# Generative AI in Cybersecurity

Module 4A: Advanced agents and agentic architectures

Henning Thomsen

hth@ucn.dk

12. May 2025

Pba IT-security @ UCN

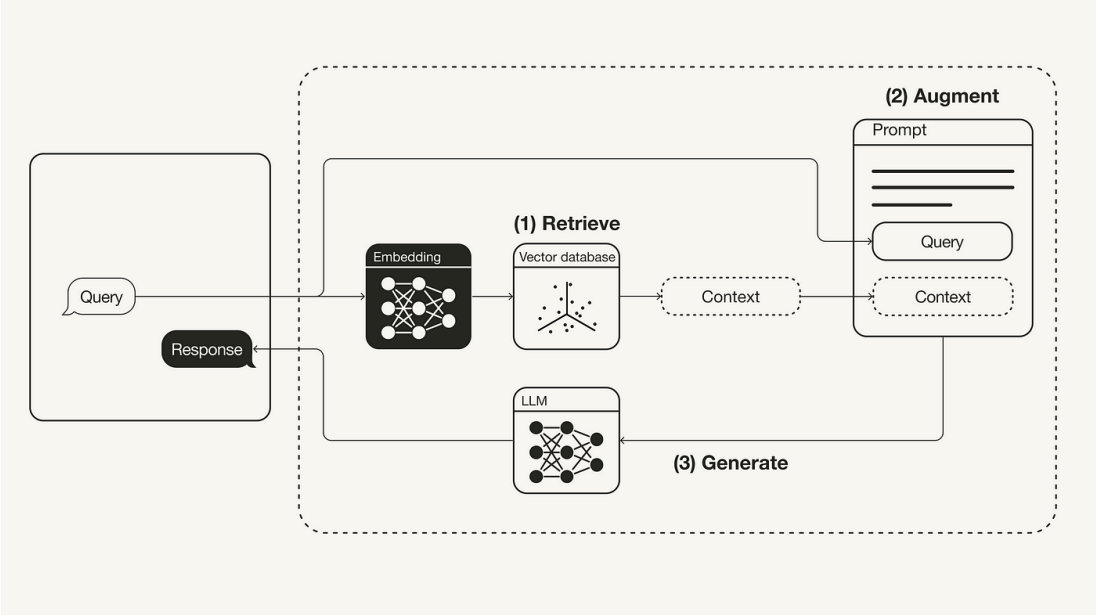
# Agenda

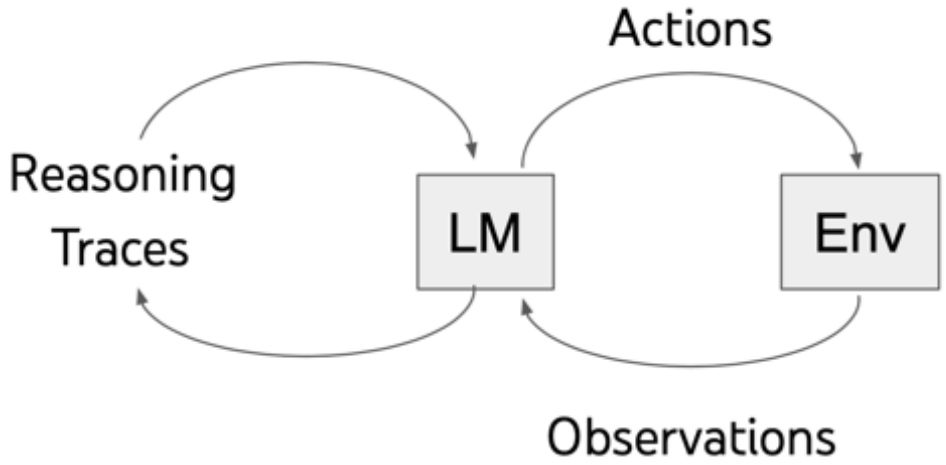
- Tool-calling agents
  - Reasoning
  - Parallelizability
- Plan-and-execute agents
- Microsoft Entra ID agent example
- Excessive agency (afternoon)
- Exam topics and extended abstract (afternoon)

