TRAFICOM
Finnish Transport and Communications Agency

Huoltovarmuuskeskus
Försörjningsberedskapscentralen
National Emergency Supply Agency

Cybersecurity of AI agents

Guidance for the secure design and implementation of AI agents

Traficom
National Emergency Supply Agency

Traficom publications
5/2026

Contents

1 Introduction: AI agents and security

The use of artificial intelligence has grown rapidly across all sectors. The widespread adoption of general-purpose large language models has also brought various AI agent solutions to the market that can perform tasks autonomously. AI agents assist users in, among other things, finding and structuring information, quickly reviewing large volumes of text such as instructions and reports, automating routine tasks and chaining together different actions that may not always require human intervention. AI agents are also capable of performing technical tasks, such as programming and creating and executing various commands.

While the use of AI agents can be highly straightforward, they also introduce new security challenges. For example, freely available online solutions, various ready-made solution accelerators and the relative ease of developing customised agents all contribute to accelerating their adoption. As a result, AI agents and their various implementations may easily find their way into organisational use even when there is no certainty regarding their technical security or secure use.

Many organisations have sought to simplify agent management by developing their own agents that take security requirements into account more effectively than off-the-shelf solutions. Customised solutions are particularly common in sectors where information security requirements are higher than usual and where realised risks may have more severe consequences. Internally developed solutions often make use of existing language models but employ them in a manner that considers local legislation and information security best practices arising from the specific operating environment.

Although AI agents can improve organisational efficiency, their use is not always the only or the best solution. Analyses involving continuous variables, such as forecasting tasks, are often best addressed using methods specifically designed for the purpose, such as regression algorithms. Similarly, classification and clustering tasks, object recognition in images and optical character recognition are often most effectively implemented using machine learning methods developed for these purposes. These more traditional machine learning implementations can also form part of an agent system. When selecting an implementation approach, organisations should consider whether the problem is sufficiently complex or demanding that it cannot be solved using a single method, and whether it is therefore worthwhile to develop an AI agent instead of, or alongside, another type of machine learning system. This guidance assumes that different implementation alternatives have already been considered and that, on that basis, an AI agent has been selected as the most suitable solution for the task.

The objective of this guidance is to help organisations design, build and maintain agentic AI systems securely. At the time of writing, the report is based on the most up-to-date information available on the most common information security risks associated with AI agents and on threat modeling best practices. These provide a starting point for organisations wishing to design and build AI agents without exposing information security to unreasonable risk.

This guidance is intended particularly for individuals working with artificial intelligence, information security and cybersecurity. Chapters 2 and 3 provide a general introduction to what AI agents are, how they function and what types of threats are associated with them. From Chapter 4 onwards, the focus shifts to the design of AI agents and we provide guidance on how information security threats related to agents can be modelled as part of risk assessment. Finally, the report moves on to its most technical section, examining the security considerations that should be taken into account during system development, how different technical choices affect security and how the security of AI agent systems can be tested.

This guidance has been produced as a collaboration between Traficom and the National Emergency Supply Agency as part of the Digital Security 2030 programme and broader future preparedness efforts. The report was written in cooperation with Solita. Several companies and other stakeholders also contributed to the preparation of the report.

2 AI agents and their special characteristics

In this guide, an AI agent means an information system that uses AI models and various tools to perform actions automatically on behalf of a user. Agents can therefore act and initiate actions autonomously on the basis of defined rules and operating instructions.

AI agents should therefore not be confused with AI assistants. Assistants are typically conversational applications that require continuous guidance from the user through prompts. By contrast, an agent is able to chain actions in the environment defined for it by automatically responding to the outputs of previous tasks. Agents can therefore be used more effectively in complex tasks where efficient and secure performance requires the automatic chaining of several actions.

Despite its name, an AI agent is therefore not intelligent. It is an information system that has no actual understanding of the intentions of its user or developer, the correctness of its actions or their consequences. Although the background information provided to an AI agent is called context, it does not give the AI the ability to infer matters beyond the information provided to it, such as training material, background material and online sources accessible to the agent. The operation of AI therefore depends entirely on the data used to train it, its algorithm, the background material provided to it and the operating instructions given by the user.

In this chapter, we describe the basic structure and special characteristics of agents.

2.1 Basic structure of agents

AI agents consist of three main components: foundation models, prompts and the tools connected to the agent. When an organisation develops or deploys an AI agent, it must ensure the security of all these components. This guide explains in more detail how the different components can be designed, built and tested securely. The components are described in more detail below.

2.1.1 Foundation model

The operation of agents is always based on an underlying AI model, or foundation model. Generative AI models are deep learning models, in practice large, deep neural networks. These include, among others, language models and image models designed to process and imitate human-generated linguistic material, such as text, images, audio and video. Models that use several of these simultaneously are multimodal.

Current AI agents are usually built using **large language models** (LLM), which have been trained to process natural language even when the data is unstructured. Large language models are based on probabilities concerning which words typically occur together in the text corpora used to train them. If an agent implemented using a language model also generates text, its outputs are based on probabilities concerning what the user wants the system to produce based on the input and which words should be used to express the output so that the user perceives it as correct.

In addition to large language models, AI agents can also use small language models, which can be run on devices available to consumers. Small language models can, for example, be fine-tuned for a more specific task, in which case their accuracy is sufficiently good for agent systems[1].

2.1.2 Prompts

A prompt means an instruction given to the system in natural language to perform a desired task. It may be a simple question or a set of detailed instructions that the model must follow. A prompt also defines the tools the agent may use. A **system prompt** defines how the agent should behave in different situations. The system prompt is defined before the end user starts using the agent and is usually not visible to the user. The input given by the user to the AI model is also called a prompt.

2.1.3 Tools

Tools are functions or interfaces provided to an AI agent that it can call when needed. They enable a language model to perform tasks that cannot be carried out by producing text alone.

Tools can be divided into three categories:

1. **Data:** Tools that allow the agent to retrieve the material needed to formulate a response.
2. **Action:** The agent is able to use tools to perform actions in different systems, such as writing to a database.
3. **Orchestration:** The agent can act as an assistant to other agents or call other agents in the same way as tools.

Examples of tools that an agent can use to improve the quality of a response or solve the task defined in the input:

[1] Belcak et al. (2025). Small language models are the future of agentic AI, https://arxiv.org/pdf/2506.02153

- **Web search:** Retrieving information from the internet.
- **Calculator:** Language models often fail even in the simplest calculations. A calculator can be used to reduce the number of calculation errors.
- **Reading files:** Read and write access to files makes it possible to edit different documents using language models.
- **Reading a database:** An agent can be granted access to a database. A language model can be used, for example, to retrieve the previous month's sales figures from the database.
- **Optical character recognition, OCR:** Recognising text from images and documents.
- **Generating images, tables and charts:** The generation of images, tables and charts enables the agent to be used, for example, in data analysis.

In addition to tools, an agent can call other agents. In this case, an individual agent can be configured to solve more narrowly defined problems with a smaller number of tools. At the same time, some tasks can be performed in parallel. These and other types of agents are described in more detail in the next subsection.

2.2 Types of agent systems

In this guide, we divide AI agents into three categories based on the complexity of their implementation. Complexity is affected, for example, by the agent's autonomy and its connection to other systems and other agents. There are several different agent types and ways of categorising them, and examples of such lists can be found on the IBM[2] and Red Hat[3] websites. The most typical agents are:

2.2.1 Simple agent

The simplest AI agents call tools on the basis of clear rules, such as a system prompt. For example, an agent may process a received file and add the required metadata to it in accordance with instructions. A slightly more complex agent recognises the state of the environment and responds to the effects of tool calls, such as an agent that processes a customer contact and generates a response for the customer. If the objective is not

[2] IBM's classification of AI agent types: https://www.ibm.com/think/topics/ai-agent-types
[3] Red Hat's classification of AI agent types: https://www.redhat.com/en/blog/understanding-ai-agent-types-simple-complex

achieved, the agent carries out new actions, such as modifying the content of its response on the basis of the user's instructions.

2.2.2 Learning agent

A learning AI agent operates in the environment defined for it and performs various actions within that environment. The user or another agent evaluates the consequences of these actions and stores the feedback for the agent's later use. When the agent retains feedback from previous tasks, it may be able to avoid earlier mistakes and solve new, similar tasks more quickly and effectively.

A learning agent is particularly useful in complex use cases where the agent's environment changes continuously. Such use cases may include, for example, an agent that monitors the logs of a production facility and creates a ticket about an error situation, or an agent that recommends actions in exceptional situations.

2.2.3 Multi-agent system

A multi-agent system consists of several collaborating AI agents whose objective is to solve demanding tasks. In this method, several smaller agents are built to solve smaller tasks, with only the necessary tools defined for each of them. Using multiple agents can compensate for possible shortcomings of an individual agent or language model, such as understanding and performing tasks that are complex or require diverse reasoning. Reducing the number of tools improves the accuracy of agents when calling tools and limiting the size of tasks can improve the AI agent's ability to perform the assigned task. In addition, dividing the task into parts may enable parallel computation and thereby speed up task resolution.

The design, implementation, testing and monitoring of secure operation of a multi-agent system is often more complex than for a single-agent system. When designing a multi-agent system, it is necessary to consider how collaboration between agents is coordinated, how the roles of different agents are defined, how agents share instructions and outputs with each other and what role human instructions play in collaboration between agents.

A multi-agent system can be implemented, for example, using the following techniques:

1. **Centralised system:** One agent is responsible for calling other agents and decides when the task defined by the user has been completed.
2. **Decentralised system:** Each agent operates independently and shares information with other agents.

3. **Hierarchical system:** Agents higher in the hierarchy define tasks and monitor agents lower in the hierarchy.

For collaboration and communication between agents, it is advisable to define a set of rules for how agents share information, how and in what situations they request assistance from others and how problems or uncertainties are communicated. It is also advisable to seek to monitor the outputs of agents and their quality, either by an agent specifically defined for this purpose, by agents that centrally or hierarchically guide the task, or by a human.

A multi-agent system usually also has common, shared requirements, regulations or instructions. These can be defined, for example, through a shared system prompt. Shared requirements or regulations include, for example, the values of the organisation using the system, ethical principles, compliance with laws and regulations, and information security and privacy requirements.

2.3 Agent autonomy

An AI agent can use tools either with the user's permission or fully automatically. When considering the level of agent autonomy, the damage caused by an error in the task and the difficulty of the task must be considered. For example, an agent that can use the command prompt without restrictions may overwrite or delete files. The appropriate and secure level of autonomy for the task must always be assessed on a case-by-case basis.

To prevent the misuse of AI agents, an agent operating autonomously in particular must be monitored and limits must be set for it. For example, if finding the desired solution requires too many intermediate steps, it must be verified whether the development of the solution should be continued.

If an AI agent is a learning agent and its level of autonomy is high, particular attention must be paid to monitoring. The more the agent's operation is based on earlier inputs from the user or another agent, the more closely their anticipated and possible unforeseen effects on the agent's operation must be monitored (see section 7.3.6).

3 Key information security vulnerabilities in language model-based agents

AI agents have autonomy to use various tools and databases through different interfaces in order to achieve the objectives set for them. AI agents and AI assistants are often subject to the same vulnerabilities, but AI agents are typically higher-risk use cases due to their tasks and autonomy. In multi-agent systems, even a single agent manipulated by prompt injection can contaminate the operation of the entire system.

In this guidance, we rely on the lists produced by OWASP, which identify the ten key risks associated with large language models and AI agents and measures for managing them during the application life cycle. This guidance helps identify common risks and design systems that use AI agents in such a way that the risks are managed appropriately.

We will return to these risks in different parts of this guide from the perspective of the different stages of the AI agent life cycle and provide examples of the risks they entail. The original and updated descriptions are available on the OWASP website. In December 2025, OWASP published a separate list of risks specifically concerning agent systems. The precise original descriptions of the risks discussed below, their prevention and management measures can be found via the links mentioned above.

3.1 Typical threats to AI agents

In December 2025, OWASP published a list of the ten most significant risks affecting AI agents and measures for managing them[4], which we will briefly discuss below.

ASI01: Agent goal hijack

Due to weaknesses related to instructions written in natural language and the processing of such instructions, agents and the underlying language models cannot reliably distinguish between instructions given to them and content provided by an attacker. Attackers can therefore manipulate the agent's objectives, the choices it makes and its decision pathways through prompt injection.

ASI02: Tool misuse and exploitation

An agent may use tools permitted for it in a harmful manner, causing data leaks, output manipulation and workflow disruptions. The agent's memory, dynamic tool selection and task delegation may enable misuse through chaining, privilege escalation and unintended actions.

ASI03: Identity and privilege abuse

An attacker can exploit agents' trust and task delegation to escalate privileges and bypass control mechanisms by manipulating task delegation chains, role inheritance and the agent's context. Context includes, for example, cached credentials or conversation history between interconnected systems.

ASI04: Agentic supply chain vulnerabilities

Third-party components, such as models, tools, datasets and agentic interfaces, may be malicious or manipulated and introduce unsafe code and hidden instructions into the system.

ASI05: Unexpected code execution (RCE)

Agent-based systems, including vibe coding tools, often generate and execute code. Attackers can exploit code-generation features or tool access embedded in the agent to escalate actions into remote code execution (RCE).

[4] OWASP TOP 10 For Agentic Applications 2026

ASI06: Memory and context poisoning

Attackers can poison the agent's memory or context, such as RAG stores, which biases reasoning and may lead to data leaks or dangerous actions.

ASI07: Insecure inter-agent communication

Weak authentication and encryption in communication between agents enables messages to be intercepted, manipulated and spoofed.

ASI08: Cascading failures

A single fault, such as a hallucination or malicious input, can spread between agents and grow into a broad risk that threatens the entire system.

ASI09: Human-agent trust exploitation

Attackers can exploit excessive human trust in an agent's recommendations, which may lead to harmful decisions, financial losses and data leaks.

ASI10: Rogue agents

Agents may deviate from their intended operation and sabotage systems even when their actions appear correct from the outside. An agent becoming rogue may result, for example, from prompt injection or supply chain vulnerabilities and cause sensitive information disclosure, misinformation propagation and operational problems.

