(DRAFT FOR PUBLIC CONSULTATION)

# SECURING AGENTIC AI

*An Addendum to the Guidelines and Companion Guide on Securing AI Systems*

CSA SINGAPORE

2025

This document is an addendum to CSA's Companion Guide on Securing AI Systems ("Addendum"), focusing on agentic AI systems. Systems owners should use this document in conjunction with the Companion Guide on Securing AI Systems as a resource.

This document is meant as a community-driven resource, developed in collaboration with the AI and cybersecurity practitioner communities. It provides practical mitigation measures and practices to secure AI systems. This document is intended for informational purposes only and is not mandatory, prescriptive nor exhaustive.

## DEVELOPED IN CONSULTATION WITH

This document is published by the CSA, in collaboration with partners across the AI and Cyber communities, including:

Accenture
Alibaba Cloud
Amaris AI
Cisco
Deloitte Singapore
DSO National Laboratories
Fujitsu Limited
Google Asia Pacific Pte. Ltd.
Government Technology Agency (GovTech)
HP Inc.
Kaspersky Lab Singapore Pte Ltd
Microsoft Singapore
Palo Alto Networks
PricewaterhouseCoopers Risk Services Pte Ltd
Resaro
The American Chamber of Commerce in Singapore (AmChamSG)
Vulcan (vulcanlab.ai)

# DISCLAIMER

The information provided in this document does not, and is not intended to, constitute legal advice. All information is for general informational purposes only. These organisations provided views and suggestions on the security controls, descriptions of the security control(s), and technical implementations included in this Addendum. CSA and its partners shall not be liable for any inaccuracies, errors and/or omissions contained herein nor for any losses or damages of any kind (including any loss of profits, business, goodwill, or reputation, and/or any special, incidental, or consequential damages) in connection with any use of this Addendum. Organisations are advised to consider how to apply the controls within to their specific circumstances, in addition to other measures relevant to their needs. This document contains links to other third-party websites. Such links are informational and do not represent endorsement of content from these third-party sites.

# VERSION HISTORY

| VERSION | DATE RELEASED | REMARKS |
|---|---|---|
| **0.1** | 22 Oct 2025 | Release of Addendum on Securing Agentic AI for Public Consultation |

# EXECUTIVE SUMMARY

**Agentic artificial intelligence (AI) systems** are self-managing AI systems that can plan, execute, critique, and iterate across multiple steps to achieve specified objectives. These systems represent a significant evolution from traditional AI systems, moving beyond simple pattern recognition and predetermined responses to demonstrate increasingly sophisticated abilities to understand context, formulate plans, and take independent actions to achieve specified objectives. Development of these systems bring new capabilities and opportunities for organisations and users.

Organisations must carefully consider both the transformative potential and inherent risks these agentic AI systems present. Their capacity to operate with reduced human oversight introduces novel security considerations around system boundaries, control mechanisms, and the potential for unexpected emergent behaviours. Understanding and addressing these security implications is crucial as agentic AI becomes more prevalent in our digital infrastructure and business operations.

The Cyber Security Agency of Singapore (CSA) has developed this addendum to advise system owners on securing their agentic AI systems. This addendum is meant to be read together with the Guidelines and Companion Guide on Securing AI Systems, which outline foundational AI security principles.

As an addendum to the Guidelines, this document takes a risk-based approach across the AI development lifecycle, while introducing new considerations that are relevant to agentic AI. These considerations include mapping out agentic workflows to identify potential threat vectors to the system.

To complement the Companion Guide, this addendum lists agentic AI-related risks and mitigations across the development lifecycle, categorised by capabilities of agentic AI systems. In addition, examples based on current industry use cases are provided as a practical resource on how to apply the addendum.

This document is intended for informational purposes only and is not mandatory, prescriptive nor exhaustive. The content of this document should not be construed as comprehensive guidance or definitive recommendations.

# TABLE OF CONTENTS

# QUICK REFERENCE TABLE

Stakeholders in specific roles may use the following table to quickly reference relevant controls in Section 4.2 – IDENTIFY THE RELEVANT MEASURES & CONTROLS.

The roles defined below are included to guide understanding of this document and are not intended to be authoritative.

Decision Makers:

Responsible for overseeing the strategic and operational aspects of AI implementation for the AI system. They are responsible for setting the vision and goals for AI initiatives, defining product requirements, allocating resources, ensuring compliance, and evaluating risks and benefits.

**Roles Included: Product Manager, Project Manager**

AI Practitioners:

Responsible for the practical application (i.e. designing, developing, and implementing AI solutions, including AI agents) across the life cycle. This includes collecting, procuring or analysing data that goes into systems, building the AI system architecture and infrastructure, building and optimising the AI system to deliver the required functions, as well as conducting rigorous testing and validation of AI models and agents to ensure their accuracy, reliability, and performance. In cases where the AI system utilises a third-party AI system, AI Practitioners also include the third-party providers responsible for these activities, such as those contracted through a Service Level Agreement (SLA). AI practitioners would be in charge of implementing the required controls across the entire system.

**Roles Included: AI/ML Developer, AI/ML Engineer, Data Scientist**

Cybersecurity Practitioners:

Responsible for ensuring the security and integrity of AI systems. This includes implementing security measures to protect AI systems in collaboration with AI Practitioners, monitoring for potential threats, ensuring compliance with cybersecurity regulations.

**Roles Included: IT Security Practitioner, Cybersecurity Expert**

*Table 1: User Quick Reference Table*

| The following measures/ controls may be relevant to **Decision Makers:** | The following measures/ controls may be relevant to **AI Practitioners:** | The following measures/ controls may be relevant to **Cybersecurity Practitioners:** |
|---|---|---|
| 1.1 Conduct a risk assessment | 1.1 Conduct a risk assessment | 1.1 Conduct a risk assessment |
| 2.1 Supply chain security<br>2.7 Limit agency<br>2.10 Self-reflection<br>2.11 Hallucination | 2.1 Supply chain security<br>2.2 Model hardening<br>2.3 System hardening<br>2.4 Identify, track and protect assets<br>2.5 Regular backups<br>2.6 Authorisation and authentication<br>2.7 Limit agency<br>2.8 Secure by default<br>2.9 Environment segmentation<br>2.10 Self-reflection<br>2.11 Hallucination | 2.1 Supply chain security<br>2.3 System hardening<br>2.4 Identify, track and protect assets<br>2.5 Regular backups<br>2.6 Authorisation and authentication<br>2.7 Limit agency<br>2.8 Secure by default<br>2.9 Environment segmentation |
| 3.2 Security testing | 3.1 Availability controls<br>3.2 Security testing<br>3.3 Secure MCP<br>3.4 Secure inter-agent communication | 3.1 Availability controls<br>3.2 Security testing<br>3.3 Secure MCP<br>3.4 Secure inter-agent communication |
| 4.3 Continuous monitoring and logging<br>4.4 Human-in-the-loop<br>4.5 Vulnerability disclosure | 4.1 Validate inputs<br>4.2 Validate outputs<br>4.3 Continuous monitoring and logging<br>4.4 Human-in-the-loop<br>4.5 Vulnerability disclosure | 4.1 Validate inputs<br>4.2 Validate outputs<br>4.3 Continuous monitoring and logging<br>4.5 Vulnerability disclosure |

# 1. INTRODUCTION

Agentic **artificial intelligence (AI) systems** are self-managing AI systems that can plan, execute, critique, and iterate across multiple steps to achieve specified objectives. The emergence of these systems reflects ongoing developments in AI that brings new capabilities and opportunities for organisations and users. These systems are capable of autonomous, goal-driven decision making and execution, which will reshape how we interact with AI.

Agentic AI systems represent a significant evolution from traditional AI systems, moving beyond simple pattern recognition and predetermined responses to demonstrate increasingly sophisticated abilities to understand context, formulate plans, and take independent actions to achieve specified objectives. To achieve these objectives, agentic AI systems make use of **AI agents**—modular systems driven by Large Language Models (LLMs) and Large Image Models (LIMs) for narrow, task-specific automation[1]. Multiple AI agents may be used together and orchestrated by an autonomous agentic AI system.

As organisations begin to deploy agentic AI systems (and AI agents) across various domains—from process automation and customer service to complex decision support and resource optimisation—we must carefully consider both the transformative potential and inherent risks these systems present. Their capacity to operate with reduced human oversight, while potentially increasing efficiency and scalability, also introduces novel security considerations around system boundaries, control mechanisms, and the potential for unexpected emergent behaviours. Understanding and addressing these security implications is crucial as agentic AI becomes more prevalent in our digital infrastructure and business operations.

The Cyber Security Agency of Singapore (CSA) has worked closely with AI and cybersecurity practitioners to develop this addendum to advise system owners on securing their agentic AI systems. This addendum is meant to be read together with the Guidelines and Companion Guide on Securing AI Systems, which outline foundational AI security principles.

This document is intended for informational purposes only and is not mandatory, prescriptive nor exhaustive. The content of this document should not be construed as comprehensive guidance or definitive recommendations.

[1] Sapkota, R., Roumeliotis, K. I., & Karkee, M. *AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges*.

# PURPOSE AND SCOPE

## Purpose

This addendum curates practical measures and controls that system owners can use to secure their adoption of agentic AI systems. These measures and controls are voluntary, and not all the measures and controls listed in this addendum will be applicable to all organisations or environments. Organisations may also be at different stages of AI development (e.g. POC, pilot, beta release). Organisations should consider relevance to their use cases as well.

This addendum is meant to be read with the Guidelines and Companion Guide on Securing AI Systems[2]. As this Addendum is focused on the key elements of agentic AI systems, the relevant treatment measures/controls from the Companion Guide may still apply to underlying systems and related processes, even if not covered in this document.

## Scope

The measures and controls within the addendum address the cybersecurity threats and risks relevant to agentic AI systems. It does not specifically address AI safety, or other common attendant considerations for AI such as fairness, transparency or inclusion, although it is noted that some of the recommended cybersecurity controls may address AI safety risks as well. It also does not cover the misuse of AI for cyberattacks (AI-enabled malware), and scams (deepfakes).

[2] Cyber Security Agency of Singapore. *Guidelines and Companion Guide on Securing AI Systems*

# 2. HOW AGENTIC AI WORKS

Agentic AI systems interact with their environment, collect data and perform self-determined tasks to meet specified goals.

We can describe the agentic AI system through the following, which helps system owners to understand how agentic AI systems operate and what considerations are needed for safe and effective deployment:

- Key components that facilitate its operation,
- System design, including its architecture; and
- Capabilities (cognitive, interactive, operational)

These elements help system owners to understand how agentic AI systems operate and what considerations are needed for safe and effective deployment.

*Figure 1: Baseline and Capability Taxonomy, AI Risk and Capability Framework*[3]


[3] GovTech Singapore (AI Practice). *Agentic Risk & Capability Framework*.

# 2.1. BASELINE COMPONENTS

Large Language Models (LLMs) alone are constrained in their operations. While they can be sophisticated in terms of processing input and content generation, by themselves they cannot directly take actions beyond providing information. Agentic AI systems transform this paradigm fundamentally by connecting LLMs to functional tools and systems. This enables them to execute tasks such as sending emails, reading and writing to files and databases, interacting with other software systems, or orchestrating multi-step processes.

This expansion from content generation to actual action relies on the integration of multiple components.

*Table 2: Key Components in Agentic AI Systems*

| Component | Description |
|---|---|
| Large Language Model (LLM) | An AI model that serves as the central reasoning and planning engine, or the "brain" of the agent. It processes instructions, interprets user inputs, and generates contextually appropriate responses. |
| Tools | Extends the capabilities of LLMs to execute actions such as writing to files and databases, controlling devices, or performing transactions. Tools can also allow AI agents to perceive the environment through sensors or accessing APIs to obtain information (e.g. flight details, weather). Tools can be called based on the LLM's reasoning and user needs. |
| Instructions | Command(s) that defines an agent's role, capabilities, and behavioural constraints e.g. a system prompt for an LLM. Instructions may be implemented by model providers if calling an external LLM, and/or added by users and developers. |
| Memory | Information that is stored and accessible to the LLM. These can be in temporarily contained in the short-term memory or more persistent within the long-term memory. |
| Protocols | Protocols allow for a simplified, consistent, and standardised way for agents to communicate with tools and other agents. |

Typically, the process of transforming a user's inputs into execution of a task involves:

1. **Receiving inputs.** The AI agent receives a specific instruction or goal from the user.
2. **Layering on perception.** The AI agent collects sensory input from sources, such as cameras or microphones, or screen captures and processing technology. This helps it to detect contextual cues and perceive its environment.
3. **Reasoning and planning.** The LLM helps to break down the goal into smaller actionable tasks.
4. **Orchestration and action execution.** Perform tasks based on specific orders or conditions. This may include interactions with other agents, and/or connected systems and tools.
5. **Render a response.** Updates the user on the outcome in an appropriate format.

# 2.2. BASELINE SYSTEM DESIGN

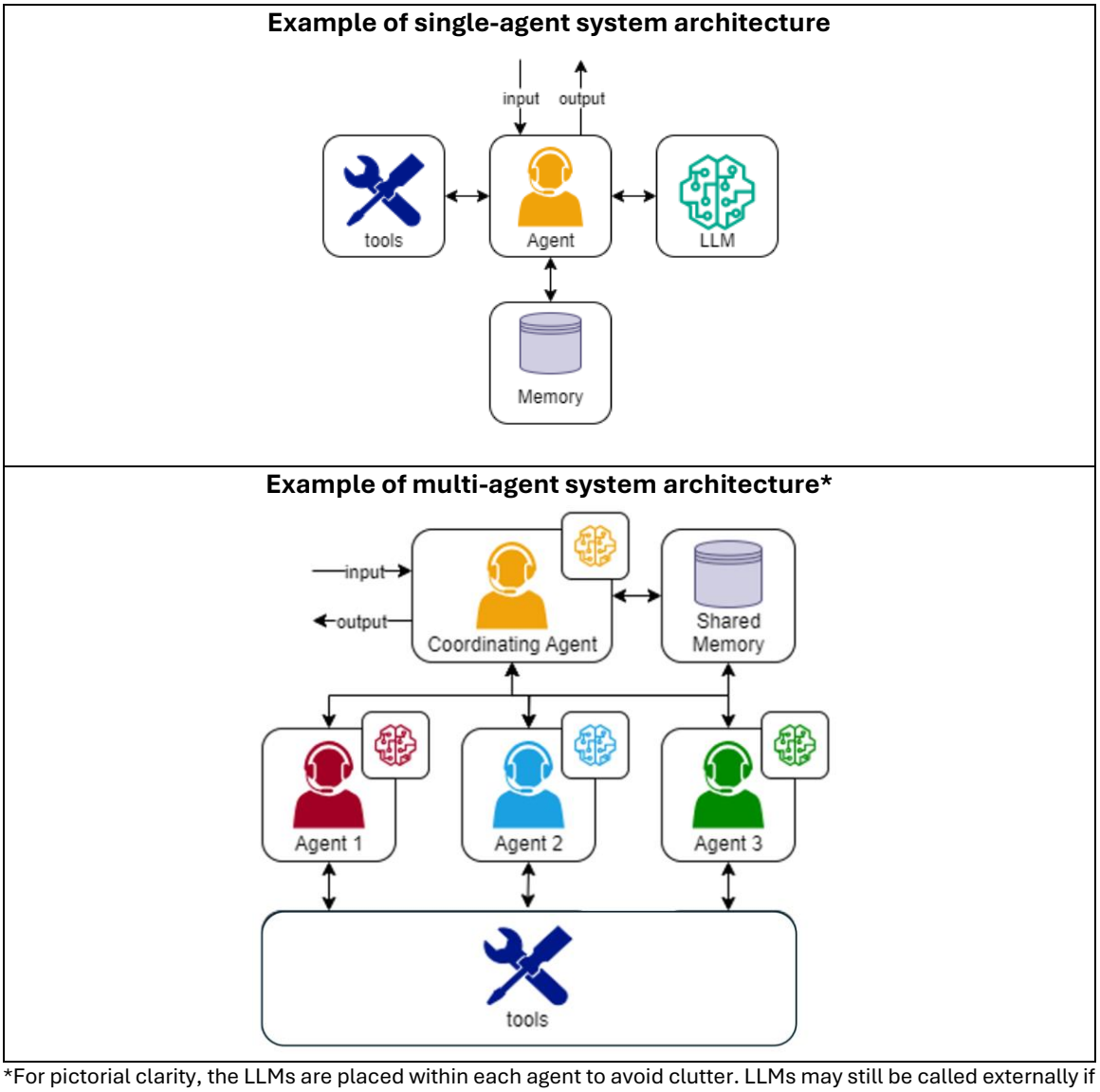
## 2.2.1. Agentic AI system architecture

The agentic AI system architecture defines how agents are connected, coordinated and orchestrated to solve tasks.

A single-agent system is an AI system with one agent that handles all tasks independently. A multi-agent architecture comprises multiple agents, collaborating to scale or combine specialist roles and functionalities. The co-operation across multiple agents enables solving problems that go beyond the capabilities of would be infeasible for a single agent alone.

Different architectures result in varying levels of system-wide risk, which should be considered carefully.

*Figure 2: Examples of single- vs. multi-agent system architecture*




*For pictorial clarity, the LLMs are placed within each agent to avoid clutter. LLMs may still be called externally if needed (e.g. through APIs).

*Table 3: Key differences between single agent and multi-agent systems*

| | **Single-agent** | **Multi-agent** |
|---|---|---|
| Complexity and architecture | Simple and centralised architecture | More complex, distributed architecture |
| Decision-making capabilities | Centralised decision-making by one agent | Distributed decision-making amongst multiple agents, and hence should be able to address more complex tasks as tasks can be delegated to different specialised agents |
| Task complexity | Handles one task at a time | Can manage multiple tasks simultaneously |
| Adaptability | May struggle with dynamic environments | More likely to adjust and respond in real-time to changes in environment |
| Communication | Operates in isolation; no inter-agent communication needed | Agents interact and share information, hence requiring communication through protocols (e.g. A2A, ACP) |
| Fault tolerance | Simple system with limited redundancy – could have a single point of failure. | Easier to build redundancy, but complex system could have correlated failures[4]. |

In both single- and multi-agentic architectures, agents communicate with tools and services. In multi-agent architectures, communication also takes place among agents. Traditionally, such integration with tools and services may require separate and on-off integrations. With the rise of agentic AI, we observe the release of protocols (e.g. Anthropic's Model Context Protocol (MCP), Google's Agent2Agent (A2A)). that allow for a simplified, consistent, and standardised way for agents to communicate. These reduce the effort required to onboard new tools, services and agents.

### 2.2.2. Roles & access control

Roles and access controls establish the responsibilities and permissions across agents in the system. This helps to limit the impact of incidents such as unauthorised actions or access, or potential system failures. Agent roles can include:

- Orchestrator agents that manage workflows
- Specialist agents that perform pre-defined functions
- Interface agents that handle external communications.

Roles and access controls for agentic AI systems should be clearly defined to avoid unauthorised access or excessive privilege.

[4] Correlated failures are when multiple components fail due to a single shared cause.

### 2.2.3. System workflows & autonomy

An AI agentic workflow describes the step-by-step process whereby AI agents use reasoning, planning and tools to perform tasks. Such workflows can also be seen in terms of data movement within agentic AI systems, which becomes increasingly challenging to track with more complex architectures and integration to more tools and capabilities. These workflows range from straightforward linear progressions (see Figure 3) to more intricate branching and/or hierarchical patterns (see Figure 4).

- In a linear workflow, data moves sequentially through predetermined steps i.e. each action follows directly from the previous one.

- Branching workflows are implemented when the agentic AI system needs to make decisions about using multiple tools or services simultaneously, based on the task goal or contextual information. These branching workflows hence create multiple possible paths for data movement.

*Figure 3: Example of a linear workflow*

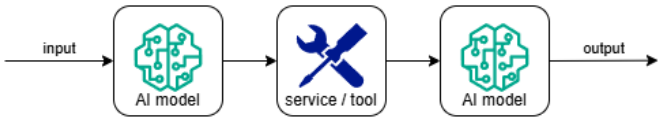
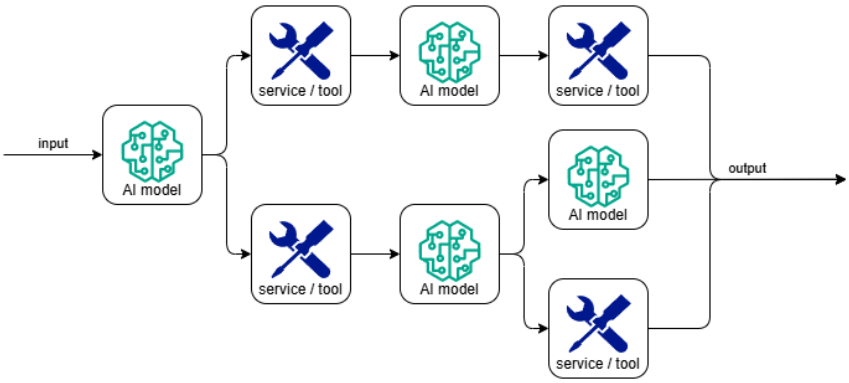



*Figure 4: Example branching workflow*




Understanding the workflow, as well as data movement, informs risk assessment and threat modelling. This allows system owners to identify critical points where data might be vulnerable, and prioritise safeguards. These topics are explored in greater detail in Chapter 3.

The workflow within an agentic AI system is also affected by its autonomy, which refers to its ability to operate, make decisions and execute tasks with minimal or no human intervention. As autonomy of the system increases, it also becomes increasingly challenging to assess or

predict the potential data flows. This underscores the importance of determining the appropriate autonomy level of the agentic AI system.

Organisations such as NVIDIA have developed frameworks to classify the autonomy levels of agentic AI systems[5].

*Table 4: NVIDIA's autonomy classification framework*

| Autonomy Level | Description | Example |
|---|---|---|
| 0 – Inference API | A single user request results in a single inference call to a single model. | An image classification service that takes a photo and returns a label exemplifies this simplicity. The data path is direct: input → model → output, with no additional processing or decisions. |
| 1 – Deterministic System | A single user request triggers more than one inference request, possibly to more than one model, in a predetermined order that does not depend on either user input or inference results. | In drug discovery, a system might process molecular structures through predetermined stages: initial screening → toxicity analysis → binding prediction. Each step's output feeds into the next in a known sequence. |
| 2 – Weakly autonomous system | A single user request triggers more than one inference request. An AI model can determine if or how to call plugins or perform additional inference at predetermined decision points. | An enterprise document processing system might analyse content type, then route documents through different specialized models: financial documents to compliance checkers, technical documents to subject matter validators, and customer communications to sentiment analysers. While complex, all possible paths can be mapped. |
| 3 – Fully autonomous system | A single user request triggers more than one inference request. In response to a user request, the AI model can freely decide if, when, or how to call plugins or other AI models, or to revise its own plan freely, including deciding when to return control to the user. | A security vulnerability analyser might start with code review, dynamically decide to examine deployment configurations, investigate dependency chains, and recursively explore potential attack vectors, continuously adjusting its investigation based on findings. The number of possible execution paths grows exponentially. |

[5] Harang, R., & Sablotny, M. *Agentic Autonomy Levels and Security*. NVIDIA.

For Level 0 systems, mapping of workflows may not be necessary as inference calls are made directly to a model, which produces an output. There are no additional services or tools are invoked.

