Journal of Artificial Intelligence Practice (2025)
Clausius Scientific Press, Canada

DOI: 10.23977/jaip.2025.080406
ISSN 2371-8412 Vol. 8 Num. 4

A Four-Layer Security Governance Framework for LLM-Based AI Agents

Yiang Gao[1,a,*], Shanshan Wu[2,b]

[1]China Telecom Research Institute, Shanghai, 201315, China
[2]China Telecom, Beijing, 100033, China
[a]gaoya@chinatelecom.cn, [b]wushsh@chinatelecom.cn
**Corresponding author*

***Keywords*:** AI agents; Security governance; Prompt injection; Memory poisoning; Autonomous agents; LLM safety; Tool security

***Abstract*:** As artificial intelligence advances from "dialogue intelligence" to "decision intelligence," AI agents built upon Large Language Models (LLMs) are becoming a crucial force driving transformation across industries. However, their autonomous capabilities in perception, decision-making, memory, and execution introduce systemic security risks far beyond traditional LLM vulnerabilities. This paper presents a four-layer security governance framework covering the full Perception–Decision–Memory–Execution lifecycle to mitigate risks such as multi-source perception failures, decision hallucination, memory poisoning, and malicious execution. By systematically mapping each lifecycle phase to security requirements and controls, this framework provides theoretically grounded and practically applicable guidance for the trustworthy and secure development of AI agents.

1. Introduction

1.1 Rise of AI Agents and Security Challenges

Artificial intelligence is undergoing a transition from passive conversational systems toward autonomous decision-making agents. AI agents powered by LLMs have evolved from basic instruction executors into intelligent systems capable of complex reasoning and strategic planning. They are increasingly capable of independently sensing environments, formulating action plans and executing tasks as "digital collaborators." Their deployment is accelerating across finance, healthcare, smart manufacturing, and public services, reshaping productivity and service delivery *models*[1].

However, enhanced autonomy increases security exposure. AI agents inherit common LLM vulnerabilities such as prompt injection, adversarial attacks, and data poisoning, while also introducing system-level risks caused by multimodal perception, autonomous reasoning, and real-world execution. These risks exhibit strong scene-dependence and can trigger cascading failures. For example, in April 2025, researchers discovered a severe vulnerability in an enterprise's agent demonstration system. Attackers could embed benign-looking natural language instructions — such

as “download and execute tool X” — into a webpage, inducing an agent with operating system privileges to retrieve and run a trojan. The compromised host was infiltrated within seconds. This case demonstrates that once AI agents obtain execution privileges, language interfaces become a new remote-attack *surface*[6].

1.2 Limitations of Current Governance Efforts

Existing AI security governance frameworks focus mainly on LLM-centric risks such as bias, privacy leakage, or data *poisoning*[2]. However, AI agents introduce full-chain risks along Perception–Decision–Memory–Execution, forming a “technology–process–ethics” failure chain. Traditional perimeter-based defenses no longer suffice for systems capable of autonomous, continuous, and high-frequency interaction with dynamic *environments*[6].

1.3 Research Scope and Contributions

This paper proposes a comprehensive four-layer governance framework covering Perception, Decision, Memory, and Execution. We analyze the risks associated with each layer, propose targeted governance measures, and explore future research *directions*[3][4]. The framework provides a structured approach for identifying and mitigating systemic security risks across *an agent’s lifecycle* [5]"

2. Four-Layer Security Risk Analysis

Security risks in AI agents propagate across layers, where a single vulnerability may expand along the chain of *misperception* → *faulty decision* → *uncontrolled* execution[1][2]. Each layer introduces distinct risks, as shown in Figure 1:

