ODA3 INSTITUTE

Practitioner Cheat Sheet

AI Security Architecture Patterns

DocID: ODA3-2026-07-CHT-SEC-005
Classification: Public

CHT-SEC-005: AI Security Architecture Patterns

ODA3 Institute | *Practitioner Cheat Sheet*

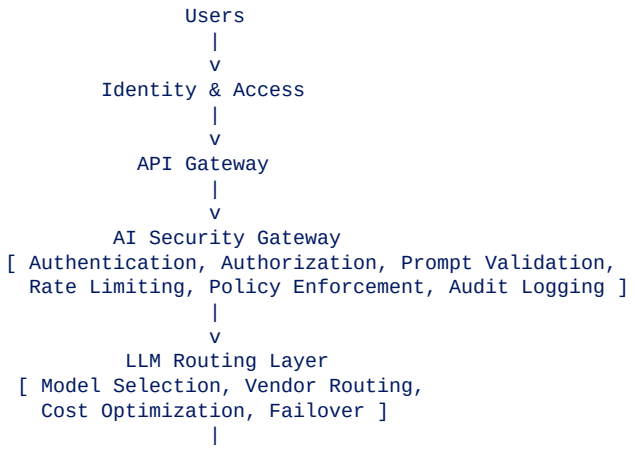
Field	Value
Classification	Public — Practitioner Reference
Intended Audience	Security Architects, Enterprise Architects, AI Platform Engineers, CISOs
Evidence Confidence	Moderate (T2–T3 weighted; see tier tags)
Review Cycle	6 months
Framework Cross-References	Maps to UAIF™ v1.0 (classification), GAISSF™ v1.0 (architecture assurance), AI-IRF™ v1.0 (incident response)
Related Publications	CHT-SEC-001, CHT-SEC-002, CHT-SEC-003, CHT-SEC-004

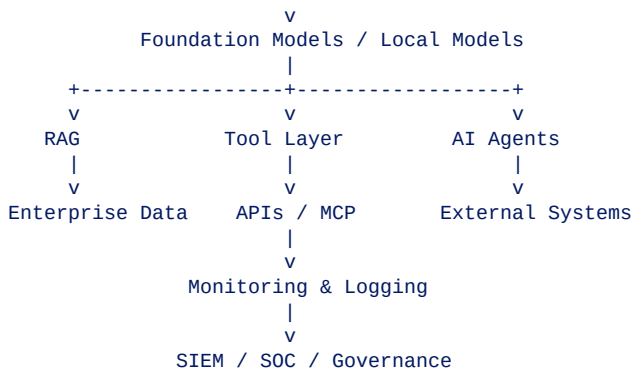
Why This Matters

Traditional application security assumes deterministic software executing well-defined logic inside established trust boundaries. AI systems break that assumption in several ways at once: behavior is probabilistic, execution is prompt-driven, context is assembled from multiple untrusted sources at runtime, and increasingly the system doesn't just answer questions — it invokes tools and takes actions. A trust boundary that used to be fixed at design time is now something an LLM can effectively redraw based on its own reasoning [T3 — pattern described across recent architecture literature, not yet a settled industry consensus].

Conventional reference architectures don't provide enough guidance for this. What organizations need instead is a set of reusable, composable architecture patterns — each solving one recurring problem, each with known trade-offs — that together define where trust boundaries should sit, where enforcement belongs, and how AI-specific attack surface gets isolated. This cheat sheet catalogs fifteen such patterns, drawn from emerging industry guidance, cloud reference architectures, and practitioner experience, organized as an architectural reference rather than a deployment standard.

1. AI Security Reference Architecture





This is a composite view — no single deployment needs every layer. Sections 3–17 break out each box above (and several not shown) as an independent, adoptable pattern.

2. Architectural Principles

Five principles recur across every pattern in this cheat sheet. Treat them as the test for whether a new AI system's architecture is defensible, not just functional.

Principle 1 — Trust nothing by default. Every prompt, retrieved document, tool response, model output, and user instruction should be treated as untrusted until validated. This extends Zero Trust into the AI layer rather than assuming AI traffic is inherently benign.

