oneconsult
together against cyberattacks

Agentic AI Security

Threats, Attack Vectors, and Practical Security Strategies for AI Applications



White paper by

Dr. Marco Bertenghi

April 2026

Abstract

The rapid spread of artificial intelligence (AI) is fundamentally transforming business processes. AI agents are increasingly acting autonomously, calling APIs, using internal tools, and making independent decisions. While this autonomy offers enormous potential, it also creates a new and extensive attack surface for cybercriminals that cannot be addressed by conventional security tests.

This white paper provides a comprehensive overview of the threat landscape for agentic AI systems. It defines key terms, analyzes the most significant attack vectors, and maps them to current security standards such as the OWASP security catalogs for AI and agentic systems. In addition, it presents practical and realistic security strategies.



Key Findings

Prompt injections remain the most dangerous attack vector for AI applications and are massively amplified by agentic systems.

Agentic frameworks with tool access, planning capabilities, and memory create an entirely new category of security risks, ranging from agent goal hijacking to memory poisoning.

Conventional security testing is not sufficient. Novel vulnerabilities require specialized penetration tests focused on AI security.

Defense in depth is the key, as individual protection mechanisms are insufficient. Only multi-layered approaches combined with the principle of least privilege offer the best protection.

Proaktive measures are crucial. Organizations must test and secure their AI applications before attackers can actively exploit vulnerabilities.

Dr. Marco Bertenghi

Senior Penetration Tester



Education

Dr. Marco Bertenghi earned his doctoral degree from the Zurich Graduate School in Mathematics (ZGSM, UZH/ETHZ) in the field of mathematics, specializing in memory and diffusion processes, inspired by the dynamics of artificial intelligence (AI) systems. Since then, he has seamlessly applied this solid analytical and mathematical foundation to the field of cybersecurity.

Practical Experience

Since 2023, Marco Bertenghi has been working as a penetration tester (since 2026 Senior) at Oneconsult AG, where he actively contributes to strengthening companies' cyber resilience through security testing. He has already led several projects focused on AI security – including in the insurance industry, the education sector, and the banking industry.

Certifications

Marco Bertenghi is certified OSSTMM Professional Security Tester (OPST) as well as Burp Suite Certified Practitioner (BSCP) – certifications that specifically complement his academic training and practical experience.

Engagement

In addition, Marco Bertenghi is actively involved in the internal Virtual Competence Center (VCC) at Oneconsult, promoting the use of AI, and serves as a guest lecturer at the University of Applied Sciences and Arts Northwestern Switzerland (FHNW), teaching courses on penetration testing and red teaming with a focus on AI.

> "With specialized AI expertise in the field of agentic systems and large language models (LLMs) as well as extensive experience in cybersecurity, Marco Bertenghi is a proven expert in agentic AI security.