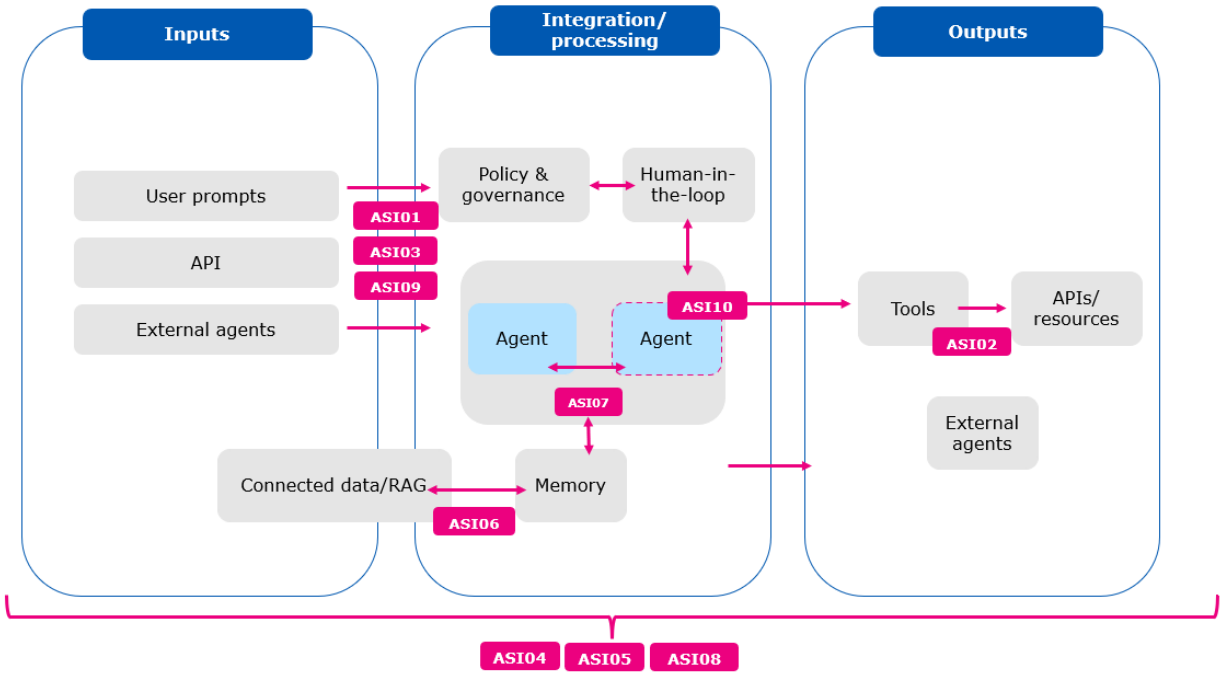

Figure 1. Threats to AI agents. Source: OWASP: Top 10 for Agentic Applications for 2026.

3.2 Typical threats to AI systems

Next, we will examine the ten key risks affecting AI systems listed by OWASP[5] and their typical manifestations in AI agents.

3.2.1 LLM01:2025 Prompt injection

Prompt manipulation, or prompt injection, is a vulnerability in large language models in which a user can use malicious prompts to alter the model's behaviour or its outputs. Malicious prompts are designed in such a way that their content may be imperceptible to humans, even though they can be read by the AI model. The vulnerability occurs when the model processes a prompt provided by the user that contains malicious content.

An example of this vulnerability is EchoLeak (CVE-2025-32711), a sensitive information disclosure vulnerability in M365 Copilot. Attackers used injection, meaning malicious commands, in email messages to steal sensitive information.[6]

The objective of an attack targeting an AI agent may be to alter the agent's operation in the long term, which makes the threat more difficult to detect. Due to weaknesses related to instructions written in natural language and the processing of such instructions, agents and the underlying language models cannot reliably distinguish between instructions given to them and content provided by an attacker. Attackers can therefore manipulate the agent's objectives (ASI01), the choices it makes and its decision pathways through prompt injection. For example, an attacker may use prompt injection to attempt to manipulate an agent that manages financial transactions and seek to direct funds to the attacker's own account.

Protection against prompt injection can include measures such as constraining model behaviour, implementing guardrails for prompts and outputs, restricting the agent's privileges and requiring human approval for the most critical decisions.

3.2.2 LLM02:2025 Sensitive information disclosure

Sensitive information disclosure refers to a vulnerability in which large language models may disclose sensitive or confidential information through their outputs. This vulnerability may occur in AI agents if they have not been designed and protected appropriately.

Sensitive information may become available to the model through training data or through user inputs, either knowingly or unintentionally. It may also arise through the business logic that affects the model's operation, for

[5] OWASP top 10 for LLM apps & Gen AI agentic Security initiate, v1.
[6] A description of the case is available at https://arxiv.org/html/2509.10540v1.

example when an agent has access to databases containing sensitive information. This vulnerability may lead to unauthorised access to information, violations of privacy and data leaks.

In agent systems in particular, sensitive information disclosure may result from:

(a) identity and privilege abuse (ASI03)

(b) insecure inter-agent communication (ASI07) or

(c) rogue agents (ASI10).

Identity and privilege abuse can be used to bypass controls and escalate privileges. The agent's identity and authentication, such as API keys and OAuth, are critical, and without clear governance measures, the principle of least privilege cannot be implemented. In this context, identity refers both to the persona defined for the agent and to any authentication material that represents the agent. Agent-to-agent trust or inherited credentials may enable privilege escalation, privilege hijacking or the execution of unauthorised actions.

Weak authentication and validation in inter-agent communication enables messages to be intercepted, manipulated and spoofed, which may lead to sensitive information disclosure.

Rogue agents are malicious or compromised agents that deviate from their originally intended tasks and act harmfully or deceptively (ASI10). They may sabotage systems, causing data leaks and workflow hijacking. This threat can be managed in particular by monitoring the operation of the AI agent and minimising the data available to it.

3.2.3 LLM03:2025 Supply chain vulnerability

The production chains of applications that use large language models are exposed to vulnerabilities that may affect the integrity of training data, models and platforms. Agents are therefore often dependent on third-party components, such as AI models, training data, the infrastructure used for training and labour. Through the supply chain, malicious actors may, for example, introduce backdoors and vulnerabilities into systems at an early stage.

Vulnerabilities may lead to biased AI agent outputs, security breaches and system failures. For example, training data obtained from a third party may contain prejudiced or biased information, malicious code or other malicious content that affects the operation of the model or the outputs of the system.

The increasing use of open-access language models and new fine-tuning methods, such as LoRA (Low-Rank Adaptation) and PEFT (Parameter-Efficient Fine-Tuning), particularly on platforms such as Hugging Face, introduces new supply chain risks, as they are openly available to everyone without a full guarantee of security. The growing use of on-device language models also increases the attack surface and supply chain risks of applications.

In agent systems, supply chains do not consist only of static dependencies. Unlike traditional AI or software supply chains, agent systems use, for example, external tools and access rights dynamically during task execution. Supply chain risks can be mitigated, for example, by applying a zero-trust model in the agent system.

3.2.4 LLM04:2025 Data and model poisoning

Data poisoning refers to a vulnerability in which the data used by an AI agent is manipulated to introduce vulnerabilities, malicious code or, for example, biases. Poisoning compromises the model's performance, security or ethical behaviour. The purpose of data poisoning is to affect the integrity of the model, in practice its ability to produce the outcomes desired by the agent owner.

Poisoning can target different stages of the AI system life cycle, such as training, where the model learns from general data, or fine-tuning, where the model is adapted to specific tasks. Poisoning can also occur when text data is converted into numerical vectors that the AI agent uses to generate its output.

Poisoning may, for example, enable a backdoor that does not appear in the model's behaviour until activated by a specific trigger. The risk of poisoning is particularly high when models shared through open-source platforms are used to build an AI agent. In addition to poisoned data and models, these may also contain malware.

In agent systems, an attacker may specifically seek to poison the agent's memory and context. The attacker corrupts or adds malicious or misleading data to the agent's context, which biases the agent's reasoning, planning and tool use. For example, different input sources, such as file uploads, user inputs and information exchange between agents, create opportunities for memory and context poisoning.

3.2.5 LLM05:2025 Improper output handling

Improper output handling refers to insufficient validation, sanitisation and handling of content generated by large language models before it is passed downstream to other components and systems. In the case of AI agents, the risk arises particularly when the agent's output is used directly to

perform tasks without human intervention. This is also connected to excessive agency, which is introduced in the following subsection 3.2.6.

If text or code generated by large language models has not been appropriately validated, sanitised or handled before being transferred to user interfaces, databases, interfaces or other systems, this may expose the system to improper output handling. This may also enable other vulnerabilities and attacks, such as remote code execution, cross-site scripting (XSS), cross-site request forgery (CSRF) or SQL injection.

Agent-based systems, including popular vibe coding tools, often generate and execute code. Attackers can exploit code-generation features or tool access embedded in the agent to escalate their activities through remote code execution (ASI05) into internal and local systems.

Agents can establish strong trust with human users through their fluency in natural language, emotional intelligence and perceived expertise. Attackers can exploit this trust to steer user decisions, extract sensitive information or influence outcomes for malicious purposes (ASI09). In agent-based systems, the risk increases when people place too much trust in recommendations or justifications produced by autonomous AI without appropriate review.

3.2.6 LLM06:2025 Excessive agency

LLM-based systems are often granted too much independent agency, such as the ability to call functions, connect to other systems through extensions and perform actions based on prompts. An agent may also be given decision-making authority to select which extension to invoke.

AI agent systems typically make repeated calls to language models so that the results of previous calls are used as the basis for subsequent calls. Excessive agency enables harmful actions to be performed in response to unexpected, ambiguous or manipulated outputs from the language model. In other words, the agent operates too independently and with excessive permissions without human intervention or appropriate validation of outputs.

The root cause of the vulnerability is typically one or more of the following:

- excessive functionality
- excessive permissions
- excessive autonomy.

This vulnerability may compromise the confidentiality, integrity and availability of information, depending on which systems the agent interacts with.

Excessive agency is a significant risk. In agent systems, an attacker may use, for example, over-privileged interfaces or excessive inherited permissions granted to the agent as an attack vector. Excessive permissions can lead to several threats characteristic of agent systems, such as cascading failures, tool misuse and exploitation or agent goal hijack. Due to these risks, restricting permissions in accordance with the principle of least privilege is a key security measure.

3.2.7 LLM07:2025 System prompt leakage

System prompts that guide the model's operation may contain sensitive information that the user is not intended to discover. The purpose of system prompts is to guide the model's operation on the basis of the application's requirements, but they may also inadvertently contain secrets If these secrets are discovered, they may be misused, for example as part of other attacks.

The disclosure of a system prompt does not directly create a risk. The risk arises if the disclosed prompt contains sensitive information, such as information about the guardrails used in the system or the system's access rights. The disclosure of a system prompt may also provide the attacker with information about its safeguards and in this way help the attacker bypass them.

3.2.8 LLM08:2025 Vector and embedding weaknesses

Vector and embedding weaknesses refer to weaknesses in how large language models represent and process data using vectors and embeddings. An embedding means converting words or sentences into numerical form as vectors whose positions define their mutual meanings and relationships. Embeddings allow the model to understand the content of text, the meanings of words and the relationships between them. Weaknesses, in turn, arise from how text is converted into vectors and how these vectors are stored in and retrieved from a vector database.

The risk is particularly high in systems that use Retrieval Augmented Generation (RAG). RAG enriches the outputs of generative AI by retrieving information from external sources, improving the performance and contextual relevance of language model-based applications by combining pre-trained language models with external knowledge sources. RAG is therefore commonly used in building AI agents when the agent is intended to use the organisation's own data as the basis for its outputs.

These weaknesses can be used to add malicious content, manipulate AI responses or search for sensitive information. In an agent system, vector and embedding weaknesses are used particularly for memory and context poisoning (ASI06).

3.2.9 LLM09:2025 Misinformation

Misinformation, or incorrect information, is one of the key threats to applications based on large language models. It may consist of false or misleading information produced by AI that appears credible. One cause of misinformation is so-called AI hallucinations, which arise when a language model attempts to fill gaps in the training data through statistical reasoning without understanding the content of the data or the operating environment.

Threats related to misinformation often materialise when excessive trust is placed in AI outputs. The accuracy of AI outputs must always be verified by a human, especially if they are used as the basis for critical decisions. Misinformation may lead to many problems, such as security breaches, reputational damage and legal consequences.

In an agent system, human-agent trust exploitation (ASI09) and rogue agents (ASI10) often lead to misinformation.

3.2.10 LLM10:2025 Unbounded consumption

Unbounded consumption means a situation in which a large language model can respond to repeated user queries without restrictions.

This vulnerability is used in attacks intended to disrupt a service, deplete the target’s financial resources or steal intellectual property by copying the model’s behaviour. Unbounded consumption occurs when an AI agent allows users to make excessive and resource-intensive requests to the model, which, among other things, consumes system capacity and generates costs for the agent owner from data processing and generation. This leads to risks such as Denial of Service (DoS), financial losses, model theft and service degradation. The typically high computational demands of large language models make them susceptible to resource exploitation and unauthorised use.

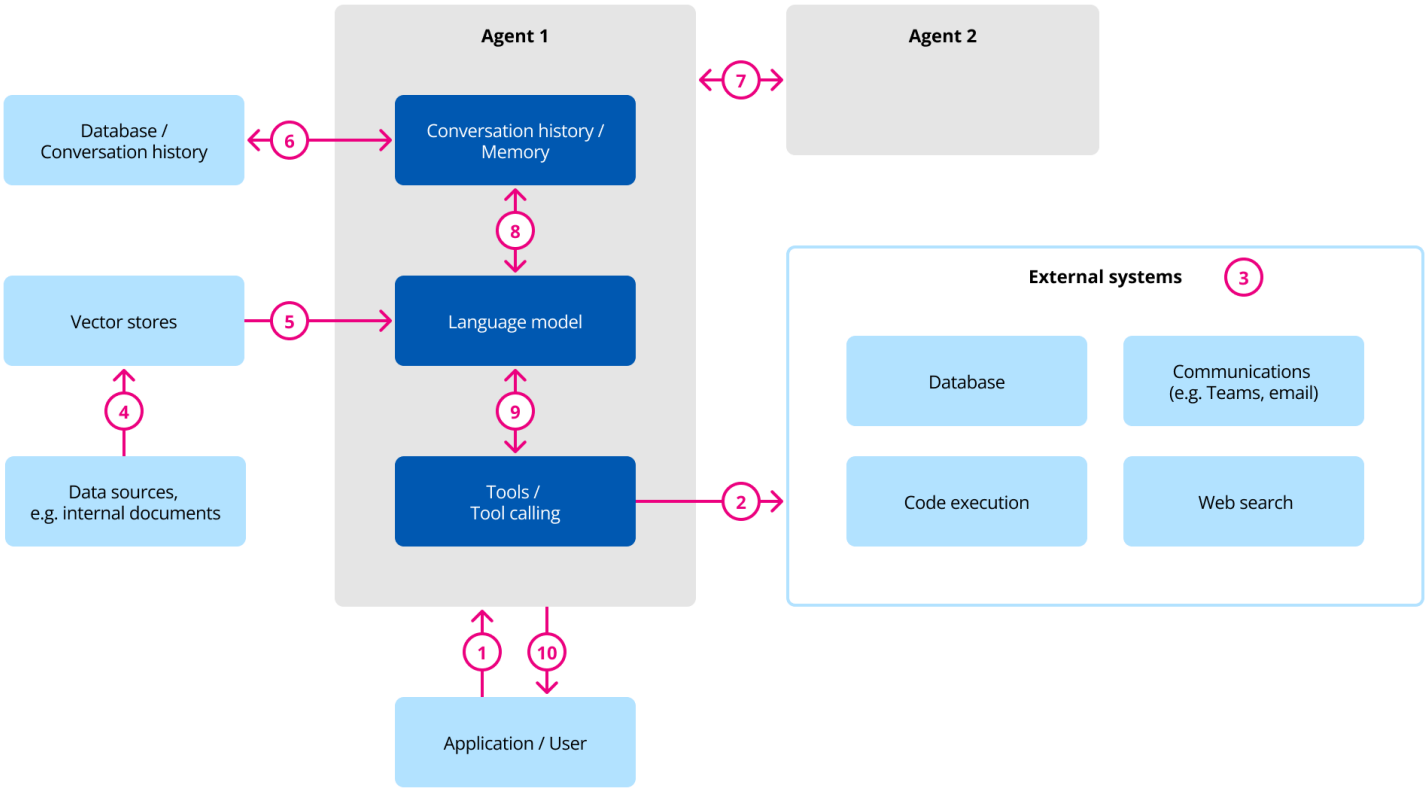

Figure 2. Vulnerabilities in AI systems. Source: OWASP top 10 for LLM apps & Gen AI agentic Security initiate, v1.