# Tool-calling agents

Reasoning and action, planning and paralellizability

# RAG vs. Tool-calling

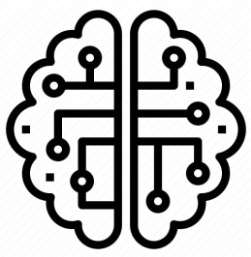



**ReAct** (Reason + Act)

# Tool-calling

User



LLM

Tools

Examples: 04_tool_call_prompt.py and 04_tool_call_llm.py

# Agent with several tools

- Suppose we want the agent to have access to two tools
  - Identify IP address
  - Identify Hash

- How should we do this?
  - 04_tool_calling_two_tools.py

- What *type* of tool-calling is this?

# Agent with several tools

- Suppose we want the agent to have access to two tools
  - Identify IP address metadata (location, registration information etc.)
  - Identify whether IP address is flagged as malicious

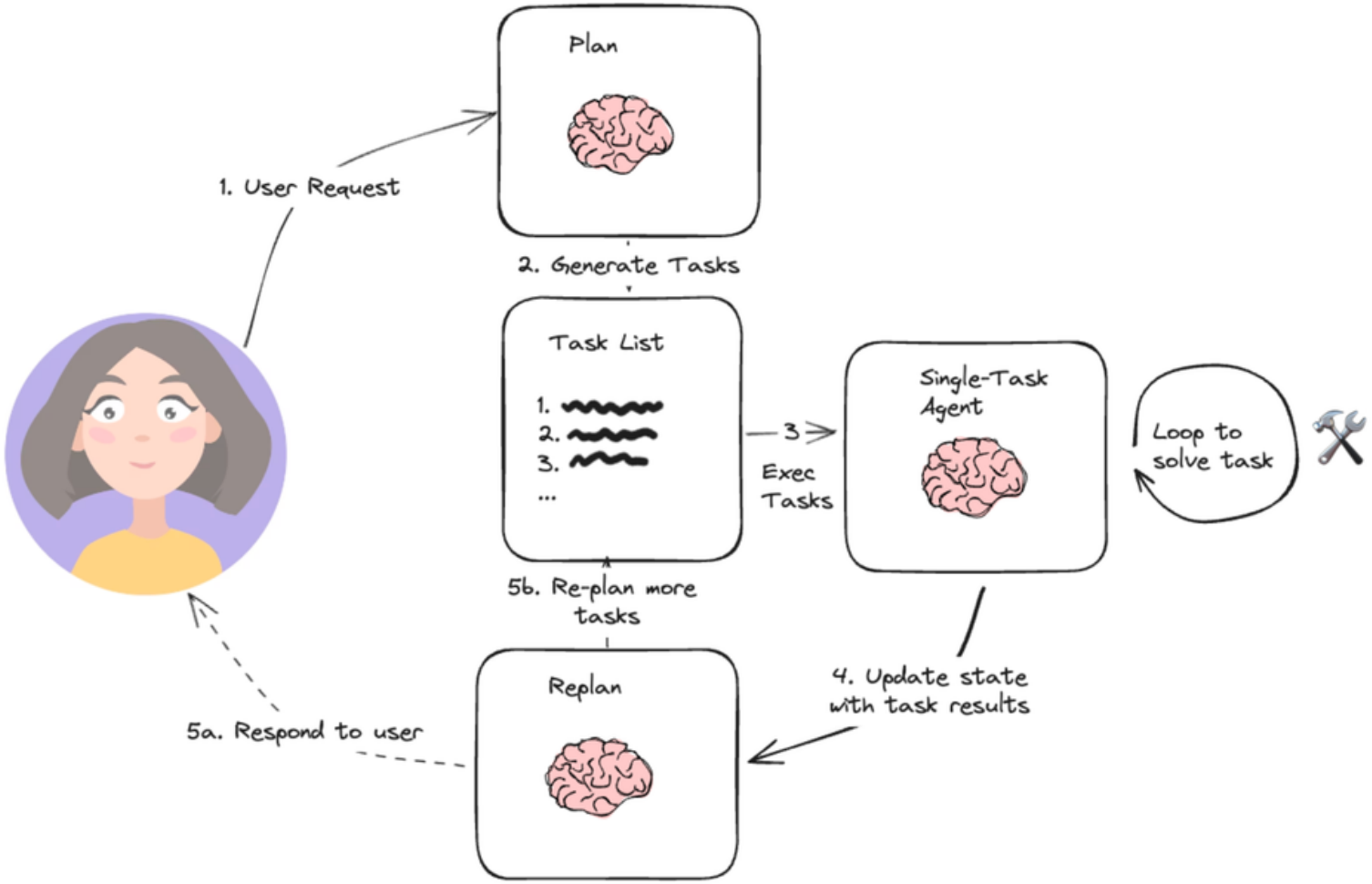
- How should we do this?
  - 04_tool_calling_identify_ip_address.py
  - 04_tool_calling_identify_ip_address_ipdb.py