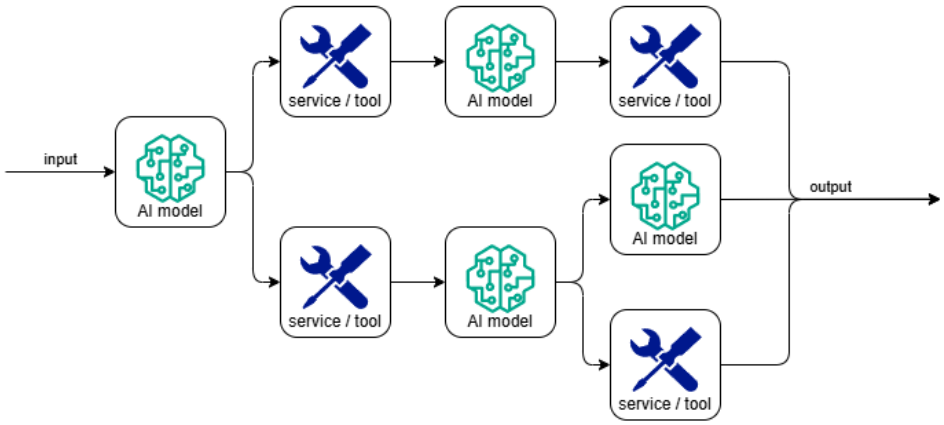
For Level 1 systems and above, mapping of workflows is highly recommended.

- Level 1 systems usually present as a linear chain of calls in which the output from one AI call or tool response is passed on to the next step in a deterministic manner. The complete workflow is known beforehand.

*Figure 5: Autonomy Level 1 – Deterministic system, linear workflow*

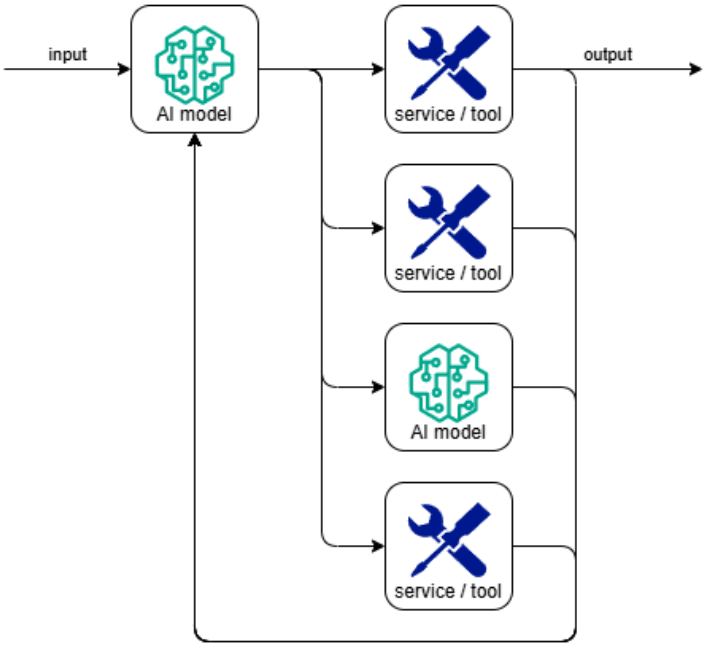

- Level 2 systems have outputs that can be sent along various paths though the workflow, based on task requirements and the orchestrator agent's decision. Every execution path can be determined, but the actual path can only be identified when the workflow is executed.

*Figure 6: Autonomy Level 2 – Weakly autonomous system, branching paths at predetermined points*



- Level 3 systems have significantly more potential execution paths, as more models and tools are invoked. This complexity can be seen in the cyclical path, which indicates a potentially unbounded number of execution paths. It is generally not possible to enumerate all the paths in advance or specific paths which will be used.

*Figure 7: Autonomy Level 3 - Fully autonomous system, flows branch to different paths and can be cyclical*




**Agent Design Patterns**

Agent design patterns define how an agentic AI system's components are organised, integrated, and orchestrated to accomplish a task. Unlike system workflows that only describe the sequence of steps an agent takes, agent design patterns provide reusable architectural templates that determine the fundamental structure and interaction model for an agentic AI system. These templates systematically provide different approaches to organise agents based on specific workload characteristics and requirements. This helps with scalability, and is more easy to maintain implementations (similar to how software design patterns like Model-View-Controller provide standardised approaches to building applications, though agent patterns are still being refined as the field matures).

Examples of these agent design patterns include:

| Agent design pattern | Description |
|---|---|
| Sequential | Specialised agents execute in a predefined, linear order with each agent's output serving as direct input for the next agent, using predefined workflow logic and no AI model orchestration. |

| | |
|---|---|
| Parallel | Multiple specialised sub-agents perform tasks independently and simultaneously, with outputs then synthesised to produce a final consolidated response, using predefined workflow logic and no AI model orchestration. |
| Loop | Repeatedly executes a sequence of specialised subagents until a specific termination condition is met, using predefined logic and no AI model orchestration. |
| Reason and act (ReAct) | Uses iterative loops of thought (reasoning about next steps), action (tool selection or final answer), and observation (saving tool outputs) for dynamic planning and continuous adaptation. |
| Coordinator | Uses a central coordinator agent, with AI model orchestration, to analyse requests, decompose into sub-tasks, and dynamically route these to specialised agents. |
| Swarm | Uses collaborative all-to-all communication, where a dispatcher routes requests to specialised agents that can communicate with each other and hand off tasks. Lacks central orchestration and requires explicit exit conditions. |

System owners should choose an agent design pattern based on the nature of tasks involved (e.g., whether they are predictable and sequential, or complex problems requiring autonomous decision-making with outputs achieved through iterative refinement cycles). Each pattern involves trade-offs: simpler patterns like sequential offer lower complexity and cost but limited flexibility, whilst advanced patterns like swarm provide exceptional capability for complex problems but require significant computational resources and sophisticated orchestration logic.

From a security perspective, agent design patterns can affect the likelihood and impact of attacks such as prompt injection, where malicious instructions embedded in processed content manipulate agents to perform rogue actions or sensitive data disclosure. Agentic AI systems can build resilience through agent design patterns that enforce strict isolation between untrusted data and agent control flow. This should be layered on with relevant security controls (discussed in Chapter 4) for more comprehensive defence.

# 2.3. CAPABILITIES

AI systems differ in their **capabilities, which can be seen as the general classes of actions that an agentic AI system can perform.**

There are three key categories of capabilities: **cognitive, interaction, and operational**[6]. Each category present distinct functions and interactions with their environment. As each type of capability presents its own value and risks, agentic AI systems with more capabilities can also incur more risks that need to be addressed.

*Cognitive capabilities*

Cognitive capabilities mimic human thinking. For example:

- **Reasoning and problem-solving**. The capability to perform structured, multi-step reasoning that demonstrates deeper understanding, problem-solving, and decision-making.
- **Planning & goal management**. The capability to develop detailed, step-by-step, and executable plans with specific tasks in response to broad instructions.
- **Agent delegation.** The capability to assign subtasks to other agents and coordinate their activities to achieve broader goals.
- **Tool use.** The capability to evaluate available options and choose the best tool for specific subtasks.

[6] GovTech Singapore (AI Practice). *Agentic Risk & Capability Framework*.

*Interaction capabilities*

Interaction capabilities describe how the agentic AI system exchanges information with users, other agents, and external systems. These capabilities below are broadly differentiated based on how and what they interact with:

- **Natural language communication.** The capability to fluently and meaningfully converse with human users, handling a wide range of situations such as explaining complex topics, generating documents or prose, or discussing issues with human users.
- **Multimodal understanding & generation**. The capability to take in image, audio, or video inputs and / or generate image, audio, or video outputs.
- **Official communication**. The capability to compose and directly publish communications that formally represent an organisation to external parties (e.g., customers, partners, regulators, courts, media) via approved channels and formats without human oversight or approval.
- **Business transactions**. The capability to execute transactions that involve exchanging money, services, or commitments with external parties.
- **Internet and search access**. The capability to access and search the Internet for services or resources, especially for up-to-date information to supplement its knowledge and provide more accurate answers.
- **Computer use**. The capability to directly control a computer interface by moving the mouse, clicking buttons, and typing on behalf of the user.
- **Other programmatic interfaces**. The capability to interact with external systems through APIs, SDKs, or backend services.

*Operational capabilities*

Operational capabilities focus on the agentic AI system's ability to execute actions safely and efficiently within its operating environment. This can include:

- **Agent communication**. The capability to communicate with other agents within the system, either through natural language or a predefined protocol, and to coordinate with other agents to accomplish complex tasks that require multiple specialties.
- **Code execution.** The capability to write, execute, and debug code in various programming languages to automate tasks or solve computational problems.
- **File & data management. The capability to create, read, modify, organise, convert, query, and update information across both unstructured files** (e.g., PDFs, Word docs, spreadsheets) **and structured data stores** (e.g., SQL/NoSQL databases, data warehouses, vector stores).
- **System management.** The capability to adjust system configurations, manage computing resources, and handle technical infrastructure tasks.

*Figure 8: Baseline and Capability Taxonomy, AI Risk and Capability Framework*

# 3. SECURITY THREATS TO AGENTIC AI SYSTEMS

Agentic AI systems face both traditional and novel security challenges. This can be seen as a cumulation across different layers of risks.

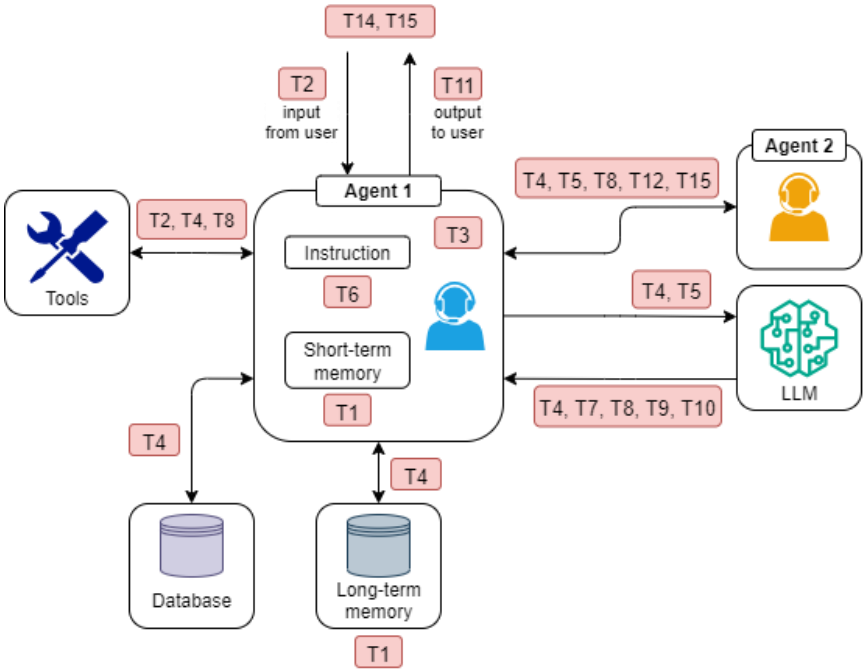
- **Classical cybersecurity risks**. This is because agentic AI systems have underlying software infrastructure and components, and can be vulnerable to threats such as remote code execution and SQL injection (if connected to a structured database).
- **Inherited risks from LLMs,** including prompt injection, jailbreaking and data leakage. Refer to CSA's Guidelines and Companion Guide on Securing AI systems, *Section 2.2.2 – Development* for a fuller articulation.
- **New risks arising from agentic AI systems.** The two primary security concerns in agentic AI systems are rogue actions and sensitive data disclosure.
    - Rogue actions occur when agents perform unintended, or harmful tasks. These can arise through prompt injection, where malicious instructions hidden within normal-looking inputs manipulate the agent's behaviour. They can also occur through simple misunderstandings, if the agent misinterprets ambiguous instructions or handles complex interfaces incorrectly. The impact of these rogue actions directly correlates with the agent's capabilities – more powerful agents pose greater risks when they malfunction.
    - Sensitive data disclosure through agent manipulation. This occurs when attackers exploit agents to reveal private information when agentic workflows are executed. The agent can be guided through a series of seemingly legitimate actions that ultimately leak protected information. Attackers can also manipulate the agent to include sensitive data in its responses.

As with all digital capabilities, there is a balance between utility and risk. For agentic AI systems, increasing the agent(s)'s autonomy, access and capabilities can enhance its usefulness. However, this can simultaneously expand the attack surface of the agentic AI system, as well as its potential for causing harm or other undesired actions if they malfunction or are maliciously exploited.

There is a growing body of resources on the risks to agentic AI systems. This includes OWASP's threat taxonomy for agentic AI systems that highlights 15 threats[7]:

T1 – Memory Poisoning
T2 – Tool Misuse
T3 – Privilege Compromise
T4 – Resource Overload
T5 – Cascading Hallucination Attacks
T6 – Intent Breaking & Goal Manipulation
T7 – Misaligned & Deceptive Behaviours
T8 – Repudiation & Untraceability
T9 – Identity Spoofing & Impersonation
T10 – Overwhelming Human in the Loop
T11 – Unexpected RCE and Code Attacks
T12 – Agent Communication Poisoning
T13 – Rogue Agents in Multi-Agent Systems
T14 – Human Attacks on Multi-Agent Systems
T15 – Human Manipulation

*Figure 9: Example of threats to agentic AI systems*




For more details on the OWASP ASI threat taxonomy, refer to ANNEX A - Threats to Agentic AI Systems or https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/

[7] OWASP. *OWASP Top 10 for LLMs - Agentic AI - Threats and Mitigations.*

# 4. SECURING AGENTIC AI

## 4.1. TAKE A LIFECYCLE APPROACH, AND START WITH A RISK ASSESSMENT

CSA's Guidelines and Companion Guide to Securing AI Systems lay out the two key principles to securing AI systems, including taking a lifecycle approach and starting with a risk assessment. This continues to be relevant for agentic AI systems. The approach to securing AI systems is included here for easy reference. Given the dynamic nature of agentic AI systems, we recommend additional considerations in Steps 1 and 3 to support the risk assessment.

### STEP 1 – Conduct a risk assessment, focusing on security risks to agentic AI systems

Conduct a risk assessment, either based on best practices or your organisation's existing Enterprise Risk Assessment/Management Framework. Risk assessment can be done with reference to CSA's published guides[8], if applicable:

- Guide to Cyber Threat Modelling
- Guide to Conducting Cybersecurity Risk Assessment for Critical Information Infrastructure

Focus on the security risks related to AI systems. For agentic AI systems, we also recommend:

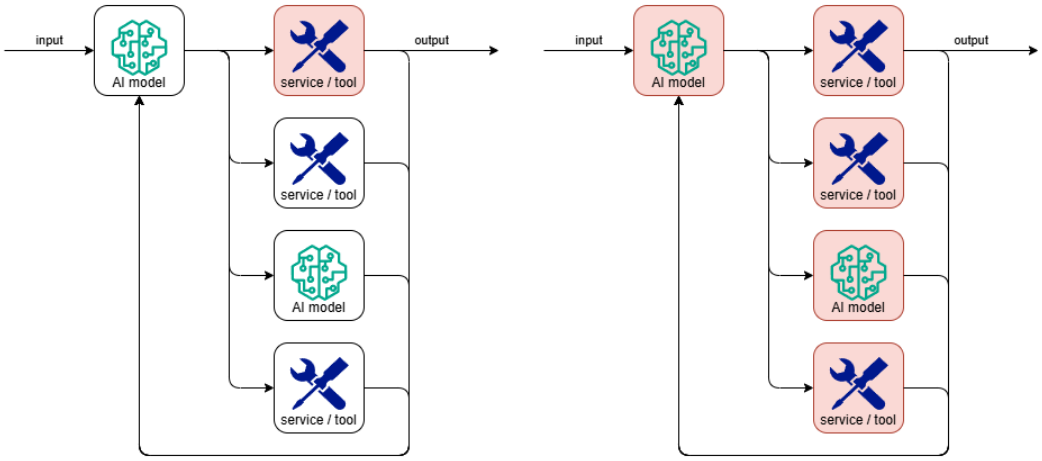
- **Assessing the autonomy level of the system.** This will assess how independently the system operates, how it makes decisions, and how complex its workflows might become. A Level 0 system making straightforward inference calls presents vastly different security challenges compared to a Level 3 system that can dynamically modify its own execution paths.
- **Perform threat modelling to identify areas of interest.** Threat modelling identifies where security risks might occur in the system's workflows. This can be complemented with taint tracing, which is a methodology to track how untrusted data moves through the system. For instance, in a customer service AI system, we can map how user inputs might flow through various decision points and tools, to identify and implement appropriate controls at critical junctures.

[8] Cyber Security Agency of Singapore. *Supplementary references*

- **Identify the risks associated with the agent(s)'s capabilities.** Each capability results in different consequences, and hence different associated risks. Taking a capability-centric approach helps to: (i) be precise about the impact of an agent's operation and potential failure; (ii) identify the different actions involved in realizing the capability, and in turn identify the potential risks. Given that agentic AI system capabilities continue to grow, a capability-centric framework helps to provide a scalable foundation for managing diverse systems.

**Taint tracing** – tracking data flows from untrusted sources through agentic workflows – enables security teams to identify when systems have been compromised and which actions require additional scrutiny or manual approval[9].

*Figure 10: Enumerating taints in Level 3 systems (tainted flows marked in red)*




Once untrusted data enters the system, the execution flow is marked as tainted, and every downstream tool and resources are also considered to be untrusted. Tainted components should be isolated from the rest of the system, to mitigate downstream impact to the system.

[9] Harang, R., & Sablotny, M. *Agentic Autonomy Levels and Security*. NVIDIA.

## STEP 2 – Prioritise areas to address based on risk/impact/resources

Prioritise which of the identified risks to address, based on the likelihood, impact, available resources, and risk appetite.

## STEP 3 – Identify and implement the relevant actions to secure the agentic AI system

Identify relevant actions and control measures to secure the agentic AI system, such as by referencing those outlined in CSA's **Companion Guide on Securing AI Systems** as well as in Section 4.2 of this Addendum and implement these across the AI life cycle.

## STEP 4 – Evaluate residual risks for mitigation or acceptance

Evaluate the residual risk after implementing security measures for the AI system to inform decisions about accepting or addressing residual risks.

**Risk Management for SaaS Environments**

For organisations using Software-as-a-Service (SaaS) agentic AI systems, detailed threat modelling and taint tracing may prove impractical due to limited visibility into third-party system architectures and data flows. Many security controls identified through these processes may be unimplementable, as they remain under the vendor's control rather than the organisation's direct management. However, understanding these risks remains crucial for informed decision-making.

The threat identification and assessment processes outlined in this document enable organisations to articulate specific security concerns to vendors, demanding appropriate mitigations or transparency about existing controls.

Where vendors cannot or will not address identified risks, organisations must escalate these findings to management for formal risk acceptance decisions. Additionally, red teaming exercises become essential for SaaS deployments, as they can uncover practical vulnerabilities and attack paths that theoretical threat modelling cannot reveal—particularly important when organisations have limited insight into the actual implementation of third-party systems. These empirical testing approaches help validate whether vendor-claimed security measures actually protect against real-world threats.

**Implementing Controls for Visibility at Enterprise-scale**

A key consideration for organisations is how to implement these steps practically, meaningfully, and at scale. One example mechanism is through the implementation of a middleware providing a single enforcement plane where identity and access management (agents identified with service principals, assigned roles in accordance with the least privilege principle, authenticated through OAuth2/OIDC with short-lived and scoped tokens), guardrails (input and output), data loss prevention, and policy controls apply consistently. Organisations adopting this mechanism route all agent-initiated calls (to SaaS APIs, internal services, data lakes, etc.) through a central gateway (API gateway, MCP gateway (if using an agentic runtime), service mesh ingress (for agent-to-microservice calls), etc.). Further, logs from the middleware are streamed into a SIEM for SOC monitoring, and processes are in-place to revoke agent access when anomalous access is detected.

## Periodic re-evaluation

The risk assessment should not be a one-time activity, but done throughout a system's operational lifetime. It is important to periodically re-evaluate threat models and controls, especially after significant system changes (e.g., updates to agent workflows, capabilities, or autonomy levels).

# 4.2. IDENTIFY THE RELEVANT MEASURES & CONTROLS

Based on the risk assessment, system owners can identify the relevant treatment measures/controls from the following tables. Each treatment measure/control plays a different role, and should be assessed for relevance and priority in addressing the security risks specific to your agentic AI system and context (Refer to Section 4.1).

As a start, we recommend users to consider all controls related to the baseline elements, and then to layer on those specific to each capability.

- **Related threats/risks** indicated serve as examples and are not exhaustive. They might differ based on your organisation's use case.
- **Related components/capabilities for each measure/control** are also provided to help you quickly identify what is relevant. Baseline risks are applicable to most, if not all agentic AI systems and should be addressed if possible.
- **Example implementations** are included for each measure/control as a more tangible elaboration on how they can be applied. These are also not exhaustive.
- Additional **references and resources** are provided for users of this document to obtain further details on applying the treatment measure/control if required.

As with the Companion Guide, the controls are organised using a lifecycle approach to systematically enumerate every potential mitigation throughout the development lifecycle.