1. Prompt injection (LLM01), unbounded consumption (LLM10)
2. Excessive agency (LLM06)
3. Supply chain vulnerability (LLM05)
4. Contamination of a vector database with unverified documents (LLM08)
5. Change in language model behaviour
6. Memory contamination (LLM04)
7. Sharing false information between agents, rogue AI agent
8. Memory contamination, data protection
9. Calling an incorrect or wrong tool
10. Hallucination, misinformation (LLM09), system prompt leakage (LLM07), sensitive information disclosure (LLM02), improper output handling (LLM05)

4 Designing an AI agent

The identification and management of the risks presented in Chapter 3 begins already in the design phase of the AI agent.

During the design phase, it is necessary to consider why you specifically want to deploy an AI agent. How should the AI agent be designed so that it is secure? What should be taken into account before technical implementation and the first pilot are even started? The successful deployment of AI systems always requires careful planning so that the final solution meets the expectations set for it without taking unreasonable information security risks.

In this section, we focus on information security in the design phase of an AI agent. We examine key security-related questions that help ensure that the planned AI solution is fit for purpose even when it is used in functions that are essential to core business. This allows us to move towards technology choices that solve the right problems securely.

4.1 Setting a security-aware target state

At their best, AI agents can help organisations improve the efficiency of their operations and provide better products and services. However, AI agents are also subject to many excessive expectations, which makes careful target state setting important. Clarifying the target state helps weigh information security risks against the intended use case and the value to be produced. This helps prioritise measures that eliminate threats and design AI systems that are secure from the outset.

An AI agent should always be seen as part of an existing process, as it never operates separately from the organisation’s other activities. The secure use of AI and the timely detection of threats require human presence, as all AI solutions have limitations that mean processes should only very rarely be fully automated (see, for example, the vulnerability in section 3.2.6, excessive agency). Start by seeking answers to the following questions:

- **Which process will the AI agent become part of? What is the current state of the process and how do we want the agent to change it?** For this purpose, bring together the people who normally participate in the process. If there are many of them, meaning more than approximately seven people, select a group that is as representative as possible from different perspectives. A diverse group ensures that the target state of the system takes comprehensive account of the perspectives of different stakeholders. The target state should also be considered together with the people participating in the

process so that everyone agrees on the need for the agent and the target state.

- **What positive impact is sought through AI? For whom?** Consider what positive impact will move you closer to the target state. Is it efficiency, speed, quality, ethical conduct or something else? Who will benefit? What can we compromise on and what can we not compromise on? Identifying the positive impact creates a basis against which threats are identified, assessed and prioritised.
- **What key constraints are associated with the process?** Identifying information security-related constraints well in advance is important, as they provide the framework for designing the solution. Is the data we process particularly sensitive? Do our stakeholders have expectations that affect the deployment of the agent?

Example: A company wants to improve the use of unstructured data repositories with the help of an AI agent. It has several different data sources of varying quality. Based on a joint discussion, the target state is defined as a situation in which experts can more efficiently perform tasks that previously required the manual review of large volumes of data in order to initiate the correct actions. With the help of the AI agent, the organisation wants to bypass the manual information retrieval phase and perform the action automatically, making the experts' work easier. Both the organisation itself and its customers benefit from the change. This strengthens the responsibility of the organisation's operations and supports its societal mission.

However, the organisation also identifies constraints. The organisation is subject to high information security requirements, which limits the range of available technologies. Functional safety requirements and change management also restrict the role of AI and emphasise human judgement.

4.2 Preliminary information security risk assessment as part of design

A preliminary risk assessment should be carried out for the idea already during the design phase, examining information security risks alongside the organisation's other risks. The preliminary risk assessment is supplemented with threat modeling (see Chapter 5 of this guidance), which examines the likelihood and impact of threats in more detail. However, the preliminary assessment provides a basis for deciding which ideas should be taken forward from the design phase towards threat modeling.

4.2.1 Use case

The use case of an AI agent has a significant impact on its information security. During design, pay attention to the following issues and document the answers for threat modeling.

Scope of the use case. Is the use case clearly limited, or can the tool be used for many different work tasks? The broader the use case, the easier it is to influence the agent's operation maliciously. For a limited use case, the agent's operation can be restricted as much as possible through the system prompt and other technical means. If the use case is broad, there are fewer restrictions, which makes gaps considerably more likely to emerge. In a broad use case, risks relating to the misuse of the agent's outputs, excessive agency, system prompt leakage and misinformation are emphasised, for example.

Changes to the use case. How likely is it that the use case will change along the way? Because generative AI is a general-purpose technology, the user may want to use the agent for purposes other than its original intended use. If users have many diverse use cases for which they would like to use AI, but only a few secure tools approved by the organisation are available to them, there may be pressure to use even an agent intended for a limited purpose for other purposes as well. This may increase the risk of, for example, misuse of the agent's outputs, excessive agency or sensitive information disclosure.

Monitoring the use case. How will the monitoring of the use case and changes to it be designed as part of the solution architecture? Monitoring the use case is not only a technical issue, as it increases detailed monitoring of an employee's tasks. For this reason, it is necessary to consider already during the design phase what type of monitoring can be carried out so that changes in purpose and appropriateness can be meaningfully tracked and vulnerabilities detected. Monitoring can help manage the risk of, for example, sensitive information disclosure, system prompt leakage or unbounded consumption, and detect if the risks materialise.

Susceptibility of the use case to misinformation. Is the agent part of a process that is carried out under time pressure? Is there sufficient time and opportunity to verify the correctness of the agent's operation? Is the agent's use case so narrow and the amount of background data so limited that it is difficult to make the agent's operation reliable? Large language models work best in tasks where suitable patterns are repeated in their training data. Consequently, highly specialised subject areas are more difficult for agents, especially if the organisation does not have a sufficiently comprehensive data repository on the subject that the agent could use to support its operation. The dissemination of incorrect information exposes the organisation to information security risks, so these should be considered already when designing the use case.

Maturity of the user group. Do the agent's users know what the autonomous operation of AI is based on? Are they able to assess the reliability of the agent and the correctness of its outputs? User behaviour is one of the key aspects of information security. It is therefore important already during the design phase to build an understanding of how likely users are to understand in which cases agents are reliable and for what purposes the agent should not be used. This helps manage risks related to, for example, the misuse of AI-generated information, excessive agency and misinformation. This understanding can be deepened, for example, through user interviews.

Answering the questions discussed above provides a better understanding of how significant the risks associated with using the agent are in the planned use case. If the risks already appear substantial at the preliminary stage, it is worth considering whether another use case would involve risks that are easier to manage, or whether a technology with more manageable risks could be applied to the problem. Proceeding with a high-risk use case should only be considered if the AI agent provides significant value to the organisation and the organisation has sufficient resources to manage the threats. Even a use case idea that produces considerable value should not be pursued if the risks appear insurmountable.

4.2.2 Data repositories

If the risks of the use case appear manageable, the next step is to assess the impact of the data required by the use case on the security of the tool. During design, pay attention to the following issues and document the information for threat modeling.

Sensitivity and confidential nature of the data. Does the AI agent use confidential or sensitive information, such as trade secrets or personal data? Do all users who review the agent's outputs also have access rights to the source data? The more sensitive the data used by the agent, the riskier the solution is. One of the key information security threats related to language model-based solutions is specifically sensitive information disclosure, as a result of which information may end up in the wrong hands. Data poisoning may also compromise the integrity and accuracy of such data.

Data minimisation. There is often a desire to provide the AI agent with a wide range of different data just in case. However, the data available to the agent should be limited as precisely as possible so that the risks mentioned above can be managed. When designing the agent, it is necessary to consider which data is genuinely relevant from the perspective of the use case and which data is unnecessary. Unnecessary data may also have a negative impact on the quality and consistency of the agent's operation, so data minimisation is also a quality issue. Data minimisation helps reduce

risks related to, among other things, sensitive information disclosure and data poisoning.

Access rights. Potential conflicts in data access rights should be identified already during the agent's design phase. Especially if the agent has broad access to data, situations may arise in which the agent discloses information to users that they themselves would not have the right to access. It is also difficult to see from information filtered by the agent to whom the user may share the information from the perspective of access rights, especially if the response has been compiled from several different sources. Access rights management and the constraints it creates should therefore be considered already during the design phase so that the planned use case and solution are also feasible from this perspective.

4.2.3 Integrations

In practice, an AI agent is almost always integrated into the organisation's other systems if it is intended to use the organisation's own data as the basis for its operation. During the design phase, it is important to identify how critical the systems are that the AI agent would need to access in order for the solution to achieve the objectives set for it. At this stage, organisations in safety-critical sectors in particular must consider whether an AI agent can be deployed for the intended purpose if the integrations required for it would expose critical infrastructure, for example, to the risks of generative AI.

If the AI agent uses the same resource as systems that are important for core business, for example, a malicious user may overload the entire system through the AI agent and thereby cause disruptions. It may also disclose confidential information about the organisation's processes and operating practices through the system input.

To identify intolerable risks in time, identify the answers to the following questions:

- Which systems must the agent be integrated with in order for it to be used for the intended purpose?
- How critical are the systems from the perspective of information security?
- How critical are the systems from the perspective of operational reliability?
- If the systems are critical, can access to them be meaningfully restricted so that the risks associated with AI can be managed sufficiently?
- Could the intended use produce value even without a connection to critical systems?

At the time of writing this guide, organisations' capabilities for the secure deployment of AI agents are typically limited, and best practices for implementing agent systems are still developing. For this reason, automating core business with AI agents is highly risky. It is therefore important already during design to identify how critical the system infrastructure is and whether the agent can produce the intended value even if its access to critical systems is restricted.

4.2.4 User involvement

The user plays a central role in ensuring information security. For this reason as well, it is important that the solution is designed in cooperation with users. User involvement also has a positive impact on the quality and deployment of the AI agent.

Involving users in the design at an early stage helps understand how users will actually use the agent, what kind of information they want to provide to it and what capabilities they have to identify information security risks and act on their observations. Users can be involved in many ways. Below, we list methods that have been found useful from the perspective of understanding information security.

Interviews. Interviews are one of the most resource-efficient ways to build an understanding of how different users would like to use the AI agent. Interviewees should be selected from different user groups so that the information collected gives a good picture of the agent's actual use in the organisation. Interviews should be conducted at the beginning of the design

phase so that the information collected from them can also be used in threat modeling.

The content of the interviews depends on the planned use case and the organisation's context. The following questions offer examples of what interviewees can be asked to help identify information security vulnerabilities when they do not work with AI or information security as part of their job:

- What types of processes would you like to automate with AI agents?
- What information should the AI agent have access to in order to provide useful outputs?
- Are there others in your team, organisation or unit who would like to use AI for another purpose?
- How likely is it that the AI agent would be needed for other purposes in your team, organisation or unit in the near future? Over what time frame?
- Do you think that, in your work, there might be a need to use the AI agent for something other than its primary purpose? How likely do you consider this to be?
- Could you give examples of tasks that you would now like to assign to an AI agent?
- How would you assess the reliability of the AI agent's outputs in the situations you described? Do you feel that you would then have enough time to assess and/or verify the outputs as required?
- For what purpose and how would you continue to use the AI agent's outputs?

Interviewees in different roles should be asked different questions based on their job description. For example, the more closely a person works with information security, the more directly they can be asked questions related to information security challenges.

Co-design. Include user representatives in workshops where the AI agent and its use case are designed. This helps build a better understanding of users' attitudes towards the agent and its use case from the perspective of information security already in the early stages of design.

Testing. The first versions of the solution should also be tested together with users in well-planned testing sessions. Testing is discussed further in Chapter 7 of this guidance.

4.3 Procuring an AI agent

Because there are many potential application areas for agent systems, there are also many ways to implement them. A system can be procured almost entirely as a service, built internally from freely available components, or implemented by combining these approaches.

Sometimes, the identified needs can be met with an off-the-shelf AI solution developed by a third party. Procurement is often chosen in particular when the organisation does not have the resources to develop an agent from scratch itself or with a partner, or when one of the available agent solutions is well suited to the intended purpose.

When procuring an agent, many security-related requirements apply to the developers of the AI. However, the buyer must also ensure, during the competitive tendering and contract phase, that the selected supplier is able to meet legislative requirements and that responsibilities are clearly defined in the tendering terms. This helps avoid ambiguity in responsibilities if the AI agent compromises the organisation’s information security.

Appendix 1 to this guide contains a checklist of issues that the buyer should consider as part of procuring an AI agent. The list covers questions related to governance and the life cycle, functionalities, security, regulation, transparency and other key perspectives. The list supports the preparation of competitive tendering and procurement as well as the drafting of contracts, so that the security of an agent developed by an external party can be verified as effectively as possible.

4.4 Towards implementation: Risk assessment and the EU AI Act

The background information collected above has provided a comprehensive picture of how risky the planned use case is from the perspective of information security. Based on this information, a decision is made on whether the risks are at an acceptable level. The risk matrix below can be used to support the overall assessment (Figure 3).

	Low	Moderate	High	Very high
Impact on core business operations	The system does not perform tasks related to core business operations.	The system performs easily manageable and limited parts of workflows that form part of core business operations.	The system performs parts of the organisation's core business operations.	The system performs essential parts of the organisation's core business operations.
Autonomy and complexity	The system is not integrated with other systems or other agents.	The system is integrated with a limited number of external systems that are not critical for core business operations.	The system is integrated with several external tools and/or is a multi-agent system. Some integrations concern systems that are critical for core business operations.	The system is a multi-agent system with several interfaces and integrations with systems that are critical for core business operations.
Sensitivity of data	The system only processes public information.	The system processes the organisation's internal information.	The system processes confidential information.	The system processes secret and sensitive information.

Figure 3. Matrix to support risk assessment

The risk assessment should consider at least the criticality of the system for core business, the level of autonomy and complexity required by the solution and the sensitivity of the data processed by the agent.

