** The PEP executes the tool call and ensures any obligations (like data masking) are applied during the process.

Appendix A — F7-LAS Maturity Model (Version 2.0)

The **F7-LAS Maturity Model** extends the core seven-layer framework into a practical assessment tool for organizations deploying agentic AI.
It provides a four-tier progression, from *Ad hoc* to *Optimized*, across each layer of the F7-LAS model, enabling practitioners to benchmark readiness, identify control gaps, and plan incremental improvements.

This appendix is designed for use during design reviews, red-teaming, and governance assessments.
It can be used alongside the NIST AI RMF functions (Govern, Map, Measure, Manage) and ISO/IEC 42001 / 23894 as a way to translate governance intent into technical maturity.

Purpose: Help enterprises assess *where* they stand within each F7-LAS layer and *what* steps will move them toward safe, auditable, and resilient agentic AI operations.

Layer / Control Domain	Level 0 – Ad hoc / Absent	Level 1 – Initial / Defined	Level 2 – Managed / Measured	Level 3 – Optimized / Assured
1 – System Prompt (Soft Policy)	No defined system prompt; behavior uncontrolled; no abstain / escalate logic.	System prompt defines basic role & scope; limited safety clauses.	Prompt pattern standardized across agents; includes safety & abstain directives; peer-reviewed for risk.	Prompt governance lifecycle in place — versioned, red-teamed, and aligned to Responsible AI policies.
2 – RAG / Grounding (Epistemic Guardrail)	Model answers purely from pretraining; no grounding or data lineage.	Basic RAG setup with unverified sources; limited access control.	Curated, access-controlled RAG sources; provenance tracked; injection testing performed.	Enterprise-wide RAG registry; automated source validation, content trust scoring, and continuous monitoring for data poisoning.
3 – Agent Planner / Controller	Ad hoc planner logic (e.g., ReAct loop) with no control limits.	Planner bounded by max steps / timeouts; limited oversight.	Formal orchestration policies; safe-planning patterns; unit-tests for planner behavior.	Verified planning framework with formal stop conditions, policy-aware loop control, and automated drift detection.
4 – Tools & Integrations (Action Surface)	Tools added ad hoc with broad	Tool catalog defined; basic access control.	Least-privilege tool design; change	Dynamic tool governance platform; risk-tiering of tools; automated credential

Layer / Control Domain	Level 0 – Ad hoc / Absent	Level 1 – Initial / Defined	Level 2 – Managed / Measured	Level 3 – Optimized / Assured
	privileges; no inventory.		management & logging enforced.	rotation & blast-radius simulation.
5 – Policy Engine outside the LLM (Hard Guardrails)	No external policy enforcement: model decisions execute directly.	Rule-based policy layer for select actions; partial approvals.	Centralized policy gateway mediating all tool calls; human-in-the-loop support; auditable logs.	Policy as code framework integrated with GRC and AI risk registers; continuous compliance testing.
6 – Sandboxed Execution Environment (Blast Radius Control)	Agents run with unrestricted network / tenant access.	Isolated environment for non-prod; basic least privilege.	Segmented tenants, VNETs, and identities; automated provisioning / teardown; role separation.	Dynamic sandbox management with context-aware scoping and automated risk containment simulations.
7 – Monitoring & Evaluation (Detection & Assurance)	Minimal logging; no behavioral visibility.	Standard logs collected; ad hoc review of incidents.	Continuous telemetry; structured evaluations / red team tests; drift tracking.	Full observability stack with XAI-style explanations and automated feedback loops to layer 1-5 controls.
Multi-Agent Governance	Agents operate independently with no delegation rules or shared visibility.	Basic agent identity defined; limited cross-agent logging; no structured delegation or handoff model.	Coordinator–Investigator–Remediator roles separated; delegation and human-approval paths enforced by the policy engine.	Full cross-agent telemetry correlation; automated detection of anomalous multi-agent interactions; role-based tool catalogs maintained at scale.