- What *type* of tool-calling is this?

# Limitations with ReAct agents?

- Can you identify some limitations?

# Plan and execute agents




From https://blog.langchain.dev/planning-agents/

# Plan and execute prompt

- Core idea: Mimic behaviour of plan-and-execute agent
  - Plan: First prompt the LLM to break down a task into steps.
  - Execute: Then walk through each step one-by-one, prompting the LLM or calling tools manually.
- 04_plan_and_execute_prompt.py

# Plan and execute prompt

```
You are a cybersecurity analyst tasked with investigating digital indicators using external
tools and data sources.

OBJECTIVE:
{input}

INITIAL PLAN:
{plan}

INVESTIGATION LOG:
{past_steps}

YOUR TASK:
- If the objective has been fully addressed, provide a final summary or conclusion.
- If further steps are needed, return only the steps that remain.
- Ensure that each step includes all necessary information (e.g., IP, domain, source).
- Avoid repeating previously completed steps.
- Be precise and concise, as if updating an analyst runbook.

Think like a methodical investigator. Each step should move the investigation forward.
```

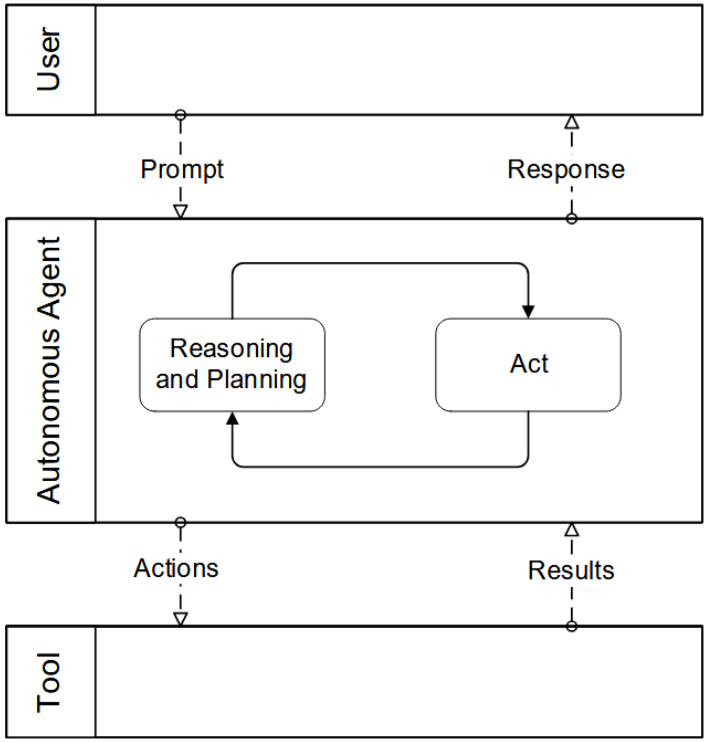
# Microsoft Entra ID Administration Agent

- Uses LLMs for user management in a Microsoft environment
  - https://github.com/OTRF/MEAN