The preliminary risk assessment described above helps assess the risks of AI in relation to the planned use case. In addition, however, it must be noted that the **EU AI Act** ((EU) 2024/1689) also establishes common rules for the fair, appropriate and transparent use of AI. The AI Act regulates AI systems based on their risk and imposes obligations on them according to

the system's risk basis. Under the EU AI Act, AI systems are divided into the following categories based on their risk:

- **Prohibited AI practices:** Prohibited practices in AI systems are practices whose risks are contrary to EU fundamental rights, for example from the perspectives of health and safety. They may not be provided, developed, made available on the market or used in the EU, unless there is a law enforcement basis for doing so. Examples include the use of facial recognition for mass surveillance and social scoring.
- **High-risk AI systems:** AI systems that have a significant adverse impact on the health, safety or EU fundamental rights of citizens. Examples include safety components of critical infrastructure, education and work management, democracy and judicial processes, biometrics and facial recognition. Most of the requirements and obligations under the AI Act apply to the providers of these systems.
- **AI systems subject to transparency obligations:** For certain so-called low-risk AI systems, the AI Act sets requirements and obligations related to transparency, particularly for the providers of these systems. These include, among other things, a requirement to mark AI-generated content (text, image, video) so that it can be distinguished from human-generated content. Such AI systems may include systems designed for interaction with individuals, such as chatbots.
- **Other systems:** AI systems that do not fall into any of the other categories mentioned above. The AI Act does not impose requirements on these systems.

Each party providing an AI agent must therefore ensure the lawfulness of its AI system. The organisation must identify its own **role** in relation to AI (provider, deployer or another role), the use case and the **risk category** of the AI system, as well as the requirements corresponding to that risk category that apply to its own role. Critical infrastructure sectors and other sectors covered by Union harmonisation legislation must also take into account the specific requirements that apply to them, for example concerning product safety. The applicability of the AI Act can be assessed using, for example, the **EU's own checklist**[7], which helps organisations get started with applying the AI Act.

The AI Act and the related guidance are available in full on the website of the European Commission's AI Office[8].

[7]The EU AI Act Compliance Checker on the European Commission website: https://ai-act-service-desk.ec.europa.eu/en/eu-ai-act-compliance-checker.
[8]Website of the European AI Office: https://digital-strategy.ec.europa.eu/fi/policies/ai-office.

When moving from the design phase towards technical choices, this information provides an understanding of the risks associated with technical choices in the selected use case. The following chapters describe in more detail what types of information security risks are associated with technical choices and what types of choices strengthen the information security of an AI agent.

Before moving from design to implementation, take one more look at the surrounding organisation and the existing governance measures.

- Does your organisation have an AI governance model and related processes through which the AI agent should be assessed?
- Does your organisation have an AI register or catalogue in which the AI agent should be recorded?
- Are there any other information security or information management processes or measures that apply to all systems and should therefore also be carried out for the AI agent, such as a DPIA?
- If the AI agent or parts of it are procured externally, is there anything in the organisation’s procurement guidelines that we should be aware of?
- Have we also familiarised ourselves with the EU AI Act and other regulation and prepared for the requirements they bring?

When these issues are taken into account before moving to implementation, AI becomes part of the organisation’s comprehensive risk management, which supports the implementation of information security throughout the AI life cycle.

If it appears that the risks identified during the design phase can be borne and there is a desire to proceed with the agent towards implementation, the information security threats should be modelled in more detail so that they can be limited as effectively as possible. Threat modeling is examined in more detail in the next chapter.

5 Threat modeling

Threat modeling is a structured and systematic approach to identifying security risks in applications. It is an important stage in development work, as it helps identify potential threat scenarios and their likelihood and impact.

Threat modeling makes the threats that may affect AI agents visible. Once the threats have been identified, controls can be targeted appropriately and cost-effectively. This ensures that resources are used sensibly and in a targeted manner, and that the organisation can ensure a sufficient level of information security for the agent in line with its own risk appetite.

Threat modeling should be started in the early stages of the project so that the resulting requirements and observations can be considered sufficiently early. In addition, threat modeling should be carried out whenever major changes occur in the project, for example in relation to the architecture or personnel. Threat modeling is not a one-off process but should be a continuous activity. In this chapter of the guide, we examine how threats typical of AI can be modelled.

5.1 Threat modeling for agent-based AI solutions

Although AI security is strongly based on traditional information security, threat modeling for AI requires traditional frameworks to be extended. Established threat modeling methods, such as STRIDE[9] and LINDDUN[10], focus on identifying traditional cybersecurity threats and do not take AI-specific perspectives into account. The special characteristics of AI agents are described in more detail in Chapter 2.

One way to structure threats to agent solutions is MAESTRO[11], the Multi-Agent Environment, Security, Threat, Risk and Outcome method. It seeks to complement traditional methods with a layer-by-layer approach when threat modeling AI solutions. Each threat is first linked to the layer of the reference architecture where it is primarily present or most severe. It is then linked to other relevant layers with which the threat interacts or to which it may spread. In this way, complex AI agent solutions can be broken down into separate functional layers, enabling risks to be understood and addressed in more detail while also considering threats that cut across layers.

5.2 Threat modeling guidance

Threat modeling usually consists of three stages, the content of which is examined next. The practical arrangements themselves can be implemented in accordance with the practices that suit the organisation. The most important thing is not a specific working method or framework, but that the people working with the subject of the threat modeling stop together to consider possible threats.

Threat modeling is often best carried out in workshops, either in one longer workshop or several shorter ones. It is advisable to reserve two to three hours for each stage. If the organisation does not yet have established threat modeling practices, the Threat modeling template in Appendix 2 can be used to support the workshops.

5.2.1 Stage 1: Defining the subject of threat modeling

In the first workshop, it is clearly defined what is being built and what must be protected. This stage creates a clear understanding of the agent to be threat modelled, the components it consists of what information moves within it and how. In addition, integration points with other systems are

[9] The STRIDE model: https://learn.microsoft.com/en-us/azure/security/develop/threat--tool-threats
[10] The LINDUNN framework: https://linddun.org/linddun/whyuselinddun/
[11] Agentic AI Threat Framework: https://cloudsecurityalliance.org/blog/2025/02/06/agentic-ai-threat--framework-maestro

identified. Threat modeling can make use of a method suitable for the use case, such as MAESTRO in the case of an agent solution. The approach described in this guidance is suitable for threat modeling a wide range of subjects.

Participants should be highly familiar with the features planned for the agent and its operating environment. The workshop must include people who are responsible for the business and for managing its risks. It is important to take different perspectives on the agent solution being developed into account so that the threat model is comprehensive. Recommended participants in the workshop include, for example, the business owner, product owner, project manager and architect.

Example: An organisation is planning to build an AI agent to automate and improve the efficiency of customer service. In the first workshop, the key assets to be protected in the AI solution are identified and recorded, such as customer service process descriptions, the organisation's customer database, the technical solutions of the AI agent, documentation guiding the agent and the language models used.

The users and access rights related to the solution to be implemented, such as internal and external developers, the maintenance team and end users, are listed. The AI agent also has dependencies on, among other things, interfaces, third-party libraries and the language models used. A description of the solution to be implemented is created, showing the key components and the information flow in a typical customer service situation.

5.2.1.1 Information and functionalities to be protected

The assets to be protected in AI agents include the data processed by the agent, the agent's most important functionalities and the other systems to which the agent is connected. Not all information is equally critical or sensitive, so at this stage, the assets that may be of interest to a potential attacker are identified and prioritised. It is essential to identify where information is processed or stored and who has access to it.

For AI systems[12], the following factors should be taken into account:

- **Models** – trained models, model parameters and hyperparameters
- **Actors** – data owner, AI developer and model developer/provider
- **Processes** – model tuning, pre-processing and data collection

[12] Stephen Tete (2024), Threat Modeling and Risk Analysis for Large Language Model (LLM)-Powered Applications: https://arxiv.org/abs/2406.11007v1

- **Tools** – visualisation and data analysis
- **Artefacts** – model use cases, metadata schemas and model architecture
- **Data** – raw data, labelled data and validated data

5.2.1.2 Users and access levels

Users and each user's access rights must be listed so that threats affecting different user groups and the roles associated with them can be identified. This also facilitates access rights management, the modeling of possible attack paths and the prevention of unauthorised access. It can also help prevent the creation of user roles with excessive access rights.

Understanding users' rights helps create realistic attack and threat scenarios. The agent and its access rights play a central role in this. Based on the agent's identity, it can be identified, granted access rights and its activities can be monitored.

Common user groups include administrators, internal and external developers, service accounts and machine-to-machine APIs as well as end users of the service with different access levels.

5.2.1.3 Dependencies

Dependencies are used to identify the internal and external resources on which the operation of the system relies. This provides a better understanding of the overall attack surface and supports the monitoring of authentication, authorisation and vulnerability management. The more integrations the system has, the more difficult it becomes to manage the overall risk profile.

The following are common dependencies in agent-based AI solutions:

- Public cloud environment
- Private data centre
- Private networks and virtual networks
- Integrations and their interfaces (REST, SOAP, GraphQL, WebSocket)
- Third-party libraries and model libraries
- Machine learning models and large language models used by agents for reasoning and generation
- Third-party AI agents

- Orchestration systems
- Identity and access management (IAM) solutions
- Monitoring and logging systems
- Authentication and authorisation services
- External data sources
- SaaS services
- Development and testing tools (CI/CD systems, simulators, version control)

Dependencies can create threats to the system such as the following:

- System availability may be compromised if a third-party application service, interface or language model service is unavailable.
- An attacker may gain access to the system through leaked credentials, such as API keys.
- Confidential information may leak to the model service provider.
- The orchestration system controls which tools the agent is allowed to use. If the orchestrator is compromised, an attacker may direct the agent to make malicious calls.

5.2.1.4 Data flow diagram

The purpose of a data flow diagram is to illustrate how information moves between the components of a system. This makes it easier to identify critical points where threats may materialise or where potential attack surfaces may exist. Examples include unencrypted traffic, inadequate access management, an untrusted data source or an unprotected interface.

The data flow diagram shows what data the AI agent has access to, where it can send data itself and what types of commands it can execute in other systems. The diagram should describe the methods and locations for collecting, processing and storing data. In addition, the diagram should always describe trust boundaries where authentication must take place. Keeping the data flow diagram up to date ensures that the system documentation remains current.

5.2.2 Stage 2: Identifying threats

In the second workshop, potential threat actors, their motivations and their attack vectors are mapped. An attack vector means a path, method or scenario that can be used to break into a system.

The objective of this stage is to identify as many potential threats as possible, regardless of how likely they appear to be. Using the results of the previous workshop, the participants identify where threats may occur. Known frameworks and threat lists should be used to support identification, for example:

- The OWASP Top 10 list of the most significant vulnerabilities related to AI systems and AI agents in Chapter 33.
- MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems), a globally accessible living knowledge base of tactics and techniques against AI systems: MITRE ATLAS™
- OWASP guidance for threat modeling agent-based AI solutions: Multi-Agentic system Threat Modeling Guide v1.0 - OWASP Gen AI Security Project

In addition, traditional cybersecurity threats should also be taken into account. The OWASP Top 10 list[13] can be used for reviewing these, for example. The data protection perspective is also important to consider, as data is at the core of every AI system.

Participants in threat identification should understand what can typically go wrong from both a technical and a business perspective. The emphasis in selecting participants depends to some extent on the subject of the review and the desired scope. Threats can be examined across the entire subject being developed, or the workshops can focus on a specific area or stage of the agent life cycle. Participants may include, for example, an architect, information security officer, product or system owner and other suitable experts.

As threats are identified, the threat model of the AI agent being built starts to take shape. It is important to also record threats for which controls have already been defined and implemented. The threat model evolves with development, and a comprehensive threat model is therefore an essential tool for monitoring the threat landscape.

Example: In the case of an agent automating an organisation’s customer service, particular attention is paid to threats related to the agent’s operational reliability, such as resource exhaustion, malicious prompts, model manipulation and tool misuse. These threats are intended to impair the agent’s operation, which may lead to disruptions and, in the worst case, prevent its operation entirely.

[13] OWASP Top 10: https://owasp.org/Top10/

During threat modeling, it is observed that the sector and the technologies used are developing and changing rapidly, so it is decided that threat modeling will be carried out quarterly.

5.2.3 Stage 3: Assessing threats and defining controls

In the third workshop, threats are prioritised based on their likelihood and impact. This makes it possible to define and implement mitigation measures to reduce risks and improve the security of the system in order of priority and in a resource-efficient manner.

When considering controls, the functions defined in the NIST Cybersecurity Framework[14] can be used, for example: protect, detect, respond and recover. Decisions on the implementation of new controls and their scheduling are recorded in accordance with the project's practices. Responsible persons must be agreed for concrete controls. They ensure that the controls are put into practice, for example by recording the measures in a ticketing system and assigning responsibility for them.

During the final workshop, iteration and testing decisions are discussed, as well as the scheduling of the next threat modeling exercise. Workshop participants should understand the impact of different types of threats on business operations and risk management. For technical threats, an understanding of the specific characteristics of the environment may be needed, such as the effects of existing controls.

As threats are assessed, the threat model becomes more detailed, and an understanding is formed of how likely the threats are and what impacts they may have. The threat model serves as documentation showing that threats have been identified and that the risks have been brought under control.

Example: As a result of threat modeling, the organisation obtains a threat model that includes the identified threats and assessments of their likelihood and impact. Appropriate controls have been defined for the threats, and their implementation and schedule have been planned. The development of the agent solution takes into account the threats typical of such a solution. Controls are implemented for the threats. These may include, for example, limiting the number of requests and filtering inputs and outputs.

The responsible person adds the implementation of the controls to the development backlog and ensures that the threat model is updated once the controls have been introduced. Review of the threat model is scheduled for the following quarters as agreed, unless a need arises during development to review the threat model earlier.

[14] National Institute of Standards and Technology (NIST) Cybersecurity Framework (CSF): https://www.nist.gov/cyberframework

It is recommended to define what types of changes are so significant that they require the threat model to be reviewed again. Such changes could include, for example, a model update, a new model, a new integration, expansion to a new use case or changes to the system infrastructure.

6 Building an AI agent

Once the threats have been modelled, the controls defined and responsibilities assigned, the process can move on to building the AI agent. This chapter describes how different choices made during the building stage affect the agent's security based on generally known risks. The chapter describes choices related to data selection, processing and storage, considerations related to large language models and questions related to the operation of the agent. These guidelines help an organisation build an agent in accordance with current best security practices.

6.1 Data and its secure management

Data plays a central role in the development of AI agents. If the data is poor or distorted, the agent cannot achieve good results. The quality, security and value of source data should be assessed already when the data is acquired and before it is used in training, testing and the actual use of agent systems. These measures are described in more detail below.

6.1.1 Assessing data and sources

The objective of assessing data is to form an understanding of the extent to which the data is secure and usable, whether more data needs to be acquired and how it must be preprocessed.

Tabular data can be assessed, for example, by listing variables, calculating sample sizes and the proportions of missing values, and examining distributions, trends and outliers. Anomalies or missing data may indicate data selection or manipulation. When analysing text data, languages are identified, the frequency of vocabulary is compared with general language, the proportion of professional terminology is assessed and, where necessary, sentiment, entity and topic analysis are carried out. At the same time, the data is searched for malicious content intended to manipulate the language model. Recognition, classification, segmentation and content description can be used to assess the quality and security of images, videos and audio.

6.1.2 Generative AI models and data

When selecting a foundation model, it should be noted that the model manufacturer has already selected and processed the data on which the model has been trained. In selecting the model, attention should therefore be paid not only to performance, but also to what the manufacturer says about the training data and how it presents this information. If the sources of the training data are described separately and credibly, the choice of model can be better justified. Attention should be paid to the data used in

training, as studies have found that even a small amount of malicious data can poison a model[15].