Appendix B — F7-LAS Threat–Control Map

This appendix aligns each F7-LAS layer with representative **attack techniques**, **threat categories**, and **control focuses** with representative attack techniques and threat categories **informed by concepts in** MITRE ATLAS and OWASP Security Projects for LLMs

F7-LAS Layer	Typical Threats	Attack Vector / Failure Mode	Primary Defensive Controls
1 System Prompt (Soft Policy)	Prompt Injection: Direct, Indirect, Triggered	Malicious instructions override role; model outputs secrets or policy-violating content.	Prompt hardening, role segmentation, input sanitization, prompt change control, red-team prompt testing.
2 RAG / Grounding (Epistemic Guardrail)	Data Poisoning: RAG Poisoning	Corrupted or adversary injects malicious content/instructions into agent's external knowledge base or vector store, memory, indexed by RAG, leads to fabricated or adversarially steered outputs or misleading information.	Data pipeline validation (checksums/digital signatures), data provenance logging, access-controlled indexes, content trust scoring, regular audits, check grounding, filtering & verification.
3 Agent Planner / Controller	Goal Hijacking, Reward Hacking	High-level missions never completed, agent actively resists shutdown/reconfiguration. Leads to dysfunctional or harmful behavior where the AI achieves highest possible reward by unintended, often trivial or hazardous means.	Loop bounds, policy-aware planning logic, anomaly detection on task chains, safe termination criteria.
4 Tools & Integrations (Action Surface)	Insecure plugins/Tool use or Excessive Agency	Agent successfully compromised via prompt injection; Attacker injects code that tells agent to use tools (blast radius-tool dependent). Agent possesses too much autonomy (large powerful action surface)	Least privilege design, tool whitelisting, API token rotation, strong auth and audit of tool calls, HITL, Functional argument validation, tool sandboxing, policy gateway, behavioral anomaly detection
5 Policy Engine Outside the LLM (Hard Guardrails)	Policy Bypass/evasion, Policy Drift	Attacker uses complex prompt injection or prompt chaining to trick the Agent/LLM into generating an action that circumvents the policy engine or output checks.	Deterministic code Policy base, strict serialization and deserialization, Security as a code (SaC) for policies, automated policy compliance checks, version control and peer review for all policy updates, runtime authorization checks,

F7-LAS Layer	Typical Threats	Attack Vector / Failure Mode	Primary Defensive Controls
			human-in-the-loop approvals, tamper-proof logging.
6 Sandboxed Execution Environment (Blast Radius Control)	Uncontained remote code execution	Agent or tool breaks isolation (prompt injection) and accesses other tenants or networks. agent generates malicious code, when executed breaks out of insufficient isolation layer, compromises, steals or deletes critical files and secrets.	Network segmentation, Micro VMs, Isolate execution per-session/per user; Destroy Sandbox, network egress allow listing, Filesystem read-only policies, strict timeouts, memory caps, and CPU throttling.
7 Monitoring & Evaluation (Detection & Assurance)	*Runtime observability*	*Evidence erasure, tampering, Assurance Gap /Policy failure, Convert manipulation/subversion*.	KPI, drift monitoring, WORM logging, separation of duties, cryptographic hashing, policy compliance auditing, Drift detection, AI Red teaming automation, Comprehensive telemetry, ATLAS-aligned detections, human oversight dashboards.

Appendix C — F7-LAS Key Performance Indicators (KPI Table)

These KPIs provide a non-prescriptive way to quantify maturity and measure control effectiveness across the seven layers of the F7-LAS™ model.

Layer	Suggested KPIs / Metrics	Measurement Goal	Typical Data Sources
1. System Prompt	% of prompts with versioned policies; prompt drift rate	Stability of intent and compliance	Prompt repository, version control, policy docs, prompt-change logs
2. RAG / Grounding	% of responses with verified sources; source trust score	Data provenance integrity	RAG service logs, retrieval metadata, source catalogs, content trust labels
3. Agent Planner	Avg. plan length before oversight; unsafe-plan rejection rate	Planner boundedness and policy adherence	Planner traces, decision logs, policy-check outcomes, review/override records
4. Tools & Integrations	% of tool calls via approved registry; token rotation success rate	Action-surface control	Tool registry, API gateway logs, token/secret management logs, CI/CD config
5. Policy Engine	Policy-enforcement success rate; human-override frequency	Hard guardrail effectiveness	Policy engine decision logs, deny/allow statistics, override workflows, audit logs
6. Sandbox	% of actions confined to isolated runtime; blast-radius reduction delta	Containment reliability	Sandbox runtime logs, environment isolation reports, before/after change impact analyses
7. Monitoring & Evaluation	Mean time to detect agent drift (MTTD); incident false-negative rate	Assurance visibility and responsiveness	SIEM/XDR alerts, monitoring dashboards, evaluation runs, incident postmortems