**Exploring Applicability of LLM-Powered Autonomous Agents to Solve Real-life Problems**

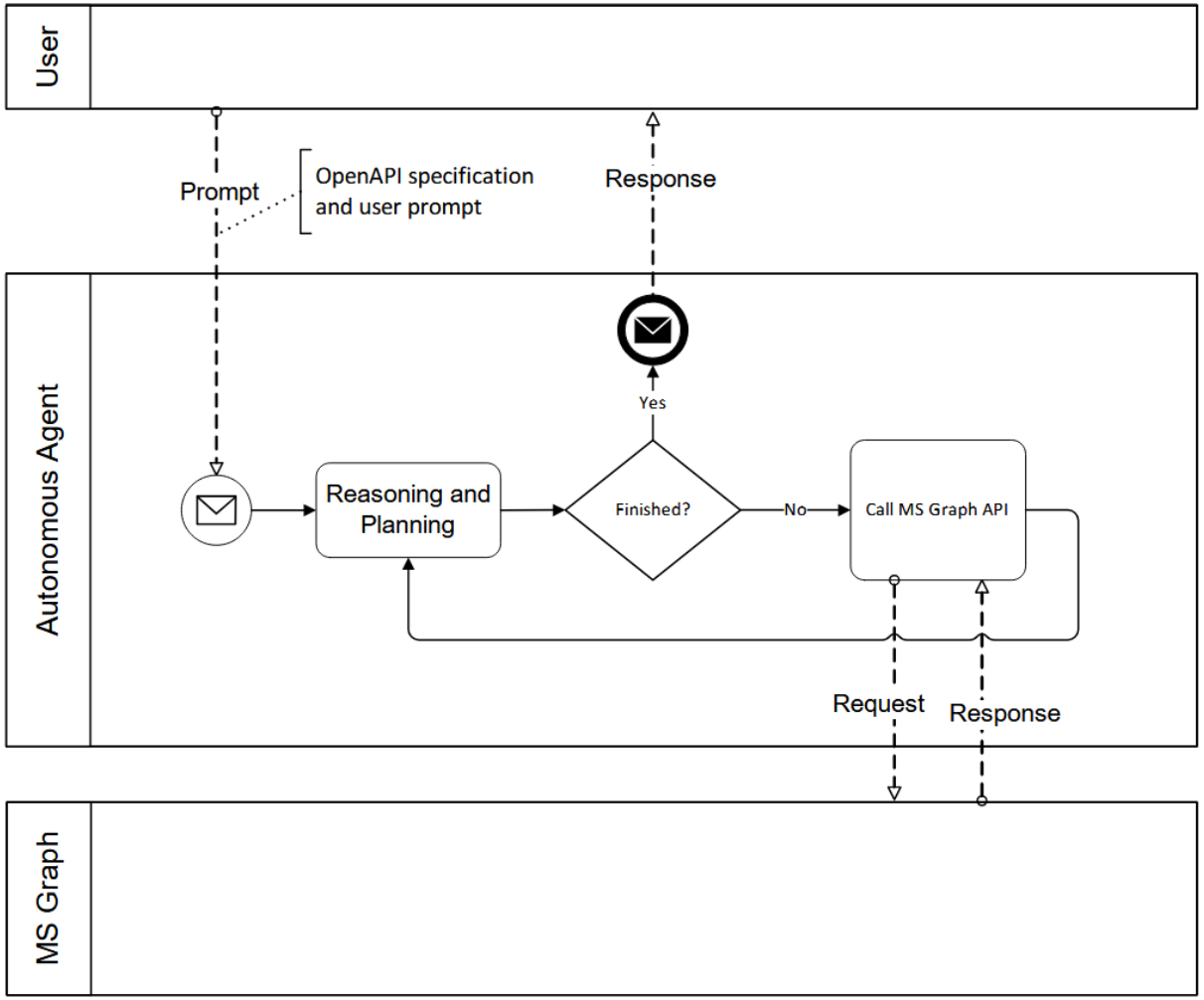
*MEAN: Microsoft Entra ID Administration Agent*

Roberto Rodriguez[1], Nestori Syynimaa[1,2]
[1]Microsoft Security Research (MSecR)
[2]Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland
{rorodri, nsyynimaa}@microsoft.com