oneconsult
together against cyberattacks

Contents

05 Growing Threat to AI Applications

06 - 07 Fundamentals of Agentic AI Systems

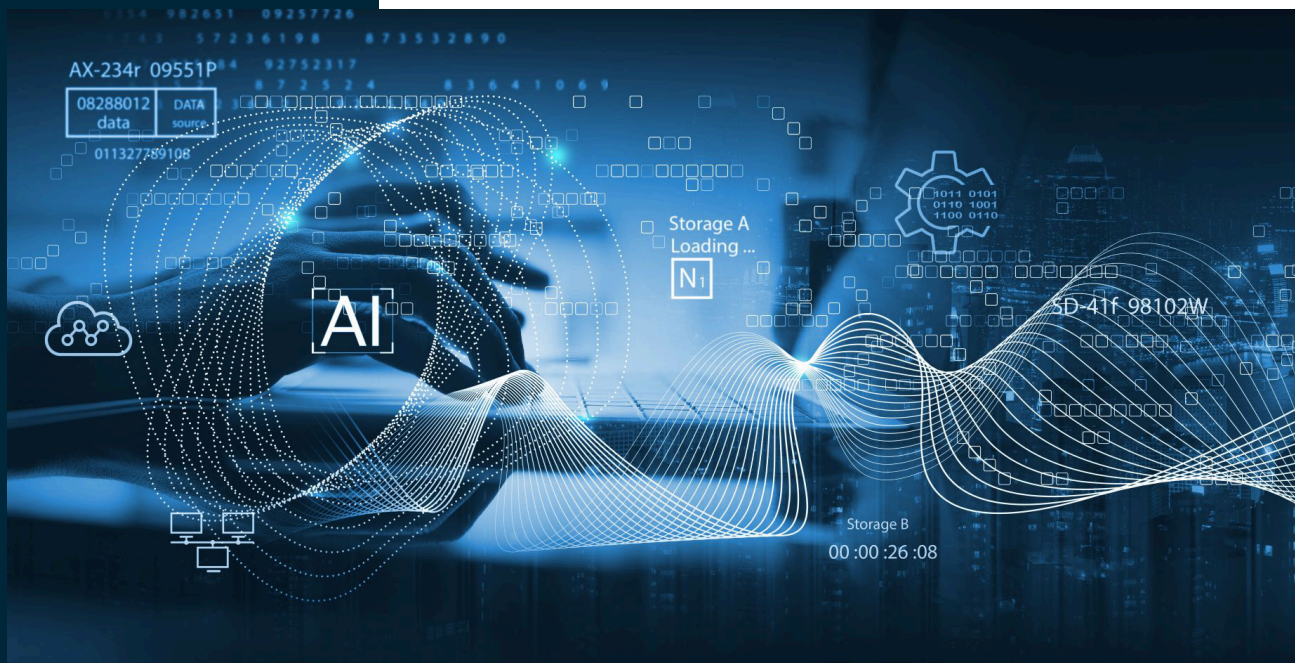
08 - 10 Attack Vectors in Detail

11 - 13 Practical Security Strategies

14 Limits of Traditional Security Testing

15 - 16 Conclusion & Recommendations

17 - 20 Appendix (Glossary, Sources, Resources)



The Growing Threat to AI Applications

Artificial intelligence (AI) has undergone unprecedented development in recent years. While in 2023 it still consisted mainly of simple chatbots – with OpenAI's ChatGPT being the prime example – it has evolved into an ecosystem of highly complex, autonomous systems. Today, companies are deploying AI applications across various business areas: from intelligent customer support chatbots and retrieval-augmented generation (RAG) systems to fully autonomous AI agents that independently plan tasks, access tools, and make decisions.

As these systems become more widespread, the attack surface increases dramatically. The OWASP LLM Top 10[1] – an internationally recognized catalog of the most critical security risks for AI language models – identifies targeted manipulation through injected instructions as the greatest risk. However, agentic AI systems amplify this threat many times over: A successfully manipulated agent can not only disclose information but also actively carry out malicious actions, access internal systems, and abuse its privileges.

Reality shows that these risks are not merely theoretical scenarios. In January 2026, exposed admin panels, stolen API keys, and active infostealer campaigns were discovered for the AI agent OpenClaw within 72 hours of its release[2]. Security researchers had previously demonstrated that popular coding agents such as GitHub Copilot and Google Jules could be completely compromised through prompt injections, including the execution of malicious code.

Agentic AI systems operate autonomously, with extensive access privileges and often deep within the corporate infrastructure. **A compromised agent is therefore not a passive data leak, but an active attacker within your network**

oneconsult
together against cyberattacks



This white paper is intended for decision-makers, security managers, and technical leaders who are implementing AI applications in their organizations or planning to do so.

There are three main objectives:

1. **Raising awareness:** Providing a thorough understanding of the threat landscape for agentic AI systems, based on current research findings and real-world security incidents.
2. **Guidance:** Introducing the most important security standards and assessment frameworks, including the OWASP LLM Top 10[3], the OWASP Top 10 for Agentic Applications[4], and the OWASP MCP Top 10[5].
3. **Actionability:** Providing concrete, practical, and realistic security strategies that organizations can implement immediately.

After reading this white paper, you will be able to realistically assess the security risks of your AI applications, prioritize appropriate protective measures, and make informed decisions regarding specialized security testing.