Closing line:

These KPIs can feed directly into existing GRC dashboards, bridging AI safety with traditional enterprise security metrics.

Appendix D — Agentic AI Red Team Lifecycle

A compact framework for testing resilience of agentic AI systems.

Stage	Objective	Example Techniques
1 Reconnaissance	Identify exposed agent interfaces, prompts, and RAG sources.	Prompt injection, data poisoning, tool discovery.
2 Exploitation	Simulate adversarial behaviors and unsafe actions.	Goal hijacking, privilege escalation, bypassing policy engine.
3 Observation	Evaluate system responses and control activations.	Monitor sandbox behavior, tool audit logs, policy triggers.
4 Evaluation	Assess drift tolerance, control response time, and containment.	Drift replays, cross-layer correlation.
5 Remediation	Update system prompts, RAG sources, and policies.	Implement findings into governance loop.

Cycle Outcome:
Establishes a continuous red team and control tune with assurance validate loop integrated into the F7-LAS lifecycle (**see Section 6.2**).

Appendix E — 7-Layer Control Review Worksheet

Practitioners can use this as a quick audit or design review checklist.

Layer	Checklist Questions
1 System Prompt	Is the prompt version-controlled, peer-reviewed, and aligned to policy?
2 RAG / Grounding	Are all data sources verified, access-controlled, and logged?
3 Agent Planner	Does the planner include stop conditions and escalation triggers (marker)?
4 Tools & Integrations	Are all tools registered, least-privilege, and audited?
5 Policy Engine	Is every action filtered through a centralized policy layer?
6 Sandbox	Are runtime environments isolated and blast radius-limited?
7 Monitoring & Evaluation	Are telemetry, explainability, and drift metrics collected continuously?

Use this worksheet during design reviews, internal audits, or red-team pre-assessments.

Appendix F — Using F7-LAS in Incident Response

Even with layered controls, agentic systems can misbehave, through faulty reasoning, prompt injection, or compromised tools. When that occurs, F7-LAS provides a structured way to diagnose and contain incidents by tracing the agent's decisions and control surfaces.

Incident response for agentic AI should mirror digital forensics and security operations processes, but with **layer awareness**.
When an anomaly or unsafe behavior is detected:

1. **Start at Layer 7 (Monitoring and Evaluation)** - Identify what triggered the alert. Review logs, telemetry, and feedback models to reconstruct the sequence of actions.
2. **Inspect Layer 5 (Policy Engine)** - Determine whether the action violated policy or if the policy was misconfigured.
3. **Review Layer 4 (Tools and Integrations)** - Assess what tools were called, with which parameters, and whether their scopes were exceeded.
4. **Contain via Layer 6 (Sandbox)** - Revoke agent tokens, freeze its environment, or isolate it from production systems.
5. **Assess upstream causes (Layers 1–3)** - Examine prompts, retrieval chains, and planning logic for malicious input or unintentional bias.

Each layer leaves a distinct forensic footprint, prompts, retrieved documents, tool call logs, policy decisions, sandbox IDs, and evaluation metrics. Together, these artifacts allow investigators to reconstruct what the agent "knew," "decided," and "did."

Finally, insights from incident response should loop back into **continuous assurance (Lifecycle Integration, Section 6)** feeding lessons learned into updated prompts, policies, and evaluation criteria. This closes the feedback cycle, turning every incident into a reinforcement opportunity for safer agentic design.