From the paper

# Microsoft Entra ID Administration Agent




From the paper

# OpenAPI

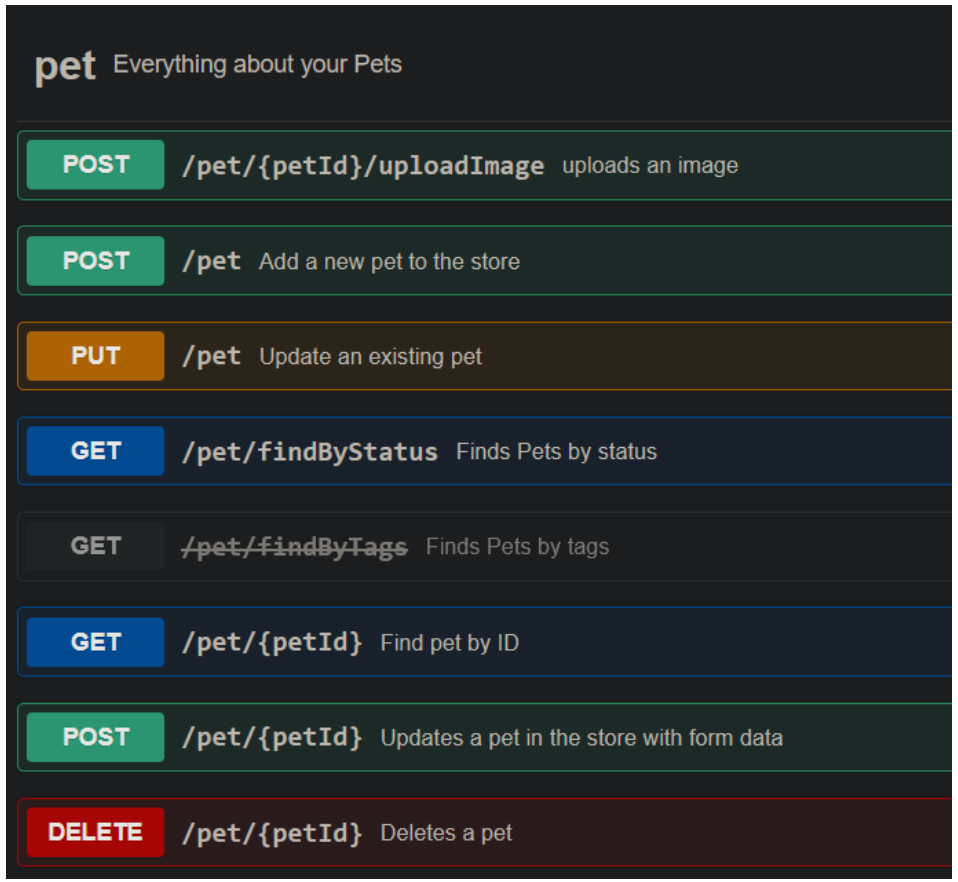
- Standard for describing RESTful APIs
- Machine-readable format (typically JSON or YAML)
- Describes endpoints, parameters, request/response schemas
- Enables automated tool usage by **agents**