Fundamentals of Agentic AI Systems

To understand the security risks of agentic AI systems, it is necessary to clearly distinguish between the different system types and their architectural components.

From Chatbots to Autonomous Agents

AI applications can be divided into three categories based on their level of autonomy:

Chat applications (LLMs)

The user interacts directly with a language model, such as ChatGPT.

The application processes text inputs and returns text responses. In some cases, multiple modalities are supported, such as images, audio, and video. Security risks primarily concern prompt injection and the disclosure of sensitive data.

RAG systems (retrieval-augmented generation)

The language model is supplemented with external data sources, such as in Google NotebookLM.

For each query, relevant documents are retrieved from a knowledge base and provided as context. In addition to the LLM risks, attack vectors arise from manipulated documents and insecure data pipelines.

Agentic systems

AI agents, such as OpenClaw or GPT-Codex, act autonomously.

They plan multi-step tasks, call upon external tools and APIs, store information in memory, and delegate tasks to other agents. These systems provide the largest attack surface, as successful attacks can lead to uncontrolled actions with real-world consequences.

The autonomy of agentic systems is not an optional feature, but their core purpose. Aside from simple chatbot applications, LLMs alone offer only limited value to businesses. It is only through the ability to independently plan, make decisions, and intervene in systems that they realize their full economic potential.

However, it is precisely this autonomy that places agentic systems in a highly privileged position within the IT landscape, making them an extremely attractive target for attackers.



Architectural Components of Agentic Systems

A typical agentic AI system consists of several core components, each of which poses its own security risks:

Components	Functions	Security Risks
LLM (thinking engine)	Natural language processing, decision-making	*Prompt injections, jailbreaks*
LLM (planning module)	Breaking down complex tasks into sub-steps	*Goal hijacking, plan manipulation*
Tool integration	Access to APIs, databases, file systems	*Tool misuse, rug pull, privilege escalation*
Memory/context	Storing information across sessions	*Memory poisoning, context leakage, and context rot*
Multi-agent orchestration	Delegation and communication between agents	*Agent communication poisoning, rogue agents*

The Threat Landscape: Attack Vectors in Detail

The threat landscape for AI applications is complex. The following section systematically outlines the most important attack vectors, starting with the most commonly exploited and moving on to the most technically sophisticated.

Prompt Injections: The Most Dangerous Attack Vector

Prompt injections are ranked first in the OWASP LLM Top 10 and remain the most commonly used attack vector against AI applications. In these attacks, attackers manipulate inputs to an AI system to bypass security mechanisms, alter the intended behavior, or extract sensitive data.

There are three main categories:

Direct prompt injection:

The attacker enters malicious instructions directly into the user input field to expose system prompts, bypass security rules, or trigger unauthorized actions. **Despite numerous protective mechanisms, recent studies show that even advanced defenses can be bypassed with a success rate of over 80 percent.**

Indirect prompt injections:

Malicious instructions are embedded in data that the AI system processes from external sources, such as websites, documents, emails, or database entries. This is particularly critical for RAG systems and agents with internet access, as attackers do not require direct access to the system.

Multi-turn attacks:

The attacker distributes the malicious instruction across several consecutive messages to bypass context-based protection mechanisms. Each individual message appears harmless, but taken together, they lead to the desired manipulation.



Current research shows that most published defense mechanisms provide a false sense of security. In the study "The Attacker Moves Second"[6], all 12 defenses examined were bypassed with a success rate of over 70 percent, and in some cases even over 90 percent.

Attacks on Agentic Frameworks

Agentic AI systems significantly increase the attack surface.

The OWASP Top 10 for Agentic Applications identifies the following attack vectors, among others:

Agent goal hijacking (intent hijacking):

Through manipulated inputs, the agent is tricked into pursuing a goal entirely different from what the developers intended. For an agent with access to tools, this can lead to it performing malicious actions, such as deleting files, exfiltrating data, or executing malicious code.

