Security Challenges and Solutions for LLM-Driven AI Agents

Sri Charan Reddy Teegala
CSE 565 Computer Security Survey 1
50681752
teegala@buffalo.edu

***Abstract*—Large Language Models (LLMs) such as GPT and LLaMA have revolutionized natural language understanding, enabling autonomous AI agents that reason, use tools, and conduct multi-step planning. Agentic systems, while beneficial, also inherit and amplify standalone LLMs' security weaknesses such as data leakage, prompt injection, and privilege escalation. This survey summarizes the main threats, defense techniques, and future research directions in securing LLM-driven AI agents. It highlights the need for secure protocols such as the Model Context Protocol (MCP), privilege control systems, and standard evaluation procedures for the safe deployment of intelligent agents.**

***Index Terms*—Large Language Models, AI Agents, LLM Security, Prompt Injection, Model Context Protocol**

I. Introduction

Large Language Models (LLMs) such as GPT-4, Claude, and Gemini have revolutionized artificial intelligence by enabling natural language understanding, reasoning, and generation at scale. However, their increasing deployment in critical domains like medical field has raised serious concerns regarding their security and robustness. Existing research has identified vulnerabilities in LLMs, including prompt injection, data exfiltration, model inversion, and jailbreak attacks [1]. As these models become foundational components in autonomous systems, ensuring their security is no longer optional—it is essential for reliable AI deployment.

A. Agentic AI and Multi-Agent Systems

Current advancements in LLM structures and orchestration frameworks have led to the creation of *agentic AI* systems where LLMs exist as autonomous agents able to reason, plan, and collaborate with humans and virtual worlds. Such tools as LangChain, AutoGen, CrewAI, and OpenDevin enable multi-agent systems where LLMs collaborate to achieve complex goals. These systems can chat, pass memory, and invoke external tools or APIs, forming dynamic *multi-agent networks*.

While the amount of such autonomy introduces unheard of flexibility, it opens up more points of attack. Security threats now extend not only across model-level vulnerabilities but also across inter-agent communication channels, shared memory spaces, and external API interfaces. The attackers can use these as entry points to trigger cascading errors or manipulate decision chains in a series of agents [4], [5]. For example, He et al. [5] demonstrated how communications-based red-teaming could breach multi-agent coordination and expose the weakness of trust mechanisms in autonomous LLM systems.

B. Model Context Protocol (MCP) and Ecosystem Security

To standardize communication between agents and external tools, the *Model Context Protocol (MCP)* was recently introduced as an emerging interoperability layer. MCP defines a structured framework for exchanging contextual data, managing memory, and enforcing access controls among LLMs and connected systems [3]. By treating context as a secure, structured API, MCP aims to mitigate risks from untrusted input and reduce the potential for prompt injection attacks.

However, while MCP improves modularity, it also introduces new security considerations. Improper context isolation, inadequate boundaries, or insecure context serialization may lead to leakage of sensitive data or unauthorized tool calls. As LLM-driven agents increasingly depend on standardized context protocols, ensuring the security of these interaction layers becomes a key challenge for the next generation of AI systems.

C. Scope and Contributions

This survey provides an overview of recent developments in securing LLM-driven AI agents. We explore (1) emerging techniques for securing agent communication and orchestration, (2) vulnerabilities and challenges unique to Agentic systems, and (3) the evolving role of context protocols such as MCP in enabling trustworthy multi-agent collaboration. By consolidating current findings, this work aims to highlight open research challenges and guide the development of resilient, transparent, and secure LLM-based ecosystems.

II. Main Techniques

This section outlines the main approaches and defense mechanisms that have been proposed to secure Large Language Model (LLM) driven AI agents. These techniques address vulnerabilities in data integrity, communication, and execution environments across both single-agent and multi-agent settings.

A. Prompt Injection and Context Sanitization

Prompt injection remains one of the most prevalent attack vectors in LLM systems. Malicious actors can insert hidden or deceptive instructions into model inputs, causing the LLM to

reveal confidential information or perform unintended actions [1].

To mitigate such attacks, recent research has focused on *context sanitization*—the process of inspecting, filtering, and restructuring input prompts before they reach the model. Common methods include:

- Static Filtering: Using regular expressions or pattern-based filters to detect potentially harmful instructions.
- Context Segmentation: Structuring conversation history or task memory into distinct trust zones to prevent cross-context leakage.

While effective against low-level injections, these methods struggle with adversarially constructed prompts or obfuscated attack payloads, suggesting the need for deeper semantic filtering and behavioral modeling.

B. Sandboxed Tool Calls

In agentic systems, LLMs often invoke external tools, APIs, or operating system commands to perform complex tasks. However, improper execution boundaries can lead to unauthorized actions or data exfiltration. To prevent this, researchers recommend a *sandboxed tool invocation* approach [3].