Appendix G — Example Implementation Patterns

Table G.1 — Example Implementation Patterns for F7-LAS

Focus Area	Pattern / Component	Example Questions for Review
Policy Engine (Layer 5)	PDP/PEP with policy-as-code	Is every tool call mediated by a PDP/PEP? Are policies versioned and testable?
	Human-in-the-loop approvals	Which actions always require approval? How is approval evidence logged?
Monitoring (Layer 7)	Standard tool call JSON schema	Can we reconstruct who did what, where, when, and under which policy?
Tools (Layer 4)	Tiered tool model (Explore / Enrich / Act)	Are high-impact tools clearly separated from low-impact tools?
	Tool registry with risk metadata	Do we know which tools exist, their owners, data access, and blast radius?
Sandbox (Layer 6)	Per-environment execution scopes	Does each agent run in a scoped IAM tenant or virtual network with least privilege?
Governance Integration	Mapping to NIST AI RMF / ISO/IEC 42001	Can we show which layers implement which governance requirements?

In an agentic policy decision flow (**PDP/PEP**), this pattern transforms the agent's autonomy from a potential security risk into a governed, auditable action, ensuring every step complies with organizational policy.

Appendix H — Organizational Role Mapping

Securing agentic AI systems requires not only layered technical controls but also clearly owned responsibilities. Each layer of the F7-LAS model aligns with specific organizational functions, ensuring accountability for policy definition, implementation, and continuous assurance.
This mapping helps governance bodies confirm that every control plane has an identified owner and escalation path.

Layer	Primary Control Domain	Typical Organizational Owner(s)	Key Responsibilities
L1 System Prompt (Soft Policy)	Context initialization, alignment with enterprise policy	AI Developers, Prompt Engineers, Product Owners	Define prompt templates, align objectives with enterprise policy, document change control.
L2 Retrieval / Grounding (RAG)	Source validation and factual assurance	Data Engineering, Knowledge Management	Manage trusted corpora, approve retrieval connectors, enforce data-classification boundaries.
L3 Planner (Reasoning, Task Control)	Decision logic, decomposition, and reasoning constraints	AI Engineering, Applied Research, Security Architecture	Design plan-formation logic, validate reasoning traces, verify safe planning boundaries.
L4 Tools and Integrations (Action Surface)	Authorized actions and external APIs	Platform Engineering, DevOps, SOC Automation	Maintain tool catalogs, enforce least-privilege permissions, monitor usage scopes.
L5 Policy Engine (Hard Policy Enforcement)	Authorization, constraint enforcement, rule logic	Security Architecture, Risk and Compliance	Implement PDP/PEP mechanisms, approve policy rule sets, integrate with IAM controls.
L6 Sandbox (Environment Containment)	Execution isolation and privilege management	Cloud Security Ops, Infrastructure Engineering	Manage agent tenants, limit network scope, enforce environment reset and isolation controls.
L7 Monitoring and Evaluation (Continuous Assurance)	Observability, metrics, and red-team feedback	SOC, Threat Intelligence, RAI Governance Board	Collect telemetry, detect abnormal patterns, manage agent red-team exercises, track KPIs.

In more mature enterprises, these roles can converge into a cross-functional **Agentic AI Assurance Group** (**AAAG**), a coalition of security, governance, and engineering stakeholders that meets regularly to review performance metrics, incident learnings, and control updates. This formalized ownership model ensures that every safeguard, technical or procedural, has a responsible steward and a documented review cadence.

Key Principle: Agentic AI security is not just a system architecture; it is an organizational discipline. Clear role assignment turns the F7-LAS model from a conceptual framework into something organizations can use to structure an operational governance program.

APPENDIX I - Implementation Patterns

SOC Multi-Agent Pattern: Coordinator, Investigator, Remediator

This implementation pattern applies the F7-LAS model to security operations via a three-agent design:

- **Coordinator Agent:**
  - Layer-4: Ticketing, messaging, agent directory
  - Responsibilities: plan → delegate → route human approvals
  - Risk: decision loops, over-delegation
- **Investigator Agent:**
  - Layer-4: Read-only SIEM, EDR, threat-intel
  - Responsibilities: evidence gathering, correlation, attribution
  - Risk: rabbit-hole looping, excessive query volume
- **Remediator Agent:**
  - Layer-4: Identity + EDR + Firewall write actions
  - Responsibilities: containment and eradication under strict policy
  - Risk: destructive action (isolate wrong host, revoke wrong user)

The pattern uses a **structured communication protocol** (Tasks → Artifacts → Outcomes), governed by Layer-5 and executed within segmented sandboxes (Layer-6). Agents collectively support the incident lifecycle while maintaining strict privilege separation and human oversight for high-risk actions.