Tool misuse and privilege escalation:

Agents with access to tools and APIs can be manipulated to use them beyond their intended scope. Particularly in cases of insufficient access control, an agent may perform privileged actions for which it is not authorized.

Memory poisoning:

Attackers manipulate an agent's persistent memory to influence its future behavior. Attacks such as SpAIware[7] and ZombieAgent[8] demonstrate that once injected, malicious code remains persistent across session boundaries and is activated during subsequent interactions.

Agent communication poisoning:

In multi-agent systems, attackers can manipulate communication between agents. Research findings show that all of the LLMs examined are vulnerable to such inter-agent compromises[9].

Overwhelming human-in-the-loop:

Agentic systems can overwhelm the human monitoring process by bombarding users with an excessive number of confirmation requests. This leads to decision fatigue and, ultimately, to the blind approval of potentially harmful actions.

MCP Protocoll: New Attack Surface Due to Standardization

The Model Context Protocol (MCP)[10] from Anthropic has established itself as the de facto standard for tool integration in AI systems. However, its growing adoption brings specific security risks, which are documented in the OWASP MCP Top 10:

Tool poisoning:

A malicious MCP server provides manipulated tool descriptions or tool results that contain hidden instructions. The agent executes these instructions without users noticing, as the malicious instructions are hidden within the metadata.

Rug-pull attacks:

An MCP server that initially appears trustworthy changes its behavior after installation and performs malicious actions. Since MCP servers can be hosted remotely, the server operator can change the behavior at any time.

Transitive trust attacks:

Trust granted to an MCP server is implicitly transferred to other, potentially untrusted servers. This enables attack chains across multiple servers.

RAG Pipeline and Output Handling

Retrieval-augmented generation (RAG) extends AI systems with external data sources, thereby creating new attack vectors:

Document poisoning:

Attackers inject manipulated documents into the knowledge base. The PoisonedRAG attack[11] achieves a success rate of over 90 percent with just five poisoned documents by specifically manipulating the context. The U.S.-based company Anthropic, which focuses on AI, has also conducted research in this area and demonstrated that even a small number of poisoned examples are sufficient to specifically influence a model's behavior[12].

Insecure output handling:

The AI system's outputs are further processed by downstream components such as renderers, browsers, or shells. Manipulated outputs can trigger classic attack vectors such as cross-site scripting (XSS), server-side request forgery (SSRF), or remote code execution (RCE).

Data leakage via the context:

Sensitive data from the knowledge base can be exposed through targeted prompt manipulation, especially when document-level access control is lacking.

Practical Security Strategies

Securing agentic AI systems requires a multi-layered approach. It's not possible for a single security measure to cover all attack vectors. Based on current research findings and practical experience, we recommend a three-tiered model of measures prioritized according to implementation speed and effectiveness.

Defense in depth is the key. You should never rely on a single security measure. Instead, combine architectural measures with operational controls and continuous monitoring.

Week 1-4

Immediate measures: The following measures offer the greatest security benefit with relatively little effort and should be implemented as a priority:

Principle of least privilege

- Limit each agent's access rights to the absolute minimum required for their task.
  - Start by basing this on the authorizer's permissions. An agent should never be granted more rights than the users who assign tasks to them. This serves as an upper limit from which the agent's rights must be further restricted.
  - Take special care to ensure that even privileged users, such as administrators, do not receive automatically extended agent rights. Task-based restrictions should apply regardless of the client's status
- Implement granular permission models for every single tool access.
- Use short-lived, task-specific tokens instead of long-lived API keys.

Input validation and sanitization

- Implement strict input validation at all levels: user prompts, system prompts, tool inputs, and tool outputs.
- Consider all inputs as potentially malicious, regardless of their source.
- Set realistic length limits for input tokens.

Secure output processing

- Consider all LLM outputs to be untrustworthy.
- Implement output encoding and sanitization before passing results to downstream systems.
- Validate structured outputs against defined schemas. By default, reject all requests that do not conform to the expected schema.
- Define realistic length limits for output tokens as well as rate limits, particularly for tool calls. Limits that are set too generously or are missing not only enable resource abuse but can also lead to a denial-of-wallet (DoW) attack due to excessively long outputs.