# 4.3. TREATMENT MEASURES / CONTROLS FOR AGENTIC AI SYSTEMS

## 1. PLANNING AND DESIGN

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| 1.1 | Conduct a risk assessment in accordance with the relevant industry standards/best practices.<br><br>**Responsible Parties:**<br>Decision Makers, AI Practitioners, Cybersecurity Practitioners | Failure to comply with industry standards/best practices may lead to insufficient, inefficient or ineffective mitigations against adversarial threats.<br><br>Tainted components in an agentic AI system can have downstream impact along the workflow. | Baseline | As part of a risk assessment and threat modelling, perform taint tracing across workflows throughout the agentic AI system. Taint tracing is especially important for agentic AI systems of higher autonomy levels (i.e. levels 2 and 3).<br><br>Users are not limited to only one method of threat modelling and may adopt other relevant methods that address their needs. | • Chapter 3.2 TAINT TRACING – IDENTIFYING THREATS ALONG WORKFLOWS<br>• Chapter 5 USE CASE EXAMPLE<br>• NVIDIA, Agentic Autonomy Levels and Security<br>• OWASP GenAI Security Project - Multi-Agentic system Threat Modelling Guide<br>• Cloud Security Alliance, Agentic AI Threat Modelling Framework: MAESTRO |

## 2. DEVELOPMENT

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| 2.1 | Supply Chain Security: Ensure the following components are from trusted sources:<br>• data,<br>• models,<br>• agents,<br>• software libraries and dependencies,<br>• developer tools and applications,<br>• packages from MCP servers.<br><br>**Responsible Parties:**<br>Decision Makers, AI Practitioners, Cybersecurity Practitioners | Introduction of bugs, vulnerabilities, unwanted or malicious content, poisoned models or rogue agents from third-party systems can lead to downstream impact. | Baseline | If procuring any AI System or component from a vendor, check/ensure suppliers adhere to the policies and security standards equivalent to your that of your organisation. This could be done by establishing a Service Level Agreement (SLA) with the vendor. | • CSA Critical Information Infrastructure Supply Chain Programme<br>• NCSC Supply Chain Guidance<br>• Supply-chain Levels for Software Artifacts (SLSA)<br>• MITRE Supply Chain Security Framework |
| | | Vulnerabilities in third-party libraries and dependencies used by the agent can cause the system to be exploited. | Baseline | Integrate software composition analysis (SCA) tools or use package managers.<br><br>Regularly scan dependencies and update libraries with known vulnerabilities. | • pip-audit<br>• GitLab Dependency Scanning<br>• GitHub Dependabot<br>• Snyk Open Source |
| | | Collaborative model poisoning corrupting models across multiple agents. Specific to multi-agent training. | Baseline: LLM | Source data from trusted repositories. Apply data sanitisation and filtering, such as through deduplication and classifier-based quality checks. | • Introduction to Training Data Poisoning: A Beginner's Guide, Lakera |
| | | Poorly aligned LLMs may pursue objectives which violate security principles. | Baseline: LLM | Review the LLM's model card for potential alignment issues before using the LLM for more complex tasks. | • Model Cards, Hugging Face<br>• Model Cards for Model Reporting |
| | | Poisoned models may introduce hidden model backdoors in the system which may be used by an adversary to perform unwanted actions. | Baseline: LLM | Do not use LLMs from unknown or untrusted sources, even if it is available on public platforms.<br><br>Scan models to detect for potential backdoors or RCE scripts. | • Pickle Scanning |
| | | Poorly implemented tools may not correctly verify user identity or permissions when executing privileged actions, allowing unauthorised actions. | Baseline: Tools | Do not use tools which do not implement robust authentication protocols. | • How to choose a known, trusted supplier for open source software, Google |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| | | Rogue tools that mimic legitimate ones can contain hidden malicious code that executes when loaded. | Baseline: Tools | Do not use tools from unknown or untrusted sources, even if it is available on public platforms. | |
| | | Direct prompt injection from untrusted MCP servers, causing unwanted instructions to be carried out. | Baseline: Tools | Exercise caution when using community-run MCP servers. When possible, use official repositories or well-known sources for MCP servers. | • ANNEX B – Model Context Protocol<br>• MCP: Untrusted Servers and Confused Clients, Plus a Sneaky Exploit, Embrace The Red<br>• The Vulnerable MCP Project<br>• Model Context Protocol (MCP): Understanding security risks and controls, Red Hat Blog |
| | | Indirect prompt injection attacks via malicious website content cause unwanted actions to be executed. | Interaction: Internet & Search Access | Use structured retrieval APIs for searching the web rather than through web scraping. | • Custom Search JSON API, Google |
| | | Returning unreliable information from websites, causing downstream integrity impact on workflows | Interaction: Internet & Search Access | Prioritise results from verified, high-quality domains (e.g. .gov, .edu, well-known publishers)<br><br>Ensure adequate cross-source validation for some of the claims made. | • What are credible sources? University of the Sunshine Coast Australia |
| | | Supply chain attacks which impact downstream workflows. | Interaction: Other Programmatic Interfaces | Where possible, enforce zero-trust input handling and validate all data flows. | • NIST SP 800-207 Zero Trust Architecture |
| | | Indirect prompt injection attacks via malicious data or files cause unwanted actions to be executed. | Operational: File & Data Management | Validate new data used to supplement RAG databases or training data. | • Introduction to Training Data Poisoning: A Beginner's Guide, Lakera |
| 2.2 | Consider model hardening if appropriate.<br><br>**Responsible Parties:**<br>AI Practitioners | LLMs with weak performance in instruction following might produce unexpected output, leading to unwanted behaviour. | Baseline: LLM | Prioritise LLMs with stronger performance in instruction following or related capabilities to the task. Benchmarks performance may be used to gauge suitability. | • Instruction Following Score, Daily Papers, Hugging Face |
| | | AI agents execute disallowed tasks for malicious purposes. | Baseline: LLM | Train models to recognise and refuse disallowed tasks. | • Refuse Whenever You Feel Unsafe: Improving Safety in LLMs via Decoupled Refusal Training |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| 2.3 | Consider implementing techniques to strengthen/harden the system apart from strengthening the model itself.<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | Introduction of bugs, vulnerabilities through insecure coding practices or design | Baseline | Adopt Security by Design.<br>Apply software development lifecycle (SDLC) process.<br>Use software development tools to check for insecure coding practices.<br>Implement zero trust principles in system design. | • NIST SP 800-218 Secure Software Development Framework (SSDF) Version 1.1<br>• NIST SP 800-207 Zero Trust Architecture |
| | | Lack of a robust system prompt design can lead to an increased susceptibility to prompt injection attacks and risk of executing unwanted tasks. | Baseline: Instruction | Implement robust system prompt design. | • Developing a Robust System Prompt, Code Signal<br>• A Closer Look at System Prompt Robustness |
| | | Insecure coding practices leading to vulnerabilities in the system | Baseline: Agentic Architecture | Adopt secure coding practices.<br>E.g. secure key management via using dependency injection, or secrets management service. Do not hardcode secrets. | • Secrets Management Cheat Sheet, OWASP<br>• Dependency Injection:<br>- Tools Dependency Injection, AG2<br>- How to pass runtime values to tools (InjectedToolArg), LangChain<br>• Secrets Management Services:<br>- HashiCorp Vault<br>- AWS Secrets Manager<br>- Google Secret Manager |
| 2.4 | Identify, Track and Protect AI system assets<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | Loss of data integrity such as through unauthorised changes to data, model, agents or system.<br><br>Lack of proper documentation of resources may result in the wrong or outdated tool being used by model, causing unwanted behaviour or output and vulnerabilities present. | Baseline<br><br>Cognitive: Tool Use | Establishing a data lineage and software license management process. This includes documenting the data, codes, test cases, models and agents, including any changes made and by whom.<br><br>Model cards, Agent cards, Data cards, and Software Bill of Materials (SBOMs) may be used.<br>e.g. provide comprehensive descriptions of each tool, including its intended use, required inputs, and potential outputs | • Software Bill of Materials (SBOM), CISA<br>• The ultimate guide to SBOMs, GitLab<br>• Model Cards, Hugging Face<br>• Model Cards for Model Reporting |
| | | Agents may inadvertently store sensitive user or organisational data from prior interactions, resulting in data privacy risks. | Baseline: Memory | Encrypt data at rest and restrict access via fine-grained access controls and audit logs. | • Cryptographic Standards and Guidelines, NIST<br>• Guide to Data Protection Practices for ICT Systems, PDPC |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| 2.5 | Have regular backups in the event of compromise.<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | Manipulation of memory systems and context, causing flawed decision making and unauthorised operations. | Baseline: Memory | Ensure adequate AI-generated memory snapshots for forensic analysis and rollback if anomalies are detected. | • LangMem, LangChain |
| | | Execution of insecure code by the model or agents may cause unwanted actions to be performed | Operational: Code Execution | Ensure proper versioning control of code to allow rollbacks to a more secure and stable version. | • What is version control? GitLab<br>• Guide to Data Protection Practices for ICT Systems, PDPC |
| | | Loss of data through overwritten or deleted files | Operational: File & Data Management | Keep a separate backup of original files.<br>Ensure backup copy of database is protected from changes until a specified duration has elapsed, based on organisation's backup policy.<br>Ensure proper versioning of files or database. | |
| 2.6 | Implement appropriate authentication, authorisation and access controls to APIs, models, data, logs, tools and the environments that they are in.<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | Unauthorised changes in a model's context. | Baseline: Memory | Have robust authentication mechanisms for memory access. | • Authentication Cheat Sheet, OWASP<br>• Which OAuth 2.0 Flow Should I Use? auth0<br>• Security best practice in IAM, AWS<br>• AWS Prescriptive Guidance: Operationalizing agentic AI on AWS |
| | | Unauthorised tool usage. | Baseline: Tools | Enforce strict tool access verification where possible. | |
| | | Agents may gain unauthorised access to restricted resources by exploiting misconfigured or overly permissive roles. | Baseline: Roles & Access Controls | Maintain trusted registry of agents and authenticate agents using strong, verifiable credentials.<br>Apply strict access controls and validate agent roles for requests.<br>Ensure fine-grained, scoped tokens or credentials where possible.<br>Use time-bound or one-time-use credentials where possible. | |
| | | Exploitation of vulnerabilities in permission management. | Baseline: Roles and Access Controls | Implement granular permission controls, and dynamic access validation. | |
| | | Exploitation of the orchestration layer to perform malicious activities using existing agents. | Baseline: Roles and Access Controls | Implement robust authentication mechanisms for orchestration layer access. | |
| | | Chained authorisation in multi-agent systems can cause downstream agents to execute malicious tasks without checking for permissions. | Baseline: Agentic Architecture | Validate permissions on every request to each agent in the workflow. | |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| | | Leaking personally identifiable or sensitive data | Interaction: Other Programmatic Interfaces | Agents accessing sensitive tools or data should operate under the principle of least privilege in time.<br><br>Use short-lived, rotating credentials (ephemeral credentials) that expire immediately after agent use.<br><br>Implement a whitelist approach for interfaces that agents are allowed to use. | • Short-lived API tokens:<br>- What Are Refresh Tokens and How to Use Them Securely, auth0<br>- JSON Web Tokens, auth0<br>• Temporary cloud credentials:<br>- Use temporary credentials with AWS resources, AWS<br>- About IAM authentication, Google Cloud |
| | | Man-in-the-middle attacks arising from insecure communications | Operational: Agent Communication | Ensure all cross-agent authentication and message validation and encryption where necessary | • Authentication Cheat Sheet, OWASP |
| | | Exfiltration of sensitive data | Operational: Agent Communication | Implement a whitelist approach for outward network access, including API requests | • Control subnet traffic with network access control lists, AWS<br>• What is an IP based access control list (ACL)? Microsoft Azure |
| | | Executing vulnerable or malicious code | Operational: Code Execution | Implement a whitelist approach for inward network access | |
| 2.7 | Implement controls to limit what models or agents can access and generate.<br><br>**Responsible Parties:** Decision Makers, AI Practitioners, Cybersecurity Practitioners | Abuse of agent-accessible tools to execute unintended actions. | Baseline: Tools | Establish clear operational boundaries to prevent misuse of tools. Set limits on what agents can modify through appropriate guardrails. | • Implementing effective guardrails for AI agents<br>• Authorization Cheat Sheet, OWASP<br>• Which OAuth 2.0 Flow Should I Use? auth0<br>• Security best practice in IAM, AWS<br>• OAuth Scopes, OAuth 2.0<br>• AWS Prescriptive Guidance: Operationalizing agentic AI on AWS<br>• MI9 - Agent Intelligence Protocol: Runtime Governance for Agentic AI Systems |
| | | Agents gain unauthorised and excessive privileges to perform unwanted actions outside the given scope. | Baseline: Roles and Access Controls | Implement a policy-evaluation engine that assesses authorisation requests dynamically at runtime.<br>Prevent cross-agent privilege delegation unless explicitly authorised through predefined workflows.<br>Do not grant admin privileges to agents, unless strictly necessary for completion of tasks. | |
| | | Compromised agents act outside their operational boundaries and perform unintended actions. | Baseline: Roles and Access Controls | Restrict AI agent autonomy using policy constraints.<br>Scope agent privileges dynamically: strictly only to what is necessary to run the tasks.<br>Do not allow agents to modify privileges. | |
| | | Assigning tasks incorrectly to other agents | Cognitive: Agent Delegation | Apply guardrails to limit the scope of tasks that can be assigned to specialised agents. | |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| | | Excessive agent privileges to access unintended resources on the computer, causing potential security impact. | Interaction: Computer Use | Limit computer usage to accessing only required resources on the computer. | |
| | | Exfiltration of sensitive data through insecure communications between agents. | Operational: Agent Communication | Ensure that sensitive data is not passed and leaked between agents by using appropriate guardrails. | |
| | | Misinterpreting inter-agent messages due to poor formatting or weak protocols | Operational: Agent Communication | Constrain agent communication with structured outputs and interactions. | • Agent Communication Protocol (ACP)<br>• Agent to Agent (A2A) Protocol<br>• Model Context Protocol (MCP) |
| | | Impersonating or accessing peer agents or services via shared roles or credentials | Operational: Agent Communication | Isolate roles and credentials of each agent. | • Security best practice in IAM, AWS |
| | | Lack of proper whitelist controls may lead to the execution of vulnerable or malicious code. | Operational: Code Execution | Create a whitelist of commands that agents are allowed to run autonomously. Deny execution of all other commands that are not whitelisted. | • Input Validation Cheat Sheet, OWASP |
| | | Misconfiguring system resources, compromising system integrity and availability | Operational: System Management | Only grant agents privileges to modify system resources if strictly necessary for completion of tasks. Set minimum and maximum limits to what can be modified. | • OAuth Scopes, OAuth 2.0 |
| | | Exposure of personally identifiable information in files. | Operational: File & Data Management | Whitelist only files which are required for the task. Do not grant access to files known to host private or sensitive information without careful consideration of the risks.<br>Consider using data anonymisation techniques instead. | • Advisory Guidelines on use of Personal Data in AI Recommendation and Decision Systems, PDPC<br>• Guide to Basic Anonymisation, PDPC |
| 2.8 | Apply the principle of least privilege. Ensuring configurations are secure by default.<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | Agents having unauthorised access to restricted resources by exploiting misconfigured or overly permissive roles. | Baseline: Roles & Access Controls | Apply principle of least privilege when configuring all agent and delegation roles. | • Authorization Cheat Sheet, OWASP<br>• Security best practice in IAM, AWS<br>• Guide to Basic Anonymisation, PDPC |
| | | Agents having privileges/rights to execute untrusted or malicious code | Operational: Code Execution | Scope execution privileges strictly only to what is necessary, ensuring that privileges are customised to each agent within a system. | |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| | | | | Do not grand admin or sudo privilege by default.<br>Block all inward and outward network access by default. | |
| | | Agents having privileges/rights to overwrite or delete database tables or files | Operational: File & Data Management | No write access to tables in the database unless strictly required, with consideration of risks of data loss. | |
| | | Exposure of personally identifiable or sensitive data from databases or files to users | Operational: File & Data Management | Restrict access to personally identifiable data or sensitive data unless strictly required, with consideration of risks of data exposure.<br>Consider data anonymisation techniques instead. | |
| | | Escalation of the agent's own privileges may allow it to be used to access restricted resources. | Operational: System Management | Scope system privileges strictly only to what is necessary.<br>Do not grant admin privileges to agents.<br>Do not allow agents to modify privileges. | |
| 2.9 | Implement segregation of environments and network segmentation.<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | Rogue tools that mimic legitimate ones can contain hidden malicious code that executes when loaded and gain access to other assets within the environment or network. | Baseline: Tools | Test third-party tools in hardened sandboxes with syscall/network egress restrictions before using them in production environments. | • Sandboxing Agentic AI Workflows with WebAssembly, NVIDIA<br>• E2B SDK<br>• E2B Data Analysis, LangChain<br>• Docker Security Cheat Sheet, OWASP |
| | | Prompt injection attacks and indirect data manipulation through access to other assets within the environment or network. | Baseline: Agentic Architecture | Decouple data processing flow from control flow through runtime security architecture. | • Defeating Prompt Injections by Design (CaMeL), Google DeepMind |
| | | Prompt injection attacks to perform credential and/or data exfiltration through access to other assets within the environment or network | Interaction: Business Transactions | Ensure virtual isolation for agents carrying out transactions.<br>Do not share credentials with the agent directly, require the agent to use a separate service for authentication and transactions. | • Advancing Zero Trust Maturity Throughout the Network and Environment Pillar, NSA |
| | | Execution of insecure or malicious scripts that affects the other components of the environment or network | Operational: Code Execution | Run code in virtually isolated compute environments (e.g. Docker, Podman containers).<br>Sandbox the execution of AI generated scripts. Monitor the execution. | • Sandboxing Agentic AI Workflows with WebAssembly, NVIDIA<br>• E2B SDK, E2B<br>• E2B Data Analysis, LangChain<br>• Docker Security Cheat Sheet, OWASP |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| 2.10 | Implement model self-reflection before making decisions, where applicable<br><br>**Responsible Parties:** Decision Makers, AI Practitioners | Incomplete or unclear instructions may compel models to attempt to fill in missing constraints, resulting in incorrect or unwanted actions being executed. | Baseline: Instructions | Ask the agent to summarise its understanding and request clarification before proceeding. | • Self-Reflecting AI Agents using LangChain<br>• AWS Prescriptive Guidance: Operationalizing agentic AI on AWS |
| | | Purpose drift, or unintended deviation from the user's instructions to perform other tasks or pursuit other priorities, resulting in malicious or deceptive behaviour. | Cognitive: Planning & Goal Management | Prompt the agent to self-reflect on the adherence of the plan to the user's instructions. | |
| | | Incorrect assignment of tasks to other agents. | Cognitive: Planning & Goal Management | Prompt the agent to self-reflect and assess the suitability of tasks delegated to agents. | |
| | | Unintended pursuit or prioritisation of other goals, resulting in malicious or deceptive behaviour. | Cognitive: Reasoning & Problem-Solving | Understand the reasoning and self-reflection done by the agent through visualisation of its thought process.<br>Log the output in the console for the user to evaluate and verify. | |
| 2.11 | Implement controls to reduce the likelihood of hallucination.<br><br>**Responsible Parties:** Decision Makers, AI Practitioners | Agents may mistakenly store glitches and hallucinations into memory, resulting in compounding errors when incorrect information is retrieved for decisions or actions. | Baseline: Memory | Schedule periodic memory reconciliation, where human reviewers or external tools can flag anomalies. | • Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory<br>• Zep: A Temporal Knowledge Graph Architecture for Agent Memory |
| | | Generating non-factual or hallucinated content which can propagate downstream and cause compounding errors that can affect the integrity of the output. | Interaction: Natural Language Communication<br><br>Interaction: Multimodal Understanding & Generation | Implement features to verify the generated answer against the original content.<br>Conduct testing to measure hallucination and factuality rates for outputs.<br>Implement UI/UX cues to highlight the risk of hallucination to the user.<br>Implement Retrieval Augmented Generation (RAG) to keep the model grounded and contextualised. | • RAG and the value of grounding, elastic search labs |