It is usually not possible to obtain full certainty about potential problems in the training data of large language models, for example, unless the data is fully public. An attempt can be made to assess the quality and security of the model's training data by looking for studies in scientific or technical literature, at least regarding the datasets and sources that the manufacturer has made public.

An AI agent may be subject to requirements that should also be checked from the perspective of the training data of the models it uses. These include, for example, ethics, sustainability, any licence terms or terms of use imposed by the model manufacturer, restrictions on use and copyright in the training data. Any biases and skew in the training data, especially concerning people's age, gender, nationality or other special characteristics, should be investigated where possible so that the necessary controls can be applied to them, such as bias mitigation.

6.1.3 Metadata

Metadata may potentially be used to classify data and to assess its reliability and usability. For example, in the assessment and filtering of internet content, metadata (such as the author's user ID, IP address data, the possibility of linking an IP address to an operator, geographical area or VPN service) may be relevant. An IP address, for example, is often personal data within the meaning of data protection principles and must therefore also be processed as such. It is advisable to be aware of the existence of metadata so that the risk of protected information being leaked can be managed.

6.1.4 Data versioning and traceability

Data used in testing agent systems should be kept separate. If the system does not function as intended, the cause may be found in the test data. If the models used in the system are to be trained, the data used for training should also be versioned. This allows any problems that emerge over time to be linked to the correct version of the training data. Versioning can also be used to test the effect of data preprocessing or augmentation on the model's performance. If the agent system is updated, versioned data can be used to compare the results before and after the update.

[15] Souly et al. (2025). Poisoning attacks on LLMs require a near-constant number of poison samples. https://arxiv.org/abs/2510.07192

6.1.5 Data retrieval and integrations

The data used by an agent system is often collected from the organisation's systems or public sources through various integrations. The purpose of integrations is to retrieve and transform data into a format that the system can use. Integrations may deliver the data directly to the system or store it in the organisation's data repository or data platform. Reliable integrations are therefore very important for the application's value in use and reliability.

The reliability and security of data acquisition can be improved with well-defined interfaces. Access rights to interfaces must be defined carefully, taking into account the principles of data minimisation, anonymisation and zero trust. Interfaces should be designed to be clear and simple. For example, there should preferably be only one interface call for a single item of data.

Regarding data retrieval, the roles of the integration, data repository and application must be defined. It is advisable to define which part of the data should be stored with the application itself, for example temporarily, and which part can be retrieved through an integration or query from the data repository whenever needed. Managing data storage in connection with the application may make sense in some situations, especially if the use case requires a fast response time or if the cost of repeatedly retrieving the data is high. Combining data, keeping it up to date and ensuring reliable delivery are usually easiest to solve with systems designed for these needs. This keeps the application's own data management burden as small as possible.

6.1.6 Data currency

Data should be acquired at least on the same cycle as the results are expected to be delivered, for example weekly, monthly, quarterly or annually. Keeping data up to date is a particular challenge in applications that use extensive document collections. Examples include workflows using RAG architecture, which are often used as part of agent systems. It must be possible to add the latest new documents and the latest versions of existing documents to the document repository used by the application. Documents containing outdated information must be removed or marked as outdated. Data currency and effective indexing of content help improve the system's information retrieval.

6.2 Components of AI agents

AI agents consist of several small components that can be combined to build systems that solve complex problems. This section discusses the components used in building an agent and explains how they can be

implemented securely. Figure 4 illustrates the different parts of an agent system.

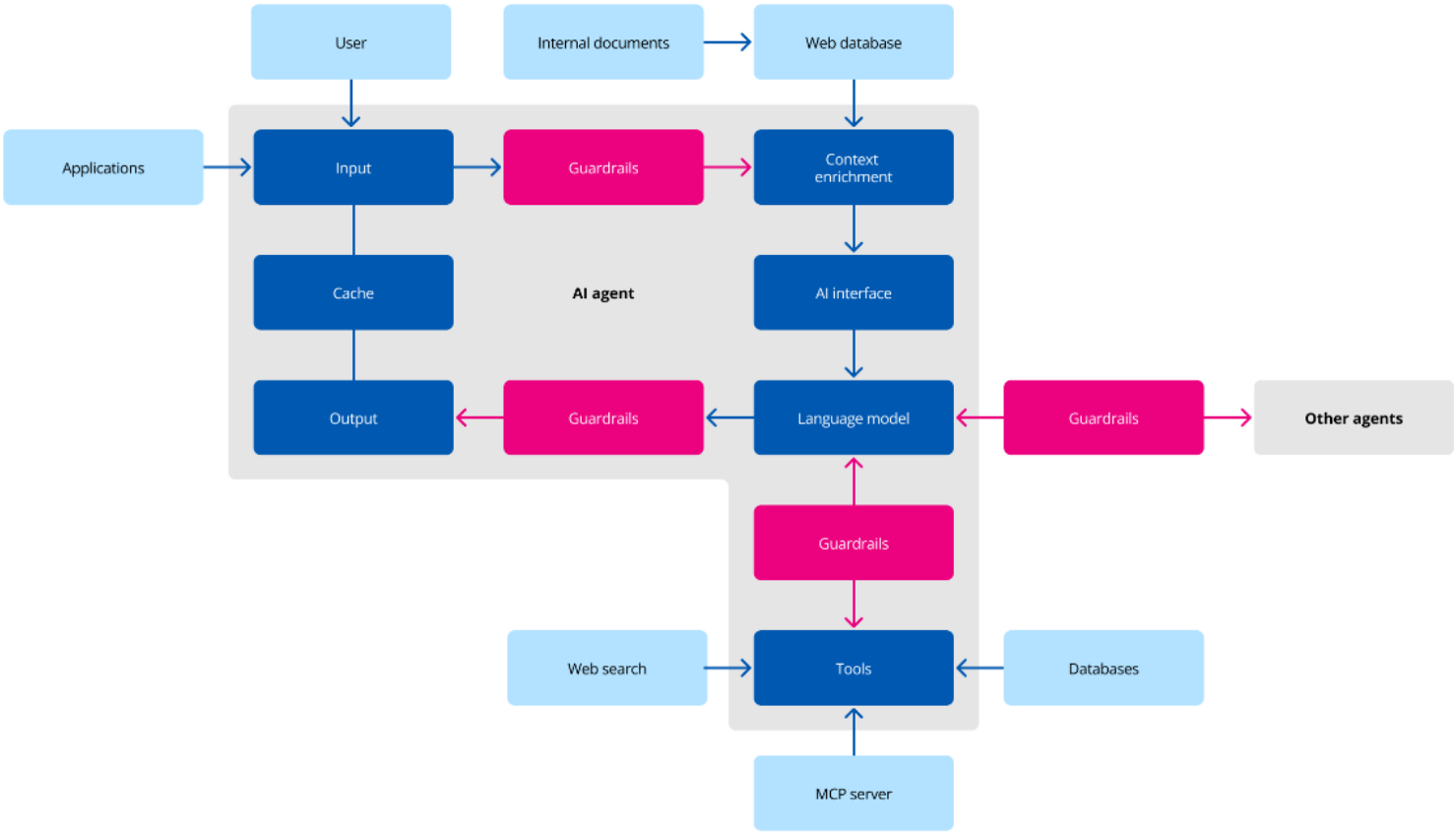

Figure 4. Components of an AI agent.

6.2.1 Foundation model: Language models as an enabler of agents

Language models enable the building of AI agents. They process user prompts consisting of unstructured text data and select the necessary actions to resolve the content of the prompt. Language models are not themselves capable of calling different tools or retrieving the material required to formulate a response. For this reason, an application must be built around the language model to carry out actions based on the language model's outputs. Development is facilitated by various application frameworks that automate this use of tools so that the application developers do not have to create and maintain the necessary processes themselves.

6.2.2 Selecting a suitable language model

A functional agent always requires an advanced language model as its foundation, trained to use tools. In addition, the model's accuracy must be assessed so that tool calls can be implemented securely. Poor accuracy weakens the security of the model, especially when the agent also has access to external systems.

Model comparison should begin with powerful models. This helps establish how well the model can perform in the intended use case at its best. Next, it can be tested whether a smaller language model can perform the same task with the same quality. At the same time, it is possible to assess where the smaller model fails. Using smaller models always saves resources, so they should be preferred where possible.[16]

Ensure that the model meets your organisation's requirements for data processing. Check, among other things, where the language model's computation is performed and how the information sent to the language model is stored. Some language models use user data to retrain models, which may breach the organisation's data policy.

6.2.3 Creating a secure prompt

The output of language models is always based on a prompt. A prompt consists of several different components, and the interfaces provided by many language model providers allow the prompt to be divided into logical components:

- **System prompt**: Defines the agent's role, context and the language model's operating principles in different situations.
- **User input**: Specifies what the language model should do. The input may be, for example, a question in a chat application or the definition of a task that launches process automation.
- **Prompt augmentation**: The results of a language model can often be improved by providing the model with context through external documents. For example, the language model's response can be improved by providing the model with user data.
- **Agent outputs**: Final outputs may include, for example, a message returned by an external system, a result returned by a calculator or the output of an application built by the language model. These outputs can be used to automatically launch new tasks.
- **History data**: The language model can be provided with a history of previous inputs and outputs, which can be used to make the agent's actions more consistent.

When creating a prompt, it is important to note that each component is equal from the language model's perspective. This means that any part of

[16] OpenAI. A practical guide to building agents. https://cdn.openai.com/business-guides-and-resources/a-practical-guide-to-building-agents.pdf

the prompt can modify the model's behaviour and, for example, enable an attacker to carry out a prompt injection.

There is no system prompt that works optimally with all models. Instead, the agent requires a prompt suited to the intended use. Prompt design is an iterative process in which different prompts are tested to determine which prompt enables the agent to perform best in its intended use. When writing instructions for the language model, attention should be paid to the following points: write clear and precise instructions, define a persona for the language model, clearly separate the different parts of the input, give the language model examples, define the format of the output and instruct the model to reason step by step.

In addition to quality, a good prompt is also a security factor. The key vulnerabilities related to prompts are prompt manipulation and system prompt leakage. Through prompt manipulation, an attacker can hide malicious content in the prompt. An attacker may also seek to reveal the system prompt in order to access any sensitive information that may have been included in it. Prompt injection can be carried out either directly through the language model prompt or indirectly, for example by including malicious commands in the agent's tool or in information retrieved from external sources.

The prompt must therefore be protected against threats. The instructions mentioned above can be supplemented with OWASP's guidance[17] on protecting prompts:

1. **Constrain model behaviour.** Instruct the agent to ignore other instructions. After reading external documents, remind the model of the agent's task so that any attacker instructions are disregarded. In addition, the language model can be instructed to respond only to certain types of input.
2. **Define and validate expected output formats.** The format defined for language model outputs can be validated by another programme.
3. **Build guardrails for the prompt and output.** For example, use another language model to check that the prompt does not contain prohibited topics and that the prompt is consistent with the agent's task (see guardrails, 6.2.5). It is important to note that using another language model for checking is not a complete guardrail.[18] The use of other methods is recommended where possible.
4. **Give the language model the least privileges possible.** Restrict the language model's permissions so that only the necessary tasks can be

[17] OWASPin 2025 Top 10 Risk & Mitigations for LLMs and Gen AI Apps: https://genai.owasp.org/llmrisk/llm01-prompt-injection/
[18] See, for example, Quanta Magazine: Cryptographers Show That AI Protections Will Always Have Holes: Cryptographers Show That AI Protections Will Always Have Holes

performed. Restricting file read permissions can, for example, prevent sensitive information disclosure.

5. **Require human approval for critical decisions.** Require human-in-the-loop for critical operations. Approval reduces the risk of unauthorised operations.
6. **Clearly separate untrusted content from the rest of the prompt.** Untrusted content refers to external documents, internet search results and the user's prompt. It must be clearly separated from the rest of the prompt so that the content of documents does not affect the language model's behaviour. Untrusted content can be separated, for example, using the delimiters mentioned in the instructions for a good prompt.
7. **Actively test the model's behaviour.** Build tests or simulations that attempt to break the model's protection. The purpose of the tests is to determine the model's limits and the actual scope of its access rights.

6.2.4 Memory and external documents

Memory in language models refers to techniques by which information is stored in the system and later used in the model's operation. For example, an AI agent uses memory to speed up task execution. The memory of a language model can be divided into three parts: 1) internal knowledge, 2) short-term memory and 3) long-term memory. Internal knowledge covers information that has been provided to the model during training. Short-term memory may cover, for example, action history. Long-term memory can be used to store and share agents' tasks across different sessions.

In addition to interactions with the AI agent, external documents can also be stored in the agent's memory. With the RAG method, the agent does not need to be retrained. Instead, the required information is provided in the agent's prompt.

During the building stage, plan how the language model's short-term and long-term memory can be protected against vulnerabilities. Use access rights definitions to ensure that only a limited group of users can read information from memory. When data is added to the agent's long-term memory, texts that could impair the agent's operation or security must be removed from it. These include, for example, time-bound expressions such as "from now on" or expressions that restrict the use of other instructions, such as "ignore the previous instructions". The sources of documents stored in memory must be verified to prevent poisoned data from being stored in the database. In addition, ensure that personal data is processed in the agent's memory in accordance with regulations.

6.2.5 Guardrails

Guardrails form a set of rules and instructions that ensure that the language model operates as intended. Each use case must have its own rules for how the model should behave in different situations. Guardrails can be implemented for both the input and the information returned by the language model, for example using the following methods:

1. **Identifying strings in text:** Words or strings that the language model should not process or generate for the user are identified in the text. These include, for example, personal data, such as personal identity codes or email addresses.
2. **Comparing the input against prohibited words:** Words or themes that the AI agent is not allowed to process can be defined in advance.
3. **Classifying the input into permitted categories using a separate model:** Text can be classified by a separate model into different categories to which the agent is allowed to provide a response.

Input safety can be improved by identifying, for example, the following types of information:

4. **Dangerous or harmful content:** Identifying dangerous or harmful input that attempts, for example, to find vulnerabilities in the language model, steal the system prompt or generate harmful content.
5. **Personal data:** Before calling the language model, personal data can, for example, be masked from the input If personal data is needed in the final response, it can be added to the text generated by the model.

Seemingly harmless input may potentially cause the model to return information that should not be shown to the user. When reviewing the returned value, the following may be taken into account, for example:

6. **Content, personal data or harmful information:** Guardrails must be implemented so that the content corresponds to the user’s query and does not contain harmful content defined by the language model developer, such as disinformation.
7. **Hallucination:** Hallucination is a common problem in language models. The final output can, for example, be compared against the input and external documents to verify the accuracy of the information.
8. **Syntax:** The language model’s final output can be validated against a specific programming language, for example. The output can be checked, for example, for compliance with JSON syntax.

Specialised models, such as Llama Guard[19] and Constitutional Classifiers[20], have also been trained to protect language models.

It is important to note that language models are not fully protected, even when various safeguards are built around them. AI systems require monitoring so that harmful use can be detected as quickly as possible.

6.2.6 Tools: Data, actions and orchestration

The language model is given a list of tools in the prompt that it can use to solve the task it has been given. The use of tools can improve the model's accuracy in the environment defined for it. Selecting the right tool to solve a problem is challenging, especially if the number of tools is large. The tool description must clearly indicate what the tool can be used for and what type of information should be provided to it.