Human-in-the-loop (HITL) for critical actions

- Enforce human confirmation (HITL) for all safety-critical actions, such as database write operations, file modifications, and external API calls.
- Design confirmation prompts to avoid decision fatigue.

Month 2-3

Mid-term measures: These measures require more planning effort but offer a lasting security benefit:

Sandboxing and isolation

- Run all tool invocations in isolated environments. Use containers, virtual machines, or micro-VMs to contain the impact of compromised agents.
- Implement network segmentation to prevent lateral movement.

Comprehensive logging and monitoring

- Log all agent actions seamlessly, including planning steps, tool calls, inter-agent messages, and decision paths. Ensure that logs do not contain personally identifiable information (PII) or other sensitive information – for example, through automated masking or filtering prior to storage. Since complete avoidance is not technically possible, appropriate protection and access controls must be applied to the log data.
- Implement real-time notifications for anomalous behavior patterns.

Centralizing policy enforcement

- Implement a dedicated policy and gateway layer that consistently enforces authentication, authorization, consent management, tool filtering, and audit logging. By centrally evaluating all requests, you ensure that your security policies are applied uniformly and without gaps across all agents, servers, and upstream services.

MCP server hardening

- Pin MCP server versions, validate tool descriptions against known patterns, and implement server integrity checks.
- Use only trusted, audited MCP servers over encrypted channels (TLS 1.2+).

Specialized security testing

- Conduct regular AI penetration tests specifically designed to address the novel attack vectors of agentic systems. **Conventional vulnerability scans cannot reliably cover these risks.**

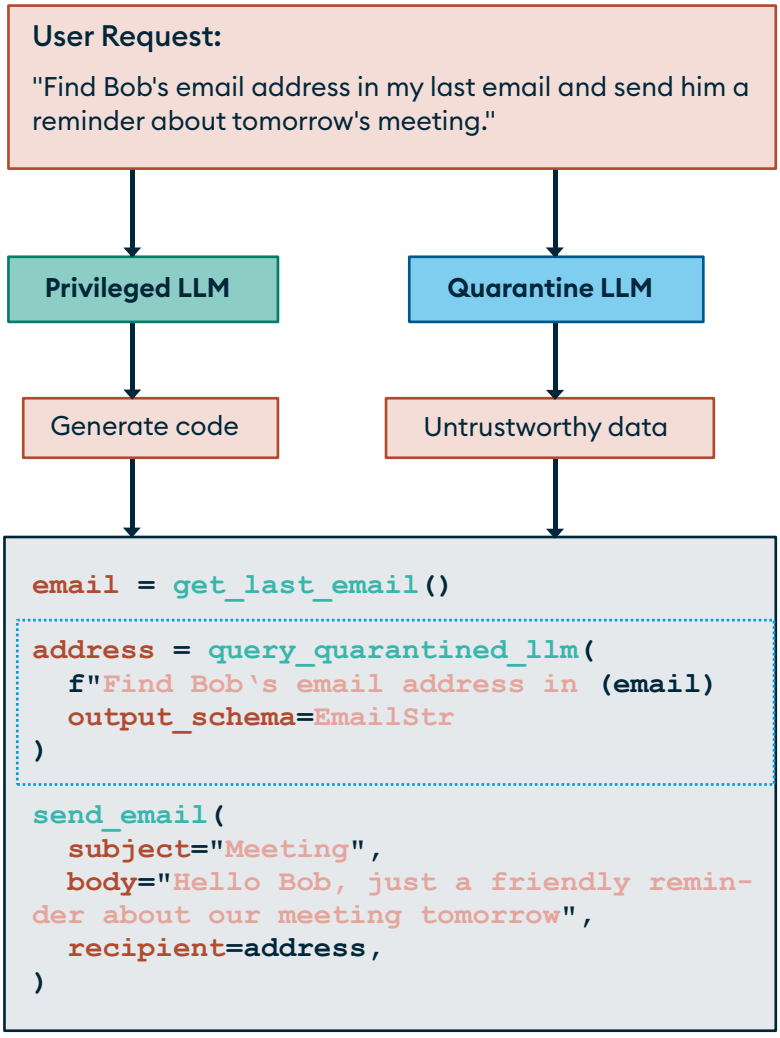
From second quarter

Strategic measures: These measures require architectural changes as well as long-term investments and must therefore be strategically planned and implemented.