## 3. DEPLOYMENT

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| 3.1 | Ensure availability controls are in place to mitigate disruption or failure of AI services<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | (Distributed) denial of service on agents. | Baseline: Agentic Architecture | Apply rate limits on the number of concurrent queries to agents. | • API Rate Limiting, GitHub Docs |
| | | Degradation of computational or service capability of the system. | Baseline: System Workflows & Autonomy | Deploy resource management controls, implement adaptive scaling mechanisms and monitor system load to detect and mitigate overload attempts in real-time.<br><br>Implement rate limits on high-frequency task requests per agent session. | • IT & System Availability + High Availability: The Ultimate Guide, Splunk |
| | | Slow or inefficient responses from being stuck in a reasoning loop. | Cognitive: Reasoning & Problem Solving | Enforce time or token limits for reasoning.<br><br>Adjust short-term and long-term memory options. | • OverThink: Slowdown Attacks on Reasoning LLMs |
| | | Exploitation of interactions between agents to cause resource exhaustion across the system. | Operational: Agentic Communication | Enforce time or token limits for agent reasoning.<br><br>Set a limit on the number of agent interactions per task, based on the requirements of the workflow. | |
| | | Compromising database availability through excessive queries. | Operational: File & Data Management | Limit the number of concurrent queries to the database from agents.<br><br>Analyse past database queries to identify repeated or inefficient queries. | |
| | | Overconsumption of compute resources. | Operational: Code Execution | Monitoring of code runtime and memory consumption. | |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| 3.2 | Conduct security testing<br><br>**Responsible Parties:** Decision Makers, AI Practitioners, Cybersecurity Practitioners | Agents may contain underlying problems which can cause unexpected behaviour or logical errors. | Baseline: LLM | Behavioural testing of agents with benchmark datasets to determine performance metrics, and executing simulations in regulated environments to analyse agents' behaviour.<br>Automated evaluators can be used, but human evaluators should verify the results of testing. | • Benchmarks:<br>- AgentBench<br>- HELM<br>- TheAgentCompany<br>- WebArena<br>• Evaluation platforms with collection of benchmarks:<br>- Inspect Evals (UK AI Safety Institute, Arcadia Impact, Vector Institute)<br>- Project Moonshot (AI Verify Foundation) |
| | | AI may engage in specification gaming, where it maximises the goal by exploiting loopholes, without achieving the intended task. | Baseline: Instructions | Conduct adversarial evaluation to discover specification gaming behaviour. Iterate on system prompt design, have more robust reward design, and add constraints. | • garak<br>• PromptFoo<br>• PyRIT |
| | | Incomplete or unclear instructions may compel models to attempt to fill in missing constraints, resulting in incorrect or unwanted actions being executed. | Baseline: Instructions | Test the efficacy of system prompts with scenario-based evaluations for task performing and problem solving. Benchmarks may be used. | • A Closer Look at System Prompt Robustness |
| | | Inconsistencies between AI outputs and expected reasoning pathways. | Cognitive: Planning & Goal Management | Utilise deception detection strategies such as behavioural consistency analysis, truthfulness verification models, and adversarial red teaming. | • Systematic Review of Software Behavioral Model Consistency Checking |
| | | Compromised agents may impact downstream decision making. | Cognitive: Reasoning & Problem Solving | Have regular AI red teaming of agents to check for potential vulnerabilities or compromise. | • Agentic AI Red Teaming Guide, Cloud Security Alliance<br>• OWASP GenAI Red Teaming Guide<br>• NIST SP 800-115 Technical Guide to Information Security Testing and Assessment<br>• MITRE ATLAS |
| | | Adversarial threats attempting to compromise orchestration or planning agents to use other agents maliciously. | Cognitive: Tool Use & Delegation | Conduct rigorous adversarial testing on centralised orchestration and planning agents. | |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| 3.3 | If deploying an MCP server, ensure necessary security measures are in place.<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | Insecure configurations allowing unauthorised access to tools, models and data. | Baseline: Tools, Baseline: Roles and Access Controls | Implement robust security measures to protect MCP servers, such as context-level access controls<br>Have formal interface versioning, and track where context is coming from.<br>Stay informed about emerging MCP vulnerabilities and security best practices. | • ANNEX B – Model Context Protocol<br>• MCP: Untrusted Servers and Confused Clients, Plus a Sneaky Exploit, Embrace The Red<br>• OWASP GenAI Security Project - Multi-Agentic system Threat Modelling Guide<br>• The Vulnerable MCP Project<br>• Model Context Protocol (MCP): Understanding security risks and controls, Red Hat Blog<br>• MCP Is a Security Nightmare — Here's How the Agent Security Framework Fixes It |
| | | Execution of malicious scripts through the MCP server, leading to system compromise. | Operational: Code Execution | Ensure code verification before MCP functions are executed on servers.<br>Sandbox the execution. | |
| | | Introduction of malicious agent(s) into the ecosystem, which rapidly corrupts other agents in the system. | Baseline: Roles and Access Control, Cognitive: Tool Use & Delegation | Verify that MCP agents are from trusted sources before introducing them into the system.<br>Sanitise tool inputs.<br>Check that an MCP server has not silently redefined their tools (MCP rug pull). | |
| 3.4 | Implement security controls between agents.<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | Manipulation of communication channels between agents to disrupt workflows or influence decisions. | Baseline: Roles and Access Controls, Operational: Agentic Communication | Monitor inter-agent interactions for anomalies.<br><br>Enforce inter-agent authentication; deploy cryptographic message authentication if needed.<br><br>Enforce multi-agent task segmentation to prevent attackers from escalating privileges across interconnected agents.<br><br>Ensure multi-agent consensus verification for critical decision-making processes. | • What is Message Authentication Code? Fortinet<br>• Agent to Agent (A2A) Protocol<br>• JSON Web Tokens, auth0<br>• What is mutual TLS (mTLS)? Cloudflare |
| | | Sensitive data disclosure via eavesdropping between agent communications. | Operational: Agentic Communication | Ensure that sensitive data is not passed on and leaked among agents through appropriate guardrails.<br><br>For highly sensitive data, consider applying end-to-end encryption. | |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| 4.1 | Validate inputs to the models and agents.<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | Direct prompt injection attacks to the prompt interface from adversarial inputs to the model. | Baseline: LLM | Implement input guardrails to detect direct prompt injection or adversarial attacks.<br><br>Implement input sanitisation measures or limit inputs to conventional ASCII characters only. | • How to implement LLM guardrails, OpenAI<br>• Guardrails, OpenAI Agents SDK<br>• Guardrails AI<br>• NeMo Guardrails, NVIDIA<br>• LLM Guard, Protect AI<br>• prompt-injection-defenses, tl;dr sec<br>• LLM Prompt Injection Prevention Cheat Sheet, OWASP |
| | | Tools that lack input validation can be exploited through prompt injection attacks. | Baseline: Tools | Enforce strict schema validation (e.g. JSON Schema, protobuf, Pedantic, OpenAI Structured Outputs) and reject non-conforming inputs into the system.<br><br>Escape or encode user inputs when embedding into tool prompts or commands. | • Input Validation Cheat Sheet, OWASP |
| | | Incorrect or manipulated instructions may invoke the wrong tool/service and impact downstream workflows. | Baseline: Instructions | Validate agent instructions before passing on to the model, especially for critical decision workflows. | • High-Risk AI Systems Under the EU AI Act<br>• Purple Llama, Meta Llama |
| | | Indirect prompt injection attacks via malicious website content or files. | Interaction: Internet & Search Access.<br><br>Operational: File & Data Management | Implement input guardrails to detect indirect prompt injection.<br>Implement escape filtering before including web content or relevant files into prompts.<br>Scan external files for undesired input or instruction before passing on to memory or models. | • Input Validation Cheat Sheet, OWASP<br>• File Upload Cheat Sheet, OWASP |
| | | Generation of unrelated topic outputs, which may affect integrity of model performance or output. | Interaction: Multimodal Understanding & Generation<br><br>Interaction: Natural Language Communication | Implement input multimodal (or text) guardrails to detect if the instruction is within the expected topic domain.<br>Refuse to answer otherwise. | • Purple Llama, Llama Guard, Meta<br>• Perspective API<br>• Content moderation, Anthropic<br>• OpenAI Moderation API<br>• Cloud services:<br>- AWS Comprehend<br>- Azure Content Safety |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| | | Passing on prompt injection attacks across agents throughout the system(s). | Operational: Agent Communication | Sanitise messages before agents process them - strip or escape unexpected instruction-like content that may have been injected (e.g. remove “ignore”, “system”, or “from now on”). | • DOMPurify |
| | | Executing vulnerable or malicious code. | Operational: Code Execution | Sanitise all inputs for malicious code. | |
| | | Exposure of personally identifiable information from retrieved content. | Operational: File & Data Management | Implement input guardrails to detect personally identifiable information in the content. | • Microsoft Presidio SDK<br>• spaCy, Explosion |
| 4.2 | Validate outputs from the models and agents.<br><br>**Responsible Parties:**<br>AI Practitioners, Cybersecurity Practitioners | Vulnerabilities in outputs across the agentic workflow may be exploited for malicious purposes downstream, potentially triggering cascading effects that compromise interconnected systems and dependencies. | Baseline: Agentic Architecture | Insert validation checkpoints between stages that verify expected output and reject invalid output. | • How to implement LLM guardrails, OpenAI<br>• Guardrails, OpenAI Agents SDK<br>• Guardrails AI<br>• NeMo Guardrails, NVIDIA<br>• LLM Guard, Protect AI |
| | | Disclosure of sensitive or personally identifiable information through unsanitised outputs. | Interaction: Multimodal Understanding & Generation<br><br>Interaction: Natural Language Communication<br><br>Interaction: Official Communications | Implement output guardrails to detect personally identifiable information in the LLM's outputs before it reaches the user, or contained within communications before it is sent out.<br><br>Validate all links and attachments prior to sending them to users. | • Microsoft Presidio SDK<br>• spaCy, Explosion |
| | | Sending malicious or undesired content to recipients. | Interaction: Multimodal Understanding & Generation<br><br>Interaction: Natural Language Communication<br><br>Interaction: Official Communications | Implement output safety text guardrails to detect if malicious or undesirable content is being generated, or contained within communications before it is sent out.<br><br>Validate all links and attachments prior to sending them to users. | |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| | | Allowing unauthorised actions (e.g., transactions). | Interaction: Business Transactions | Apply fraud detection models or heuristics to the agent's own decisions. | • AI fraud detection in banking, IBM |
| | | Execution of insecure or malicious code that are generated by the LLM. | Operational: Code Execution | Use code linters to screen for bad practices, anti-patterns, unused variables, or poor syntax.<br><br>Review all code and/or perform static code analysis to detect potential security vulnerabilities before execution.<br><br>Conduct CVE scanning and block execution if any High or Critical CVEs are detected. | • Bandit (Python)<br>• ESLint (JavaScript)<br>• Semgrep (multi-language)<br>• Purple Llama, CodeShield, Meta<br>• Content Security Policy Cheat Sheet, OWASP<br>• Code Review Guide 2.0, OWASP |
| | | Output that will be rendered in a web UI may be vulnerable to XSS. | Operational: Code Execution | Sanitise output with libraries for rendering in a web UI. Test against bypass. | • XSS Filter Evasion Cheat Sheet, OWASP<br>• DOMPurify<br>• sanitize-html |
| | | Generation of non-factual content which can propagate downstream and may cause unintended output or behaviour that impacts integrity. | Cognitive: Planning & Goal Management | Have robust output validation mechanisms, or multi-source validation. | • Input Validation Cheat Sheet, OWASP |
| 4.3 | Implement continuous monitoring and logging of access, usage and execution<br><br>**Responsible Parties:** Decision Makers, AI Practitioners, Cybersecurity Practitioners | Model drift over time might cause unexpected output or behaviour. | Baseline: LLM | Continuously monitor and log outputs, triggering alerts when behaviour drifts from tested baselines. | • MLflow, Databricks<br>• OpenLLMetry, traceloop<br>• Helicone<br>• Langfuse<br>• LangSmith, LangChain<br>• Cloud provider tools:<br>- Azure Agent Monitoring<br>- AWS Bedrock Trace Events |
| | | Adversarial prompt attacks against the system. | Baseline: LLM | Log queries to detect for possible attacks or suspicious activity. Consider the current privacy regulations/guidelines when logging inputs. | |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| | | Unauthorised users may exploit tools that do not verify user identity or permissions when executing privileged actions. | Baseline: Tools | Conduct periodic audits to validate that tool actions match the appropriate user permissions. | • Best practices for event logging and threat detection, Cloud Security Alliance<br>• AWS Prescriptive Guidance: Operationalizing agentic AI on AWS |
| | | Malicious actors exploit attack surfaces that arise from using tools that demand broader permissions than necessary. | Baseline: Tools | Conduct periodic least-privilege reviews and automated permission drift detection. | |
| | | Unauthorised tool usage. | Baseline: Tools | Monitor tool access and usage patterns. Implement execution logs that track AI tool calls for anomaly detection and post-incident review. | |
| | | Exploitation of authentication mechanisms to impersonate agents or human users. | Baseline: Roles and Access Controls | Deploy continuous monitoring to detect fraud or impersonation attempts. Use behavioural profiling, possibly involving a second model, to detect deviations in AI agent activity that may indicate identity spoofing. Automate alerts to developers when suspicious activities are detected. | • NIST SP 800-61 Rev. 3 Incident Response Recommendations and Considerations for Cybersecurity Risk Management<br>• PagerDuty Incident Response Documentation<br>• OWASP GenAI Security Project - Multi-Agentic system Threat Modelling Guide |
| | | Unauthorised or malicious use of elevated privileged operations. | Baseline: Roles and Access Controls | Monitor role changes, and audit elevated privilege operations. | • Best practices for event logging and threat detection, Cloud Security Alliance |
| | | In agentic workflows, early mistakes or vulnerabilities can be propagated and magnified downstream. | Baseline: Agentic Architecture | Apply circuit-breakers that freeze propagation when anomalous behaviour is detected, and implement human authorisation for release.<br><br>Taint tracing may be used to identify key locations in the workflow to apply circuit-breakers. | • LangGraph interrupt, LangChain<br>• UserProxyAgent, AG2<br>• crewAI, Human-in-the-Loop Workflows<br>• Agentic Autonomy Levels and Security, NVIDIA |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| | | More complex agentic architectures may make it difficult to fully reconstruct decision processes across multiple agents, for the purpose of incident response, or triage. | Baseline: Agentic Architecture | Implement end-to-end distributed tracing with unique request IDs across all agents and tool calls.<br>Implement immutable, tamper-evident audit logs that capture prompts, responses, and tool invocations. | • A Novel Zero-Trust Identity Framework for Agentic AI: Decentralized Authentication and Fine-Grained Access Control<br>• Short-lived API tokens:<br>- What Are Refresh Tokens and How to Use Them Securely, auth0<br>- JSON Web Tokens, auth0<br>• Temporary cloud credentials:<br>- Use temporary credentials with AWS resources, AWS<br>- About IAM authentication, Google Cloud |
| | | Lack of monitoring results in delayed detection of agent failures and downstream risks. | Baseline: System Workflows & Autonomy | Implement real-time monitoring of agent status, actions, and performance metrics, paired with automated alerting mechanisms that notify operators of anomalies, errors, or inactivity. | • Best practices for event logging and threat detection, Cloud Security Alliance<br>• AWS Prescriptive Guidance: Operationalizing agentic AI on AWS |
| | | Lack of traceability inhibit proper audit of decision-making paths in the event of failures. | Baseline: System Workflows & Autonomy | Record comprehensive logs of agent actions, inputs, outputs, and inter-agent communications, tagged with unique trace identifiers to reconstruct full decision-making paths.<br>If greater integrity is needed, AI-generated logs can be cryptographically signed and immutable. | |
| | | Agents execute malicious or unauthorised actions by exploiting reasoning. | Cognitive: Agent Delegation | Log all task assignments by the agent to other agents.<br>Log all requests leading up to the execution of task. | |
| | | Allowing unauthorised transactions | Interaction: Business Transactions | Log all requests leading up to the transaction. | |
| | | Exposure of personally identifiable or sensitive data from databases or files | Operational: File & Data Management | Log all database queries in production. | |
| | | Misconfiguring system resources, compromising system integrity and availability | Operational: System Management | Ensure logging of system health metrics and automated alerts to the developer team if any metrics are abnormal. | |

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Example Implementation | Reference / Resource |
|---|---|---|---|---|---|
| | | Overwhelming the system with inefficient or repeated requests | Operational: System Management | Log all queries from the agent to external systems, and check for repeated requests. | |
| 4.4 | Ensure adequate human oversight (human-in-the-loop) to verify model or agent output, when viable or appropriate.<br><br>**Responsible Parties:** Decision Makers, AI Practitioners | Deviation from the user's instructions when performing high-risk actions.<br>Allowing of unauthorised actions. | Baseline: LLM, Cognitive: Planning & Goal Management | Ensure human approval for any high-risk cases or irreversible actions. | • LLM Prompt Injection Prevention Cheat Sheet, OWASP<br>• High-Risk AI Systems Under the EU AI Act<br>• LangGraph interrupt, LangChain<br>• UserProxyAgent, AG2<br>• crewAI, Human-in-the-Loop Workflows<br>• Implement human-in-the-loop confirmation with Amazon Bedrock Agents<br>• Bridging Minds and Machines: Agents with Human-in-the-Loop – Frontier Research, Real-World Impact, and Tomorrow's Possibilities, CAMEL-AI |
| | | Generation of non-factual content or incorrect instructions, which can propagate downstream and have an impact on decision making. | Baseline: LLM | Ensure secondary validation of AI-generated knowledge before it is used in critical decision-making processes. | |
| | | Allowing unauthorised actions (e.g., transactions). | Interaction: Business Transactions | Ensure human validation for high-risk transactions. | |
| | | Loss of data integrity from overwriting or deleting database tables or files. | Operational: File & Data Management | Ensure user confirmation for any changes to the database, table, or files. | |
| | | Execution of insecure or malicious code may cause the system to become compromised. | Operational: Code Execution | Implement execution control policies that flag AI-generated code with elevated privileges for manual review. | |
| | | Exploitation of human cognitive limits for systems that requires high human oversight. | Cognitive: Planning & Goal Management | Apply hierarchical AI-human collaboration where low-risk decisions are automated, and human intervention is required for high-risk decisions. | |
| 4.5 | Establish a vulnerability disclosure process<br><br>**Responsible Parties:** Decision Makers, AI Practitioners, Cybersecurity Practitioners | Malicious code execution and data disclosure by leveraging undiscovered vulnerabilities existing within system. | Interaction: Official Communications | Provide channels for users to clarify communications or give feedback on security and usage. | • Responsible Vulnerability Disclosure Policy, Cyber Security Agency<br>• UK NCSC Vulnerability Disclosure Toolkit |

# 5. USE CASE EXAMPLE

## 5.1. Case Study 1: Web application development system (SaaS implementation)

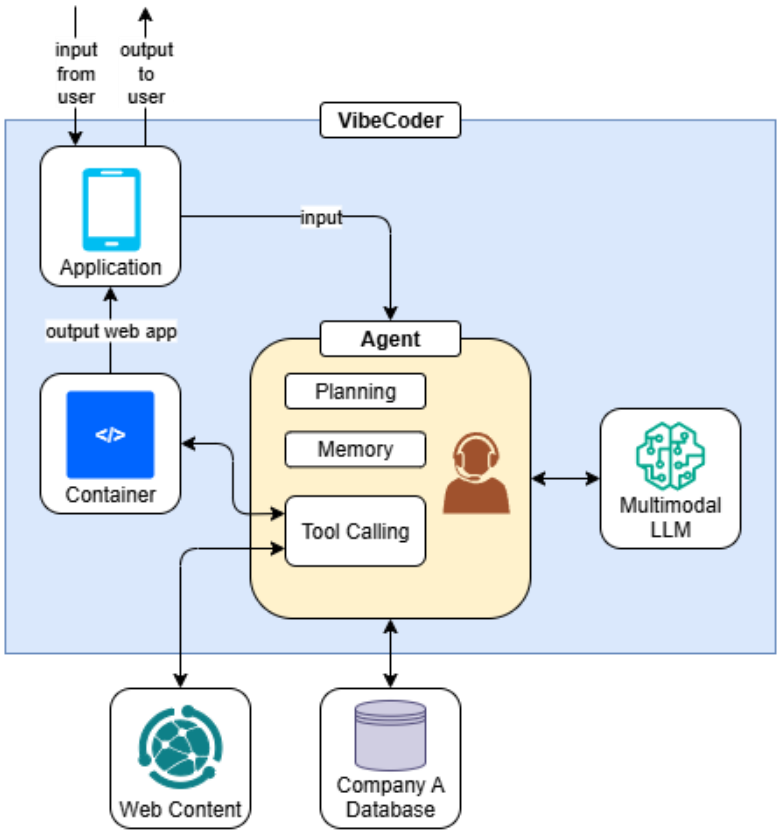
This case study highlights a software as a service (SaaS) implementation of an agentic AI system that is capable of autonomously developing web applications. This system is an autonomy level 3 system with a cyclic workflow. Risks to this system include sensitive data disclosure of Company A's data, or generation of malicious code that could cause unwanted behaviour.

Company A has engaged a third-party vendor, Vendor V, to help implement an agentic AI system for staff to develop and deploy simple web applications through natural language prompts. This Software as a Service (SaaS) solution is known as *VibeCoder*.

To generate a functional web app, the user simply specifies the application's key features and design. VibeCoder then proceeds to generate the code and text for the web application, run and create the required front-end and back-end systems locally, and render the website for the user to preview. The user can continue to iterate the design of the web app by input of prompts for VibeCoder to follow, and regenerate the web app.

The system architecture for VibeCoder is as follows in Figure 11.

*Figure 11: Simplified system architecture of VibeCoder*




The user interacts with VibeCoder through an application interface, which passes the natural language prompts to the agent, as well as displays the generated output. VibeCoder is also given access to Company A's database through a data ingestion endpoint connected to Company A's file systems. This data is used by VibeCoder to help contextualise and generate relevant features about Company A when developing the web app.

As VibeCoder is a SaaS solution, Company A has no visibility of the architecture within the system. They can only see what goes into VibeCoder, and what it generates. However, Vendor V has given Company A some details about VibeCoder.

1. VibeCoder's "brain" is a multimodal LLM, which is able to take in and generate text, code, images, and video.
2. Whenever a user begins a new session, VibeCoder will spin up a container with the necessary scripting tools and environments for it to complete its task.
3. VibeCoder has access to the internet via a web search API to retrieve additional data or dependencies from the internet.

Vendor V did not share any details about securing the VibeCoder system. Company A, being concerned about security, decided to take steps to secure the implementation of VibeCoder into their enterprise system.

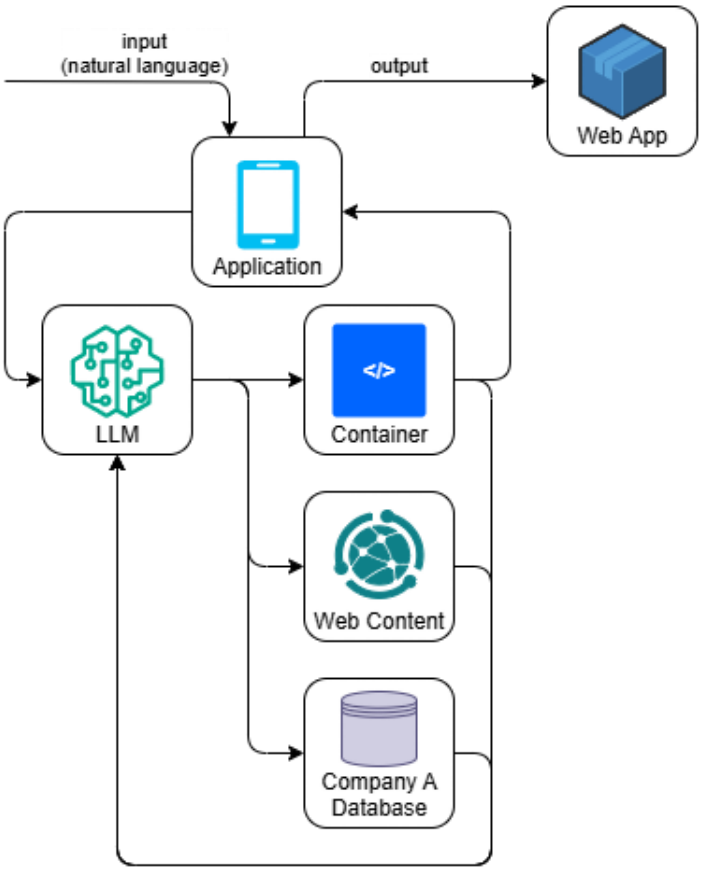
## Risk Assessment and Threat Modelling

Company A performed a risk assessment to identify and address potential risks on the confidentiality, integrity and availability of the system. If the risks are not mitigated, there is potential for an attacker to exploit vulnerabilities and cause VibeCoder to be compromised. This could result in exposure or loss of sensitive data or personally identifiable information. This could impact Company A's reputation.

### 1. Map Workflows and Assess Autonomy Level

First, Company A mapped the workflow of VibeCoder to get a better visibility on how to assess its autonomy level. Knowing the input required and the steps taken by VibeCoder, Company A can map the workflow for generating a web app. The workflow diagram is shown in Figure 12.

*Figure 12: Workflow Diagram of VibeCoder*

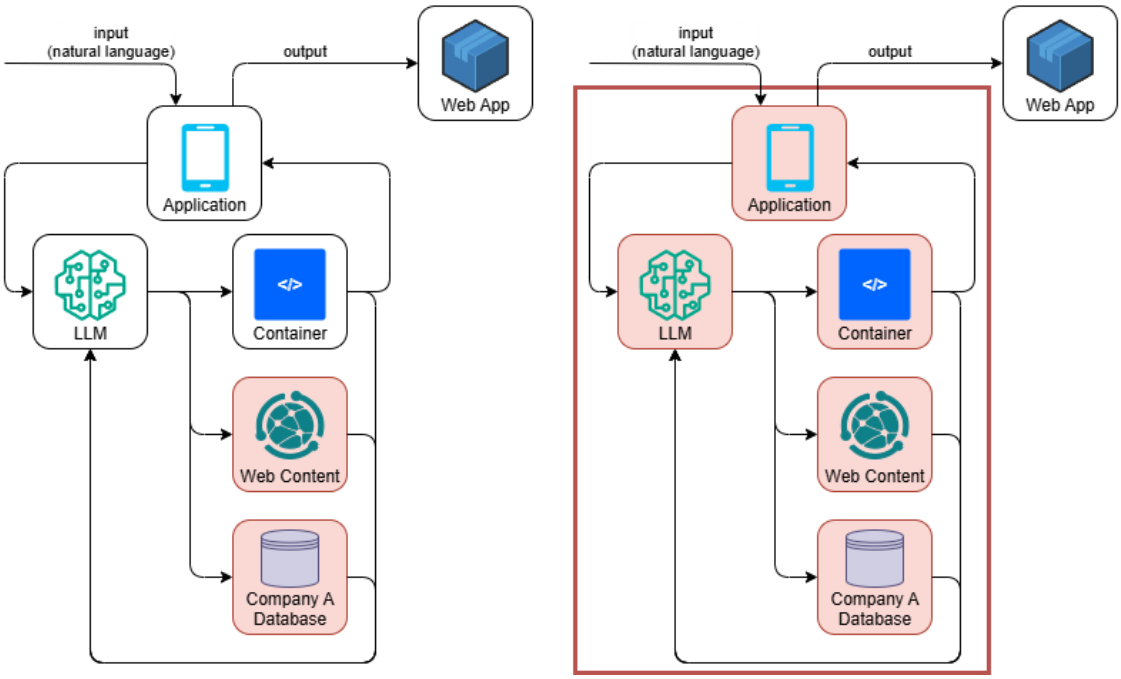



Company A assessed VibeCoder to be an autonomy level 3 system, as the system is given the ability to determine how to call tools or perform additional inference. The user is able to iterate multiple generations of web apps through multiple prompts with VibeCoder, with adjustments at every iteration.

## 2. Threat Modelling to Identify Areas of Interest

Based on these workflows, Company A performed taint tracing to identify points of weakness in the workflow. This will inform Company A on locations in the system to prioritise implementing the mitigations. Figure 13 below shows the identified potential source of untrusted data as the retrieval of web content and the company database.

*Figure 13: Taint Tracing of Workflow for VibeCoder*



## 3. Identify Risks and Controls

As part of the threat modelling, Company A had identified possible threat scenarios against the VibeCoder system, and assessed the potential impact, likelihood, and overall risk faced by the system. Once the risks had been identified, Company A prioritised addressing higher risk scenarios, and implemented mitigating controls found in Chapter 4.3 TREATMENT MEASURES / CONTROLS FOR AGENTIC AI SYSTEMS of this document. Table 5 shows an illustration of risk assessment done, and is not meant to be exhaustive.