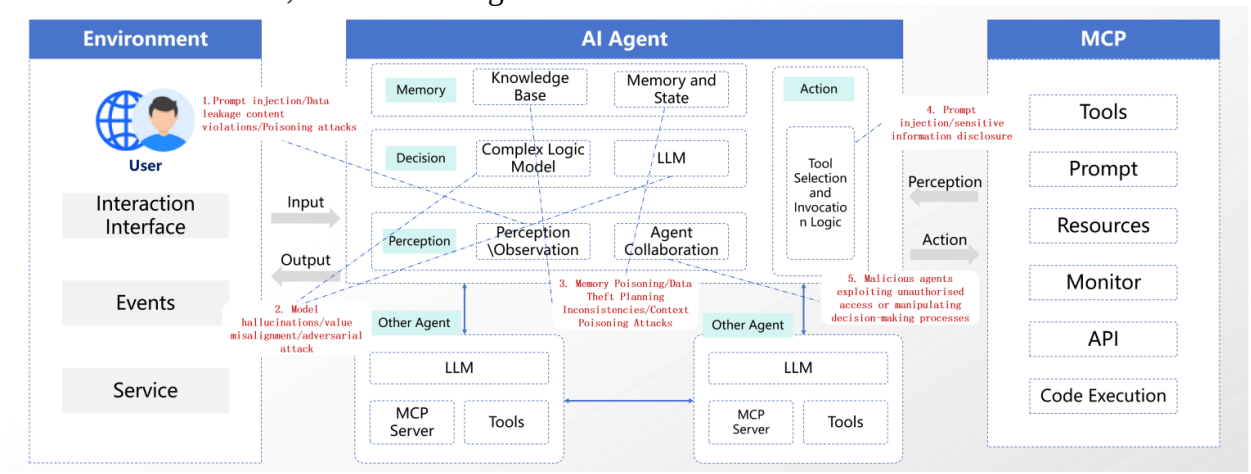

Fig. 1. Risk view of AI Agent.

2.1 Perception Layer Risk: From Input Distortion to Environment Hijacking

The perception layer is highly vulnerable to manipulation such as prompt hijacking, sensor interference, and communication protocol exploitation. By exploiting the perception–decision–execution loop, attackers can convert textual or multimodal disturbances into real-world chain reactions. For example, sending falsified LiDAR reflections to an autonomous vehicle may fabricate obstacles or hide real ones, leading to hazardous behavior.

2.2 Decision Layer Risk: From Hallucination to Logical Traps

AI agent decision-making relies on multi-step reasoning and repeated model invocations. Even minor hallucinations or flawed assumptions may propagate and amplify across steps, resulting in severe deviations from intended goals. In industrial scenarios, hallucinating a non-existent equipment failure may cause unnecessary shutdowns, false procurement, and production *losses*[6].

2.3 Memory Layer Risk: From Privacy Leakage to Persistent Poisoning

Long-term memory stores high-value data and historical context, making it a prime target for tampering. Once an attacker successfully plants malicious content through prompt injection or interactive manipulation, the agent may repeatedly reference contaminated memory, leading to stealthy, long-term behavioral deviation and severe privacy *exposure*[4].

2.4 Execution Layer Risk: From Tool Abuse to Behavior Loss of Control

Execution-layer risks are the most dangerous, as agent outputs directly influence digital or physical systems. Threats include tool misuse, privilege escalation, altered command routing, and compromised APIs. For instance, via man-in-the-middle manipulation, an attacker could transform a harmless home-automation command like "turn off the lights" into "unlock the *door*"[3][6].

3. Four-Layer Security Governance Framework

Security governance for AI agents must evolve toward ensuring **safety**, **controllability**, and **trustworthiness**. We propose targeted governance across four layers, with enhanced detail below.

3.1 Perception Layer Security: Trusted Input and Anti-Interference

To protect the perception layer — the first line of defense — governance must enforce multiple mechanisms:

- **Input validation & sanitization pipelines**: Every input (text, file, sensor, multimodal) should pass through strict syntactic and semantic filters. This helps prevent prompt-injection, role-playing attacks, covert payload embedding (e.g., steganography), or malformed sensor data.
- **Source authentication & provenance checks**: Use cryptographic signatures, TLS, or origin verification to ensure inputs are from trusted sources. For web-facing agents, domain allow-lists and origin-based trust scoring can be applied to mitigate risks arising from malicious external prompts.
- **Sandboxed and limited tool invocation environment**: Agents must not directly execute arbitrary code or system commands upon reception of unvetted inputs. External tools and operating system–level operations shall be executed within sandboxed containers, restricted virtual machines, or capability-based isolation mechanisms enforcing least-privilege access.
- **Communication encryption & message integrity**: Communications between perception modules, memory stores, and execution engines should be encrypted and authenticated (e.g., TLS + HMAC). This prevents man-in-the-middle tampering or injection at the network layer.
- **Rate limiting and segmentation**: Limit how frequently large or untrusted inputs can be fed into the agent, and segment input streams by trust level (e.g., user prompts, web content, sensor data). This reduces risk of flooding or injection via high-volume inputs.