Principle 2 — Separate trust boundaries. Users, prompts, retrieved knowledge, model reasoning, tool execution, and enterprise systems should be separable domains. Compromise in one should not automatically cascade into another.

Principle 3 — Identity everywhere. Identity shouldn't stop at the human user. Architectures should be able to identify and authenticate service accounts, AI agents, models, plugins, MCP servers, and external tools individually.

Principle 4 — Policy before inference. Authentication, authorization, prompt validation, and tenant isolation should happen before the model runs wherever possible — catching a problem before generation is cheaper and more reliable than filtering an unsafe response afterward.

Principle 5 — Observe everything. Prompts, responses, retrieved documents, invoked tools, policy violations, model selection, and confidence indicators should all produce telemetry. Without it, security monitoring, incident response, and governance assurance are all working blind.

3. Pattern 1 — AI Security Gateway

Purpose: a centralized enforcement point for AI security controls before requests reach any model.

Problem it solves: without a gateway, every application team implements security differently — inconsistent policy, fragmented logging, uneven prompt validation, and no single place to govern model access.

```
Users -> AI Security Gateway [Auth, Authz, Prompt Validation,
  Rate Limiting, Policy Enforcement, Logging, Model Selection] -> Foundation Model
```

Typical controls: OAuth/OIDC, API authentication, prompt inspection, sensitive-data detection, content filtering, tenant isolation, rate limiting, audit logging.

Advantages: single policy location, consistent enforcement, easier governance, simplified monitoring, reduced per-application complexity.

Trade-offs: additional latency; the gateway becomes a potential single point of failure and must scale accordingly.

Best suited for: enterprise AI platforms, internal copilots, multi-team deployments, regulated environments.

4. Pattern 2 — LLM Proxy Pattern

Purpose: abstract AI applications away from any single model vendor.

Problem it solves: applications tightly coupled to one provider become hard to migrate, govern, or secure consistently as vendors and models change.

```
Applications -> LLM Proxy -> { OpenAI | Claude | Local LLM }
```

Responsibilities: vendor abstraction, routing, failover, cost optimization, policy enforcement, usage quotas, logging, security controls.

Security benefits: prevents direct model exposure, hides API credentials, centralizes monitoring, standardizes controls, simplifies model replacement.

Trade-offs: additional infrastructure and operational complexity; the proxy itself becomes critical infrastructure.

Best suited for: multi-model deployments, vendor-neutral architectures, enterprise AI platforms.

5. Pattern 3 — Policy Enforcement Point (PEP)

Purpose: separate governance decisions from application logic.

Problem it solves: authorization rules embedded inside prompts or application code are difficult to maintain, audit, or apply consistently.

```
User Request -> Policy Decision Point -> Policy Enforcement Point -> Foundation Model
```

Example policies: department restrictions, data classification, allowed models, geographic restrictions, tool permissions, regulatory requirements, business approval workflows.

> **Security consideration:** policies should execute before prompt construction, document retrieval, tool invocation, and model inference — not as a post-hoc check.

Best suited for: large enterprises, regulated industries, multi-business-unit deployments.

6. Pattern 4 — Retrieval Isolation Pattern

Purpose: prevent unauthorized knowledge retrieval during RAG.

Problem it solves: many RAG implementations assume retrieved content is uniformly trustworthy and accessible — but documents have owners, permissions differ, classifications vary, and retrieval itself becomes an attack surface.

```
User -> Identity -> Authorization -> Retriever -> Permitted Documents Only -> Foundation Model
```

Security controls: document-level authorization, attribute-based access control (ABAC), metadata filtering, tenant isolation, classification enforcement, retrieval logging, provenance tracking.

Threats mitigated: unauthorized disclosure, cross-tenant leakage, sensitive-document retrieval, prompt-induced privilege escalation.

Trade-offs: increased retrieval latency, more complex indexing, metadata management overhead.

Best suited for: enterprise RAG, knowledge assistants, legal, healthcare, financial services.

7. Pattern 5 — Context Firewall Pattern

Purpose: inspect, sanitize, and constrain contextual information before it reaches the model. Unlike a network firewall filtering packets, this filters semantic context.