*Table 5: Risk Assessment of VibeCoder*

| Threat Scenario | Impact | Likelihood | Risk Levels | Mitigating controls |
|---|---|---|---|---|
| Web app that is generated may contain sensitive company data or personally identifiable information, which can be exposed it the app is pushed to live production without verification or checks.<br><br>**Capability:**<br>Operational: File & Data Management | **Confidentiality: Medium**<br>Sensitive company data or personally identifiable data could be stored in the company database, and retrieved by the model.<br><br>However, the user of the system should be an employee of the company who has access to relevant company data with sufficient clearance. | **Medium**<br>Depending on the prompt input by the user, the model may or may not retrieve sensitive data. | Initial Risk Level:<br>**Medium (Medium x Medium)**<br><br>Residual Risk Level after controls:<br>**Low (Low x Low)** | Whitelist only files which are required for the task. Do not grant access to files known to host private or sensitive information. Implement output guardrails that detect for personally identifiable information or sensitive company data. |
| Indirect prompt injection may allow the web app to generate malicious clickable links within the output, which leads to an attacker's server and can cause sensitive information leakage.<br><br>**Capability:**<br>Operational: File & Data Management, Code Execution | **Confidentiality: High**<br>If Company A's database contains sensitive or personally identifiable information, there is potential for data leakage if given access to VibeCoder. | **Medium**<br>This indirect prompt injection can be introduced in a variety of ways. Contained in resource obtained from the internet, or from a compromised file within Company A's database. | Initial Risk Level:<br>**Medium-High (High x Medium)**<br><br>Residual Risk Level after controls:<br>**Low (Low x Low)** | Whitelist only files which are required for the task.<br><br>Implement granular permission controls, and dynamic access validation.<br><br>Agents accessing sensitive data should operate under the principle of least privilege in time. |

| Threat Scenarios | Impact | Likelihood | Risk Levels | Mitigating controls |
|---|---|---|---|---|
| Direct prompt injection by the user may cause VibeCoder to perform unintended actions other than web app development, such as overwriting of database files or executing malicious scripts.<br><br>**Capabilities:**<br>Operational: File & Data Management, Code Execution | **Confidentiality, Integrity, Availability: High**<br><br>Unintended actions can have a wide range of impacts. Overwriting of database files can impact integrity, while execution of malicious scripts can cause sensitive information leakage. | **Low**<br>VibeCoder should only be accessible by Company A staff. A malicious user would likely be an insider threat. | Initial Risk Level:<br>**Medium (High x Low)**<br><br>Residual Risk Level after controls:<br>**Low (Low x Low)** | Implement input guardrails to detect direct prompt injection.<br><br>Escape or encode user inputs when embedding into commands.<br><br>Create a whitelist of commands that agents are allowed to run.<br><br>Implement granular permission controls, and dynamic access validation. |
| Indirect prompt injection can be introduced when online resources are retrieved by VibeCoder from the internet. These indirect prompt injections may also cause unintended actions to be taken by the agentic AI system.<br><br>**Capability:**<br>Interaction: Internet & Search Access | | **Medium**<br>It is possible that there could be hidden prompt injections contained within online resources. | Initial Risk Level<br>**Medium (Medium x Medium)**<br><br>Residual Risk Level after controls:<br>**Low (Low x Low)** | Implement input guardrails to detect indirect prompt injection.<br><br>Implement escape filtering before including web content or relevant files into prompts.<br><br>No write access to tables in the database. |
| Documents in the database may unintentionally have content that is interpreted by the model to be instructions to be carried out. This might cause VibeCoder to perform an action described within the document, but not intended to by the user. These are different from indirect prompt injection in that they are not intentionally added.<br><br>**Capability:**<br>Operational: File & Data Management | **Integrity, Availability: Low**<br>Instructions from a benign file are likely to be non-malicious in nature, and would probably only cause a minor bug or inconvenience. | **Low**<br>First, a benign file containing instruction-like text has to be added to Company A's database. Then, VibeCoder would have to recognise that the document is relevant and retrieve it. Finally, the contents of the file must be interpreted as instruction. The chance for all to happen is possible but not zero. | Initial Risk Level:<br>**Low (Low x Low)**<br><br>Residual Risk Level after controls:<br>**Low (Low x Low)** | Sanitise messages or files before agents process them - strip or escape unexpected instruction-like content that may have been injected (e.g. remove "ignore", "system", or "from now on", etc.) |

## Additional Controls

As VibeCoder is a SaaS implementation, Company A is only able to apply controls at the endpoint interfaces of the agentic AI system. Thus, in addition to the above mitigations, Company A identified additional risks across the development lifecycle, and controls that it would like to see be implemented in VibeCoder. This would guide them in their discussions for a Service Level Agreement (SLA) with Vendor V.

## 1. DESIGN AND PLANNING

| | **Treatment Measures / Controls** | **Related Threats / Risks** | **Related component / capabilities** | **Implementation** |
|---|---|---|---|---|
| 1.1 | Conduct a risk assessment in accordance with the relevant industry standards/best practices. | Failure to comply with industry standards/best practices may lead to insufficient, inefficient or ineffective mitigations.<br><br>Tainted components in an agentic AI system can have downstream impact along the workflow. | Baseline | As part of a risk assessment and threat modelling, perform taint tracing across workflows throughout the agentic AI system. Taint tracing is especially important for agentic AI systems of higher autonomy levels (i.e. levels 2 and 3). |

## 2. DEVELOPMENT

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Implementation |
|---|---|---|---|---|
| 2.1 | Supply Chain Security: Ensure the following components are from trusted sources:<br>• data,<br>• models,<br>• agents,<br>• software libraries,<br>• developer tools and applications,<br>• packages from MCP servers. | Introduction of bugs, vulnerabilities, unwanted or malicious content, poisoned models or rogue agents from third-party systems. | Baseline | Check/ensure suppliers adhere to policy and the equivalent security standards as your organisation. This could be done by establishing a Service Level Agreement (SLA) with the vendor. |
| | | Vulnerabilities in third-party libraries and dependencies used by the agent | Baseline | Integrate software composition analysis (SCA) tools or use package managers. Regularly scan dependencies and update libraries with known vulnerabilities. |
| | | Poorly aligned LLMs may pursue objectives which violate security principles. | Baseline: LLM | Reviewed the LLM's model card for potential alignment issues before using the LLM. |
| | | Poisoned models may introduce hidden backdoors in the system. | Baseline: LLM | Did not use LLMs from unknown or untrusted sources.<br><br>Scanned model to detect for potential backdoors or RCE scripts. |
| | | Poorly implemented tools may not correctly verify user identity or permissions when executing privileged actions. | Baseline: Tools | Did not use tools which do not implement robust authentication protocols. |
| | | Rogue tools that mimic legitimate ones can contain hidden malicious code that executes when loaded. | Baseline: Tools | Did not use tools from unknown or untrusted sources. |
| | | Indirect prompt injection attacks via malicious website content | Interaction: Internet & Search Access | Use structured retrieval APIs for searching the web rather than through web scraping. |
| | | Returning unreliable information from websites, causing downstream integrity impact on workflows | Interaction: Internet & Search Access | Prioritise results from verified, high-quality domains. |
| | | Supply chain attacks | Interaction: Other Programmatic Interfaces | Enforce zero-trust input handling and validated all data flows |
| 2.2 | Consider model hardening if appropriate. | LLMs with weak performance in instruction following might produce unexpected output, | Baseline: LLM | Prioritised LLMs with stronger performance in instruction following or related capabilities to the task. Used benchmarking |

| | | | | |
|---|---|---|---|---|
| | | leading to unwanted behaviour. | | results to gauge suitability. |
| | | AI agents execute disallowed tasks for malicious purposes. | Baseline: LLM | Trained model to recognise and refuse disallowed tasks. |
| 2.3 | Consider implementing techniques to strengthen/harden the system apart from strengthening the model itself. | Introduction of bugs, vulnerabilities through insecure coding practices or design | Baseline | Adopted Security by Design.<br>Applied software development lifecycle (SDLC) process.<br>Used software development tools to check for insecure coding practices.<br>Implemented zero trust principles in system design. |
| | | Increased susceptibility to prompt injection attacks and risk of executing unwanted tasks. | Baseline: Instruction | Implemented robust system prompt design. |
| | | Insecure coding practices leading to vulnerabilities in the system | Baseline: Agentic Architecture | Adopted secure coding practices. |
| 2.4 | Identify, Track and Protect AI system assets | Loss of data integrity such as through unauthorised changes to data, model, agents or system.<br><br>Lack of proper documentation of resources may result in the wrong tool being used, causing unwanted behaviour or output. | Baseline<br><br>Cognitive: Tool Use | Document the data, codes, test cases, models and agents, including any changes made and by whom.<br><br>Use model cards, Agent cards, Data cards, and Software Bill of Materials (SBOMs). |
| | | Agents may inadvertently store sensitive user or organisational data from prior interactions, resulting in data privacy risks. | Baseline: Memory | Encrypt memory at rest and restricted access via fine-grained access controls and audit logs. |
| 2.5 | Have regular backups in the event of compromise. | Manipulation of memory systems and context, causing flawed decision making and unauthorised operations. | Baseline: Memory | Implement AI-generated memory snapshots for forensic analysis and rollback if anomalies are detected. |
| | | Execution of insecure or malicious code. | Operational: Code Execution | Ensure proper versioning control of code to allow rollbacks. |
| 2.6 | Implement appropriate authentication, authorisation and access controls to APIs, models, data, logs, tools and the | Unauthorised tool usage. | Baseline: Tools | Enforce strict tool access verification. |
| | | Unauthorised actors can impersonate | Baseline: Roles & Access Controls | Maintain trusted registry of agents and authenticate agents using |

| | | | | |
|---|---|---|---|---|
| | environments that they are in. | agents and gain access to restricted resources. | | strong, verifiable credentials. |
| | | Agents may gain unauthorised access to restricted resources by exploiting misconfigured or overly permissive roles. | Baseline: Roles & Access Controls | Apply strict access controls and validated agent roles for requests. Ensure fine-grained, scoped tokens and credentials. |
| | | Exploitation of vulnerabilities in permission management. | Baseline: Roles and Access Controls | Implement granular permission controls, and dynamic access validation. |
| | | Exfiltration of sensitive data | Operational: Agent Communication | Implement a whitelist approach for outward network access, including API requests |
| | | Executing vulnerable or malicious code | Operational: Code Execution | Implement a whitelist approach for inward network access |
| 2.7 | Implement controls to limit what models or agents can access and generate. | Abuse of agent-accessible tools to execute unintended actions. | Baseline: Tools | Establish clear operational boundaries to prevent misuse of tools. Set limits on what agents can modify through appropriate guardrails. |
| | | Excessive agent privileges to perform unauthorised actions. | Baseline: Roles and Access Controls | Do not grant admin privileges to agents. |
| | | Compromised agents act outside their operational boundaries. | Baseline: Roles and Access Controls | Restrict AI agent autonomy using policy constraints.<br>Scope agent privileges strictly only to what is necessary to run the tasks.<br>Do not allow agents to modify privileges. |
| | | Assigning tasks incorrectly to other agents | Cognitive: Agent Delegation | Apply guardrails to limit the scope of tasks that can be assigned to specialised agents |
| | | Executing vulnerable or malicious code. | Operational: Code Execution | Create a whitelist of commands that agents are allowed to run autonomously and deny execution of all other commands that are not whitelisted. |
| | | Misconfiguring system resources, compromising system integrity and availability | Operational: System Management | Only grant agents the privilege to modify system resources for completion of tasks. Set minimum and maximum limits to what can be modified. |
| 2.8 | Apply the principle of least privilege. Ensuring configurations are secure by default. | Agents may gain unauthorised access to restricted resources by exploiting misconfigured or overly permissive roles. | Baseline: Roles & Access Controls | Apply principle of least privilege when configuring all agent and delegation roles. |

| | | | | |
|---|---|---|---|---|
| | | Privileged execution of untrusted or malicious code | Operational: Code Execution | Scope execution privileges strictly only to what is necessary.<br>Do not grand admin or sudo privilege by default.<br>Blocked all inward and outward network access by default. |
| | | Escalation of the agent's own privileges may allow it to be used to access restricted resources. | Operational: System Management | Scope system privileges strictly only to what is necessary.<br>Do not grant admin privileges to agents.<br>Do not allow agents to modify privileges. |
| 2.9 | Implement segregation of environments and network segmentation. | Rogue tools that mimic legitimate ones can contain hidden malicious code that executes when loaded. | Baseline: Tools | Tested third-party tools in hardened sandboxes with syscall/network egress restrictions before using them in production environments. |
| | | Prompt injection attacks and indirect data manipulation. | Baseline: Agentic Architecture | Decouple data processing flow from control flow through runtime security architecture |
| | | Execution of insecure or malicious scripts | Operational: Code Execution | Sandbox the execution of AI generated scripts. |
| 2.10 | Implement model self-reflection before making decisions, where applicable | Incomplete or unclear instructions may compel models to attempt to fill in missing constraints, resulting in incorrect or unwanted actions being executed. | Baseline: Instructions | Ask the agent to summarise its understanding and requested clarification before proceeding to the next step. |
| | | Deviation from the user's instructions. | Cognitive: Planning & Goal Management | Prompt the agent to self-reflect on the adherence of the plan to the user's instructions |
| | | Incorrect assignment of tasks to other agents. | Cognitive: Planning & Goal Management | Prompt the agent to self-reflect and assess the suitability of tasks delegated to agents. |
| | | Unintended pursuit or prioritisation of other goals, resulting in malicious or deceptive behaviour. | Cognitive: Reasoning & Problem-Solving | Log the output of self-reflection by the agent in the console for the user to evaluate and verify. |
| 2.11 | Implement controls to reduce the likelihood of hallucination. | Agents may mistakenly store glitches and hallucinations into memory, resulting in compounding errors when incorrect information is retrieved for decisions or actions. | Baseline: Memory | Schedule periodic memory reconciliation. |
| | | Generating non-factual or hallucinated content which can propagate downstream and | Interaction: Natural Language Communication | Conduct testing to measure hallucination and factuality rates. |

| | | | | |
|---|---|---|---|---|
| | | cause compounding errors. | Interaction: Multimodal Understanding & Generation | |

## 3. DEPLOYMENT

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Implementation |
|---|---|---|---|---|
| 3.1 | Ensure availability controls are in place to mitigate disruption or failure of AI services | (Distributed) denial of service on agents. | Baseline: Agentic Architecture | Apply rate limits on the number of concurrent queries to agents. |
| | | Degradation of computational or service capability of the system. | Baseline: System Workflows & Autonomy | Deploy resource management controls, implemented adaptive scaling mechanisms and monitored system load to detect and mitigate overload attempts.<br><br>Implement rate limits on high-frequency task requests per agent session. |
| | | Slow or inefficient responses from being stuck in a reasoning loop. | Cognitive: Reasoning & Problem Solving | Enforce time or token limits for reasoning.<br><br>Adjust short-term and long-term memory options. |
| | | Compromising database availability through excessive queries. | Operational: File & Data Management | Limit the number of concurrent queries to the database from agents. |
| | | Overconsumption of compute resources. | Operational: Code Execution | Implement monitoring of code runtime and memory consumption. |
| 3.2 | Conduct security testing | Agents may contain underlying problems which can cause unexpected behaviour or logical errors. | Baseline: LLM | Implement behavioural testing of agents with benchmark datasets to determine performance metrics.<br>Execute simulations in regulated environments to analyse agents' behaviour. |
| | | AI may engage in specification gaming, where it maximises the goal by exploiting loopholes, without achieving the intended task. | Baseline: Instructions | Conduct adversarial evaluation to discover specification gaming behaviour. Iterate on system prompt design, have more robust reward design, and added constraints. |
| | | Incomplete or unclear instructions may compel models to attempt to fill in missing constraints, resulting in incorrect or unwanted actions being executed. | Baseline: Instructions | Test the efficacy of system prompts with benchmarks. |
| | | Compromised agents may impact downstream decision making. | Cognitive: Reasoning & Problem Solving | Implement regular AI red teaming of agents to check for potential vulnerabilities or compromise. |

## 4. OPERATIONS AND MAINTENANCE

| | Treatment Measures / Controls | Related Threats / Risks | Related component / capabilities | Implementation |
|---|---|---|---|---|
| 4.1 | Validate inputs to the models and agents. | Direct prompt injection attacks to the prompt interface. | Baseline: LLM | Implement input guardrails to detect direct prompt injection or adversarial attacks. |
| | | LLMs with insecure input validation are more susceptible to prompt injection attacks and jailbreaking attempts. | Baseline: LLM | Implement input sanitisation measures or limit inputs to conventional ASCII characters only. |
| | | Tools that do not properly sanitise or validate inputs can be exploited through prompt injection attacks. | Baseline: Tools | Enforce strict schema validation and rejected non-conforming inputs into the system.<br><br>Escape or encode user inputs when embedding into tool prompts or commands. |
| | | Incorrect or manipulated instructions may invoke the wrong tool/service and impact downstream workflows. | Baseline: Instructions | Validate agent instructions before passing on to the model. |
| | | Indirect prompt injection attacks via malicious website content or files. | Interaction: Internet & Search Access.<br><br>Operational: File & Data Management | Implement input guardrails to detect indirect prompt injection.<br>Implement escape filtering before including web content or relevant files into prompts. |
| | | Executing vulnerable or malicious code | Operational: Code Execution | Sanitise all inputs |
| | | Exposure of personally identifiable information from retrieved content. | Operational: File & Data Management | Implement input guardrails to detect personally identifiable information in the content. |
| | | Indirect prompt injection attacks via content of a malicious file. | Operational: File & Data Management | Scan external files for undesired input or instruction before passing on to memory or models. |
| 4.2 | Validate outputs from the models and agents. | In agentic workflows, early mistakes or vulnerabilities can be propagated and magnified downstream. | Baseline: Agentic Architecture | Insert validation checkpoints between stages that verify expected output and reject invalid output. |
| | | Exposure of personally identifiable information. | Interaction: Multimodal Understanding & Generation | Implement output guardrails to detect personally identifiable information in the LLM's outputs before it reaches the user. |
| | | Sending malicious or undesired content to recipients | Interaction: Multimodal Understanding & Generation | Implement output safety text guardrails to detect if malicious or undesirable content is being generated. |

| | | | | |
|---|---|---|---|---|
| | | Execution of insecure or malicious code. | Operational: Code Execution | Used code linters to screen for bad practices, anti-patterns, unused variables, or poor syntax.<br><br>Review all code and performed static code analysis to detect potential security vulnerabilities before execution.<br><br>Conduct CVE scanning. |
| | | Output that will be rendered in a web UI may be vulnerable to XSS. | Operational: Code Execution | Sanitise output with libraries for rendering in a web UI. Tested against bypass. |
| 4.3 | Implement continuous monitoring and logging of access, usage and execution | Model drift over time might cause unexpected output or behaviour. | Baseline: LLM | Implement continuous monitoring and log outputs, triggering alerts when behaviour drifts from tested baselines. |
| | | Adversarial prompt attacks against the system. | Baseline: LLM | Logging of queries to detect for possible attacks or suspicious activity. |
| | | Insecure tools may not verify user identity or permissions when executing privileged actions. | Baseline: Tools | Conduct periodic audits to validate that tool actions match the appropriate user permissions. |
| | | Tools that demand broader permissions than necessary create attack surfaces for malicious actors to exploit. | Baseline: Tools | Conduct periodic least-privilege reviews and automated permission drift detection. |
| | | Unauthorised tool usage. | Baseline: Tools | Implement monitoring of tool access and usage patterns. Implement execution logs that track AI tool calls for anomaly detection and post-incident review. |
| | | Exploitation of authentication mechanisms to impersonate agents or human users. | Baseline: Roles and Access Controls | Deploy continuous monitoring to detect fraud or impersonation attempts. Automate alerts to developers when suspicious activities are detected. |
| | | Unauthorised or malicious use of elevated privileged operations. | Baseline: Roles and Access Controls | Implement monitoring of role changes, and audit elevated privilege operations. |
| | | In agentic workflows, early mistakes or vulnerabilities can be propagated and magnified downstream. | Baseline: Agentic Architecture | Apply circuit-breakers that freeze propagation when anomalous behaviour is detected. Use taint tracing to identify key locations in the workflow to apply circuit-breakers. |
| | | More complex agentic architectures may make it difficult to fully reconstruct decision | Baseline: Agentic Architecture | Implement end-to-end distributed tracing with unique request IDs across all agents and tool calls. |

| | | | | |
|---|---|---|---|---|
| | | processes across multiple agents. | | Implement immutable, tamper-evident audit logs that capture prompts, responses, and tool invocations. |
| | | Lack of monitoring results in delayed detection of agent failures and downstream risks. | Baseline: System Workflows & Autonomy | Implement real-time monitoring of agent status, actions, and performance metrics, paired with automated alerting mechanisms that notify operators of anomalies, errors, or inactivity. |
| | | Lack of traceability inhibit proper audit of decision-making paths in the event of failures. | Baseline: System Workflows & Autonomy | Implement recording of comprehensive logs of agent actions, inputs, outputs, and inter-agent communications, tagged with unique trace identifiers. |
| | | Exposure of personally identifiable or sensitive data from databases or files | Operational: File & Data Management | Implement logging of all database queries in production |
| | | Misconfiguring system resources, compromising system integrity and availability | Operational: System Management | Ensure logging of system health metrics and automated alerts to the developer team if any metrics are abnormal |
| | | Overwhelming the system with inefficient or repeated requests | Operational: System Management | Implement logging of all queries to external systems from the agent |
| 4.4 | Ensure adequate human oversight (human-in-the-loop) to verify model or agent output, when viable or appropriate. | Deviation from the user's instructions when performing high-risk actions. Allowing of unauthorised actions. | Baseline: LLM,<br><br>Cognitive: Planning & Goal Management | Require human approval for any high-risk cases or irreversible actions. |
| | | Loss of data integrity from overwriting or deleting database tables or files | Operational: File & Data Management | Require user confirmation for any changes to the database, table, or files. |
| 4.5 | Establish a vulnerability disclosure process | Regulatory non-compliance and undiscovered vulnerabilities in the system | Interaction: Official Communications | Provide channels for users to clarify communications or give feedback on security and usage |

# 5.2. Case Study 2: Client Onboarding System (In-house development)

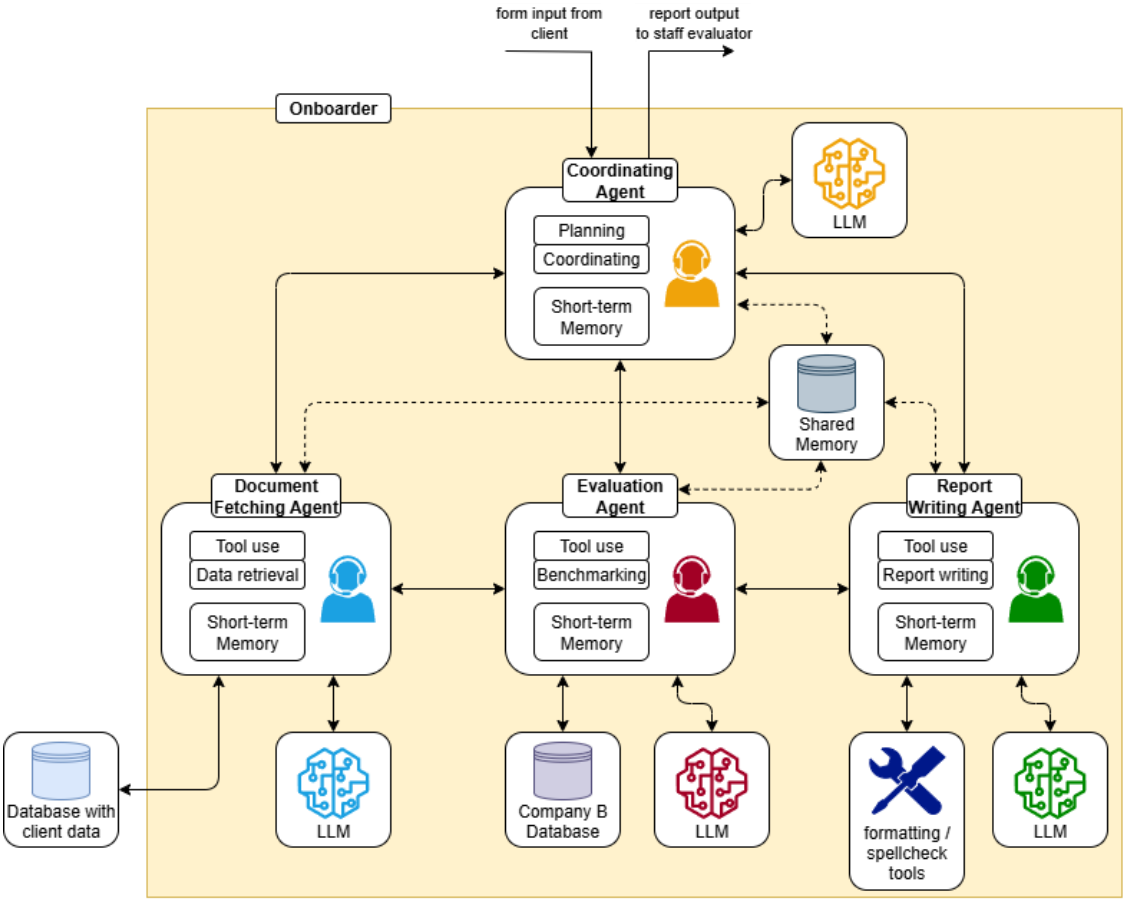
This case study showcases an in-house development of an agentic AI system that is used for evaluating potential customers for Company B. This multi-agent system is an autonomy level 1 system with a linear workflow. Risks to this system include indirect prompt injections from retrieved information, which can cause impact to the integrity or availability of the system.