Dual-LLM architecture

Separate the language model that processes user requests from the model that makes safety-critical decisions. Approaches such as CaMeL (**Ca**pabilities for **M**achin**e** **L**earning) from Google DeepMind[13] use a separate model – one that cannot be influenced by user input – to authorize tool calls.

Example of a dual-LLM architecture introduced by Simon Willison[14] and used in the CaMeL approach



The dual architecture consists of a privileged LLM and a quarantine LLM:

- The privileged LLM receives instructions, plans actions, and can execute tools.
- The quarantine LLM can be invoked by the privileged LLM whenever untrusted data needs to be processed. The quarantine LLM cannot execute tools; it merely generates text.

In the example shown, a privileged LLM (in green), possibly after prior planning, generates code to read email addresses and send emails.

The non-privileged quarantine LLM (in blue) is called by the privileged LLM solely to populate the relevant text-based data fields, without being able to generate code or execute additional tools itself. The required data comes from an untrusted source (the email), which may contain prompt injections.

Constitutional Classifiers

Implement specialized classification models that scan inputs and outputs for malicious content in real time. Approaches such as Anthropic's Constitutional Classifiers have reduced the jailbreak success rate from 86 to 4.4%[15].

Why Traditional Security Tests Are Not Enough

As demonstrated in the previous sections, both the threat landscape of AI applications and the necessary security measures are multifaceted and complex.

This is precisely why conventional vulnerability scans and application tests are insufficient to address the specific risks posed by these systems. They are designed for established attack patterns such as SQL injection, XSS, IDOR, or RCE and do not cover the complex, AI-specific attack landscape that AI applications entail.

The risks of AI applications are fundamentally different:

- **Prompt injections** are based on natural language and cannot be detected with signature-based scans.
- **Agentic vulnerabilities** such as goal hijacking or memory poisoning arise from the interplay of model behavior, orchestration logic, and system architecture.
- **The attack vectors** are highly dynamic and require creative, manual attack simulations by specialized penetration testers with expertise in AI security.
- **Tool integrations and MCP configurations** must be evaluated within the context of the overall architecture, not in isolation from it.

A specialized AI penetration test closes this gap. It combines automated attack techniques with manual verification and uncovers novel vulnerabilities that conventional tests cannot detect.



Automated attack techniques form the first layer: Specialized frameworks systematically generate thousands of attack variants and probe the model for various types of prompt injections and potential for abuse, such as jailbreaks.

However, since automated tools primarily cover known attack patterns, manual verification by specialized penetration testers is essential.

They evaluate the results in the context of the specific application architecture, identify logical vulnerabilities in the orchestration layer, and simulate targeted attacks that no scanner can anticipate.



Traditional vulnerability scans do not detect prompt injections, insecure agent logic, or manipulated RAG pipelines. These vulnerabilities require targeted, manual attack simulations by specialized penetration testers.

Conclusion & Recommendations

The threat landscape for AI applications has fundamentally changed with the emergence of agentic AI systems. The key findings of this white paper can be summarized as follows:

- **Agentic AI systems create an entirely new category of security risks that go beyond the numerous known LLM vulnerabilities.** Tool misuse, agent goal hijacking, memory poisoning, and agent communication poisoning are real threats with potentially severe consequences. Therefore, AI systems should be viewed as employees who are particularly vulnerable to social engineering attacks.
- **Prompt injections remain the most dangerous attack vector and are massively amplified by agentic systems.** Most published defense mechanisms do not provide a sufficient level of protection.
- **The MCP protocol, as a new standard for tool integrations, introduces specific security risks that must be explicitly addressed.**
- **Conventional security tests are insufficient to uncover the novel vulnerabilities of agentic AI systems.** This requires specialized AI penetration tests.