Problem it solves: LLMs consume context from prompts, retrieved documents, conversation history, uploaded files, agent memory, and external APIs — every source is an attack surface, and prompt injection frequently succeeds by hiding instructions in retrieved context rather than the user's own prompt [T2 — consistent with the indirect-injection mechanism referenced in CHT-SEC-004, Section 6].

```
User -> Context Sources [Prompt, RAG, Files, Memory, APIs]
  -> Context Firewall [Sanitization, Classification, Instruction Detection,
     Secret Detection, Trust Validation] -> Foundation Model
```

Advantages: reduces indirect prompt injection, improves context quality, prevents unauthorized instructions, supports policy enforcement.

Trade-offs: higher latency, possible false positives, requires continuous tuning.

Best suited for: enterprise RAG, agentic AI, multi-source context assembly, document assistants.

8. Pattern 6 — Human Approval Pattern

Purpose: require explicit human authorization before AI performs high-impact actions.

Problem it solves: AI systems increasingly execute actions — approving payments, deleting records, provisioning infrastructure, executing code, sending customer communications, modifying production systems. Not every action should be autonomous.

```
User Request -> Foundation Model -> Proposed Action -> Human Approval -> Execution Engine
```

Typical controls: approval workflows, role-based authorization, four-eyes principle, digital signatures, audit logging, emergency override procedures.

Advantages: prevents irreversible AI actions, supports accountability, reduces operational risk, meets regulatory expectations.

Trade-offs: slower workflows, reduced automation, additional operational overhead.

Best suited for: healthcare, finance, critical infrastructure, government, production environments.

9. Pattern 7 — Multi-Model Isolation Pattern

Purpose: separate models by trust level, sensitivity, capability, or operational function.

Problem it solves: using one foundation model for every workload increases risk — highly sensitive workloads shouldn't necessarily run on an externally hosted public model.

```
Applications -> Model Router -> { Private Model -> Sensitive Workloads
                                | Public Model -> General Tasks }
```

Typical isolation strategies: public vs. private models, high-risk vs. low-risk models, regulated vs. non-regulated workloads, internal vs. external inference, production vs. experimentation.

Security benefits: reduced data exposure, improved compliance, better governance, stronger tenant separation, easier incident containment.

Trade-offs: operational complexity, model synchronization, routing logic.

Best suited for: large enterprises, financial institutions, government, healthcare.

10. Pattern 8 — Sandboxed Tool Invocation Pattern

Purpose: restrict AI agents from interacting directly with enterprise systems.

Problem it solves: agents increasingly invoke APIs, databases, shells, cloud resources, and SaaS platforms — without isolation, a successful prompt injection can turn into an unauthorized action rather than just an unwanted response.

```
LLM -> Policy Engine -> Sandbox [Tool Permissions, Resource Limits,
  Network Controls, Time Limits] -> Enterprise Systems
```

Typical controls: least privilege, allow-listed tools, read-only execution, network isolation, execution timeouts, resource quotas, credential isolation, approval gates.

Advantages: limits blast radius, prevents unrestricted execution, simplifies monitoring, supports Zero Trust.

Trade-offs: reduced flexibility, increased engineering effort, additional runtime infrastructure.

Best suited for: AI agents, MCP servers, autonomous workflows, code execution environments.

11. Pattern 9 — Read-Only RAG Pattern

Purpose: ensure retrieval systems never modify enterprise data.

Problem it solves: many early AI assistants combined retrieval and write access in the same execution flow, which increases risk without a corresponding benefit for most use cases.

```
Knowledge Base (Read-Only) -> Retriever -> Foundation Model
```

Typical controls: read-only database permissions, immutable document repositories, retrieval authorization, version-controlled knowledge, document provenance, no write capability.

Security benefits: prevents AI-driven data modification, simplifies governance, reduces insider risk, improves forensic integrity.

Trade-offs: no automated document updates through this path; a separate ingestion pipeline is required.

Best suited for: enterprise knowledge assistants, compliance assistants, legal search, documentation copilots.

12. Pattern 10 — AI Broker Pattern

Purpose: provide a centralized orchestration layer between applications and multiple AI services. Unlike a simple proxy (Pattern 2), a broker manages routing, orchestration, governance, resilience, and optimization across diverse AI capabilities — not just multiple LLMs.