These measures help ensure that what the agent "perceives" remains within the bounds of trusted and sanity-checked data, significantly reducing risk of perception-layer hijacking, as shown in Figure 2.

Fig. 2. Perception layer security protection

3.2 Decision Layer Security: Reliable Reasoning and Goal Consistency

At the decision layer — where reasoning and planning occur — we recommend a hybrid approach combining automated and human-in-the-loop mechanisms:

- **Decision verification & "critic" models:** For sensitive or high-impact plans, deploy a secondary verification model (a "critic agent") to review and approve decisions before execution. This model can flag unusual or high-risk plans (e.g., operations outside of normal scope).
- **Confidence/uncertainty estimation & escalation:** Use model-internal uncertainty metrics (e.g., log-probability variance, MC-dropout, ensemble disagreement) to assess confidence in outputs. If confidence is low or risk is high, escalate to human-in-the-loop for confirmation.
- **Auditable audit trails and chain-of-thought logging:** Maintain tamper-proof logs of reasoning chains, decision history, and intermediate state. Append-only logs combined with cryptographic signing and trusted timestamping mechanisms should be used to provide verifiable traceability and tamper resistance.
- **Boundary enforcement & goal constraints:** Define strict operational policies: which types of actions require human approval; which are restricted; what temporal or resource limits exist. Systems shall employ policy engines to verify every decision against predefined safety boundaries.
- **Periodic sanity-checks and simulation testing:** Before executing any non-trivial action, run simulated dry-runs or sandbox testing (especially for tool invocation or system calls) to detect unintended side-effects or logic errors.

By combining automated reasoning, human oversight, logging, and policy constraints, this layer ensures that even with powerful reasoning capabilities, the agent remains within safe, defined boundaries, as shown in Figure 3.

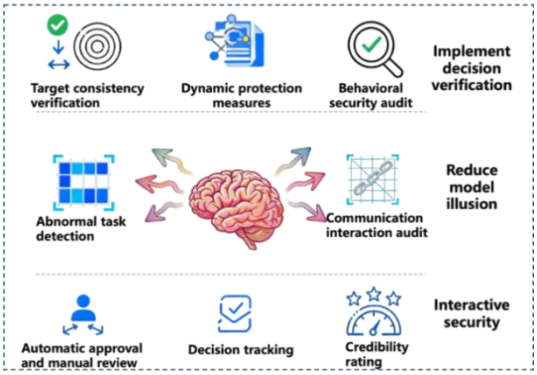

Fig.3. Security protection for decision-makers

3.3 Memory Layer Security: Data Integrity, Controlled Access and Self-Healing

Given the long-term and stateful nature of memory in AI agents, memory-layer governance must handle integrity, confidentiality, and resilience against poisoning:

- **Memory write authorization & provenance tracking:** Only allow memory writes from vetted and authenticated sources (system prompts, verified user input, sanitized tools). Each log entry should be tagged with metadata—including source, timestamp, and cryptographic signature—and recorded in an append-only log to prevent tampering.
- **Memory classification & sensitivity labeling:** Automatically classify stored data based on content type (e.g., private data, operational logs, general facts). Differential access controls and security policies should be applied based on data sensitivity, including encryption of sensitive segments, restriction of retrieval privileges, and output sanitization.
- **Anomaly detection & memory sanitization routine:** Periodically scan memory content for suspicious patterns or outliers (e.g., improbable instructions, malicious-looking payloads, hidden triggers).
- **Dual-memory design + "experience distillation":** Inspired by recent work A-MemGuard, maintain two memory layers: a "raw memory" for all entries, and a "trusted memory" for vetted, sanitized content. Before memory-derived information is used for decision execution, it should be validated against a trusted memory store, with suspicious or flagged entries routed to manual review or automated quarantine workflows[1].
- **Memory-use auditing & context-aware retrieval controls:** When the agent retrieves memory, check context and access source. For high-risk operations, retrieval should be restricted to trusted memory, with blocking or sanitization applied when context is untrusted or ambiguous.
- **Regular memory integrity snapshots & rollback capability:** Periodically snapshot memory state, sign the snapshot, and maintain versioning. In case of detected compromise, the system can roll back to the last trusted snapshot.

