

GUIDE · TOMORIS TOOLBOX

Using AI Agents Securely: Permissions, Approvals, MCP Servers and Logging

Risks according to OWASP, principles for permissions and approvals, vetting and isolating MCP servers, a three-stage rollout and a checklist before production

UPDATED

Oct 25, 2026

LENGTH

PDF, 8 pages

ONLINE

tomoris.com/en/downloads/ki-agenten-sicher-einsetzen

AT A GLANCE

3

causes of excessive agency according to OWASP LLM06

ASI01–10

OWASP Top 10 for Agentic Applications 2026

MUST NOT

token passthrough in the MCP specification

3

rollout stages, plus ongoing review

WHO IT IS FOR

- IT management and IT security who approve and operate AI agents
- Development teams that build or integrate agents, tools or MCP servers
- Business units planning agents with access to email, files or business applications
- Management deciding on use and limits

WHAT THIS COVERS

AI agents plan several steps on their own and call tools along the way: they read files, write emails, change code or query databases. That turns a wrong or manipulated answer into a wrong action. This guide classifies the key risks according to OWASP and shows how to limit permissions, require approval for consequential actions, treat third-party content as untrusted, vet MCP servers and run agents in isolation and traceably. It ends with a checklist for the move into production.

This guide is a TOMORIS recommendation based on the sources listed. Product features are described according to vendor documentation as of October 25, 2026, and may change. It does not replace a security test or legal advice.

CONTENTS

01	What makes agents riskier than chat	2
02	Six principles for permissions and approvals	3
03	Identities and credentials	4
04	Vetting and operating MCP servers	4
05	Example: how a tool implements the principles	5
06	Rollout in three stages	6
07	Checklist before production	6
08	Sources	7

01 —

What makes agents riskier than chat

Agents act: the key risks according to the OWASP Top 10 for LLM Applications 2025 and the Top 10 for Agentic Applications 2026.

A chat assistant answers; an agent acts. It receives tools, such as file access, email, a shell or APIs, and decides on its own when to call them. OWASP calls the consequence “Excessive Agency” (LLM06:2025): harmful actions in response to unexpected, ambiguous or manipulated model output, regardless of why the model misbehaves. The causes are excessive functionality, excessive permissions and excessive autonomy. The OWASP Top 10 for Agentic Applications 2026 (published on December 9, 2025) extends this with the principle of Least Agency: use agentic behavior where it is needed, because otherwise it enlarges the attack surface without benefit.

Prompt Injection (LLM01)

Inputs alter the model’s behavior in unintended ways. Indirectly, this happens through third-party content such as websites, files or emails; the instructions do not have to be visible to humans. OWASP considers it unclear whether it can be reliably prevented and recommends limiting the impact.

Agent Goal Hijack (ASI01)	Attackers alter the agent's goals, task selection or decision paths, for example through prompts, deceptive tool outputs, manipulated files or poisoned external data. Agents cannot reliably distinguish instructions from content.
Tool Misuse (ASI02)	The agent misuses legitimate tools within its plans and delegation chains, for example by doing more than intended with existing permissions. OWASP builds on the LLM06 mitigations here.
Identity & Privilege Abuse (ASI03)	Delegation chains, inherited roles or cached credentials are exploited to escalate privileges. According to OWASP, least privilege cannot truly be enforced without a dedicated, managed identity for the agent.
Agentic Supply Chain (ASI04)	Third-party agents, tools, models, MCP servers or registries are malicious, compromised or tampered with in transit. Agents often load such capabilities only at runtime.

02 —

Six principles for permissions and approvals

What OWASP recommends against excessive agency and prompt injection, as rules for your agents.

1

Limit tools to what is necessary

Offer only the extensions the agent needs for its task, and within them only the necessary functions: an agent that summarizes emails needs to read, not send or delete. Avoid open-ended tools such as arbitrary shell commands or URL fetches (OWASP LLM06, mitigations 1 to 3).

2

Minimize permissions

Give each tool only the permissions it needs, for example read access to one table instead of the whole database, and run it in the context of the respective user (LLM06, mitigations 4 and 5).