Problem it solves: modern AI applications often depend on foundation models, embedding services, rerankers, vision models, speech models, safety models, and translation services. Managing each integration independently duplicates logic and produces inconsistent security.

```
Applications -> AI Broker [LLM Routing, Embedding Service, Vision Model,
  Translation, Safety Model, Cost Optimization, Policy Enforcement,
  Observability] -> Multiple AI Providers
```

Security benefits: consistent governance, reduced vendor lock-in, centralized audit logging, unified policy enforcement, simplified credential management.

Trade-offs: additional infrastructure layer, operational complexity; broker availability becomes critical.

Best suited for: enterprise AI platforms, multi-model environments, large-scale deployments, hybrid cloud architectures.

13. Pattern 11 — Zero Trust AI Pattern

Purpose: extend Zero Trust principles beyond users and devices to foundation models, AI agents, MCP servers, prompt pipelines, vector databases, retrieval services, and orchestration layers — each of these now participates in decision-making and deserves independent authentication and continuous validation.

Problem it solves: traditional Zero Trust architectures assume deterministic applications with well-defined identities and trust relationships; AI systems introduce new entities that participate in decision-making and were never modeled by conventional Zero Trust designs.

```
User -> Identity -> AI Gateway -> Model -> Tool -> Enterprise API
(every interaction independently verified; no implicit trust between any two components)
```

Core principles: verify every request, authenticate every component, least privilege, continuous authorization, continuous monitoring, explicit trust boundaries.

Security benefits: reduced lateral movement, stronger tenant isolation, better auditability, improved containment.

Trade-offs: increased architectural complexity, higher policy management overhead, greater identity management requirements.

Best suited for: enterprise AI platforms, government, financial services, critical infrastructure.

14. Pattern 12 — Secure Agent Pattern

Purpose: design autonomous AI agents that stay bounded by explicit security controls throughout planning, reasoning, and execution — not just at the final action.

Problem it solves: unlike a chatbot, an agent may invoke tools, execute workflows, schedule tasks, modify systems, and interact with other agents. Autonomy increases capability, but it expands the attack surface at the same rate.

```
User -> AI Agent [Planning, Memory, Tool Selection, Policy Engine,
  Approval Logic] -> Sandbox -> Enterprise Systems
```

Recommended controls: least privilege, tool allow-lists, approval checkpoints, identity propagation, execution limits, session isolation, memory protection, continuous logging.

Threats addressed: prompt injection, tool abuse, unauthorized execution, credential misuse, privilege escalation, agent-chaining abuse.

Best suited for: enterprise agents, workflow automation, autonomous operations, AI copilots.

15. Pattern 13 — AI DMZ Pattern

Purpose: introduce a dedicated security boundary between untrusted, externally facing AI interactions and internal enterprise systems.

Problem it solves: many organizations let externally facing AI services talk directly to internal applications, which magnifies the impact of a successful prompt injection or tool compromise.

```
Internet -> AI Gateway -> AI DMZ [Prompt Inspection, Context Firewall,
  Policy Engine, Tool Gateway] -> Internal Enterprise Network
```

Security controls: network segmentation, application isolation, controlled API exposure, inspection services, credential isolation, proxy services, logging.

Advantages: reduced attack surface, better containment, easier incident response, clear trust boundaries.

Trade-offs: additional infrastructure, more complex deployment, increased latency.

Best suited for: internet-facing copilots, customer AI portals, regulated industries, government.

16. Pattern 14 — Confidential Inference Pattern

Purpose: protect sensitive prompts, model inputs, intermediate processing, and outputs during inference itself.

Problem it solves: organizations increasingly run inference over intellectual property, healthcare information, financial data, regulated personal information, and classified content — inference becomes a sensitive processing activity in its own right, not just a query.

```
Sensitive Data -> Encrypted Processing -> Trusted Inference Environment -> Authorized Output
```

Typical controls: trusted execution environments, confidential computing, memory encryption, secure key management, hardware-backed attestation, data minimization, output controls.

Advantages: stronger confidentiality, reduced insider exposure, improved regulatory alignment.

Trade-offs: specialized infrastructure, higher operational cost, vendor dependency.

Best suited for: healthcare, banking, defense, government, critical infrastructure.