References / Resources

1. ***Microsoft Security Blog.*** *Empowering Defenders in the Era of Agentic AI with Microsoft Sentinel. Microsoft, 2025.*
   URL: https://www.microsoft.com/en-us/security/blog/2025/09/30/empowering-defenders-in-the-era-of-agentic-ai-with-microsoft-sentinel/
2. ***Microsoft Source.*** *Microsoft Sentinel Evolves to Bring Agentic AI to Cyber Defense. Microsoft, 2025.*
   URL: https://news.microsoft.com/source/emea/2025/10/microsoft-sentinel-evolves-to-bring-agentic-ai-to-cyber-defense/
3. ***Microsoft Learn.*** *Overview of Microsoft Sentinel MCP and the Security Data Lake. Microsoft, 2025.*
   URL: https://learn.microsoft.com/en-us/azure/sentinel/datalake/sentinel-mcp-overview
4. ***National Institute of Standards and Technology (NIST).*** *Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1, 2023.*
   URL: https://www.nist.gov/itl/ai-risk-management-framework
5. ***European Union.*** *Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act).*
   URL: https://eur-lex.europa.eu/eli/reg/2024/1689/oj
6. ***ISO/IEC 42001:2023.*** *Information Technology — Artificial Intelligence — Management System. International Organization for Standardization / International Electrotechnical Commission, 2023.*
7. ***ISO/IEC 42005:2025.*** *Information Technology — Artificial Intelligence (AI) — AI System Impact Assessment. ISO/IEC, 2025.*
8. ***ISO/IEC 42006:2025.*** *Information Technology — Artificial Intelligence — Requirements for Bodies Providing Audit and Certification of Artificial Intelligence Management Systems. ISO/IEC, 2025.*
9. ***ISO/IEC 23894:2023.*** *Information Technology — Artificial Intelligence — Guidance on Risk Management. ISO/IEC, 2023.*
10. ***ISO/IEC 38507:2022.*** *Information Technology — Governance of IT — Governance Implications of the Use of Artificial Intelligence by Organizations. ISO/IEC, 2022.*
11. ***ISO/IEC 22989:2022.*** *Artificial Intelligence — Concepts and Terminology. ISO/IEC, 2022.*
12. ***ISO/IEC 23053:2022.*** *Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML). ISO/IEC, 2022.*
13. ***ISO/IEC TR 24028:2020.*** *Information Technology — Artificial Intelligence — Overview of Trustworthiness in Artificial Intelligence. ISO/IEC, 2020.*
14. ***OECD.*** *OECD Principles on Artificial Intelligence. Organisation for Economic Co-operation and Development, 2019.*
    URL: https://oecd.ai/en/ai-principles
15. ***Leike, J., Krakovna, V., Everitt, T., Orseau, L., & Legg, S.*** *Specification Gaming: The Flip Side of AI Ingenuity. DeepMind, 2018.*
    URL: https://deepmind.google/blog/specification-gaming-the-flip-side-of-ai-ingenuity/
16. ***Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D.*** *Concrete Problems in AI Safety. arXiv preprint arXiv:1606.06565, 2016.*
    URL: https://arxiv.org/abs/1606.06565

17. ***Cloud Security Alliance***. *Agentic AI Threat Modeling Framework MAESTRO (Multi-Agent Environment, Security, Threat, Risk & Outcome) 2025*.
URL: https://cloudsecurityalliance.org/blog/2025/02/06/agentic-ai-threat-modeling-framework-maestro?utm_source=chatgpt.com
18. ***Agentic AI Governance Network (AIGN. (n.d.).*** *Agentic AI Governance Framework: operationalizing trust for autonomous, self-improving, and multi-agent AI systems*.
URL: https://aign.global/ai-governance-framework/agentic-ai-governance-framework/ AIGN