Specific Recommendations for Action

Conducting an inventory: Identify all AI applications in your organization, along with their tool access, permissions, and data flows. Prioritize identifying agentic systems with a high degree of autonomy.

Implementing the least privilege principle: Review the access rights of all AI agents and reduce them to the absolute minimum required. Implement granular and dynamic authorization concepts that adapt to the respective user and their rights. Statically assigned privileges carry the risk of granting too much access in the wrong context. If an agent can access data that users should not have access to, this constitutes a design flaw

Establishing human-in-the-loop: Ensure that all security-critical actions require human confirmation before they are executed.

Validating, cleansing, and limiting all inputs and outputs: Check all inputs and outputs for compliance with defined schemas, sanitize data before passing it to downstream systems, and enforce realistic token limits

Commissioning specialized security testing: Have your AI applications tested by a specialized AI penetration test that covers novel attack vectors

Protect Your AI Applications

Minimize risks in your AI applications before attackers exploit them. Oneconsult offers specialized AI penetration tests that identify security vulnerabilities in your AI applications before attackers can actively exploit them.

AI Agent Security

As part of a gray-box testing approach, we analyze your AI applications, including architectural documentation, prompt templates, tool specifications, and configuration interfaces.

Your Advantages:

- You receive a comprehensive security assessment that covers the graphical user interface as well as REST API endpoints and administrative configuration panels.
- You receive a comprehensive report detailing prioritized vulnerabilities and specific recommendations for action.

› Learn more now

Questions about our services or interested in a quote?

Contact us for a personalized cybersecurity consultation.

Thomas Tunkel

Principal Client Services Manager

thomas.tunkel@oneconsult.com

linkedin.com/thomas-tunkel



Technical questions about AI security?

The author of this white paper is available to assist you.

Dr. Marco Bertenghi

Senior Penetration Tester

marco.bertenghi@oneconsult.com

linkedin.com/bertenghi



Glossary

Agent communication poisoning	Manipulating communication between agents in multi-agent systems to distort subsequent instructions or results
Agentic system	AI system with planning capabilities, tool access, and autonomous decision-making
AI	Artificial intelligence – systems that generate new (synthetic) content (text, code, images, audio, video)
Context leakage	Unintended disclosure of sensitive information from an agent's context to external systems or downstream agents
Context rot	Gradual deterioration in an agent's decision-making quality due to the accumulation of outdated, contradictory, or manipulated information in the context
Defense in depth	Multi-layered security approach with overlapping protection mechanisms
DoW	Denial of wallet – cost-based attack in which high inference costs are deliberately incurred through massive or excessively long requests
Goal hijacking	Manipulation of an agent to pursue a goal other than the one intended by the user
HITL	Human-in-the-loop – human monitoring and verification of critical agent actions
IDOR	Insecure direct object reference – unauthorized access to resources through the manipulation of references
Jailbreak	Attempt to completely bypass the security mechanisms of an AI system
Least Privilege	Security principle in which each component is granted only the minimum necessary access rights
LLM	Large language model – understands and generates natural language
MCP	Model context protocol – standard protocol for tool integration in AI systems
Memory Poisoning	Injecting malicious data into an agent's persistent storage
OWASP	Open Worldwide Application Security Project – non-profit organization for application security

Plan manipulation	Attack in which an agent's planning logic is manipulated to alter the execution order or target priority
Privilege escalation	Gaining higher access privileges than originally intended – in the case of agents, often by exploiting transitive permissions in the call chain; in the context of agents, this is also referred to as "privilege creep"
Prompt injection	Attack in which inputs are manipulated to alter the behavior of an AI system
RAG	Retrieval-augmented generation – extending an LLM with external data sources
RCE	Remote code execution – remote execution of arbitrary malicious code on a target system
Rogue agent	Agentic AI component that behaves outside its intended parameters – due to compromise, misconfiguration, or emergent behavior
Rug Pull	Attack in which an MCP tool or external service that initially appears legitimate is subsequently replaced with malicious functionality
Sandboxing	Executing code or processes in an isolated environment to mitigate damage
Social engineering	Manipulating individuals (or agents) through psychological deception to gain unauthorized access or information
SQL injection	Injecting malicious SQL commands into database queries to manipulate or extract data
SSRF	Server-side request forgery – attack that causes a server or agent to send requests to internal or unauthorized resources
Tool misuse	Misuse of legitimate tools by an agent for unintended or harmful purposes
Tool poisoning	Attack in which manipulated tool descriptions contain hidden instructions for an AI agent
XSS	Cross-site scripting – attack to inject malicious scripts into web applications