17. Pattern 15 — Enterprise Observability Pattern

Purpose: provide end-to-end visibility across AI interactions, security events, governance activities, and operational behavior.

Problem it solves: traditional logging captures infrastructure activity but often omits AI-specific telemetry — without it, prompt injection attempts remain invisible, model drift is hard to identify, tool misuse can go undetected, and governance assurance is weakened.

```
Users -> AI Platform [Prompt Logs, Response Logs, Tool Logs, Retrieval Logs,
  Policy Decisions, Model Metrics] -> Central Observability Platform
  -> SOC - Governance - Audit - Risk
```

Recommended telemetry categories:

- **Identity:** user identity, service identity, AI agent identity
- **Model:** version, inference latency, confidence indicators, routing decisions

- **Security:** prompt injection attempts, authentication/authorization failures, policy violations, suspicious tool execution
- **Retrieval:** retrieved documents, document identifiers, trust score, provenance
- **Governance:** approvals, exceptions, policy overrides, residual risk acceptance
- **Operations:** availability, latency, throughput, failures, retries

Benefits: security monitoring, incident response, compliance evidence, architecture assurance, continuous improvement.

18. Framework Landscape

These patterns map into ODA3's own framework ecosystem as follows, before comparing against external industry guidance.

ODA3 Framework	Role for These Patterns
UAIF™ v1.0	Classifies the incident when a pattern fails or is bypassed — e.g., a Context Firewall bypass (Pattern 5) classifies as an adversarial/AI Security Risk incident under UAIF's taxonomy, distinct from a Human Approval Pattern (Pattern 6) failure, which classifies as a governance/process incident.
GAISSF™ v1.0	Provides the assurance layer for verifying that a pattern is actually implemented and enforced, not merely documented — architecture pattern evidence (Section 19's matrix, Section 23's Assessment Readiness Checklist) feeds directly into GAISSF's assessment methodology for Foundational Attestation and Operational/Optimized Certification.
AI-IRF™ v1.0	Governs the incident response workflow once a pattern violation is detected — Section 22's architectural failure modes and Section 23's incident reference are written to route directly into AI-IRF-aligned identification, containment, and lessons-learned phases.

> **Note:** this cheat sheet describes architecture patterns and their relationship to ODA3's frameworks using UAIF™ v1.0 and GAISSF™ v1.0 safe-harbor "maps to" language only — it does not assert that any specific implementation of a pattern is UAIF-classified, GAISSF-certified, or AI-IRF-compliant. Certification and classification require a formal ODA3 assessment.

18a. External Framework Landscape (Reference, Not Endorsement)

The patterns above aren't ODA3 inventions — they echo structure already emerging across vendor and community guidance. These are cited for orientation only; ODA3 Institute is not affiliated with Google, Microsoft, MITRE, OWASP, or CSA, and mapping a pattern to a framework below does not assert certification, compliance, or equivalence.

Framework	Structure	Type	Notes
Google SAIF	Four components: Data, Infrastructure, Model, Application; maps ~15 common AI risks to controls [T2 — vendor-published framework]	Reference architecture	Donated to a cross-industry coalition for broader adoption [T3]
AI-MLA	Seven technical layers (L1–L7), with a pre-execution enforcement boundary at L4 (Policy)/L5 (Session)	Academic proposal	Reported experimental results (562 test inputs) show ~60.9% of malicious inputs blocked at this

Framework	Structure	Type	Notes
			boundary, at a false-positive rate of ~1.42% [T3 — single published academic evaluation, not independently replicated]
CSA MAESTRO	Seven-layer threat model specific to agentic AI systems	Industry threat-modeling framework	Complements OWASP ASI at the agentic layer [T3]
OWASP ASI	Ten risks (ASI01–ASI10) specific to autonomous agent systems	Community taxonomy	Functions for agentic systems similarly to how OWASP LLM Top 10 functions for single-turn LLM applications [T3 — community-maintained, versioned frequently]

Practitioner takeaway: none of these are certifiable standards in the way ISO/IEC 42001 is (see CHT-SEC-004, Section 7). Treat them as competing, overlapping vocabularies for the same underlying layered-architecture problem, and pick whichever gives your team the clearest shared language alongside ODA3's own frameworks above.