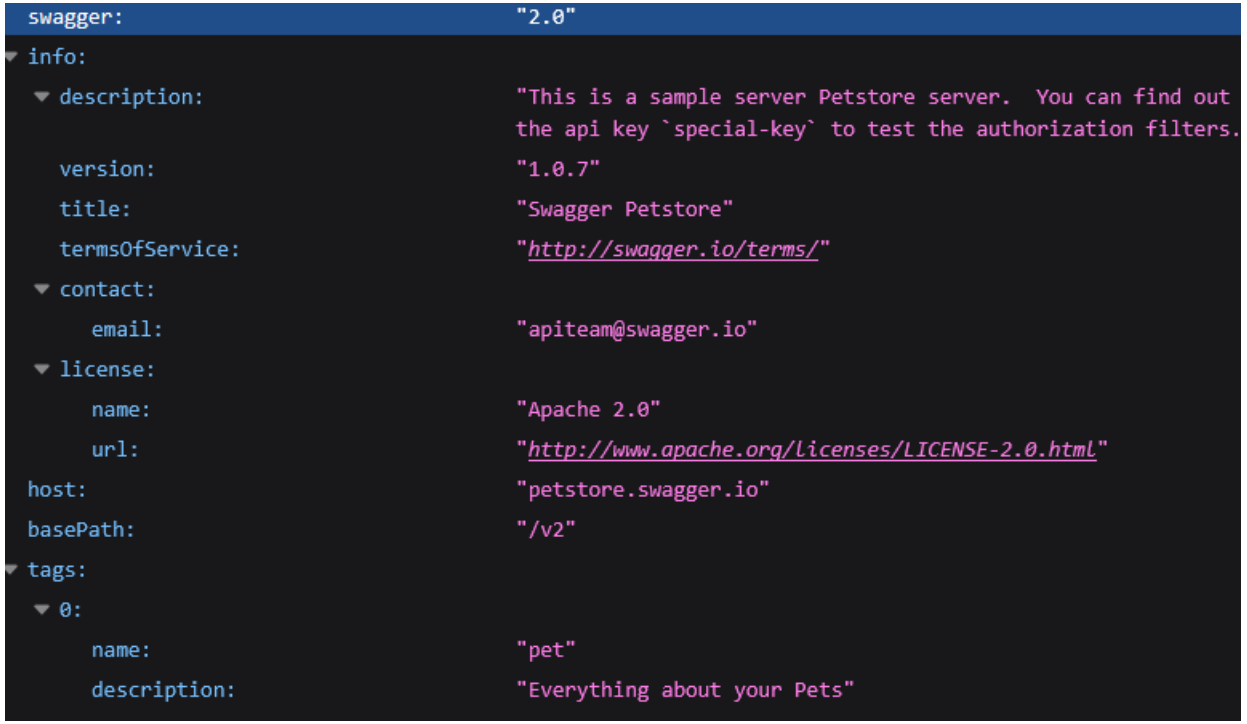
```yaml
openapi: 3.0.0
info:
  version: 1.0.0
  title: Sample API
  description: A sample API to illustrate OpenAPI concepts
paths:
  /list:
    get:
      description: Returns a list of stuff
      responses:
        '200':
          description: Successful response
```

From https://support.smartbear.com/swaggerhub/docs/en/get-started/openapi-3-0-tutorial.html

# OpenAPI - example

- Swagger Petstore: https://petstore.swagger.io/#/



# OpenAPI specification

```yaml
openapi: 3.0.1
info:
  title: Users
  version: v1.0
servers:
  - url: https://graph.microsoft.com/v1.0/
    description: Core
paths:
  /users:
    get:
      tags:
        - users.user
      summary: List users
      description: List properties and relationships of the user objects.
      externalDocs:
        description: Find more info here
        url: https://learn.microsoft.com/graph/api/intune-onboarding-user-list?view=g
      operationId: user_ListUser
      parameters:
        - name: ConsistencyLevel
          in: header
          description: 'Indicates the requested consistency level. Documentation URL:
```

From https://github.com/OTRF/MEAN

# Agent Execution

```
user_query = (
    "Show lastPasswordChangeDateTime and jobTitle of users."
)
msgraph_agent.invoke(user_query)
```

```
> Entering new AgentExecutor chain...
Action: api_planner
Action Input: Show lastPasswordChangeDateTime and jobTitle of users.
Observation: 1. GET /users to retrieve a list of user objects.
2. For each user object, extract the 'lastPasswordChangeDateTime' and 'jobTitle' properties.
Thought:I'm ready to execute the API calls.
Action: api_controller
Action Input: 1. GET /users to retrieve a list of user objects.
2. For each user object, extract the 'lastPasswordChangeDateTime' and 'jobTitle' properties.

> Entering new AgentExecutor chain...
I need to make a GET request to the /users endpoint. I will set the $top parameter to 50 to li
ll also set the $select parameter to 'lastPasswordChangeDateTime' and 'jobTitle' to only retri
ser. I don't need to use the $search, $filter, $orderby, or $expand parameters for this reques
```

From https://github.com/OTRF/MEAN