Sandboxing involves isolating tool execution environments from model logic. Techniques include:

- Execution Wrappers: Limiting LLM access to pre-approved tool functions and validating parameters before execution.
- Capability Tokens: Assigning temporary, scoped credentials for tool access that automatically expire after use.
- Audit Logging: Recording every tool call and response to trace malicious or unexpected behavior.

Such techniques ensures that even if the LLM output is not as expected, its ability to interact with external systems remains controlled.

C. Secure Multi-Agent Communication

In multi-agent ecosystems, security extends beyond a single LLM instance to the communication between agents. Message interception, tampering, or impersonation can lead to cascading system failures or coordinated misinformation [4], [5].

To address this, secure communication protocols are being developed with the following core principles:

- Authentication: Each agent must verify the identity of the sender before processing a message.
- Message Integrity: Digital signatures or hashing schemes prevent message modification in transit.
- Trust Scoring: Agents maintain trust levels for their co-agents based on historical reliability or message validation rates.

He et al. [5] demonstrated how communication-based red-teaming can expose weaknesses in these trust mechanisms, encouraging further research into decentralized verification and consensus-based communication protocols.

D. Model Context Protocol (MCP) Security Layers

The Model Context Protocol (MCP) provides a standardized interface for managing LLM context, tool access, and memory. While designed for interoperability, MCP's centralized context management also presents new attack surfaces. To address this, several MCP-specific defense layers have been proposed [3]:

- Context Isolation: Separating trusted and untrusted memory to reduce cross-context contamination.
- Permissioned Context Access: Restricting which agents or subsystems can modify shared context entries.
- Structured Context Validation: Ensuring all contextual data conforms to schema-based integrity checks before being injected into the model.

These measures align with trends in secure orchestration, emphasizing transparency and least-privilege access in Agentic AI Frameworks.

E. Red-Teaming and Continuous Evaluation

Finally, an emerging practice for maintaining security in LLM ecosystems is systematic *red-teaming*, the process of simulating attacks to test system response [5]. Red-teaming can uncover vulnerabilities in agent communication, reasoning consistency, and context handling.

Modern approaches include:

- Human-in-the-Loop Evaluation: Combining expert oversight with LLM testing to detect nuanced security gaps.
- Feedback Loops: Integrating discovered vulnerabilities into retraining or fine-tuning processes for continuous hardening.

Through iterative evaluation, companies can build more trustworthy and adaptive AI systems that evolve with the advancing threat patterns.

III. Issues and Problems

Despite recent progress in securing Large Language Models (LLMs) and agentic AI systems, several unresolved issues persist. These challenges stem from the complex interactions between agents, their dynamic use of context, and the unpredictable behaviors of LLMs in adversarial environments. This section discusses key problems that remain at the forefront of LLM security research.

A. Context Contamination and Memory Leakage

One of the most critical challenges in agentic systems is *context contamination* the unintended mixing of trusted and untrusted data within shared memory. When multiple agents operate over the same context or knowledge base, malicious inputs from one agent can propagate to others, influencing downstream reasoning or actions [2], [4].

In addition, long-term memory modules that contain contextual information across sessions increase the risk of *data leakage*. Sensitive data stored in these memory systems can be indirectly exposed through reconstruction attacks or model probing. The absence of strong encryption or access control over persistent memory worsens this issue.

B. Prompt Injection and Semantic Obfuscation

Although prompt filtering and guardrails are common, attackers continue to manipulate LLMs through indirect prompt injections and semantic obfuscation [1]. Such attacks hide malicious intent behind benign-sounding instructions or multi-turn reasoning chains, allowing harmful behaviors to bypass static or rule-based filters.

Furthermore, the lack of formal semantics for LLM prompts makes it difficult to define what an "attack" is. This makes it difficult to create universally reliable detection models, as safety violations often depend on subtle linguistic or contextual hints.

C. Tool Misuse and Execution Privilege Escalation

In agentic models, LLMs are able to execute external tools, APIs, and scripts. This, while powerful, is a significant attack surface for *tool misuse* and *privilege escalation* [3].

An attacker may be capable of crafting inputs that cause an agent to:

- Execute unforeseen commands.
- Read unauthorized files or network resources.
- Link a series of harmless actions to achieve a nefarious outcome.

In spite of sandboxing protection, subtle flaws in parameter validation, API composition, or capability delegation may be exploited indirectly. The challenge lies in balancing agent autonomy and operational security.

D. Unverified Inter-Agent Communication

Agents powered by LLMs often collaborate in unverified natural language communication or lack formal encryption protocols. This leaves multi-agent systems vulnerable to *agent-in-the-middle* attacks—where attackers intercept or inject spurious messages between agents [5].

Moreover, there are no standard messages on the world scale to validate origin or guarantee semantic intent. Without robust trust mechanisms, deception or corrupt agents can rapidly distribute false information, pollute shared memory, or affect decision conclusions.

E. Evaluation Gaps and Lack of Standard Benchmarks

One of the main limitations of existing LLM security studies is the lack of standardized benchmark metrics to assess agent-level vulnerabilities. The vast majority of current metrics are centered around static LLM performance (e.g., hallucination rate or factual accuracy), rather than on dynamic multi-agent threat scenarios [5].