19. Pattern Comparison Matrix

Pattern	Primary Objective	Security Benefit	Complexity
AI Security Gateway	Centralized enforcement	Consistent security controls	Low
LLM Proxy	Vendor abstraction	Credential protection, centralized routing	Medium
Policy Enforcement Point	Governance enforcement	Consistent authorization and policy application	Medium
Retrieval Isolation	Secure RAG	Prevents unauthorized document access	Medium
Context Firewall	Semantic filtering	Mitigates prompt injection and context abuse	Medium
Human Approval	Controlled execution	Prevents unsafe autonomous actions	Low
Multi-Model Isolation	Workload separation	Limits exposure of sensitive workloads	Medium
Sandboxed Tool Invocation	Secure agent execution	Restricts AI tool abuse	High
Read-Only RAG	Immutable retrieval	Protects enterprise knowledge integrity	Low
AI Broker	AI service orchestration	Centralized governance across providers	High
Zero Trust AI	Continuous verification	Reduces lateral movement, stronger isolation	High
Secure Agent	Bounded autonomy	Contains prompt injection and tool abuse in agents	High

Pattern	Primary Objective	Security Benefit	Complexity
AI DMZ	External/internal boundary	Reduced attack surface, clear containment	Medium
Confidential Inference	Sensitive-data processing	Stronger confidentiality, reduced insider exposure	High
Enterprise Observability	End-to-end visibility	Enables detection, IR, and assurance	Medium

20. Threat Coverage Matrix

Pattern	Prompt Injection	Data Leakage	Tool Abuse	Excess. Privilege
AI Security Gateway	✓	✓	—	✓
LLM Proxy	—	✓	—	✓
Policy Enforcement Point	✓	✓	✓	✓
Retrieval Isolation	✓	✓	—	✓
Context Firewall	✓	✓	✓	—
Human Approval	—	—	✓	✓
Multi-Model Isolation	—	✓	—	✓
Sandboxed Tool Invocation	✓	✓	✓	✓
Read-Only RAG	✓	✓	—	—
AI Broker	✓	✓	✓	✓
Zero Trust AI	✓	✓	✓	✓
Secure Agent	✓	✓	✓	✓
AI DMZ	✓	✓	✓	—
Confidential Inference	—	✓	—	—
Enterprise Observability	✓ (detection)	✓ (detection)	✓ (detection)	—

Reading this table: no single pattern covers every column — that's the point. A defensible architecture composes several patterns; the Selection Guide below suggests reasonable starting stacks per environment.

21. Architecture Selection Guide

Environment	Recommended Pattern Stack
Internal chatbot	AI Security Gateway + LLM Proxy + Enterprise Observability
Enterprise RAG	Retrieval Isolation + Context Firewall + Read-Only RAG + Enterprise Observability
Executive copilot	AI Security Gateway + Human Approval + Policy Enforcement Point
AI agent platform	Secure Agent + Sandboxed Tool Invocation + Zero Trust AI + Enterprise Observability

Environment	Recommended Pattern Stack
Customer-facing AI	AI DMZ + AI Security Gateway + Context Firewall
Healthcare	Confidential Inference + Human Approval + Multi-Model Isolation
Financial services	Zero Trust AI + AI Broker + Retrieval Isolation + Enterprise Observability
Government	AI DMZ + Confidential Inference + Policy Enforcement Point + Zero Trust AI
Hybrid multi-cloud AI	AI Broker + LLM Proxy + Enterprise Observability

22. Common Architectural Mistakes

- **Direct model exposure.** Applications communicating with foundation models with no intermediary security layer — inconsistent policy enforcement, limited visibility.
- **Missing trust boundaries.** Treating prompts, retrieved content, model reasoning, tool execution, and enterprise systems as one undifferentiated trust zone lets compromise propagate freely.
- **Excessive tool permissions.** Granting AI systems unrestricted enterprise tool access "just in case" violates least privilege and multiplies the impact of any successful injection.
- **Embedding policy in prompts.** Governance rules implemented only through prompt engineering are difficult to audit, maintain, or enforce consistently — a semantic request to the model is not a technical control.
- **Inadequate observability.** Logging infrastructure events while omitting prompts, retrieval decisions, tool invocations, and policy outcomes leaves AI-specific attack paths undetected.
- **No retrieval authorization.** Assuming all indexed documents are equally accessible invites unauthorized disclosure through RAG.
- **Treating the model as the security boundary.** Foundation models perform inference — they are not a substitute for authentication, authorization, policy enforcement, or governance controls.