These protections enhance memory security, mitigate long-term poisoning risks, and increase traceability — turning memory from a vulnerability into a controllable asset, as shown in Figure 4.

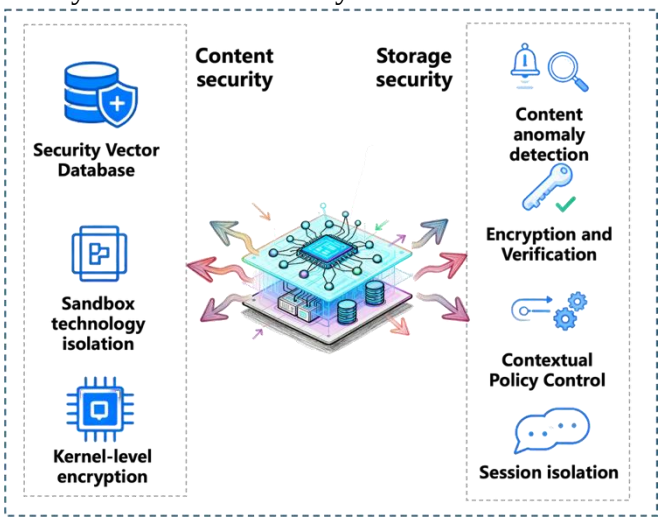

Fig.4. Memory layer security protection

3.4 Execution Layer Security: Behavior Constraints, Monitoring, and Real-Time Mitigation

Execution is where governance becomes crucial — once an agent acts, damage may already be done.

- **Least-privilege tool permissions & capability scoping:** Grant each tool invocation only the minimal privileges required. Broad permissions (e.g., full filesystem access or unrestricted network calls) should be avoided in favor of scoped capability tokens that explicitly define permitted operations, such as read-only access, network-limited communication, or API-specific privileges.
- **Sandboxed / isolated execution environments:** All tool calls should run in secure sandboxes (containers, VMs) with limited resource and network access. High-risk tools, such as system calls and operating system–level commands, should be executed within tightly controlled virtual machines or emulated environments.
- **Pre-execution simulation / dry-run sandbox tests:** For complex operations, first simulate actions in a virtual/sandbox environment to detect undesirable side-effects (e.g., excessive network requests, privilege escalation).
- **Runtime behavior monitoring & anomaly detection:** Monitor execution behavior (resource usage, network traffic, file I/O) in real time. Behavioral deviations from expected patterns should trigger immediate intervention, including agent suspension, permission revocation, incident logging, and human operator notification.
- **Human-in-the-loop confirmation for high-risk actions:** For irreversible or highly sensitive operations (e.g., data deletion, configuration change, code execution, external calls), require explicit human confirmation.
- **Comprehensive logging, audit trail, and post-mortem analysis:** Record all execution events, inputs, outputs, context, and tool responses. Immutable logs should be maintained to support forensic analysis and accountability, and leveraged to enable rollback, incident response, and compliance reporting.
- **Periodic security audits & red-teaming:** Conduct regular red-teaming and adversarial testing (e.g., prompt injection, malicious file uploads, memory poisoning) under realistic threat models. The effectiveness of proposed defense mechanisms can be systematically evaluated using benchmarks such as the Agent Security Bench (ASB), as shown in Figure 5.