This evaluation gap renders it impossible to compare defenses accurately or quantify progress. Additionally, red-teaming methods are typically ad hoc without any single taxonomy for categorizing communication-based or context-based attacks. Facilitating open, reproducible security evaluation frameworks remains an immediate research need.

F. Ethical and Governance Challenges

Finally, with ever-more autonomous LLM-based agents, accountability, transparency, and control problems emerge. No one knows who to hold responsible when an agent chooses to inflict damage or act immorally—its developer, user, or model creator.

The lack of well-defined governance structures and transparencies also creates dual-use threats: techniques designated for red-teaming or probing can themselves be dual-used against ill intent. Thus, ensuring effective responsible disclosure and collaboration between security researchers, developers, and regulators remains an unresolved problem.

IV. Future Trends

The development of LLM-based AI agents brings with it both increasing capabilities and an increasing attack surface. Future lines of inquiry highlight enhancing the security stance of such systems through secure architecture design, provable behavior, and federated coordination mechanisms.

A. Secure Multi-Agent Coordination

As LLM-based ecosystems evolve into distributed and cooperative systems, research in the future will tackle secure multi-agent coordination. Techniques such as trust scoring, reputation-based access, and dynamic threat modeling can offer real-time detection of compromised or malicious agents. Cryptographic signatures and audit trails embedded in agent communications can ensure message authenticity and provenance.

B. Formal Verification and Behavioral Guarantees

A new direction is applying formal verification to agentic AI. By combining symbolic reasoning and model interpretability tools, researchers attempt to establish mathematical guarantees of properties such as safety, integrity, and bounded reasoning. Verification frameworks that are specialized for LLM-based decision pipelines can be applied for prompt injection prevention and prevention of unwanted self-replication behaviors.

C. Context Isolation and Secure MCP Implementations

The Model Context Protocol (MCP) will continue to evolve as a primary facilitator of modular and context-aware agent systems. Later iterations of MCP may include fine-grained permission control, sandboxed execution contexts, and encrypted context exchange. These features can be employed to thwart cross-agent data leakage and context poisoning attacks.

D. Human-in-the-Loop and Adaptive Governance

Future LLM agents will probably include adaptive governance frameworks, in which human stewardship actively adjusts agent autonomy in accordance with risk estimation. Policy-driven orchestration layers can offer continuous compliance with ethical and legal norms, as well as enable situational intervention when anomalous or unsafe behavior is detected.

E. Privacy-Preserving Collaboration and Federated Security

Collaborative LLM systems will be going in the direction of employing federated learning and secure multi-party computation (SMPC) to preserve privacy while sharing intelligence among agents. The trend promotes resistance against model inversion and data extraction attacks and enables trustless collaboration without exposing data directly.

V. Conclusion

This survey explored the rapidly evolving realm of AI agent security in LLM, where the area of intersection between model safety, communication integrity, and contextual robustness was highlighted. Novel architectures such as the Model Context Protocol (MCP) were discussed, as well as the vulnerabilities that happen in multi-agent cases, including prompt injections, impersonation of agents, and cross-context contamination.

Since agentic AI makes its way toward autonomous and collaborative ecosystems, secure coordination and responsible operation will be central to its success. Current mitigation strategies, while valuable in themselves, require reinforcing governance frameworks with inherent flexibility and verifiable control.

In the coming years, integration of formal verification, context isolation, and human-in-the-loop monitoring is quite promising for enabling safe LLM ecosystems. Creating open standards and interoperable security protocols will be needed to enable safe deployment of intelligent, networked agents at scale.

References

[1] S. Abdali, R. Anarfi, C. J. Barberan, J. He, and E. Shayegani, "Securing Large Language Models: Threats, Vulnerabilities and Responsible Practices," *arXiv:2403.12503*, Mar. 2025.

[2] F. He, T. Zhu, D. Ye, B. Liu, W. Zhou, and P. S. Yu, "The Emerged Security and Privacy of LLM Agent: A Survey with Case Studies," *arXiv:2407.19354*, Jul. 2024.

[3] X. Hou, Y. Zhao, S. Wang, and H. Wang, "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions," *arXiv:2503.23278*, Mar. 2025.

[4] D. Kong, S. Lin, Z. Xu, Z. Wang, M. Li, Y. Li, Y. Zhang, H. Peng, Z. Sha, Y. Li, C. Lin, X. Wang, X. Liu, N. Zhang, C. Chen, M. K. Khan, and M. Han, "A Survey of LLM-Driven AI Agent Communication: Protocols, Security Risks, and Defense Countermeasures," *arXiv:2506.19676*, Jun. 2025.

[5] P. He, Y. Lin, S. Dong, H. Xu, Y. Xing, and H. Liu, "Red-Teaming LLM Multi-Agent Systems via Communication Attacks," *arXiv:2502.14847*, Feb. 2025.