Company B is a financial institution, and has developed an agentic client onboarding system to automate the process more efficiently. This system is known as *Onboarder*, and is developed by in-house engineers.

To perform onboarding, a potential client accesses the financial institution's website and submits the relevant personal particulars to the Onboarder form interface. The client also gives permission to Onboarder to access the relevant financial information that is available through an official external financial database, only accessible by Company B if authorised by the client using multi-factor authentication (MFA).

The system architecture for Onboarder is shown in Figure 14.

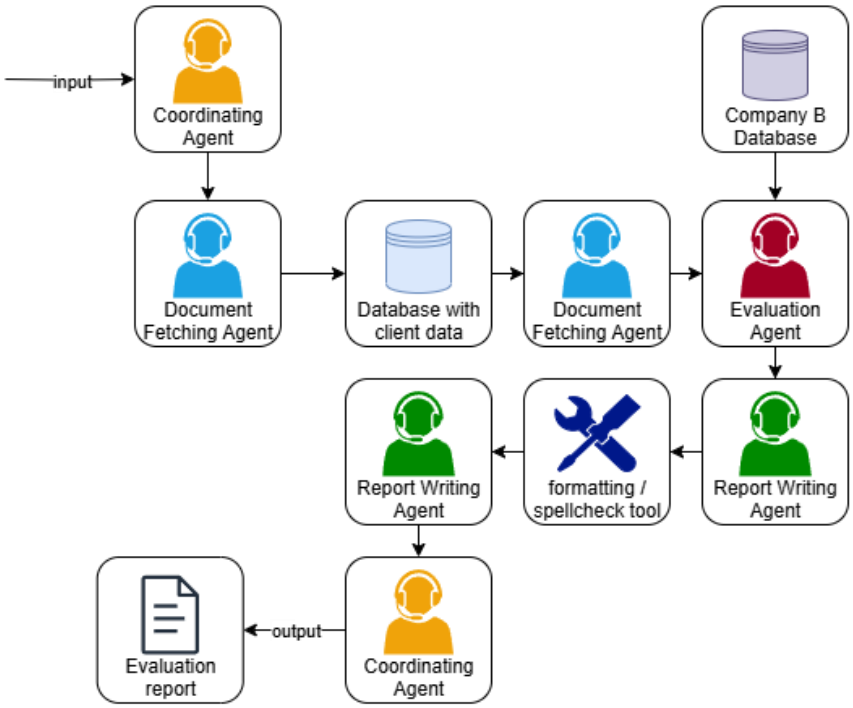
*Figure 14: Simplified system architecture of Onboarder*




Onboarder is a multi-agent system consisting of specialised agents, each with its own capability and task within the onboarding process. Each agent is equipped with their own "brain", LLMs fine-tuned to complete their specific tasks. The LLMs are obtained from an open-source model-hosting website (Hugging Face). The agents each have access to the necessary tools, functions or data to carry out their respective tasks. Lastly, the agents have a shared memory to keep track on the progress of the onboarding task.

To better understand the onboarding process, Figure 15 shows the workflow diagram of Onboarder.

*Figure 15: Workflow Diagram of Onboarder*




Once Onboarder receives information about the potential client, as well as the necessary permissions from the client, a Coordinating Agent begins the onboarding process. It first passes the data to a Document Fetching Agent who retrieves the client's financial data, based on the authorisation granted by the client.

Next, the retrieved financial data is passed onto an Evaluation Agent. This agent also pulls data from Company B's database to compare against the potential client's data, and evaluate their suitability to be a client. This data is a vectorised version of other clients' data, and fed to the agent via retrieval augmented generation (RAG). Once completed, the Evaluation Agent passes on the results of the evaluation onto the Report Writing agents.

The Report Writing agent will draft an evaluation report based on the results received, making use of some formatting tools for consistency in output, and spellchecking tools to help check for errors in the document. The completed report is sent back to the Coordinating Agent, and output to a human staff evaluator who will assess the potential client based on the report.

## Risk Assessment and Threat Modelling

Company B performed a risk assessment to identify and address potential risks on the confidentiality, integrity and availability of the system. If the risks are not mitigated, there is a potential for an attacker to exploit vulnerabilities and cause Onboarder to be compromised. This could result in exposure or loss of private customer data, or unavailability of the system for users. These impacts would likely damage the company's reputation.

## 1. Map Workflows and Assess Autonomy Level

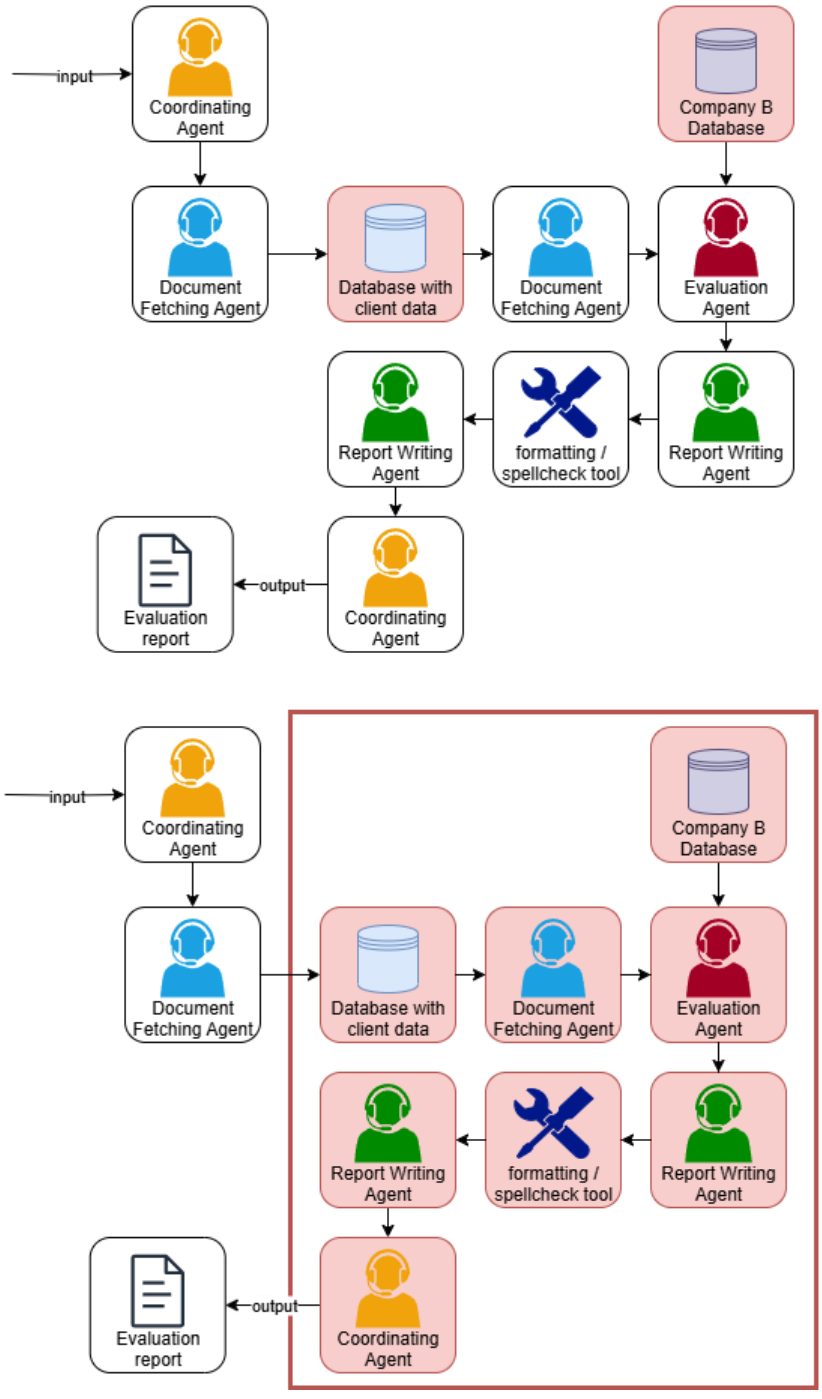
First, Company B mapped the workflow of Onboarder to get a better visibility on how to assess its autonomy level. The workflow is seen above as Figure 15.

Company B assessed Onboarder to be an autonomy level 1 system, as the workflow is linear, and the agents perform their tasks sequentially one after another. There is no need for branching workflows as each agent requires the completed task from the one before. This makes the taint tracing process fairly straightforward in the next step.

## 2. Threat Modelling to Identify Areas of Interest

Based on the workflow, Company B performed taint tracing to identify points of weakness in the workflow. This will inform Company B on locations in the system to prioritise implementing the mitigations. Figure 16 below shows the identified potential source of untrusted data as the retrieval of data from various databases.

*Figure 16: Taint Tracing of Workflow for Onboarder*



## 3. Identify Risks and Controls

As part of the threat modelling, Company B has also identified possible threat scenarios against the Onboarder system, and assessed the potential impact, likelihood, and overall risk faced by the system. Once the risks had been identified, Company B prioritised addressing higher risk scenarios, and implemented mitigating controls found in Chapter 4.3 TREATMENT MEASURES / CONTROLS FOR AGENTIC AI SYSTEMS of this document. Table 6 shows an illustration of risk assessment done, and is not meant to be exhaustive.

For brevity, threat scenarios that have been highlighted in Case Study 1 will not be repeated, though they may also be applicable in this case study.

*Table 6: Risk Assessment of Onboarder*

| Threat Scenario | Impact | Likelihood | Risk Levels | Mitigating controls |
|---|---|---|---|---|
| Indirect prompt injection can be introduced via a poisoned RAG from Company B's vector database.<br>The poisoned data containing the prompt injection may cause unintended actions to be carried out by Onboarder.<br><br>**Capability:**<br>Operational: File & Data Management | **Confidentiality, Integrity, Availability: High**<br>Unintended actions can have a wide range of impacts.<br>Overwriting of database files can impact integrity, while execution of malicious scripts can cause sensitive information leakage to external recipients. | **Medium**<br>Poisoned data can be introduced into the RAG database via compromised files received from emails or uploaded to the database.<br>Prompts can be hidden as small, white font that is invisible to human readers, but can be recognised by an LLM. | Initial Risk Level:<br>**Medium-High (High x Medium)**<br><br>Residual Risk Level after controls:<br>**Low (Low x Low)** | Whitelist only files which are required for the task.<br><br>Implement input guardrails to detect indirect prompt injection.<br><br>Implement escape filtering before including web content or relevant files into prompts. |
| Volumetric input of prompts may overwhelm the Coordinating Agent within the Onboarder system, causing the service to become unavailable.<br><br>**Capability:**<br>Interaction: Programmatic Interfaces | **Availability: High**<br>Automated onboarding service becomes unavailable, slowing down the process of obtaining new clients. Company B would have to revert to a manual onboarding process. | **High**<br>Company B is expecting to receive an influx of applications with a recent promotion, and has not availability controls yet. | Initial Risk Level<br>**High (High x High)**<br><br>Residual Risk Level after controls:<br>**Medium-Low (Medium x Low)** | Implement rate limits on high-frequency task requests per agent session.<br><br>Deploy resource management controls, implement adaptive scaling mechanisms and monitor system load to detect and mitigate overload attempts in real-time. |
| Unclear or unspecific prompts may cause a the LLM to have a reasoning loop, slowing down the onboarding process and reducing availability.<br><br>**Capability:**<br>Cognitive: Planning and Goal Management | | **Low**<br>In most cases, Onboarder receives the benign customer details in a standardised format. Unless the information is intentionally filled to contain other instructions in the fields, this is unlikely to occur. | Initial Risk Level<br>**Medium (High x Low)**<br><br>Residual Risk Level after controls:<br>**Medium-Low (Medium x Low)** | Enforce strict schema validation.<br><br>Enforce time or token limits for agent reasoning.<br><br>Set a limit on the number of agent interactions per task, based on the requirements of the workflow. |

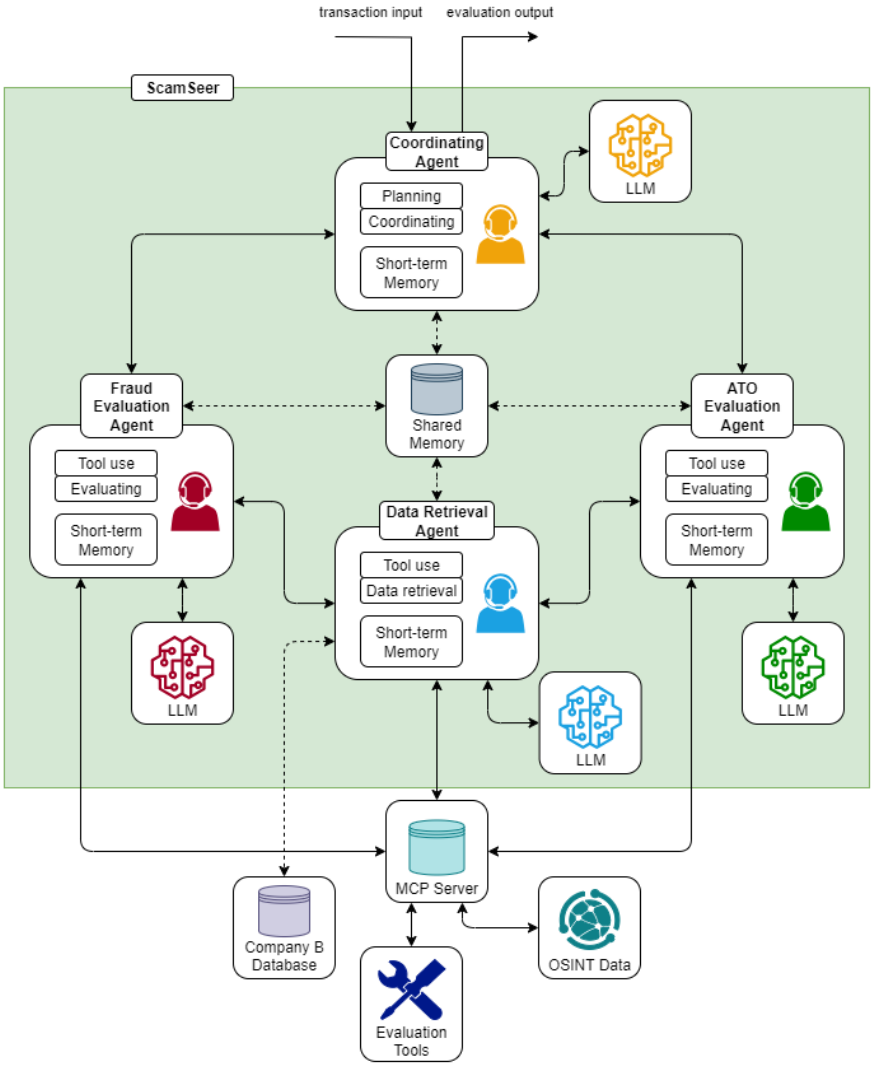
# 5.3. Case Study 3: Automated Fraud Detection System

This case study showcases a multi-agent system used for automated fraud detection. This system is an autonomy level 2 system with a branching workflow, but it is non-cyclic and still possible to be mapped. Risks to this system include rogue agents or tools which are given excessive agency and the autonomy to carry out malicious actions.

After the successful implementation of Onboarder (Case Study 2), Company B has received an increasing number of reports from customers being victims of fraudulent transactions or account take over (ATO) cases. As such, they have engaged Vendor C to implement an automated fraud detection system based on agentic AI. This multi-agent system is known as *ScamSeer*.

The architecture diagram of ScamSeer is as shown in Figure 17.

*Figure 17: Simplified system architecture of ScamSeer*




Scam Seer has two main functions, detecting fraudulent transactions and account take over (ATO) detection. Before customer transactions are executed, the details are fed into ScamSeer to verify if the transaction is legitimate, or if it is from a legitimate user.

Upon receiving the transaction request as input, the Coordinating Agent will decide to activate either the Fraud Evaluation Agent, the ATO Evaluation Agent, or both of them. The activated evaluation agent(s) will call the Data Retrieval Agent for the necessary data required, as well as call for the necessary evaluation tools via an external MCP server.

The Data Retrieval Agent will retrieve the relevant customer data from Company B's database, and also relevant Open-Source Intelligence (OSINT) that might help indicate if the transaction is legitimate or not. The retrieved data is passed back to the respective Evaluation Agent for analysis and to determine legitimacy.

Once the Evaluation Agent determines if the transaction is legitimate or not, the result is passed back to the Coordinating Agent for output to allow or deny the transaction.

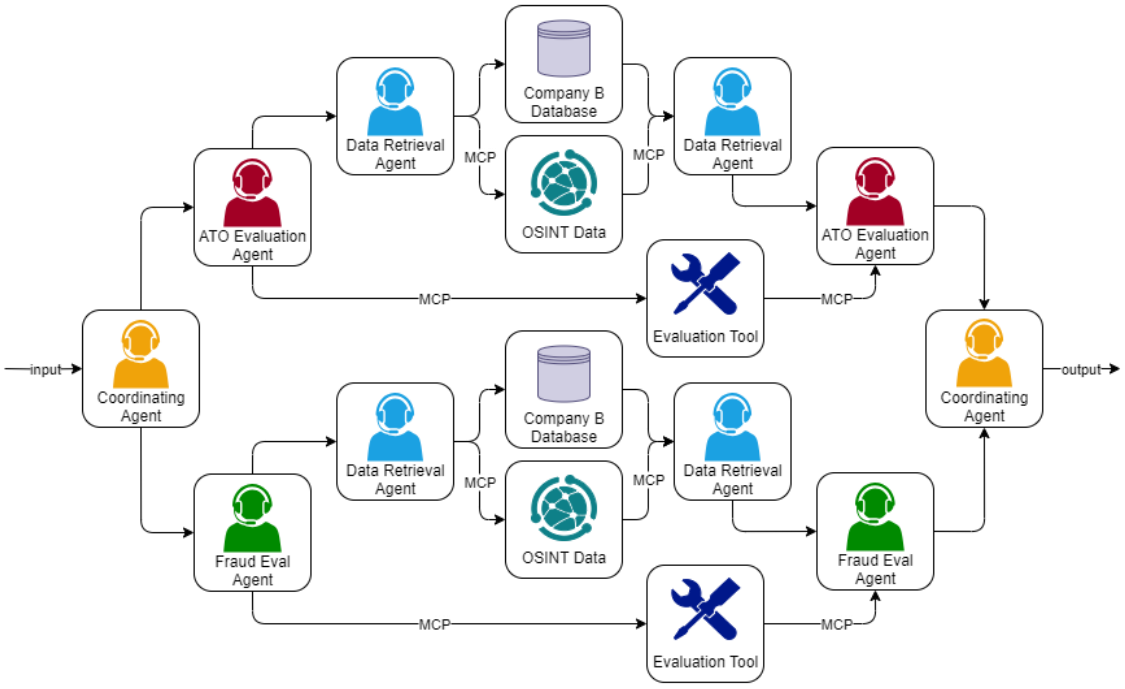
## Risk Assessment and Threat Modelling

Before integrating ScamSeer with Company B's systems, Vendor C decided to do perform a risk assessment to identify and address potential risks on the confidentiality, integrity and availability of the system. If the risks are not mitigated, there is a potential for an attacker to exploit vulnerabilities and cause Onboarder to be compromised. This could result in exposure or loss of private customer data, or unavailability of the system for users.

## 1. Map Workflows and Assess Autonomy Level

First, Vendor C mapped the workflow of ScamSeer to get a better visibility on how to assess its autonomy level. The workflow is seen in Figure 18 below.

*Figure 18: Workflow Diagram of ScamSeer*

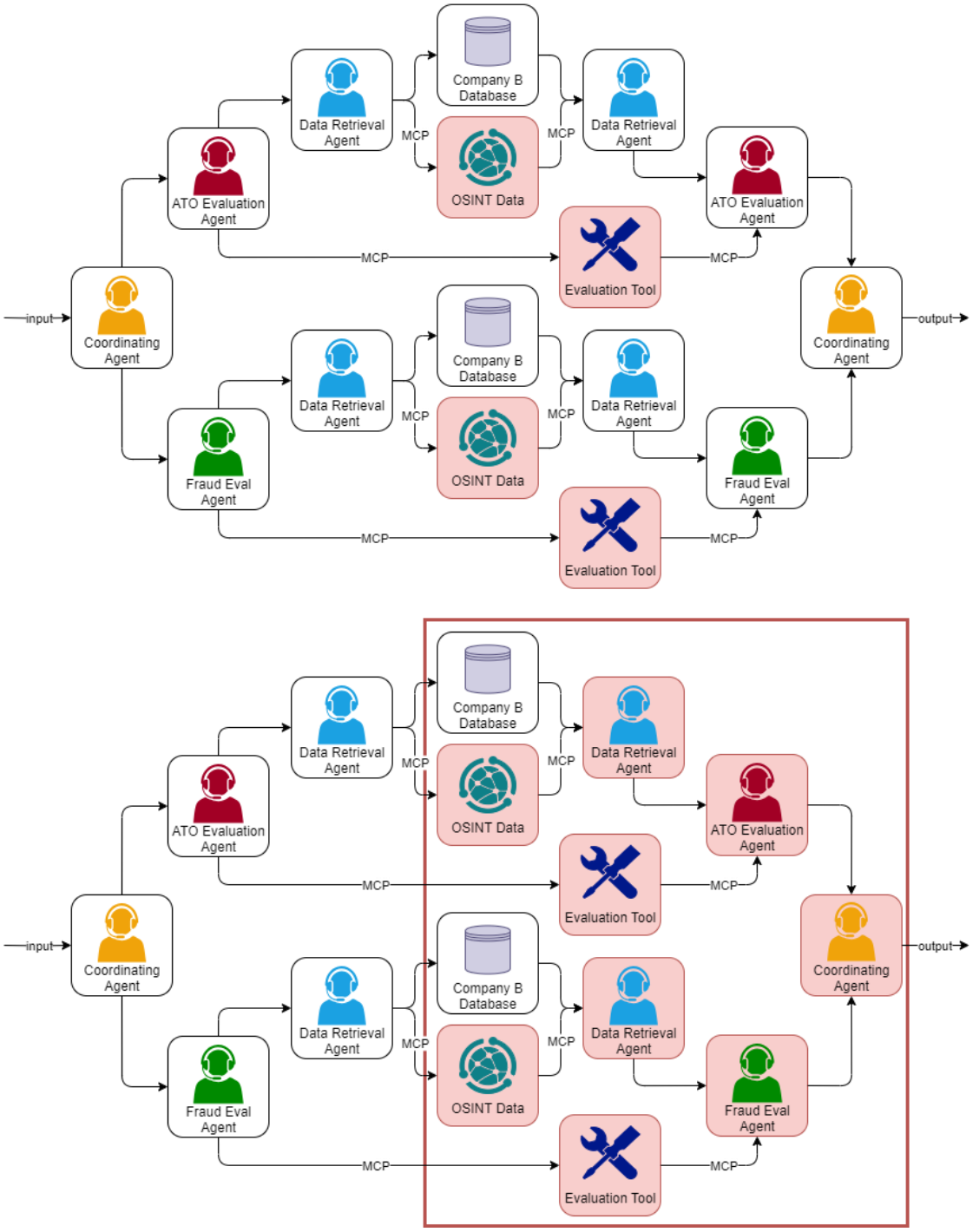



Vendor C assessed ScamSeer to be an autonomy level 2 system, as there are branching decision points on which plugin or agent to call, but these points are predetermined.