3

Require approval for consequential actions

A human confirms high-impact actions before they are executed: sending, deleting, paying, changing permissions, publishing (LLM06, mitigation 6; LLM01, mitigation 5).

4

Do not leave authorization to the model

The target system decides according to its own rules whether an action is allowed, not the language model. Every request made through a tool is checked there against the security policy (LLM06, mitigation 7: complete mediation).

5

Treat third-party content as untrusted

Websites, documents, emails and tool descriptions from MCP servers can contain instructions. They are processed separately and labeled, and must not trigger actions without review (LLM01, mitigation 6; MCP specification).

6

Log and limit

Log and monitor tool calls and actions in target systems, and limit rates and budgets so that misbehavior is noticed before major damage occurs (LLM06; OWASP ASI02, mitigation 5).

An example from OWASP LLM06: an assistant that summarizes emails is built with an extension that can only read, signs in with an OAuth grant limited to read-only scope, and every email it drafts is sent only by the human after review.

03

Identities and credentials

Dedicated, narrowly scoped and short-lived access instead of shared administrator accounts.

- A dedicated identity per agent and task instead of an administrator's or developer's personal account; this keeps it traceable who acted (OWASP ASI03).
- Short-lived, narrowly scoped tokens per task instead of permanent keys; cap permissions with boundaries (ASI03, mitigation 1; ASI02: just-in-time and short-lived access).
- Credentials separated and narrowly scoped per MCP server, nothing inherited from your own shell, nothing reused (MCP Local Server Security, checklist).
- MCP servers must not accept tokens that were not explicitly issued for them and must not pass tokens on to downstream APIs unchecked (MCP Security Best Practices: Token Passthrough, "MUST NOT").
- Secrets belong in a vault or the system's credential manager, not in prompts, chat histories or configuration files in the repository.
- It must be possible to revoke and renew an agent's access centrally if it behaves suspiciously (ASI04 and ASI10: kill switch and credential revocation).

04

Vetting and operating MCP servers

The Model Context Protocol connects agents to tools; every server is code with access. Here is how to vet one.

MCP servers provide an agent with tools, data and prompts. The specification (version 2026-07-28) states: tools represent arbitrary code execution and must be treated with appropriate caution; descriptions of tool behavior are considered untrusted unless they come from a trusted server; hosts must obtain explicit user consent before invoking any tool. The protocol cannot enforce these principles itself; applications must do so.

- ☐ Does the server need local files, tools or the local network? If not, use the remote version over Streamable HTTP with authorization instead of a local process
- ☐ Publisher identified, source code available for review, exact package name checked (typosquatting packages), version pinned

- ☐ With launchers such as npx or uvx, keep in mind that the version specifier pins only the top-level package; a container image with a digest freezes all dependencies
- ☐ Tool descriptions read before use; servers with foreign instructions in descriptions removed; client asks again when tools change
- ☐ Third-party servers isolated in a container, sandbox or virtual machine, without unnecessary volume mounts
- ☐ File access limited to specific directories, read-only where possible
- ☐ Outbound network traffic blocked or limited to a known list of destinations
- ☐ Only as many servers as necessary, because all configured servers share the same model context
- ☐ In the organization: list of approved servers with criteria, ongoing inventory of client configurations, logs of MCP activity, servers can be blocked centrally

The checkpoints follow the “Local Server Security” guide in the MCP documentation (version 2026-07-28). It calls process isolation the most effective single measure, because it protects even when the provenance check fails.

05

Example: how a tool implements the principles

Claude Code features according to Anthropic’s documentation, as an example for vetting any agent tool.