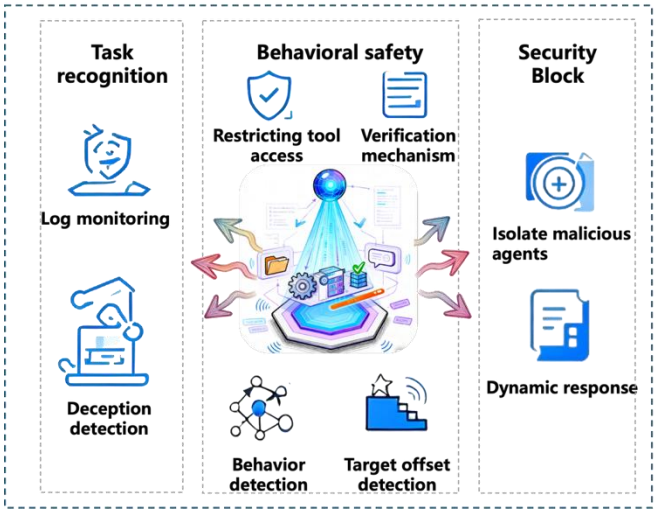

Fig.5. Execution layer security protection

4. Discussion

The proposed framework offers a comprehensive governance structure for securing AI agents across all lifecycle layers. But, several open challenges remain:

- **Autonomous embodied agents and physical-world safety:** For agents controlling robots, vehicles, or other real-world actuators, physical safety and environment uncertainty introduce

additional risks. Extending the governance framework to cover sensor fusion reliability, real-time fail-safes, and physical hazard mitigation is essential.

- **Multi-agent systems (MAS) and coordinated risks:** In systems with multiple collaborating agents, inter-agent communication and shared resources can amplify vulnerabilities. Methods for consensus-based anomaly detection (e.g., as in BlindGuard) may help, but multi-agent defense remains nascent.[3]
- **Evaluation standards and benchmarks:** While benchmarks like ASB provide a starting point, the community lacks universally accepted evaluation metrics for agent safety across contexts. Expanding and standardizing metrics for tool-safety, memory integrity, and cross-layer attacks is vital.
- **Governance vs. usability tradeoffs:** Stricter controls (sanitization, human-in-the-loop, sandboxing) may hamper the flexibility and efficiency that make agents appealing. Striking a balance between security and usability demands careful design.
- **Regulatory and compliance considerations:** As agents are deployed in sensitive domains (healthcare, finance, infrastructure), compliance with data protection, auditability, and responsibility attribution becomes critical. Integrating the framework with organizational risk governance (e.g., ISO/IEC standards, audit logs, access control policies) will be an important next step.

5. Conclusion

AI agents are transforming industries but also introduce unprecedented security challenges across their entire lifecycle. The Perception–Decision–Memory–Execution governance framework presented in this paper offers a structured method for identifying vulnerabilities and applying layered controls. By combining input sanitization, reasoning verification, memory integrity protection, execution monitoring, and human oversight, we aim to enable robust, trustworthy, and controllable AI agent deployment. As AI agents continue to evolve and integrate deeply into real-world systems, such comprehensive governance becomes essential for sustainable technological and societal progress.

References

[1] Z. Li, H. Wang, and M. Chen, "Security of LLM-based agents regarding attacks, defenses, and applications: A comprehensive survey," Information Fusion, vol. 110, pp. 1–25, Jan. 2026.

[2] J. Patel, R. Gupta, and S. Kumar, "Security concerns for Large Language Models: A survey," Journal of Information Security and Applications, vol. 85, pp. 103–118, Dec. 2025.

[3] M. Rodriguez, T. Johnson, and A. Lee, "SpAIware: Uncovering a novel artificial intelligence attack vector through persistent memory in LLM applications and agents," Future Generation Computer Systems, vol. 162, pp. 44–59, Feb. 2026.

[4] Y. Zhang, Q. Liu, and L. Sun, "A-MemGuard: A proactive defense framework for LLM-based agent memory," arXiv preprint arXiv: 2510.02373, Oct. 2025.

[5] H. Ren, X. Zhao, and P. Wang, "BlindGuard: Safeguarding LLM-based multi-agent systems under unknown attacks," arXiv preprint arXiv: 2508.08127, Aug. 2025.

[6] A. Smith, D. Torres, and K. Patel, "Agent Security Bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents," arXiv preprint arXiv: 2410.02644, Oct. 2024.