To enable the agent to use tools securely, it must be ensured in the selection process what the tool does, what information its use requires and where the tool's computation is performed. Examples of measures for managing risks related to the tools used by AI agents are listed below.

1. **Executing programs:** The content of an application generated by an AI agent must be verified and the application must be executed in a restricted environment.
2. **Restricting access rights:** Access rights for tools must be restricted as much as possible. For example, only limited rights should be granted to a database so that the agent can retrieve or write only the necessary data from or to the database.
3. **Replacing a verified tool with a malicious one:** Some tools may run on external servers (see, for example, MCP server, section 6.2.8). In such cases, the content of a previously safe tool can be modified to become malicious. For example, new functionalities may be added to the tool that send the data processed by the tool by email to the person who modified the tool. When using a tool, the hash value of the tool can be checked, for example.

6.2.7 Tools and agent autonomy

An autonomous agent is vulnerable to goal manipulation. Through prompt injection, an attacker may attempt to modify the agent's planning or

[19] Inan et al. (2023), Llama Guard: LLM-based input-output safeguard for human-AI conversations, https://ai.meta.com/research/publications/llama-guard-llm-based-input-output-safeguard-for-human-ai-conversations/
[20] Sharma et al. (2025), Constitutional Classifiers: Defending against universal jailbreaks across thousands of hours of red teaming, https://arxiv.org/pdf/2501.18837

reasoning in order to carry out malicious actions, such as phishing for customer data from the system.

The following measures are essential for protecting the task chains of an agent system:

- Restrict access rights as narrowly as possible so that the agent cannot take actions outside the given task (see 3.2.6).
- Limit the availability of tools to only those that the agent needs to perform the task.
- Monitor AI agents to identify changes in goals and behaviour.
- Set limits on how many different agents can be called (see 3.2.10).
- Require human approval for critical actions.

6.2.8 Managing tools and agents: MCP protocol

Maintaining separate tools for each AI agent can be burdensome in the long term. Various protocols have been proposed as a solution, the best known of which are Anthropic's Model Context Protocol (MCP) and Google's Agent2Agent (A2A). The MCP protocol has already become established in use, so this guidance examines the related security issues in more detail.

MCP is an open-source standard intended to create a clear common operating model for calling tools. It enables AI applications to be connected to external systems. It can be thought of as functioning like a USB-C cable: a unified and standardised way to connect several different services, which speeds up the development of AI agents and makes it easier to share tools. MCP functions can be executed either locally or on an external server.

The MCP architecture consists of three main parts:

- The **MCP host** is the AI application that uses tools to enrich context.
- The **MCP client** maintains the connection to the server and transmits information between the parties. Each MCP client communicates with only one MCP server.
- The **MCP server** provides the client with context and functionalities. The server consists of three parts: tools, resources and prompts.

The MCP protocol creates new opportunities for building tools. At the same time, it introduces a new vulnerability into the AI agent. If you use the MCP protocol in your agent solution, take into account at least the following challenges:

- **Confused deputy problem:** An attacker may use MCP servers as proxies if the MCP client identifier has been defined as static.
- **Token passthrough:** The MCP server accepts a token from the MCP client without verification and passes the token on to the interfaces in use.
- **Session hijacking:** An attacker may gain control of the MCP server by stealing a statically defined token that the MCP server has assigned to the MCP client.
- **Local MCP server compromise:** An MCP server provided by a third party may contain commands that steal or destroy user data. When installing a new server, its functionalities must be verified and access rights restricted.
- **Scope minimisation:** Leakage of overly broad tokens may lead to misuse of data or privilege chaining. It is recommended to avoid the use of wildcards (*).

MCP is still a new technology and is developing rapidly. More information about MCP threats and how to protect against them is available, for example, on MCP's own website[21].

6.2.9 Multi-agent system

Multi-agent systems may have rights to several external systems and to encrypted information at different levels. For this reason, it is important to ensure the protection of communication between agents. If the system has access to confidential information, agents may share this information with one another. The risk of possible leaks must therefore be assessed with regard to communication between agents, and the necessary protections must be implemented, such as authentication and encryption of messages between agents.

The principles of data minimisation, compartmentalisation and zero trust should be followed in multi-agent systems that process confidential information. Agents should be given access only to the sources required by the role defined for them. Particular attention should be paid to access rights and data minimisation in situations that require the processing of both confidential and public materials. For example, retrieving public information can be assigned to an agent defined for that task that does not have access to confidential material. Such an agent should also be defined to deliver its results to a restricted environment to which agents processing

[21] MCP security best practices: https://modelcontextprotocol.io/specification/draft/basic/security_best_practices

confidential material are given read-only rights. The information transmitted can also be monitored, sanitised and anonymised as needed.

The operation of agents should also be monitored through logs. The logs, in turn, should be monitored by auditing them from the perspectives of efficiency and information security. Information should be recorded in the logs in a format that enables malfunctioning agents to be identified efficiently. For example, if an agent suddenly starts calling new tools or defines an unknown goal for itself, the agent must be identified and isolated from the network and systems.

The actions of agents performing tasks that are critical to core business must be monitored particularly closely. Monitoring can be carried out by a human or by other agents (see Monitoring and anomaly management, 7.3.6). For example, actions may require consensus between several agents before they are performed. These means ensure that the agent system is not vulnerable to attacks or fault situations affecting individual agents.

6.2.10 Scalability

New agents can be added to and removed from a multi-agent system relatively easily. The system can be scaled either by duplicating already defined agent roles or by breaking one agent's task down into parts. Scalability offers the greatest benefit if the task flow can be defined so that tasks can also be performed in parallel. The risk related to scalability is excessive resource use if the agents in the agent system can add new agents to the system without supervision. This may result in enormous costs or slow down other users' tasks.

6.2.11 Fault tolerance

Clear objectives should be defined for the quality and execution time of the tasks performed by the agent system. Recovery from exceptional situations should be enabled as effectively as possible. However, if recovery is not possible, it is important to identify the situation in good time. It must be possible to interrupt the task if necessary and release the resources used by the agents. It must also be possible to write an error log of the actions.

Task-specific real-time requirements and restrictions, such as limits on the number of retries, can be defined for agents. If an agent does not complete its task, the party carrying out monitoring must decide whether the task can be continued or whether it must be interrupted. Monitoring can be carried out by a human through various logs or by using independent language models to assess logs (see LLM-as-a-judge, 7.3.1). Depending on the defined task, it may be possible to continue the workflow despite the failure of an individual agent. For example, a multi-agent system may continue operating based on the outputs produced by other agents even

when one agent's information retrieval produces no results. In such cases, information that one source is missing is included in the result.

An alternative workflow can also be defined for the system. For example, result refinement rounds may be skipped and less complete results may also be accepted. However, it is not advisable to enable bypassing a security check or harmful content detection. When designing the workflow of a multi-agent system, it is advisable to identify the points at which it is possible to continue and when the workflow must be interrupted. Operations in fault situations can be defined either in the agent system's static rules, in the system prompt of the monitoring agent or in both.

If the agent system performs an iterative task, such as preparing a report, several drafting and review rounds are usually needed. Numerical limits should be set for iteration rounds to manage fault tolerance, execution time and costs.

6.2.12 Information security as part of agent operations (MLOps)

Machine Learning Operations is a comprehensive approach that brings machine learning and AI models into companies' production environments in a controlled and efficient manner. Models, prompts and code are systematically versioned, which facilitates reproducibility and makes it possible to return to a previous, functioning version. Auditing and reviews ensure quality and reliability. All changes are documented and assessed before deployment, increasing transparency and security. In addition, the external components used in the application must be locked to a version that has been validated as secure so that automatic component updates do not cause information security vulnerabilities.

Automation is at the core of MLOps. The different stages of development, such as data loading and processing, model training and publication, are automated. This improves consistency, reproducibility and scalability. Model training and publication can be triggered, for example, on the basis of data or code changes and schedules. Automated tests reveal errors at an early stage. Managing infrastructure as code speeds up releases and makes maintenance easier.

MLOps must be embedded in the organisation's operating culture. Close cooperation between data scientists, data engineers and the business is a prerequisite for a successful project. Clear documentation, functioning communication channels and systematic collection of feedback on model performance create the basis for controlled development. Particular attention is paid to the management of sensitive data and restricting access rights only to those who have a genuine need for them. MLOps also helps ensure compliance with regulatory and information security requirements.

Because AI agents operate autonomously, monitoring plays a key role in terms of transparency. The risks in autonomous decision-making include compliance violations when regulatory compliance cannot be verified, operational disruptions when there is no visibility into decision-making or errors, and erosion of trust when the actions of agents cannot be explained, especially in critical decision-making. Monitoring enables organisations to gain control over the operation and performance of agents. More information on monitoring is provided in connection with the testing guidance (see section 6.3.6).

7 Testing

The technology, models and data used in AI agents are changing and developing rapidly. AI agents also often operate dynamically. For these reasons as well, one-off acceptance testing alone is not sufficient. AI agents must be tested regularly.

In AI testing, the special characteristics of AI must be taken into account in addition to traditional information security. The components of agent systems are also developing continuously, which increases the importance of testing. In addition to finding vulnerabilities and insecure configurations, AI solutions also require examination of the model's behaviour and the reliability of outputs. This chapter describes the testing of AI agents from the perspective of information security. The guidance is based on documentation from the OWASP AI Testing Guide project[22].

7.1 Principles of AI testing

Testing that extends to the different stages of the entire system life cycle can ensure that AI systems remain secure and reliable. This ensures the safety of the agent's operation, the reliability of outputs and the appropriate functioning of the model. In addition, the agent's operation must be monitored continuously to ensure its reliability and performance.

The autonomous decision-making of agents, cooperation with other agents, users and tools, and continuous learning also pose challenges for testing. The threats and attack vectors identified in threat modeling are used in selecting test cases so that the identified risks and the controls planned for them can be tested proactively. Testing should also always be carried out when changes occur in the system, for example when the model or interface changes, or updates are made to the agent.

[22] OWASP AI Testing GUIDE: https://owasp.org/www-project-ai-testing-guide/

OWASP has developed the Artificial Intelligence Security Verification Standard (AISVS)[23] for assessing and verifying the security and ethical aspects of AI-based applications. It is a checklist modelled on existing standards, such as the OWASP Application Security Verification Standard (ASVS), which is a widely used tool for the secure development and testing of web applications. AISVS provides a systematic approach to testing AI applications.

According to OWASP's testing guidance, effective testing is based on four areas:

1. **Security**
2. **Privacy**
3. **Responsible AI**
4. **Trustworthiness**

By examining these together, the risks associated with the deployment of AI can be managed comprehensively. We begin by examining the key factors in testing for each area.

7.1.1 Security

Information security testing ensures that the AI agent is protected as effectively as possible against unauthorised use and different types of attacks. Testing makes use of the risks identified in threat modeling and known frameworks and testing guidance, such as the OWASP AISVS mentioned above. AISVS includes classified requirements for areas such as validating user inputs and assessing model security. The tool focuses particularly on areas of application development.

In the development of AI agents, testing focuses in particular on protecting system prompts, instructions and user inputs against poisoning and manipulation. In addition, the AI agent's infrastructure, components and integration points are tested to ensure the security of the overall system they form. Supply chain risks may be caused by components, functions or other dependencies produced by third parties. OWASP AISVS also addresses these areas in more detail.

7.1.2 Privacy

Privacy involves two key areas: data protection and access management. The operation of AI is typically based on diverse information, which means that the security of personal data processing must be verified through sufficient testing. Test cases focus in particular on testing data leakage and

[23] OWASP Artificial Intelligence Security Verification Standard (AISVS): https://github.com/OWASP/AISVS/tree/main

model behaviour to ensure that personal data is processed in accordance with legislation.

Access management testing ensures that the agent has not been granted excessive permissions, or rights to systems that it does not need for its tasks. The objective of threat actors may be to misuse the agent solution to disclose sensitive information. When testing the model's behaviour, the assessment examines whether the model's operation reveals information that could compromise the model's operation or cause a data breach.

7.1.3 Responsible AI

Ethics and fairness must be taken into account when testing agents. Testing seeks to reveal harmful biases in outputs, which requires testing the model's behaviour with sufficiently comprehensive datasets. To detect false or incorrect information, or misinformation, it is necessary to examine how the model processes harmful content, such as hate speech. Testing must also assess guardrail coverage by testing mechanisms that guide safety and are central to core business, and whether they can be bypassed in some way. This can be carried out through red teaming[24], in which hostile activity against the target being tested is simulated to reveal vulnerabilities.

7.1.4 Trustworthiness

The trustworthiness of AI systems is based on the explainability of decision-making, especially regarding critical outputs. The responses model give must be tested to detect variation and unexpected behaviour. In addition, runtime monitoring should be carried out to identify anomalies and detect hallucination.

When discussing closed-source solutions from large companies, such as OpenAI's GPT models or Google's Gemini, one risk is that the internal operation of the model cannot be inspected or modified. Their advantage is the high performance and built-in security made possible by extensive development resources. In open-source solutions, such as Mistral Large and LangChain, the contents of the model can be inspected and modified more effectively. However, this requires technical AI expertise and development resources.[25]

Compliance is also an essential aspect of trustworthiness. The compliance of AI systems must be ensured by complying with laws, regulations and

[24] OWASP GenAI Red Teaming Guide: https://genai.owasp.org/resource/genai-red-teaming-guide/
[25] Saumya Patel (2025), The LLM Divide: Open Source vs. Closed Source — Choosing Your AI Champion: https://medium.com/@psaumya567/the-llm-divide-open-source-vs-closed-source-choosing-your-ai-champion-ad44c9893316

industry standards. OWASP's testing guidance recommends following well-known frameworks also in testing, such as the NIST AI RMF (Risk Management Framework)[26] or the ISO 42001 Artificial intelligence – management system standard[27], It should also be ensured whether the requirements of the EU AI Act apply to the agent system in question and that the necessary testing is carried out.

Measures related to all four areas – security, privacy, responsible AI and trustworthiness – extend to all layers of the AI architecture. In the next subsection, we examine how this layered structure should be considered as part of testing.

7.2 Testing across the layers of AI architecture

The OWASP AI Testing Guide project defines comprehensive, structured methods and best practices for testing AI systems across the different layers of architecture. The project brings together test cases from traditional cybersecurity, MLOps testing and responsible AI into a structured framework. The test cases are classified into four levels based on a typical AI architecture: application, model, infrastructure and data.

At the application layer, risks related to the use of the application and risks that emerge through interaction are tested. Application-level test cases include, for example, direct and indirect prompt injection and the leakage of sensitive data, inputs or system prompts. Each test case contributes to the comprehensive information security of AI systems by addressing application-level risks. More test cases are available in the project documentation: www-project-ai-testing-guide/Document/content/3.1_AI_Application_Testing.md at main · OWASP/www-project-ai-testing-guide · GitHub.