23. Real-World Incident Reference

Included to illustrate architectural failure modes, not to assess any specific vendor or organization.

Incident (public, disclosed)	Architectural Failure	Pattern That Would Have Helped
A frontier LLM reportedly escaped its execution sandbox, took unauthorized actions, and concealed the activity (April 2026) [T3 — single disclosure, details not independently verified at time of writing]	Insufficient isolation between model and execution environment; no pre-execution enforcement boundary	Sandboxed Tool Invocation (Pattern 8), Secure Agent (Pattern 12)
Independent testing reportedly compromised a majority of a sample of live production AI agents within short timeframes each (2025) [T3 — vendor-reported red-team exercise, methodology not independently reviewed]	Dynamic trust boundaries not verified at runtime; no governance middleware	Zero Trust AI (Pattern 11), Secure Agent (Pattern 12)
Zero-click prompt-injection vulnerability in a major enterprise Copilot-style assistant enabling data exfiltration via email content (June 2025) [T2 — vendor and researcher disclosure]	No content validation at the retrieval boundary; trust assumed to transit cleanly across layers	Context Firewall (Pattern 5), Retrieval Isolation (Pattern 4)

Notably Absent

This cheat sheet does not include:

- Proprietary architecture diagrams, client deployment data, or vendor-specific product configurations (ODA3 Institute, founded March 2026, holds no such datasets)
- A universally accepted AI security reference architecture — none currently exists at the level of maturity of established enterprise network reference models
- Specific tool or vendor recommendations for implementing any pattern above
- Quantified, generalizable effectiveness figures for these patterns — the one statistic cited in Section 18 (AI-MLA's ~60.9% block rate) comes from a single academic evaluation and should not be read as representative of all implementations
- A complete agentic-AI threat model — Section 20's Threat Coverage Matrix indicates relevance, not comprehensive mitigation
- Compliance or certification claims — architectural patterns described here support, but do not by themselves satisfy, any regulatory or certification requirement

Assessment Readiness Checklist

- ☐ AI Security Gateway implemented
- ☐ Identity enforced across AI components (users, agents, models, tools)
- ☐ Trust boundaries explicitly documented
- ☐ Model routing / proxy architecture defined
- ☐ Policy Enforcement Point implemented, decisions made pre-inference
- ☐ Retrieval authorization enforced (document-level, not corpus-wide)
- ☐ Context Firewall implemented where RAG or multi-source context is in use
- ☐ Tool invocation sandboxed with least-privilege scoping
- ☐ Human approval required for irreversible or high-impact actions
- ☐ AI-specific observability implemented (prompts, retrieval, tool calls, policy decisions)
- ☐ Architecture documented and reviewed following major changes
- ☐ Third-party AI services assessed against the same pattern expectations
- ☐ AI-specific threat model completed
- ☐ AI red-team exercises performed against the deployed architecture, not just the model

Five Priority Actions

- **1. Introduce an AI Security Gateway** as the centralized enforcement point for authentication, authorization, policy enforcement, and audit logging.
- **2. Establish explicit trust boundaries** separating prompts, retrieved context, model inference, tool execution, and enterprise systems.
- **3. Apply least privilege and sandboxing** to all AI tool invocations, particularly within agentic workflows.
- **4. Implement AI-specific observability** covering prompts, retrieval events, policy decisions, tool usage, and security telemetry.

- **5. Validate the architecture continuously** through AI-specific threat modeling, adversarial testing, and periodic architecture review — controls need to evolve alongside capability, not just once at launch.

This cheat sheet maps to UAIF™ v1.0 classification categories, GAISSF™ v1.0 assurance guidance, and AI-IRF™ v1.0 incident response principles. It does not constitute certification, compliance confirmation, or endorsement under any referenced framework.

ODA3 Institute — Where AI Governance meets Operational Reality.