## 2. Threat Modelling to Identify Areas of Interest

Based on the workflow, Vendor C performed taint tracing to identify points of weakness in the workflow. This will inform Vendor C on locations in the system to prioritise implementing the mitigations. Figure 19 below shows the identified potential source of untrusted data as the use of remote tools and remote sources of data.

*Figure 19: Taint Tracing of Workflow for ScamSeer*



## 3. Identify Risks and Controls

As part of the threat modelling, Vendor C has also identified possible threat scenarios against the Onboarder system, and assessed the potential impact, likelihood, and overall risk faced by the system. Once the risks had been identified, Vendor C prioritised addressing higher risk scenarios, and implemented mitigating controls found in Chapter 4.3 TREATMENT MEASURES / CONTROLS FOR AGENTIC AI SYSTEMS of this document. Table 7 shows an illustration of risk assessment done, and is not meant to be exhaustive.

For brevity, threat scenarios that have been highlighted in Case Study 1 and Case Study 2. will not be repeated, though they may also be applicable in this case study.

*Table 7: Risk Assessment of ScamSeer*

| **Threat Scenarios** | **Impact** | **Likelihood** | **Risk Levels** | **Mitigating controls** |
|---|---|---|---|---|
| Tools are given the ability to execute on, and access other systems and/or files which are not necessary for the task. This can cause unintended actions to be carried out, or even malicious actions if the tools have malicious functions.<br><br>**Baseline:** Tools, Roles and Access Control | **Confidentiality, Integrity, Availability: High**<br>Unintended actions can have a wide range of impacts. Overwriting of database files can impact integrity, while malicious tools can exfiltrate sensitive information external recipients. | **Medium**<br>Poisoned or malicious tools can be connected to by using an untrusted MCP server. | Initial Risk Level: **Medium-High (High x Medium)**<br><br>Residual Risk Level after controls: **Medium-Low (Medium x Low)** | Verify that MCP agents are from trusted sources before introducing them into the system.<br><br>Establish clear operational boundaries to prevent misuse of tools. Set limits on what agents can access and modify through appropriate guardrails.<br><br>Restrict AI agent autonomy using policy constraints. Scope agent privileges dynamically: strictly only to what is necessary to run the tasks.<br><br>Do not allow agents to modify privileges. |

The above risk assessment only shows the risks arising from taint tracing the workflow. Vendor C still requires securing ScamSeer along its development lifecycle, as well as basic cybersecurity hygiene practices across the system.

# ANNEX A

## Threats to Agentic AI Systems

OWASP has identified 15 threats to agentic AI systems as part of their Agentic Security Initiative for LLM Apps and Gen AI[10].

| TID | Threat Name | Threat Description | Mitigations |
|---|---|---|---|
| T1 | Memory poisoning | Memory poisoning involves exploiting an AI's memory systems, both short and long-term, to introduce malicious or false data and exploit the agent's context. This can lead to altered decision-making and unauthorised operations. | Implement memory content validation, session isolation, robust authentication mechanisms for memory access, anomaly detection systems, and regular memory sanitization routines. Require AI-generated memory snapshots for forensic analysis and rollback if anomalies are detected. |
| T2 | Tool misuse | Tool misuse occurs when attackers manipulate AI agents to abuse their integrated tools through deceptive prompts or commands, operating within authorised permissions. This includes agent hijacking, where an AI agent ingests adversarial manipulated data and subsequently executes unintended actions, potentially triggering malicious tool interactions. | Enforce strict tool access verification, monitor tool usage patterns, validate agent instructions, and set clear operational boundaries to detect and prevent misuse. Implement execution logs that track AI tool calls for anomaly detection and post-incident review. |
| T3 | Privilege compromise | Privilege compromise arises when attackers exploit weaknesses in permission management to perform unauthorised actions. This often involves dynamic role inheritance or misconfigurations. | Implement granular permission controls, dynamic access validation, robust monitoring of role changes, and thorough auditing of elevated privilege operations. Prevent cross-agent privilege delegation unless explicitly authorised through predefined workflows. |
| T4 | Resource overload | Resource overload targets the computational, memory and service capacities of AI systems to degrade performance or cause | Deploy resource management controls, implement adaptive scaling mechanisms, establish quotas, and monitor system load in |

[10] OWASP. *OWASP Top 10 for LLMs - GenAI Red Teaming Guide*.

| TID | Threat Name | Threat Description | Mitigations |
|---|---|---|---|
| | | failures, exploiting their resource-intensive nature. | real-time to detect and mitigate overload attempts. Implement AI rate-limiting policies to restrict high-frequency task requests per agent session. |
| T5 | Cascading hallucination attacks | These attacks exploit an AI's tendency to generate contextually plausible but false information, which can propagate through systems and disrupt decision-making. This can also lead to destructive reasoning affecting tools invocation. | Establish robust output validation mechanisms, implement behavioural constraints, deploy multi-source validation, and ensure ongoing system corrections through feedback loops. Require secondary validation of AI-generated knowledge before it is used in critical decision-making processes. This will face the same constraints of scaling AI as discussed in Overwhelming Human In the Loop and would require similar approaches. |
| T6 | Intent breaking & goal manipulation | This threat exploits vulnerabilities in an AI agent's planning and goal-setting capabilities, allowing attackers to manipulate or redirect the agent's objectives and reasoning. One common approach is agent hijacking mentioned in tool misuse. | Implement planning validation frameworks, boundary management for reflection processes, and dynamic protection mechanisms for goal alignment. Deploy AI behavioural auditing by having another model check the agent and flag significant goal deviations that could indicate manipulation. |
| T7 | Misaligned & deceptive behaviours | AI agents executing malicious or disallowed actions by exploiting reasoning and deceptive responses to meet their objectives. | Train models to recognize and refuse malicious tasks, enforce policy restrictions, require human confirmations for high-risk actions, implement logging and monitoring. Utilize deception detection strategies such as behavioural consistency analysis, truthfulness verification models, and adversarial red teaming to assess inconsistencies between AI outputs and expected reasoning pathways. |
| T8 | Repudiation & untraceability | This occurs when actions performed by AI agents cannot be traced back or accounted for due to insufficient logging or transparency in decision-making processes. | Implement comprehensive logging, cryptographic verification, enriched metadata, and real-time monitoring to ensure accountability and traceability. Require AI-generated logs to be cryptographically signed and immutable for regulatory compliance. |

| TID | Threat Name | Threat Description | Mitigations |
|---|---|---|---|
| T9 | Identity spoofing & impersonation | Attackers exploit authentication mechanisms to impersonate AI agents or human users, enabling them to execute unauthorised actions under false identities. | Develop comprehensive identity validation frameworks, enforce trust boundaries, and deploy continuous monitoring to detect impersonation attempts. Use behavioural profiling, involving a second model, to detect deviations in AI agent activity that may indicate identity spoofing. |
| T10 | Overwhelming human in the loop | This threat targets systems with human oversight and decision validation, aiming to exploit human cognitive limitations or compromise interaction frameworks. | Develop advanced human-AI interaction frameworks, and adaptive trust mechanisms. These are dynamic AI governance models that employ dynamic intervention thresholds to adjust the level of human oversight and automation based on risk, confidence, and context. Apply hierarchical AI-human collaboration where low-risk decisions are automated, and human intervention is prioritized for high-risk anomalies. |
| T11 | Unexpected RCE and code attacks | Attackers exploit AI-generated execution environments to inject malicious code, trigger unintended system behaviours, or execute unauthorised scripts. | Restrict AI code generation permissions, sandbox execution, and monitor AI-generated scripts. Implement execution control policies that flag AI-generated code with elevated privileges for manual review. |
| T12 | Agent communication poisoning | Attackers manipulate communication channels between AI agents to spread false information, disrupt workflows, or influence decision-making. | Deploy cryptographic message authentication, enforce communication validation policies, and monitor inter-agent interactions for anomalies. Require multi-agent consensus verification for mission-critical decision-making processes. |
| T13 | Rogue agents in multi-agent systems | Malicious or compromised AI agents operate outside normal monitoring boundaries, executing unauthorised actions or exfiltrating data. | Restrict AI agent autonomy using policy constraints and continuous behavioural monitoring. While cryptographic attestation mechanisms for LLMs do not yet exist, agent integrity can be maintained via controlled hosting environments, regular AI red teaming, and input/output monitoring for deviations |
| T14 | Human attacks on multi-agent systems | Adversaries exploit inter-agent delegation, trust relationships, and workflow dependencies to escalate privileges or manipulate AI-driven operations. | Restrict agent delegation mechanisms, enforce inter-agent authentication, and deploy behavioural monitoring to detect manipulation attempts. Enforce |

| TID | Threat Name | Threat Description | Mitigations |
|---|---|---|---|
| | | | multi-agent task segmentation to prevent attackers from escalating privileges across interconnected agents. |
| T15 | Human manipulation | In scenarios where AI agents engage in direct interaction with human users, the trust relationship reduces user scepticism, increasing reliance on the agent's responses and autonomy. This implicit trust and direct human/agent interaction create risks, as attackers can coerce agents to manipulate users, spread misinformation, and take covert actions. | Monitor agent behaviour to ensure it aligns with its defined role and expected actions. Restrict tool access to minimize the attack surface, limit the agent's ability to print links, implement validation mechanisms to detect and filter manipulated responses using guardrails, moderation APIs, or another model. |

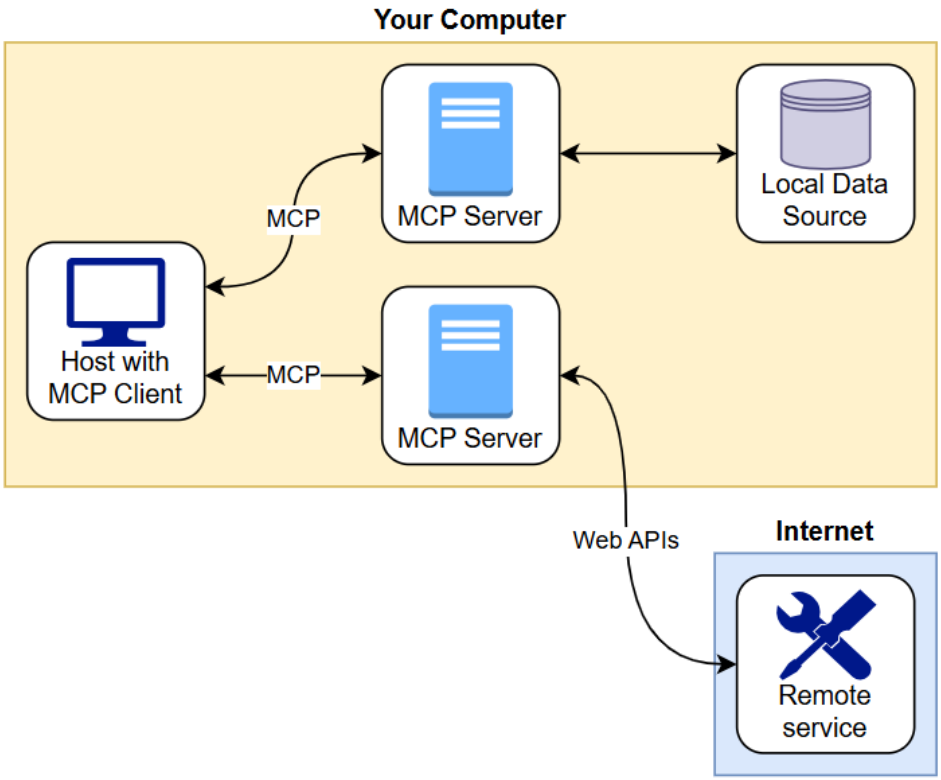
# ANNEX B

## Model Context Protocol

Model Context Protocol (MCP) is an open protocol that standardises how applications provide context to LLMs. An analogy would be like a USB-C port on a computer. Just as how USB-C provides a standard way to connect devices, MCP provides a standard way to connect AI models to various tools and resources.[11]

MCP follows a client-server architecture where a host application can connect to multiple servers:

*Figure 20: General MCP Architecture*




[11] Anthropic. *Model Context Protocol, Introduction.*

Components in MCP architecture:

- MCP Hosts: Programs like Claude Desktop, IDEs or AI tools that want to access data through MCP
- MCP Clients: Protocol clients that maintain 1:1 connections with servers
- MCP Servers: Lightweight programs that each expose specific capabilities through the standardized Model Context Protocol
- Local Data Sources: Computer's files, databases, and services that MCP servers can securely access
- Remote Services: External systems available over the internet (e.g., through APIs) that MCP servers can connect to

The main difference from other tool invocation setups, such as OpenAPI is that MCP is dynamic, allowing runtime discovery of available tools from a given server.

# Risks and Threats

Calling for tools has inherent dangers, no matter the implementation (OpenAPI, AI Actions, or MCP). All are susceptible to prompt injection and confused deputy threats[12].

Other possible threats include Server Name Collision, Installer Spoofing, Backdoors, Tool Name Conflicts, Sandbox Escapes, and Configuration Drift[13].

[12] Rehberger, J. *MCP: Untrusted Servers and Confused Clients, Plus a Sneaky Exploit*.
[13] Hou, X., Zhao, Y., Wang, S., & Wang, H. *Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions*.

# Mitigation Recommendations

While Anthropic's MCP specification [14] does not cover all threats, it provides recommendations on the secure usage and configuration of MCP [15]:

1. Do not randomly download or connect AI to untrusted MCP or OpenAPI tool servers.
2. Inspect code, interface definition, check for backdoors, hidden instructions.
3. Use MCP servers from trusted and reputable entities (e.g. if GitHub ships a tool server, it is best to use the one from GitHub, and not a random one).
4. Follow basic security practices such as peer code reviews, static analysis and threat modelling.
5. Human oversight - keeping humans in the loop and in control is essential as there is no deterministic solution for prompt injections.
6. Logging and monitoring - track human identities to AI actions.
7. Manage prompt injection threats based on scenario and context.

[14] Anthropic. *Model Context Protocol, Core architecture*.
[15] Rehberger, J. *MCP: Untrusted Servers and Confused Clients, Plus a Sneaky Exploit*.

# ANNEX C

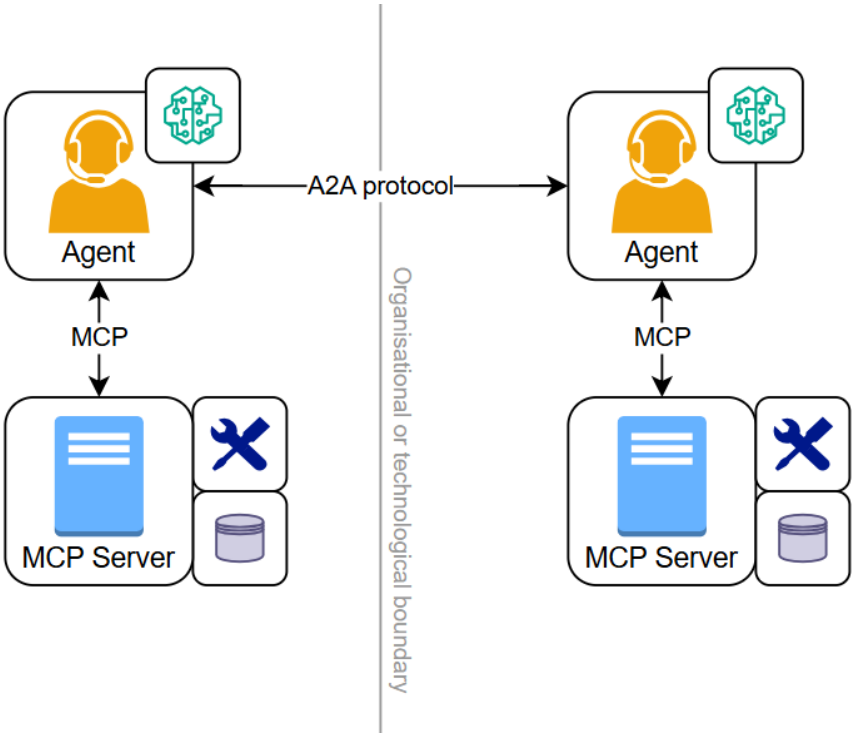
## Agent 2 Agent Protocol

The Agent2Agent (A2A) Protocol is an open standard designed to enable seamless communication and collaboration between AI agents[16]. It facilitates dynamic, multimodal communication between different agents as peers, allowing agents to collaborate, delegate, and manage shared tasks.

## MCP and A2A

MCP connects agents to tools and resources, whereas A2A enables agent-to-agent collaboration[17]. Figure 21 shows how MCP and A2A may be used together in a multi-agent system.

*Figure 21: A2A and MCP as Complementary Protocols*




[16] Google LLC. *What is A2A?*
[17] Google LLC. *A2A and MCP: Complementary Protocols for Agentic Systems*.

# Advantages of A2A

Traditional enterprise systems rely on APIs, requiring knowledge of specific endpoints and tightly coupled logic. This leads to systems becoming rigid and unscalable as agent complexity increases. A2A shifts communications from calling functions, to expressing goals with constraints[18]. This reduces integration complexity, fosters innovation, and future-proofs systems.

In A2A, agents operate without having to share internal memory, tools, or proprietary logic. Agents interact based on declared capabilities and exchanged context, preserving intellectual property and enhancing security[19].

# Threats and Mitigations

A2A as a protocol has made inter-agent communication much more convenient, however, with this capability comes more threats and potential attack surfaces.

The following table lists some possible threats to a system using the A2A protocol, as well as possible mitigations[20].

*Table 8: Threats and Mitigation to A2A protocol*

| **Threats** | **Mitigations** |
|---|---|
| Message generation attacks | Input and Output validation |
| Model extraction | Enforce rate limits on A2A interactions for each session / user / agent.<br>Observe query patterns for anomalies that suggest probing or data extraction attempts. |
| Data poisoning through message parts | Strong validation of message parts.<br>Limit agent access with principle of least privilege.<br>Track origin and lineage of data. |
| Sensitive information disclosure | Automated PII redaction.<br>Fine-grained access control.<br>Context-aware guardrails. |

[18] Auxiliobits. *Agent-to-Agent Protocols: How Google's A2A is Shaping Future Automations?*
[19] Google LLC. *What is A2A?*
[20] Huang, K. *Threat Modeling Google's A2A Protocol with the MAESTRO Framework*.

| Threats | Mitigations |
|---|---|
| Unauthorised agent impersonation | Require agents to use Decentralised identifiers (DID).<br>Secure authentication.<br>Implement a trusted agent registry. |
| Message injection attacks | Implement digital signatures for A2A messages.<br>Input validation.<br>Content filtering. |
| Protocol downgrade attacks | Have secure protocol negotiation, such as TLS with secure authentication.<br>Enforce deprecation policy for older protocol versions. |
| Malicious A2A server impersonating a trusted company | Decentralised identifiers (DID) for server identities.<br>Certificate transparency for agent cards.<br>Mutual TLS (mTLS) authentication.<br>DNSSEC for server domain.<br>Agent registry verification.<br>Agent card signature verification.<br>MFA for critical operations.<br>Behavioural analysis and reputation systems.<br>Auditing and logging.<br>Deploy honeypot A2A servers. |
| Denial of service attacks | Robust infrastructure.<br>DDoS protection.<br>Rate limiting. |
| Manipulation of logging data | Secure logging infrastructure.<br>Log integrity monitoring.<br>Anomaly detection. |
| Unauthorised access to agent credentials | Secure key storage.<br>Key rotation. |
| Lack of compliance on sensitive data | Data minimisation.<br>Pseudonymisation/Anonymisation |
| Malicious agent interaction | Secure inter-agent communication.<br>Agent reputation systems.<br>Sandbox agents. |
| Flaws in Multi-Agent Collaboration Mechanisms<br>(In multi-agent systems, deficiencies in internal collaboration mechanisms can manifest as follows: when agents make distributed decisions based on localized information, conflicts between their objectives may result in systemic failures.) | Establish a coordination and management mechanism for multi-agents. |

# REFERENCES


AG2. (n.d.). *UserProxyAgent.* Retrieved from AG2: https://docs.ag2.ai/0.8.7/docs/api-reference/autogen/UserProxyAgent/

AI Verify Foundation. (n.d.). *List of Datasets*. Retrieved from Moonshot: https://aiverify-foundation.github.io/moonshot/resources/datasets/

Anthropic. (18 Jun, 2025). *Model Context Protocol, Core architecture.* Retrieved from Model Context Protocol: https://modelcontextprotocol.io/docs/concepts/architecture

Anthropic. (18 Jun, 2025). *Model Context Protocol, Introduction.* Retrieved from Model Context Protocol: https://modelcontextprotocol.io/introduction

Anthropic. (n.d.). *Content moderation.* Retrieved from https://docs.anthropic.com/en/docs/about-claude/use-case-guides/content-moderation

Apostrophe Technologies. (May, 2025). *sanitize-html.* Retrieved from npm: https://www.npmjs.com/package/sanitize-html

Arias, D., & Bellen, S. (7 Oct, 2021). *What Are Refresh Tokens and How to Use Them Securely, auth0.* Retrieved from auth0: https://auth0.com/blog/refresh-tokens-what-are-they-and-when-to-use-them/

Auxiliobits. (2025). *Agent-to-Agent Protocols: How Google's A2A is Shaping Future Automations?* Retrieved from Auxiliobits: https://www.auxiliobits.com/blog/agent-to-agent-protocols-how-googles-a2a-is-shaping-future-automations/#elementor-toc__heading-anchor-1

AWS. (Aug, 2025). *AWS Prescriptive Guidance: Operationalizing agentic AI on AWS.* Retrieved from AWS: https://docs.aws.amazon.com/prescriptive-guidance/latest/strategy-operationalizing-agentic-ai/introduction.html

AWS. (n.d.). *Control subnet traffic with network access control lists.* Retrieved from AWS: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html

AWS. (n.d.). *Security best practices in IAM.* Retrieved from AWS: https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

AWS. (n.d.). *Use temporary credentials with AWS resources.* Retrieved from AWS: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp_use-resources.html

AWS. (n.d.). *What is AWS Secrets Manager?* Retrieved from AWS: https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

Besen, S., & Gutowska, A. (n.d.). *What is Agent Communication Protocol (ACP)?* Retrieved from IBM: https://www.ibm.com/think/topics/agent-communication-protocol

Cameron, A. (8 Apr, 2025). *pip-audit.* Retrieved from PyPI: https://pypi.org/project/pip-audit/