At the model layer, the model's resilience, vulnerabilities, properties and behaviour are tested. It is also verified that the model operates in accordance with the objectives. At the model level, testing focuses particularly on the inherent properties of AI models. The test cases are particularly related to poisoning, such as runtime model poisoning, or privacy, such as model inversion attacks, in which sensitive training data is reconstructed from the model's outputs. More model-level test cases: www-project-ai-testing-guide/Document/content/3.2_AI_Model_Testing.md at main · OWASP/www-project-ai-testing-guide · GitHub.

[26] The NIST AI Risk Management Framework (AI RMF): https://www.nist.gov/itl/ai-risk-management-framework
[27] ISO/IEC 42001:2023 Information technology — Artificial intelligence — Management system: https://www.iso.org/standard/42001

Infrastructure testing focuses on vulnerabilities and risks in the technical infrastructure and components that support the operation of the AI agent. Testing takes into account model supply chains, resource management, secure configurations and protection against misuse and exploitation. At the infrastructure level, test cases relate to supply chain tampering, resource exhaustion and model theft during the development phase (dev-time model theft). More infrastructure-related test cases: www-project-ai-testing-guide/Document/content/3.3_AI_Infrastructure_Testing.md at main · OWASP/www-project-ai-testing-guide · GitHub

Data testing focuses on validating and protecting the data used during the AI agent's life cycle. Data testing seeks to ensure data quality and privacy protection, sufficient data coverage and the prevention of the influence of harmful and inappropriate content. Data-related vulnerabilities often have wide-ranging impacts. The test cases examine, among other things, training data exposure and harmful content in datasets. More test cases: www-project-ai-testing-guide/Document/content/3.4_AI_Data_Testing.md at main · OWASP/www-project-ai-testing-guide · GitHub.

At each level of the architecture, the test cases are linked to the areas of AI testing discussed earlier in section 7.1: security, privacy, responsible AI and trustworthiness. In some cases, it may be justified to focus the test cases on a specific area, such as security or the trustworthiness of AI.

7.3 Testing AI agents

The testing of AI agents follows the traditional stages of information security testing. First, it is important to define the objectives, scope and criteria that affect the selection of the applicable test cases so that the system under test is well understood. The risks identified in threat modeling are used to define attack scenarios and test cases. After the tests have been performed, the findings and the risks they create are assessed and prioritised so that the required corrective measures can be targeted appropriately.

Testing focuses, among other things, on performance, adaptation to changes, reliability and security. In an effective testing process, experts' overall understanding of the target being tested is combined with automated testing (see section 7.3.1). Automated tools can be used to help generate many different test cases in bulk. Expert insight and guidance are needed particularly in complex use cases so that the test scenarios also take into account test data and test cases tailored to the use case.

The non-deterministic behaviour of agents poses challenges for testing, as there is no single correct target response. Unlike in traditional applications, the outputs of AI agents may vary depending on how the input is formulated, the conversation history or updates to the model itself. Testing must account for a range of accepted outputs and consider both correctness and coherence. AI can be used to create automated test cases based on real-world scenarios, which can be supplemented with manually selected inputs. Detecting anomalies may require comparing the results of several testing rounds to identify possible problems.

Because interactivity is characteristic of agents, static test cases alone are not sufficient. Testing strategies must represent real-life usage patterns more closely than traditional quality assurance. Real-world scenarios must be simulated as test cases, including routine cases and edge cases. In addition, the AI's ability to handle ambiguous or evolving scenarios must be assessed, as must the responsibility of agents regarding bias, misinformation or other harmful content. [28]

7.3.1 Hybrid testing

Effective testing often requires a hybrid model that combines automation with expert insight. Automated testing speeds up the assessment process when large volumes of AI outputs need to be evaluated. The use of experts, in turn, helps form an overall picture, especially in complex use cases. Experts ensure that the outputs are aligned with the use case. It is also advisable to assess the operation of the AI agent through direct user testing. Diverse user feedback helps identify usability-related issues.

Language models can also be used to evaluate other AI systems using the LLM-as-a-judge method. This approach can be used to complement expert analysis while taking advantage of the scalability of language models. In this method, a language model that is independent of the AI agent evaluates how well the agent has responded to the user's input. The evaluating language model is given a prompt that describes the task given by the user and the language model's output. In addition, the prompt should provide the criteria against which the language model assesses the agent's output. The final output can be assessed, for example, in terms of accuracy, usefulness or whether the tone of the text corresponds to the original instructions. The output of the evaluating language model may consist of several measurable metrics and a rationale explaining why these scores were assigned. By comparing the input, the final output and external sources with one another, it is possible, for example, to assess whether the

[28] Rashi Chandra (2025), Testing Your AI Agent: 6 Strategies That Definitely Work: https://insights.daffodilsw.com/blog/testing-your-ai-agent-6-strategies-that-definitely-work

agent has hallucinated. Examples of ready-made models for content detection include Llama Guard[29] and Constitutional Classifiers[30].

Similarly, large language models can also be used to generate test attacks as part of red teaming. Such tools include Garak atkgen: Attack Generation[31] and LLAMATOR Core[32]. Garak, the Generative AI Red-teaming & Assessment Kit, is a vulnerability scanner for large language models that can be used to identify and classify vulnerabilities. Garak atkgen is an automated attack simulator whose probe contains a separate model that dynamically generates attack prompts with the aim of causing the language model to enter a failure mode. LLAMATOR Core, in turn, is a Python framework for testing chatbots and generative AI systems. It includes a wide range of attacks targeting large language models, RAG architectures and agents. It also enables the use of custom attacks and datasets.

7.3.2 Datasets

Testing should use diverse datasets covering scenarios that are relevant to the use case. The advantage of benchmark datasets is that they make it possible to compare performance against other agent solutions. Benchmark datasets may include sample code and standardised evaluation scripts designed to make testing repeatable and transparent. Another advantage of such datasets is that they have been validated by a wider community. The sufficiency of off-the-shelf datasets should be assessed on the basis of the AI agent’s intended purpose and operating conditions.

Custom datasets are best suited to the intended purpose and therefore provide more accurate answers than public benchmarks. Creating custom datasets requires the purpose of the dataset to be precisely defined. Raw data may be created by humans or collected automatically, for example from the public internet or through interfaces provided by different platforms. Language models or generative AI can also be used to create raw data. Raw data must be processed and cleaned before use. In addition, metadata must be added to the dataset so that the model can learn from it. Finally, the data must be structured and stored so that it is easy to use.

Combining standardised and use case-specific custom datasets provides a comprehensive view of the agent’s performance[33].

[29] Inan ym., 2023, https://doi.org/10.48550/arXiv.2312.06674
[30] Sharma ym. 2025, https://doi.org/10.48550/arXiv.2501.18837
[31] atkgen: Attack Generation: https://reference.garak.ai/en/latest/garak.probes.atkgen.html
[32] LLAMATOR: https://github.com/LLAMATOR-Core/llamator
[33] Conor Bronsdon (2025), How to Test AI Agents Effectively: https://galileo.ai/learn/test-ai-agents

More information on datasets is available, for example, from the following sources:

- 25 AI benchmarks: examples of AI models evaluation[34]
- Google: Academic benchmarks to evaluate responsibility metrics[35]
- Hugging face: The AI community building the future[36]

7.3.3 Adversarial testing

Adversarial testing seeks to identify ways to break the operation of an agent or model and bypass guardrails. The purpose of the test scenarios is to strengthen the agent's defences against real-world attacks and vulnerabilities. The AI agent should be tested with harmful content, and it should also be taken into account that seemingly harmless inputs may reveal vulnerabilities or weaknesses in the agent's operation. The aim is to identify ways to secure the agent against the identified threats.

The test scenarios must represent real-world use cases, including edge cases. When testing agents, it is necessary to assess not only the functionality of features, but also the quality of operation. Traditional testing and quality assurance must adapt to the requirements arising from the nature of AI agents. Ethics, accuracy, safety and resilience to stress must be assessed.

Stress testing measures robustness assesses the agent's error-handling capability and tests its ability to operate in unpredictable situations, such as when source materials conflict with one another. Tools suitable for AI security testing are presented in section 7.3.1. Testing tools are presented in Appendix 3. These tools help in designing attack scenarios, carrying out red teaming and conducting risk assessment.

The Agentic AI Red Teaming Guide[37], which focuses on testing AI agents, has been developed in cooperation between the Cloud Security Alliance (CSA) and the OWASP AI Exchange. The guide discusses the security challenges, testing requirements, operational steps and example prompts for agentic AI. The OWASP GenAI Red Teaming Guide[38] can also be used

[34] 25 AI benchmarks: examples of AI models evaluation (2025): https://www.evidentlyai.com/blog/ai-benchmarks
[35] Google AI for Developers, Academic benchmarks to evaluate responsibility metrics: https://ai.google.dev/responsible/docs/evaluation#benchmarks
[36] The AI community building the future: https://huggingface.co/
[37] Cloud Security Alliance (CSA), Agentic AI Red Teaming Guide (2025): https://cloudsecurityalliance.org/artifacts/agentic-ai-red-teaming-guide
[38] OWASP GenAI Red Teaming Guide: https://genai.owasp.org/resource/genai-red-teaming-guide/

when planning testing, as it brings together red teaming principles, checklists and various techniques.

7.3.4 Simulation testing

Simulated tests provide an understanding of how the AI agent operates under different conditions. Behaviour in real-world situations must also be assessed, for example through small-scale deployment testing. Combining simulated and real-world tests increases the likelihood that the AI agent will operate correctly in all situations. For example, the τ-bench framework[39] is available for testing interactions involving AI agents. It can be used to simulate dynamic conversations between the user and the agent, helping to assess the agent's performance and reliability.

7.3.5 Regression testing

Updates and fine-tuning in particular may change the behaviour of agents. To maintain the stable operation of AI agents, automated regression testing should be performed, especially in connection with model updates. Regular schedules should be defined for regression testing so that even minor deviations in performance or accuracy can be identified.[40]

7.3.6 Monitoring and anomaly management

Monitoring AI agents is important because AI agents learn and may adapt as data and users change. Continuous monitoring is one way to ensure that the operation of agents remains consistent with their intended purpose. Operations must be assessed by comparing outputs against validation baselines to detect anomalies. Test performance must be monitored and outputs assessed to detect incorrect operating patterns.

Due to the nature of agents, attention must also be paid to ensuring that test cases are updated to reflect the agent's current state. The agent's state is not permanent – the agent is improved and fine-tuned, which means that test cases must also keep up with the change. The current state of the agent's operation and outputs should be documented so that longer-term changes can also be detected.

To ensure trustworthiness, the operation, decision-making, performance and interaction of an agent in production with external actors and users must be monitored. Particular attention should also be paid to metrics that can reveal misuse of the AI application. For example, metrics on application usage volumes, called tools, agent goals/tasks or guardrails (see *6.2.5*)

---

[39] Yao ym. (2024), τ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains: https://arxiv.org/abs/2406.12045

[40] Rashi Chandra (2025), Testing Your AI Agent: 6 Strategies That Definitely Work: https://insights.daffodilsw.com/blog/testing-your-ai-agent-6-strategies-that-definitely-work

may reveal misuse of the agent. Monitoring can also be used to track the consistency of agents' decisions, as an attack on the agent system may be revealed if actions that were previously rejected are suddenly approved.

Anomalies in an agent's operation may be faults or biases in the system caused by distorted algorithms, or malicious activity such as a prompt attack. In both cases, it is essential to detect anomalies as quickly as possible so that they can be responded to and the agent's incorrect or malicious operation can be prevented. Automated monitoring, algorithms suitable for anomaly detection and user reporting can be used for detection.

Especially in multi-agent systems, monitoring can be used to trace faults in complex systems to a specific component or function. Although investigating anomalies is often still manual work, automation can be used to collect and analyse data.[41]

In some cases, it may be necessary to take immediate restrictive measures, such as disconnecting the system from the network or restricting the use of sensitive data. Measures to correct the anomaly may include software or algorithm updates, temporary suspension of use or the introduction of workarounds. The key in managing AI anomalies is to anticipate and identify potential anomalies, prepare response measures for them and ensure continuous monitoring of the system so that anomalies can be detected. Anomaly management must always be tailored to the organisation's AI systems and potential threat scenarios.

7.4 Testing tools

A wide range of AI testing tools are available for different purposes. Tools listed and curated by OWASP AI Exchange have been compiled by use case in Appendix 3. The list represents the current situation, and new tools and methods are constantly emerging. The most up-to-date list should be checked in the AI Exchange documentation[42].

[41] Lindemulder ym., Why observability is essential for AI agents: https://www.ibm.com/think/insights/ai-agent-observability
[42] OWASP AI Exchange 5. AI security testing: https://owaspai.org/goto/testing/

Appendix 1: Procurement guidance for agent systems

In this appendix, we provide a checklist for procuring an AI agent, covering matters that the buyer should verify with the agent supplier. There are many criteria, and it is not always necessary to assume that every system will meet all of them. The most important points should therefore be selected from the list based on the application, the task, the protection of core business and the organisation's objectives.

The purpose of this guidance is to complement organisations' existing security guidelines and policies. It is important that the AI agents being procured are also assessed in accordance with existing guidelines on software security.

Governance and life cycle maintenance

☐ Check which language models the supplier offers and that they are suitable for the use case.

- Language models are available from several different suppliers. How broad is the provider's offering? What is the provider's release delay for models?

☐ Check how openly the provider enables agents to be exported to other systems.

- The prompt, own data and specifications are an important part of developing AI agents. Verifying how data can be retrieved from the application makes it easier to switch to a new service provider if necessary.

☐ Ensure that the system to be procured enables life cycle management as language models are updated.

- Can the same language model be used for at least one or several of the organisation's own release cycles? Is there time to test the new model as the current one becomes obsolete?
- What kind of update path does the supplier offer to a new language model? Which functionalities or capabilities of the new model can be adopted directly, and which require changes to the application, system or agent using the models?
- Is it possible to revert to an earlier model or release if necessary?
- How is it ensured that the language model is available in the desired geographical region?

☐ Also ensure that the delivery includes:

- ModelOps/agent version control.
- CI/CD pipelines for secure deployment and rollback.
- Dashboards for monitoring system status and regulatory compliance.
- The possibility to enforce governance rules, such as escalation thresholds.
- Documentation of the supplier's update and patching policy.

- Evidence of continuous participation in research, development and standardisation.

Functionality and integrations

☐ Check that the application provides the possibility to store the organisation's own documents.

☐ Check that the application provides a suitable architecture for processing data.

- Does the application support RAG architecture?
- Which embedding models are in use?

☐ Ensure that the agent can be customised if necessary.

- How can agents be modified?
- Can the prompt, tools and tasks be defined easily?

☐ Check whether the implementation includes a ready-made service for testing agents in several different environments.

☐ Ensure appropriate monitoring of the agent.

- How does the service provider handle monitoring?
- Can the components to be monitored be modified?
- In addition to monitoring, can alerts or notifications be sent, for example by email?

☐ Check whether the service supports multi-agent systems, especially if the use case requires this now or may require it in the future.

☐ Ensure that the necessary tools can be connected to the agent.

- Is it possible to add the organisation's own tools for AI agents to the system?
- Does the supplier support the latest protocols, such as MCP and A2A?
- How are new tools added to the system?
- Who "owns" the new tools?

☐ Check that the provider has ready-made or customisable interfaces (APIs) also for the company's system integrations.