Bibliography

1. **OWASP Foundation.** (2025). OWASP Top 10 for LLM Applications 2025. https://genai.owasp.org/llm-top-10/

2. **Mairie, J.** (2026, 28. Januar). The Clawdbot Dumpster Fire: 72 Hours that Exposed Everything Wrong with AI Security. Acuvity. https://acuvity.ai/the-clawdbot-dumpster-fire-72-hours-that-exposed-everything-wrong-with-ai-security/

3. **OWASP Foundation.** (2026). OWASP Top 10 for Large Language Model Applications (v2.0). https://owasp.org/www-project-top-10-for-large-language-model-applications/

4. **OWASP Foundation.** (2026). OWASP Top 10 for Agentic Applications for 2026. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/

5. **OWASP Foundation.** (2025). OWASP MCP Top 10. https://owasp.org/www-project-mcp-top-10/

6. **Nasr, M., et al.** (2025). The Attacker Moves Second: Stronger Adaptive Attacks Bypass Defenses Against LLM Jailbreaks and Prompt Injections. arXiv:2510.09023. https://arxiv.org/abs/2510.09023

7. **Herrador, M., Rehberger J.** (2026). SpAIware: Uncovering a novel artificial intelligence attack vector through persistent memory in LLM applications and agents. Future Generation Computer Systems. https://doi.org/10.1016/j.future.2025.107994

8. **Yang, X., et al.** (2026). Zombie Agents: Persistent Control of Self-Evolving LLM Agents via Self-Reinforcing Injections. arXiv:2602.15654. https://www.arxiv.org/abs/2602.15654

9. **Lupinacci, M., et al.** (2025). The Dark Side of LLMs: Agent-based Attacks for Complete Computer Takeover. arXiv:2507.06850. https://arxiv.org/abs/2507.06850

10. **Anthropic / MCP Community.** (2026). Model Context Protocol (MCP). Abgerufen am [Datum einfügen], von https://modelcontextprotocol.io

11. **Zou, W., et al.** (2025). PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. USENIX Security '25. https://www.usenix.org/conference/usenixsecurity25/presentation/zou-poisonedrag

12. **Anthropic.** (2025). Poisoning Attacks on LLMs Require a Near-constant Number of Poison Samples. arXiv:2510.07192. https://arxiv.org/abs/2510.07192

13. **Debenedetti, E., et al.** (2025). Defeating Prompt Injections by Design. arXiv:2503.18813. https://arxiv.org/pdf/2503.18813

14. **Willison, S.** (2023, 25. April). The Dual LLM Pattern for Building AI Assistants That Can Resist Prompt Injection. Simon Willison's Weblog. https://simonwillison.net/2023/Apr/25/dual-llm-pattern/

15. **Anthropic.** (2025). Constitutional Classifiers: Defending against Universal Jailbreaks across Thousands of Hours of Red Teaming. arXiv:2501.18837. https://arxiv.org/abs/2501.18837

Additional Ressources

- OWASP LLM Top 10 for Large Language Model Applications: https://owasp.org/www-project-top-10-for-large-language-model-applications/
- OWASP Top 10 for Agentic Applications: https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/
- OWASP MCP Top 10: https://owasp.org/www-project-mcp-top-10/
- MITRE ATLAS AI Security 101: https://atlas.mitre.org/resources/ai-security-101
- MITRA ATLAS Mitigations: https://atlas.mitre.org/mitigations