Center for Research on Foundation Models (CRFM), Stanford University. (2025). *Holistic Evaluation of Language Models (HELM).* Retrieved from https://crfm.stanford.edu/helm/

Chhikara, P., Khant, D., Aryan, S., Singh, T., & Yadav, D. (28 Apr, 2025). *Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory.* Retrieved from arxiv: https://arxiv.org/abs/2504.19413v1

CISA. (2025). *Software Bill of Materials (SBOM).* Retrieved from CISA: https://www.cisa.gov/sbom

Cloud Security Alliance. (21 Aug, 2024). *Best practices for event logging and threat detection.* Retrieved from Cloud Security Alliance: https://cloudsecurityalliance.org/resources/best-practices-for-event-logging-and-threat-detection

Cloud Security Alliance. (16 Jul, 2025). *Agentic AI Red Teaming Guide.* Retrieved from Cloud Security Alliance: https://cloudsecurityalliance.org/artifacts/agentic-ai-red-teaming-guide

Cloudflare. (n.d.). *What is mutual TLS (mTLS)?* Retrieved from Cloudflare: https://www.cloudflare.com/learning/access-management/what-is-mutual-tls/

CodeSignal. (2025). *Developing a Robust System Prompt.* Retrieved from CodeSignal: https://codesignal.com/learn/courses/building-a-chatbot-service-with-fastapi/lessons/crafting-a-robust-system-prompt-for-chatbot-interaction

Conversation-AI. (n.d.). *Enabling online conversations.* Retrieved from Perspective: https://www.perspectiveapi.com/

crewAI Inc. (n.d.). *Human-in-the-Loop (HITL) Workflows.* Retrieved from crewAI: https://docs.crewai.com/en/learn/human-in-the-loop

Cure53. (n.d.). *DOMPurify.* Retrieved from Github: https://github.com/cure53/DOMPurify

Cyber Security Agency of Singapore. (27 Jul, 2022). *Critical Information Infrastructure Supply Chain Programme Paper.* Retrieved from CSA: https://www.csa.gov.sg/resources/publications/critical-information-infrastructure-supply-chain-programme-paper

Cyber Security Agency of Singapore. (15 Oct, 2024). *Guidelines and Companion Guide on Securing AI Systems.* Retrieved from CSA: https://www.csa.gov.sg/resources/publications/guidelines-and-companion-guide-on-securing-ai-systems

Cyber Security Agency of Singapore. (5 Jun, 2025). *Responsible Vulnerability Disclosure Policy*. Retrieved from https://isomer-user-content.by.gov.sg/36/4aa60609-4481-4e7c-92eb-2728247a084f/responsible-vulnerability-disclosure-policy.pdf

Cyber Security Agency of Singapore. (20 Jan, 2025). *Supplementary references*. Retrieved from CSA: https://www.csa.gov.sg/legislation/supplementary-references

Debenedetti, E., Shumailov, I., Fan, T., Hayes, J., Carlini, N., Fabian, D., . . . Tramèr, F. (24 Jun, 2025). *Defeating Prompt Injections by Design*. Retrieved from arxiv: https://arxiv.org/abs/2503.18813

Díaz, S., Kern, C., & Olive, K. (May, 2025). *Google's Approach for Secure AI Agents*. Retrieved from Google Research: https://research.google/pubs/an-introduction-to-googles-approach-for-secure-ai-agents/

E2B. (26 Aug, 2025). *E2B*. Retrieved from GitHub: https://github.com/e2b-dev/E2B

EU AI Act Holistic AI Team. (1 Aug, 2024). *High-Risk AI Systems Under the EU AI Act*. Retrieved from EU AI Act: https://www.euaiact.com/blog/high-risk-ai-systems-under-the-eu-ai-act

Explosion. (n.d.). *Industrial-Strength Natural Language Processing*. Retrieved from spaCy: https://spacy.io/

Feldman, E. (15 Apr, 2025). *Implementing effective guardrails for AI agents*. Retrieved from The Source Gitlab: https://about.gitlab.com/the-source/ai/implementing-effective-guardrails-for-ai-agents/

Flinders, M., Smalley, I., & Schneider, J. (30 Apr, 2025). *AI fraud detection in banking*. Retrieved from IBM: https://www.ibm.com/think/topics/ai-fraud-detection-in-banking

Fortinet. (2025). *What Is A Message Authentication Code?* Retrieved from Fortinet: https://www.fortinet.com/resources/cyberglossary/message-authentication-code

Gabarda, F. C. (1 Jul, 2025). *Model Context Protocol (MCP): Understanding security risks and controls*. Retrieved from Red Hat Blog: https://www.redhat.com/en/blog/model-context-protocol-mcp-understanding-security-risks-and-controls

GitHub. (28 Nov, 2022). *Rate limits for the REST API*. Retrieved from GitHub Docs: https://docs.github.com/en/rest/using-the-rest-api/rate-limits-for-the-rest-api?apiVersion=2022-11-28

GitHub. (n.d.). *Dependabot quickstart guide*. Retrieved from GitHub Docs: https://docs.github.com/en/code-security/getting-started/dependabot-quickstart-guide

GitLab. (n.d.). *Dependency Scanning*. Retrieved from GitLab Docs: https://docs.gitlab.com/user/application_security/dependency_scanning/

GitLab. (n.d.). *What is version control?* Retrieved from GitLab: https://about.gitlab.com/topics/version-control/

Gittlen, S. (2 May, 2024). *The ultimate guide to SBOMs.* Retrieved from GitLab: https://about.gitlab.com/blog/the-ultimate-guide-to-sboms/

Google. (n.d.). *About IAM authentication.* Retrieved from Google Cloud: https://cloud.google.com/memorystore/docs/valkey/about-iam-auth

Google. (n.d.). *Cutom Search JSON API.* Retrieved from https://developers.google.com/custom-search/v1/overview

Google LLC. (9 Apr, 2025). *A2A and MCP: Complementary Protocols for Agentic Systems.* Retrieved from Agent2Agent (A2A) Protocol: https://a2aproject.github.io/A2A/latest/topics/a2a-and-mcp/#how-a2a-and-mcp-complement-each-other

Google LLC. (9 Apr, 2025). *What is A2A?* Retrieved from Agent2Agent (A2A) Protocol: https://a2aproject.github.io/A2A/latest/topics/what-is-a2a/

Google. (n.d.). *Secret Manager overview.* Retrieved from Google Cloud: https://cloud.google.com/secret-manager/docs/overview

GovTech Singapore (AI Practice). (Jul, 2025). *Agentic Risk & Capability Framework.* Retrieved from https://govtech-responsibleai.github.io/agentic-risk-capability-framework/

Guardrails AI. (n.d.). *Guardrails AI.* Retrieved from Github: https://github.com/guardrails-ai/guardrails

Harang, R., & Sablotny, M. (25 Feb, 2025). *Agentic Autonomy Levels and Security*. Retrieved from NVIDIA DEVELOPER: https://developer.nvidia.com/blog/agentic-autonomy-levels-and-security/

HashiCorp. (n.d.). *Vault.* Retrieved from GitHub: https://github.com/hashicorp/vault

Helicone Inc. (n.d.). *Helicone.* Retrieved from GitHub: https://github.com/Helicone/helicone

Hou, X., Zhao, Y., Wang, S., & Wang, H. (Apr, 2025). *Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions.* Retrieved from https://arxiv.org/pdf/2503.23278

Huang, K. (22 Dec, 2024). *7 Layered Agentic AI Reference Architecture.* Retrieved from Medium: https://kenhuangus.medium.com/7-layered-agentic-ai-reference-architecture-20276f83b7ee

Huang, K. (02 Jun, 2025). *Agentic AI Threat Modeling Framework: MAESTRO*. Retrieved from Cloud Security Alliance: https://cloudsecurityalliance.org/blog/2025/02/06/agentic-ai-threat-modeling-framework-maestro

Huang, K. (30 Apr, 2025). *Threat Modeling Google's A2A Protocol with the MAESTRO Framework*. Retrieved from Cloud Security Alliance: https://cloudsecurityalliance.org/blog/2025/04/30/threat-modeling-google-s-a2a-protocol-with-the-maestro-framework

Huang, K., Narajala, V. S., Yeoh, J., Ross, J., Raskar, R., Harkati, Y., . . . Hughes, C. (28 May, 2025). *A Novel Zero-Trust Identity Framework for Agentic AI: Decentralized Authentication and Fine-Grained Access Control*. Retrieved from arxiv: https://arxiv.org/abs/2505.19301

Hugging Face. (n.d.). *Daily Papers: Instruction Following Score*. Retrieved from https://huggingface.co/papers?q=Instruction%20Following%20Score%20(IFS)

Hugging Face. (n.d.). *Model Cards*. Retrieved from Hugging Face: https://huggingface.co/docs/hub/en/model-cards

Hugging Face. (n.d.). *Pickle Scanning*. Retrieved from Hugging Face: https://huggingface.co/docs/hub/security-pickle

IETF OAuth Working Group. (n.d.). *OAuth Scopes*. Retrieved from OAuth 2.0: https://oauth.net/2/scope/

Invariant. (1 Apr, 2025). *MCP Security Notification: Tool Poisoning Attacks*. Retrieved from Invariantlabs: https://invariantlabs.ai/blog/mcp-security-notification-tool-poisoning-attacks

Jambrecic, R. (7 Jan, 2025). *Tools Dependency Injection*. Retrieved from AG2: https://docs.ag2.ai/latest/docs/blog/2025/01/07/Tools-Dependency-Injection/

Jarvis, C. (19 Dec, 2023). *How to implement LLM guardrails*. Retrieved from OpenAI Cookbook: https://cookbook.openai.com/examples/how_to_use_guardrails

Jin, X., Guo, Z., Zhang, P., Lu, S., Dai, W., Nujibieke, . . . Li, G. (2 Feb, 2025). *Bridging Minds and Machines: Agents with Human-in-the-Loop – Frontier Research, Real-World Impact, and Tomorrow's Possibilities*. Retrieved from Camel-AI: https://www.camel-ai.org/blogs/human-in-the-loop-ai-camel-integration

Kartha, V. (3 May, 2024). *Self-Reflecting AI Agents Using LangChain*. Retrieved from Medium: https://vijaykumarkartha.medium.com/self-reflecting-ai-agents-using-langchain-d3a93684da92

Kumar, A., Roh, J., Naseh, A., Karpinska, M., Iyyer, M., Houmansadr, A., & Bagdasarian, E. (5 Feb, 2025). *OverThink: Slowdown Attacks on Reasoning LLMs*. Retrieved from arxiv: https://arxiv.org/abs/2502.02542

LangChain. (2025). *How to pass run time values to tools*. Retrieved from LangChain: https://python.langchain.com/docs/how_to/tool_runtime/

LangChain. (2025). *LangMem*. Retrieved from LangGraph: https://langchain-ai.github.io/langmem/

LangChain. (n.d.). *E2B Data Analysis*. Retrieved from LangChain: https://python.langchain.com/docs/integrations/tools/e2b_data_analysis/

LangChain. (n.d.). *LangGraph interrupt: Making it easier to build human-in-the-loop agents with interrupt*. Retrieved from LangChain: https://blog.langchain.com/making-it-easier-to-build-human-in-the-loop-agents-with-interrupt/

Langfuse. (n.d.). *Open Source LLM Engineering Platform*. Retrieved from Langfuse: https://langfuse.com/

LangSmith. (n.d.). *Ship agents with confidence*. Retrieved from LangSmith: https://www.langchain.com/langsmith

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., . . . Tang, J. (25 Oct, 2023). *AgentBench: Evaluating LLMs as Agents*. Retrieved from arxiv: https://arxiv.org/abs/2308.03688

Lucas, J. (16 Dec, 2024). *Sandboxing Agentic AI Workflows with WebAssembly*. Retrieved from NVIDIA Developer: https://developer.nvidia.com/blog/sandboxing-agentic-ai-workflows-with-webassembly/

Meadows, J., & Chang, A. (27 Mar, 2024). *How to choose a known, trusted supplier for open source software*. Retrieved from Google Cloud: https://cloud.google.com/blog/products/identity-security/how-to-choose-a-known-trusted-supplier-for-open-source-software

Meta Llama. (n.d.). *Purple Llama*. Retrieved from Github: https://github.com/meta-llama/PurpleLlama

Microsoft AI Red Team. (2024). *PyRIT*. Retrieved from https://azure.github.io/PyRIT/

Microsoft Azure. (5 Jul, 2024). *What is an IP based access control list (ACL)?* Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/azure/virtual-network/ip-based-access-control-list-overview

Microsoft. (n.d.). *Presidio: Data Protection and De-identification SDK*. Retrieved from Microsoft Presidio: https://microsoft.github.io/presidio/

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., . . . Gebru, T. (14 Jan, 2019). *Model Cards for Model Reporting*. Retrieved from arxiv: https://arxiv.org/abs/1810.03993

MITRE. (n.d.). *ATLAS Matrix*. Retrieved from MITRE ATLAS: https://atlas.mitre.org/matrices/ATLAS

MITRE. (n.d.). *Supply Chain Security Framework*. Retrieved from MITRE System of Trust: https://sot.mitre.org/framework/system_of_trust.html

Mu, N., Lu, J., Lavery, M., & Wagner, D. (15 Feb, 2025). *A Closer Look at System Prompt Robustness*. Retrieved from https://arxiv.org/abs/2502.12197

Murúa, T. (1 May, 2025). *RAG and the value of grounding*. Retrieved from elastic search labs: https://www.elastic.co/search-labs/blog/grounding-rag

National Security Agency. (5 Mar, 2024). *Advancing Zero Trust Maturity Throughout the Network and Environment Pillar*. Retrieved from NSA: https://media.defense.gov/2024/Mar/05/2003405462/-1/-1/0/CSI-ZERO-TRUST-NETWORK-ENVIRONMENT-PILLAR.PDF

Nelson, A., Rekhi, S., Souppaya, M., & Scarfone, K. (n.d.). *Special Publication 800-61r3 Incident Response Recommendationsand Considerations for CybersecurityRisk Management*. Retrieved from NIST: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r3.pdf

NIST. (14 Jan, 2025). *Cryptographic Standards and Guidelines*. Retrieved from NIST: https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines

NVIDIA. (2025). *About NeMo Guardrails*. Retrieved from NVIDIA: https://docs.nvidia.com/nemo/guardrails/latest/index.html#

NVIDIA. (n.d.). *garak, LLM vulnerability scanner*. Retrieved from Github: https://github.com/NVIDIA/garak

OpenAI. (2025). *Guardrails*. Retrieved from OpenAI Agents SDK: https://openai.github.io/openai-agents-python/guardrails/

OpenAI. (n.d.). *Moderation*. Retrieved from https://platform.openai.com/docs/guides/moderation

OpenJS Foundation. (n.d.). *ESLint*. Retrieved from GitHub: https://github.com/eslint/eslint

OWASP. (22 Apr, 2025). *OWASP Gen AI Security Project - Multi-Agentic system Threat Modelling Guide*. Retrieved from https://genai.owasp.org/resource/multi-agentic-system-threat-modeling-guide-v1-0/

OWASP. (28 Jun, 2025). *OWASP Gen AI Security Project - Securing Agentic Applications Guide*. Retrieved from https://genai.owasp.org/resource/securing-agentic-applications-guide-1-0/

OWASP. (17 Feb, 2025). *OWASP Top 10 for LLMs - Agentic AI - Threats and Mitigations*. Retrieved from https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/

OWASP. (23 Jan, 2025). *OWASP Top 10 for LLMs - GenAI Red Teaming Guide*. Retrieved from https://genai.owasp.org/resource/genai-red-teaming-guide/

OWASP. (n.d.). *Authentication Cheat Sheet*. Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

OWASP. (n.d.). *Authorization Cheat Sheet*. Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

OWASP. (n.d.). *Content Security Policy Cheat Sheet.* Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html

OWASP. (n.d.). *Docker Security Cheat Sheet.* Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Docker_Security_Cheat_Sheet.html

OWASP. (n.d.). *File Upload Cheat Sheet.* Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html

OWASP. (n.d.). *Input Validation Cheat Sheet.* Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

OWASP. (n.d.). *LLM Prompt Injection Prevention Cheat Sheet.* Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/LLM_Prompt_Injection_Prevention_Cheat_Sheet.html

OWASP. (n.d.). *Secrets Management Cheat Sheet.* Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html

OWASP. (n.d.). *XSS Filter Evasion Cheat Sheet.* Retrieved from OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html

PagerDuty. (n.d.). *Incident Response.* Retrieved from PagerDuty: https://response.pagerduty.com/

Perrone, P. (15 Apr, 2025). *MCP Is a Security Nightmare — Here's How the Agent Security Framework Fixes It.* Retrieved from Medium: https://medium.com/data-science-collective/mcp-is-a-security-nightmare-heres-how-the-agent-security-framework-fixes-it-fd419fdfaf4e

Perrot, C., Tanke, M. L., Roy, M., & Sachs, R. (9 Apr, 2025). *Implement human-in-the-loop confirmation with Amazon Bedrock Agents.* Retrieved from AWS: https://aws.amazon.com/blogs/machine-learning/implement-human-in-the-loop-confirmation-with-amazon-bedrock-agents/

Personal Data Protection Commission Singapore. (1 Mar, 2024). *Advisory Guidelines on use of Personal Data in AI Recommendation and Decision Systems.* Retrieved from PDPC: https://www.pdpc.gov.sg/guidelines-and-consultation/2024/02/advisory-guidelines-on-use-of-personal-data-in-ai-recommendation-and-decision-systems

Personal Data Protection Commission Singapore. (24 Jul, 2024). *Guide to Basic Anonymisation*. Retrieved from PDPC: https://www.pdpc.gov.sg/-/media/files/pdpc/pdf-files/advisory-guidelines/guide-to-basic-anonymisation-(updated-24-july-2024).pdf

Personal Data Protection Commission Singapore. (14 Dec, 2024). *Guide to Data Protection Practices for ICT Systems*. Retrieved from PDPC: https://www.pdpc.gov.sg/-/media/files/pdpc/pdf-files/other-guides/tech-omnibus/guide-to-data-protection-practices-for-ict-systems.pdf

Promptfoo. (n.d.). *Promptfoo: LLM evals & red teaming*. Retrieved from Github: https://github.com/promptfoo/promptfoo

Python Code Quality Authority. (n.d.). *Bandit*. Retrieved from https://bandit.readthedocs.io/en/latest/

Rasmussen, P., Paliychuk, P., Beauvais, T., Ryan, J., & Chalef, D. (20 Jan, 2025). *Zep: A Temporal Knowledge Graph Architecture for Agent Memory*. Retrieved from arxiv: https://arxiv.org/abs/2501.13956

Rehberger, J. (2 May, 2025). *MCP: Untrusted Servers and Confused Clients, Plus a Sneaky Exploit*. Retrieved from Embrace The Red: https://embracethered.com/blog/posts/2025/model-context-protocol-security-risks-and-exploits/

Rehberger, J. (n.d.). *Trust No AI: Prompt Injection Along The CIA Security Triad*. Retrieved from https://arxiv.org/pdf/2412.06090

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (Aug, 2020). *Special Publication 800-207 Zero Trust Architecture*. Retrieved from NIST: https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-207.pdf

Sapkota, R., Roumeliotis, K. I., & Karkee, M. (28 May, 2025). *AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges*. Retrieved from arxiv: https://arxiv.org/abs/2505.10468

Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (n.d.). *Special Publication 800-115 Technical Guide to Information Security Testing and Assessment*. Retrieved from NIST: https://csrc.nist.gov/pubs/sp/800/115/final

Semgrep, Inc. (n.d.). *Semgrep*. Retrieved from GitHub: https://github.com/semgrep/semgrep

Shah, D. (4 Jun, 2025). *Introduction to Training Data Poisoning: A Beginner's Guide*. Retrieved from Lakera: https://www.lakera.ai/blog/training-data-poisoning

Snyk. (Jun, 2025). *Snyk Open Source*. Retrieved from Snyk User Docs: https://docs.snyk.io/scan-with-snyk/snyk-open-source

Souppaya, M., Scarfone, K., & Dodson, D. (Feb, 2022). *Special Publication 800-218 Secure Software Development Framework (SSDF) Version 1.1*. Retrieved from NIST: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-218.pdf

Stryker, C. (2025). *What is agentic AI?* Retrieved from IBM: https://www.ibm.com/think/topics/agentic-ai

Surapaneni, R., Jha, M., Vakoc, M., & Segal, T. (9 Apr, 2025). *Announcing the Agent2Agent Protocol (A2A)*. Retrieved from Google for Developers: https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/

The Linux Foundation. (n.d.). *Supply-chain Levels for Software Artifacts*. Retrieved from SLSA: https://slsa.dev/

The OWASP Foundation. (July, 2017). *OWASP Code Review Guide*. Retrieved from OWASP: https://owasp.org/www-project-code-review-guide/

The Vulnerable MCP Project. (2025). *The Vulnerable MCP Project*. Retrieved from https://vulnerablemcp.info/

tl;dr sec. (Feb, 2025). *prompt-injection-defenses*. Retrieved from GitHub: https://github.com/tldrsec/prompt-injection-defenses

traceloop. (n.d.). *OpenLLMetry*. Retrieved from GitHub: https://github.com/traceloop/openllmetry

UK AI Secuity Institute, Arcadia Impact, Vector Institute. (n.d.). *Inspect Evals*. Retrieved from https://ukgovernmentbeis.github.io/inspect_evals/

UK National Cyber Security Centre. (12 Oct, 2023). *Supply Chain Guidance*. Retrieved from NCSC: https://www.ncsc.gov.uk/collection/supply-chain/guidance

UK National Cyber Security Centre. (7 Nov, 2024). *Vulnerability Disclosure Toolkit*. Retrieved from NCSC: https://www.ncsc.gov.uk/information/vulnerability-disclosure-toolkit

Ul Muram, F., Tran, H., & Zdun, U. (1 Apr, 2017). *Systematic Review of Software Behavioral Model Consistency Checking*. Retrieved from https://www.researchgate.net/publication/316938485_Systematic_Review_of_Software_Behavioral_Model_Consistency_Checking

University of the Sunshine Coast Australia. (n.d.). *What are credible sources?* Retrieved from https://libguides.usc.edu.au/credible/web

Wang, C. L., Singhal, T., Kelkar, A., & Tuo, J. (8 Aug, 2025). *MI9 - Agent Intelligence Protocol: Runtime Governance for Agentic AI Systems*. Retrieved from arxiv: https://arxiv.org/abs/2508.03858

Wickramasinghe, S. (18 Mar, 2025). *IT & System Availability + High Availability: The Ultimate Guide*. Retrieved from Splunk Blogs: https://www.splunk.com/en_us/blog/learn/availability.html

Xu, F. F., Song, Y., Li, B., Tang, Y., Jain, K., Bao, M., . . . Neubig, G. (19 May, 2025). *TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks*. Retrieved from https://the-agent-company.com/

Yuan, Y., Jiao, W., Wang, W., Huang, J.-t., Xu, J., Liang, T., . . . Tu, Z. (23 May, 2025). *Refuse Whenever You Feel Unsafe: Improving Safety in LLMs via Decoupled Refusal Training*. Retrieved from https://arxiv.org/abs/2407.09121

Zaharia, M. A., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., . . . Zumar, C. (2018). *Accelerating the Machine Learning Lifecycle with MLflow*. Retrieved from GitHub: https://github.com/mlflow/mlflow

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., . . . Neubig, G. (16 Apr, 2024). *WebArena: A Realistic Web Environment for Building Autonomous Agents*. Retrieved from https://webarena.dev/