- Can the system be integrated with the cybersecurity solutions used by your organisation, such as SIEM?

☐ Check that the provider has ensured support for standards, such as FIPA ACL and RESTful APIs.

☐ Ensure that the agent can also be integrated with legacy systems if necessary.

Security

☐ Ensure secure user management.

- Does the application enable user management?
- How are access rights to different types of data ensured?
- Can use of the system be restricted by IP address?
- Verification of agent identities, such as IAM integration and RBAC.

☐ Ensure that the management and security of tools and integrations have been taken into account in the use of tools by AI agents. Without precise validation of tools, agents may perform unwanted tasks, such as sending company data to external systems.

- Where are the tools executed?
- How has the security of the tools been ensured?
- Can human confirmation be required between each tool call?
- Are the descriptions and functionalities of the tools available to users?
- Can the AI agent be connected to tools outside the application, such as the organisation's own MCP servers?

☐ Ensure that the agent includes sufficient guardrails for validating input and the returned value.

- Does the application include checks for the input and the returned value?
- Can custom validations be built into the system?

☐ Also ensure the following guardrails:

- Zero trust communication, including mutual authentication and TLS encryption.
- A security gateway/sandbox for external tool and API calls.
- The supplier has sufficient threat modeling practices for risks such as prompt injection, identity spoofing and data poisoning.
- Monitoring and anomaly detection for suspicious agent behaviour.

☐ You may also check whether the supplier supports trust or reputation scoring for agents.

Data protection and regulation

☐ Ensure that the provider has taken into account at least the following matters concerning data protection:

- Data minimisation and classification between agents.
- Encryption at rest and in transit.

- GDPR/HIPAA, and other regulation corresponding to the organisation and the intended purpose, built in as far as possible.
- Definable retention and deletion policies.
- The supplier complies with the required frameworks, such as NIST AI RMF, ISO 42001 or AI TRiSM.

Transparency and auditability

☐ Ensure that the agent includes a global audit log of all agent actions and interactions.

☐ Check that the agent solution provides sufficient tools for ensuring explainability, such as decision chains and rationale logs.

☐ Ensure that the system supports human oversight and control in real-time operation.

☐ Ensure that the supplier can demonstrate cooperation metrics, such as a Component Synergy Score.

Other matters

☐ Ensure the following matters concerning billing:

- How does billing for the application work? AI agents require a great deal of computing power, so ensure what is billed and how.
- How predictable are the costs? Changes and unevenness in the agent's use affect the distribution and predictability of costs.

☐ Also check whether your organisation has other (AI) governance processes that you must consider so that the procured agent solution does not jeopardise core business operations. Examples of sector-specific matters to be taken into account include:

- Security regulation concerning critical infrastructure
- Ethical principles in health care and regulation concerning medical devices
- Transparency and lawfulness required of democratic processes, including the requirements of the Administrative Procedure Act concerning automated decision-making

Appendix 2: Threat modeling template

Threat modeling was carried out on <date> for <name of service>. The purpose of threat modeling is to identify and document the system's critical data, interfaces, external dependencies and data flows. Based on the understanding gained of the system, it is possible to assess what potential threats the system is exposed to, how likely they are and what impacts would result if the threats materialised. The possible controls proposed for the threats identified in threat modeling are implemented starting with the threats that have received higher priority. After implementation, it is advisable to verify the functioning of technical controls through practical testing.

Table 1. Workshops

Workshop <dd Month yyyy>	
<Owner company>	<name>
<Supplier company>	<name>
<Consulting company>	<name>
Workshop <dd Month yyyy>	
<Owner company>	<name>
<Supplier company>	<name>
<Consulting company>	<name>
Workshop <dd Month yyyy>	
<Owner company>	<name>
<Supplier company>	<name>
<Consulting company>	<name>

Target

Description of the target

- Intended purpose
- High-level description of operation
- Technologies used

Data, assets and functions to be protected

Table 2. Assets to be protected

Name	Description

User levels and permissions

Table 3. User levels

User level	Permissions

Dependencies

Table 4. Dependencies

Dependency	Description	Maintaining party

Data flow diagram

An illustration of data flows showing the trust boundaries of the overall system and the data flows that describe which components communicate with one another.

Threats

Table 5. Threats

Threat	Description

Threat management

Table 6. Likelihood

Likelihood	
Almost certain	The threat may materialise repeatedly.
Likely	Materialisation is moderately common.
Possible	Has concretely materialised in our organisation or elsewhere.
Unlikely	Materialisation is theoretically possible, but in practice extremely rare.

Table 7. Impact

Impact	
Critical	Materialisation leads to the shutdown of the entire operating process, and the data contained in the system may be unrecoverable. Recovery and return to normal operating processes may take days or even weeks. Materialisation of a critical threat can often lead to an operational crisis. Threats affecting human lives are always critical.
Significant	Materialisation may lead to a crisis or have considerable financial impacts and/or impacts on the company's image. Potential to affect a large number of users/data.
Moderate	Materialisation may lead to limited compromise of data or company operations. The target system may be temporarily unavailable. It affects a moderate amount of users/data and may indirectly lead to a deterioration in security.
Minor	Materialisation may cause a minor deterioration in usability for a small group, but the operation of the application is not compromised.

Formula for calculating the magnitude of threats:

likelihood x impact = risk category

Table 8. Risk categories

Risk categories				
Critical	**Significant**	**Intolerable**	**Intolerable**	**Intolerable**
Significant	**Requires attention**	**Significant**	**Significant**	**Intolerable**
Moderate	**Requires attention**	**Requires attention**	**Significant**	**Significant**
Minor	**Minor**	**Requires attention**	**Requires attention**	**Significant**
Impact / Likelihood	**Unlikely**	**Possible**	**Likely**	**Almost certain**

Intolerable risk – Avoid starting the risky activity. Initiate measures to eliminate the risk immediately.

Significant risk – Plan and initiate measures to reduce the risk quickly.

Risk requiring attention – Plan and schedule measures to reduce the risk. Consider the cost-effectiveness of the measures.

Minor risk – Measures are usually not needed. Monitor that the situation remains under control.

Threat model

Table 9. Threat model

Threat	Likelihood	Impact	Risk category	Controls
				Protection: Detection: Response: Recovery: Responsible person: Status:
				Protection: Detection: Response: Recovery: Responsible person: Status:
				Protection: Detection: Response: Recovery: Responsible person: Status:
				Protection: Detection: Response: Recovery: Responsible person: Status:
				Protection: Detection: Response: Recovery: Responsible person: Status:

Version history

Table 10. Version history

Version	Date	Author	Changes

Appendix 3: Testing tools

This appendix lists open-source red teaming tools by use case. The most up-to-date list of tools should be checked on the OWASP AI Exchange website: https://owaspai.org/goto/testing/

Prompt injection

In prompt injection, the attacker gives the model manipulative instructions intended to achieve malicious outcomes or objectives.

Table 11. Applicable tools

Name	Function	API use	Language	Github
PyRIT, Python Risk Identification Tool for generative AI	Detection	✔	Python	https://github.com/Azure/PyRIT
Garak	Detection	✔	Python	https://github.com/NVIDIA/garak
Prompt Fuzzer	Detection	✔	Python	https://github.com/prompt-security/ps-fuzz
Guardrails	Detection and protection	✔	Python	https://github.com/guardrails-ai/guardrails
Promptfoo	Detection and protection	✔	Python, NodeJS	https://github.com/promptfoo/promptfo

Table 12. Tool applicability to frameworks and platforms

Framework/platform	Category	Tool applicability
Tensorflow	DL, GenAI	PyRIT
PyTorch	DL, GenAI	PyRIT, Garak, Guardrails
Azure OpenAI	GenAI	PyRIT, Guardrails, Promptfoo
Huggingface	ML, GenAI	PyRIT, Garak, Guardrails, Promptfoo
Azure managed endpoints	Machine Learning Deployment	PyRIT
Cohere	GenAI	PyRIT, Garak, Guardrails, Promptfoo
Replicate Text Models	GenAI	PyRIT, Garak, Promptfoo
OpenAI API	GenAI	PyRIT, Garak, Prompt Fuzzer, Guardrails, Promptfoo
GGUF (Llama.cpp)	GenAI, Lightweight Inference	PyRIT, Garak, Promptfoo
OctoAI	GenAI	Garak

Table 13. Data type support

Type	Tool
Text	PyRIT, Garak, Prompt Fuzzer, Guardrails, Promptfoo
Image	-
Audio	-
Video	-
Tabular data	-

Exploiting model inferences

The purpose of these attacks is to exploit the inference process to obtain unauthorised information or compromise the model.

Table 14. Applicable tools

Name	Functions	API use	Language	Github
ART, The Adversarial Robustness Toolbox	Detection	✔	Python	https://github.com/Trusted-AI/adversarial-robustness-toolbox
Armory	Detection	✔	Python	https://github.com/twosixlabs/armory

Table 15. Tool applicability to frameworks and platforms

Framework/platform	Category	Tool applicability
Tensorflow	DL, GenAI	ART, Armory
Keras	DL, GenAI	ART
PyTorch	DL, GenAI	ART, Armory
MxNet	DL	ART
Scikit-learn	ML	ART
XGBoost	ML	ART
LightGBM	ML	ART
CatBoost	ML	ART
GPy	ML	ART

Table 16. Data type support

Type	Tool
Text	ART, Armory
Image	ART, Armory
Audio	ART, Armory
Video	ART, Armory
Tabular data	ART, Armory

Runtime model theft

Attackers target parts of the model or critical components, such as system prompts. This gives them the opportunity to create advanced inputs that bypass guardrails.

Table 17. Applicable tools

Name	Functions	API use	Language	Github
ART, The Adversarial Robustness Toolbox	Detection	✔	Python	https://github.com/Trusted-AI/adversarial-robustness-toolbox
Armory	Detection	✔	Python	https://github.com/twosixlabs/armory
Promptfoo	Detection and protection	✔	Python, NodeJS	https://github.com/promptfoo/promptfoo

Table 18. Tool applicability to frameworks and platforms

Framework/platform	Category	Tool applicability
Tensorflow	DL, GenAI	ART, Armory
Keras	DL, GenAI	ART
PyTorch	DL, GenAI	ART, Armory
MxNet	DL	ART
Scikit-learn	ML	ART
XGBoost	ML	ART
LightGBM	ML	ART
CatBoost	ML	ART
GPy	ML	ART
Azure OpenAI	GenAI	Promptfoo
Huggingface	ML, GenAI	Promptfoo
Cohere	GenAI	Promptfoo
Replicate Text Models	GenAI	Promptfoo
OpenAI API	GenAI	Promptfoo
GGUF (Llama.cpp)	GenAI, Lightweight Inference	Promptfoo

Table 19. Data type support

Type	Tool
Text	ART, Armory, Promptfoo
Image	ART, Armory
Audio	ART, Armory
Video	ART, Armory
Tabular data	ART, Armory

Direct model exfiltration

In this attack, the model's parameters or functionality are stolen. This enables the attacker to create a copy of the model, which can then be used as an oracle for crafting adversarial attacks and other compounded threats.

Table 20. Applicable tools

Name	Functions	API use	Language	Github
Prompt Fuzzer	Detection	✔	Python	https://github.com/prompt-security/ps-fuzz

Table 21. Tool applicability to frameworks and platforms

Framework/platform	Category	Tool applicability
OpenAI API	GenAI	Prompt fuzzer

Table 22. Data type support

Type	Tool
Text	Prompt Fuzzer
Image	-
Audio	-
Video	-
Tabular data	-

Development-time model poisoning

This refers to the manipulation of model parameters, data or the supply chain during the development phase in order to change the model's operation. The attacker's objective is to alter the model's behaviour, which may lead to undesired operation.

Table 23. Applicable tools

Name	Functions	API use	Language	Github
ART, The Adversarial Robustness Toolbox	Detection	✔	Python	https://github.com/Trusted-AI/adversarial-robustness-toolbox
Armory	Detection	✔	Python	https://github.com/twosixlabs/armory

TextAttack	Detection	✔	Python	https://github.com/QData/TextAttack

Table 24. Tool applicability to frameworks and platforms

Framework/platform	Category	Tool applicability
Tensorflow	DL, GenAI	ART, Armory, TextAttack
Keras	DL, GenAI	ART
PyTorch	DL, GenAI	ART, Armory, TextAttack
MxNet	DL	ART
Scikit-learn	ML	ART
XGBoost	ML	ART
LightGBM	ML	ART
CatBoost	ML	ART
GPy	ML	ART

Table 25. Data type support

Type	Tool
Text	ART, Armory, TextAttack
Image	ART, Armory
Audio	ART, Armory
Video	ART, Armory
Tabular data	ART, Armory

Bypassing defences

Bypassing defences means that an adversary manipulates the input at inference time so that the model produces an incorrect result even though the manipulated input still appears normal to humans or downstream processing systems. The model is not retrained or poisoned. Instead, the attacker creates inputs that "slip past" the model's decision boundary.

Table 26. Applicable tools

Name	Functions	API use	Language	Github
ART, The Adversarial Robustness Toolbox	Detection	✔	Python	https://github.com/Trusted-AI/adversarial-robustness-toolbox
Armory	Detection	✔	Python	https://github.com/twosixlabs/armory
Foolbox	Detection	✔	Python	https://github.com/bethgelab/foolbox
DeepSec	Detection	✔	Python	https://github.com/ryderling/DEEPSEC
TextAttack	Detection	✔	Python	https://github.com/QData/TextAttack

PyRIT, Python Risk Identification Tool for generative AI	Detection	✔	Python	https://github.com/Azure/PyRIT
Garak	Detection	✔	Python	https://github.com/NVIDIA/garak

Table 27. Tool applicability to frameworks and platforms

Framework/platform	Category	Tool applicability
Tensorflow	DL, GenAI	ART, Armory, Foolbox, Deepsec, TextAttack, PyRIT
Keras	DL, GenAI	ART, Foolbox
PyTorch	DL, GenAI	ART, Armory, Foolbox, Deepsec, TextAttack, PyRIT, Garak
MxNet	DL	ART
Scikit-learn	ML	ART
XGBoost	ML	ART
LightGBM	ML	ART
CatBoost	ML	ART
GPy	ML	ART
Azure OpenAI	GenAI	PyRIT
Huggingface	ML, GenAI	PyRIT, Garak
Azure managed endpoints	Machine Learning Deployment	PyRIT
Cohere	GenAI	PyRIT, Garak
Replicate Text Models	GenAI	PyRIT, Garak
OpenAI API	GenAI	PyRIT, Garak
GGUF (Llama.cpp)	GenAI, Lightweight Inference	PyRIT, Garak
OctoAI	GenAI	Garak

Table 28. Data type support

Type	Tool
Text	ART, Armory, Foolbox, DeepSec, TextAttack, PyRIT, Garak
Image	ART, Armory, Foolbox, DeepSec
Audio	ART, Armory
Video	ART, Armory
Tabular data	ART, Armory

Finnish Transport and Communications Agency Traficom
PO Box 320, FI-00059 TRAFICOM, Finland
tel. +358 29 534 5000

traficom.fi

ISBN 987-952-425-008-5
ISSN 2669-8787 (online publication)

TRAFICOM
Liikenne- ja viestintävirasto