Principle and implementation in Claude Code according to Anthropic’s security documentation (as of October 25, 2026)

PRINCIPLE	IMPLEMENTATION PER DOCUMENTATION
Approvals	Permission modes define what happens without asking; in manual mode, Claude Code asks before file changes, tests and commands, and commands without a matching rule require approval
Isolation	Sandbox for shell commands with file system and network isolation; for scripts and tool calls involving external services, Anthropic recommends virtual machines
Third-party content	Commands that fetch content from the web are not approved automatically by default; Anthropic advises against feeding untrusted content in directly
MCP servers	Servers from the project configuration require separate consent; Anthropic recommends your own servers or those from trusted providers
Policies and monitoring	Managed settings enforce organizational policies, permissions can be reviewed regularly, and usage can be monitored via OpenTelemetry

The example describes features according to the vendor documentation; it is not an assessment or endorsement of a product. Anthropic itself points out that no system is immune to all attacks and that suggestions should be reviewed before approval.

06 —

Rollout in three stages

Expand autonomy step by step and review continuously, each stage with logs and clear accountability; a TOMORIS recommendation.



Regardless of the stage, Art. 4 of the AI Act as amended by Regulation (EU) 2026/1744 requires providers and deployers to take measures to support the AI literacy of the people working with AI systems on their behalf. Anyone who approves or monitors agents must know their limits.

07 —

Checklist before production

To tick off before an agent works with real data and systems.

☐ The agent's task, limits and responsible person defined in writing
☐ Agentic behavior is necessary for the task; simpler, fixed workflows were considered
☐ Only necessary tools and functions enabled, no open shell or fetch tools without good reason
☐ Dedicated identity with least privilege, short-lived access, no shared administrator accounts
☐ Consequential actions require human approval; the list of these actions is defined
☐ Target systems check every request against their own permissions

- ☐ Third-party content is treated as untrusted; tests with injected instructions carried out
- ☐ MCP servers vetted for provenance, version pinned, isolated, with their own narrowly scoped access
- ☐ Network access of the agent and its tools limited to known destinations
- ☐ All tool calls and actions logged, analysis and retention defined
- ☐ Budgets and rate limits set, kill switch and access revocation tested
- ☐ Data protection reviewed: legal basis, contract with the provider, data protection impact assessment if required
- ☐ Users trained; reporting path for suspicious behavior known

This checklist is a TOMORIS recommendation based on the OWASP and MCP publications listed; it does not replace a security test of the specific agent.

NEXT STEP

Check It Yourself or Put It into Practice Together

DISCUSS IMPLEMENTATION

Get in Touch

tomoris.com/en/contact

contact@tomoris.com

08

Sources

All statements are supported by sources. Legislation applies in its current version.

- 1 **OWASP Top 10 for LLM Applications 2025: LLM06 Excessive Agency**
Causes, mitigations 1 to 7, limiting through logs and rate limits, email assistant example
genai.owasp.org/llmrisk/llm062025-excessive-agency
- 2 **OWASP Top 10 for LLM Applications 2025: LLM01 Prompt Injection**
direct and indirect prompt injection, mitigations 4 to 6
genai.owasp.org/llmrisk/llm01-prompt-injection
- 3 **OWASP Top 10 for Agentic Applications for 2026 (December 9, 2025)**
Least Agency, ASI01 to ASI04 and ASI10 with mitigations
genai.owasp.org/resource/owasp-top-10-for-agentic-applications-...
- 4 **Model Context Protocol, specification 2026-07-28**
Section Security and Trust & Safety
modelcontextprotocol.io/specification/2026-07-28
- 5 **Model Context Protocol: Security Best Practices (2026-07-28)**
Token Passthrough, Local MCP Server Compromise
modelcontextprotocol.io/docs/2026-07-28/tutorials/security/secu...
- 6 **Model Context Protocol: Local Server Security (2026-07-28)**
Provenance, tool descriptions, isolation, credentials, network, checklist
modelcontextprotocol.io/docs/2026-07-28/tutorials/security/loca...

- 7 **Anthropic: Claude Code Security**
Permissions, sandbox, prompt injection, MCP, team security
code.claude.com/docs/en/security
- 8 **Regulation (EU) 2026/1744 (Digital Omnibus Regulation on AI)**
new Art. 4 of Regulation (EU) 2024/1689, in force since July 27, 2026
eur-lex.europa.eu/eli/reg/2026/1744/oj/deu